الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Faculté des Sciences de l'Ingéniorat

Département d'Informatique

THESE

Présentée en vue de l'obtention du diplôme de

DOCTORAT 3ième cycle

Utilisation des Algorithmes Evolutionnaires dans la Conception des Logiciels

Filière: Informatique

Option: Ingénierie des Logiciels Complexes

Par:

Mme BEDBOUDI Amina

Directeur de Thèse

Mr **Mohamed Cherif BOURAS** MCA, Université Badji Mokhtar –Annaba-

Devant le jury

Président :	Salim GHANEMI	Prof,	Université Badji Mokhtar –Annaba-
Examinateurs :	Djamel MESLATI	Prof,	Université Badji Mokhtar –Annaba-
	Mohamed REDJIMI	Prof,	Université 20 Août 1955 –Skikda-

Année 2018/2019

REMERCIEMENT

Merci Allah (notre Dieu) de m'avoir guidé vers la religion et la science et m'a donné la capacité d'écrire et de réfléchir. La science sans religion est boiteuse et la religion sans science est aveugle.

Je tiens à remercier sincèrement le Docteur Cherif Bouras, qui, en tant que Directeur de thèse, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce travail, ainsi que pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer.

Je tiens aussi à remercier professeur Kimour Mohamed Tahar qui m'a donné de précieux conseils et m'a guidé dans l'élaboration de cette thèse.

J'adresse mes remerciements les plus sincères au professeur Meslati Djamel, Professeur Hayet Merouani, professeur Ghanemi Salim, professeur Ghoualmi, professeur Mohamed-BenAli Yamina, et Docteur Benouhaiba.

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

Je tiens à remercier vivement et chaleureusement tout l'encadrement de la spécialité du Master ILC du département d'Informatique, tous les membres du CSD et du CSF, qui m'ont donné une chance inestimable de reprendre le travail avec force, après un dur moment de désespoir.

J'adresse mes vifs remerciements à tous les enseignants du département d'informatique.

DEDICACE

À ma chère mère à, mon cher père, les deux seules personnes qui m'ont donné le soutien matériel et moral jusqu'à la dernière minute, je dédie cette thèse.

À mon mari, à ma petite famille, mes frères et mes sœurs, et à tous ceux qui ont participé de près ou de loin à l'aboutissement de cette thèse.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION GENERALE

Les algorithmes évolutionnaires constituent une classe d'algorithmes dont le principe s'appuyé sur la théorie de l'évolution pour la résolution de problèmes divers; Ce sont des techniques bio-inspirées de calcul [Siarry, 2014]. Le principe de base de leur fonctionnement est de faire évoluer une population de solutions à un problème donné, en vue de trouver les meilleurs résultats. Etant donné qu'ils utilisent itérativement des processus aléatoires, ils sont appelés algorithmes stochastiques.

La grande majorité de ces méthodes sont utilisées pour résoudre des problèmes d'optimisation, elles sont en cela des méta-heuristiques, bien que le cadre général ne soit pas nécessairement dédié aux algorithmes d'optimisation au sens strict [Yu et al, 2008]. On les classe également parmi les méthodes d'intelligence computationnelle.

Ces dernières années, de nombreux algorithmes de résolution de problèmes ont été proposés et ils ont fait leurs preuves dans différents domaines de l'ingénierie des systèmes, en particulier dans les systèmes informatiques où l'optimisation occupe une place de choix dans le développement de logiciel.

Les algorithmes évolutionnaires sont efficacement appliqués à de larges problèmes d'optimisation dans les domaines de l'ingénierie, du marketing, de la recherche opérationnelle et des sciences sociales, tels que l'ordonnancement, la génétique, la sélection de matériaux, la conception structurelle, etc. Outre les problèmes d'optimisation mathématique [Delfour, 2012], les algorithmes évolutionnaires ont également été utilisés comme cadre expérimental dans l'évolution biologique et la sélection naturelle dans le domaine de la vie artificielle.

Quatre grandes classes de méthodes de résolutions de problèmes peuvent être définies [Allaire, 2006]. Un problème n'est pas nécessairement un problème d'optimisation, mais s'y ramène très souvent, ainsi dans les jeux ou dans le fameux problème des reines de Gauss, il est toujours possible de trouver une

fonction à minimiser. La difficulté réside le plus souvent dans la formalisation (définition de la fonction, puis définition d'un codage des solutions) :

1 - méthodes combinatoires, où un nombre déterminé d'éventuelles solutions peuvent être examinées [Bernard, 2005].

2 - méthodes constructives (exploration d'arborescence avec retour en arrière), la méthode de l'élastique pourrait être assimilée à une méthode constructive.

3 - méthodes locales (les méthodes mathématiques classiques de descente de gradient, mais aussi des heuristiques stochastiques comme le recuit simulé).

4 - méthodes évolutionnaires [Siarry, 2014], ce sont des méthodes stochastiques et globales: faisant intervenir une population de points en s'inspirant de l'évolution des espèces vivantes. Parmi ces méthodes nous pouvons distinguer celles qui utilisent des opérateurs explicites (la mutation, le croisement ...) et celles ayant implicitement des règles de transition.

La nécessité de rechercher de nouvelles techniques d'optimisation découle du fait que les techniques traditionnelles sont devenues inefficaces pour résoudre des problèmes pratiques [Angeline, 1998a].

La complexité des problèmes d'optimisation à résoudre pour des systèmes de plus en plus complexes, les nombreuses variables de conception ainsi que leurs exigences pratiques rendent ces problèmes difficiles à gérer en utilisant des méthodes d'optimisation traditionnelles [Randy et al, 2004].

Récemment, les techniques méta-heuristiques ont connu un engouement fort tant au niveau académique qu'industriel à la recherche de solutions robustes à des problèmes complexes. S'inspirant des phénomènes naturels, ces techniques permettent d'obtenir des solutions convenables à des problèmes d'optimisation NP difficiles [Mridul et al, 2015].

Cependant, ces algorithmes ont tendance à produire des solutions différentes même lorsque leurs conditions initiales restent constantes à chaque exécution et ce, en raison de leur nature aléatoire. Ils sont préférés pour ces

fonctions qui ont plusieurs optimaux locaux, car ils peuvent échapper facilement aux minimums locaux en dépit de leur lenteur de convergence [Sayadi et al, 2013].

La majorité des méta-heuristiques s'inspirent des phénomènes biologiques et physiques de la nature, telle que l'optimisation par essaim particulaires (PSO) [Cesare, 2015], basée sur le comportement d'essaim d'oiseaux ou de poissons, les algorithmes génétiques, dérivés de la théorie d'évolution de la nature et le recuit simulé fondé sur un processus de métallurgie.

L'idée fondamentale derrière ces techniques est de simuler des systèmes biologiques et physiques dans la nature, tels que l'évolution naturelle, le système immunitaire, l'intelligence d'essaim, le processus de recuit, etc., dans un algorithme numérique. Ces algorithmes diffèrent dans la façon avec laquelle ils évoluent dans l'espace de recherche, en se basant sur une stratégie associée, souvent inspirée de la nature [Parrill, 2008].

Le principal objectif de cette thèse est d'utiliser les algorithmes évolutionnaires dans les activités de conception de logiciel de façon à accélérer le processus de conception, gagner du temps, réduire les coûts de développement et augmenter la productivité en optimisant rapidement les modèles de conceptions au niveau du coût, des délais et du fonctionnement du logiciel lui-même.

A la lumière de ces éléments, nous avons mené une synthèse de l'état de l'art sur les algorithmes évolutionnaires et leur utilisation dans les processus de développement de logiciel en général. Suite à quoi, nous avons proposé un algorithme génétique modifié de façon prendre au mieux en charge le problème de clustering de données. En outre, nous avons introduit une technique de croisement à l'algorithme chauve-souris de façon à le rendre plus efficace.

Après étude des processus de conception de logiciels les plus notables [Molina et al, 2008], [Mridul et al, 2015], [Molina et al, 2008], [Jarke et al, 1984], [Dong et al, 2007], nous avons proposé une modification de processus de conception de base de données en introduisant une nouvelle technique d'optimisation de requête de jointure large dans les transactions. Cette technique

utilise notre modification de l'algorithme à évolution différentielle [Qin, 2009].

La structure de la présente thèse est la suivante :

Chapitre I présente une synthèse de l'état de l'art sur les algorithmes évolutionnaires.

Chapitre 2 est consacré à la présentation des principales applications de ces méthodes de résolution de problèmes dans les activités de conception de logiciels.

Chapitre 3 décrit une contribution portant sur une amélioration de l'algorithme de chauve-souris par hybridation avec les algorithmes génétiques.

Chapitre 4 développe une deuxième contribution portant sur le clustering des données par utilisation d'un algorithme génétique adapté.

Chapitre 5 une troisième contribution porte sur l'optimisation de requêtes de base de données on utilisant l'algorithme d'Evolution Différentielle.

Une conclusion et des perspectives sont présentées à la fin du manuscrit.

PARTIE I :

ETAT DE L'ART

CHAPITRE I:

ALGORITHMES EVOLUTIONNAIRES

1.1.Introduction

L'évolution biologique a engendré des êtres vivants autonomes extrêmement complexes qui peuvent résoudre des problèmes extraordinairement difficiles, tels que l'adaptation continuelle à un environnement (complexe, incertain et en constante transformation).

La grande variété des situations auxquelles la vie s'est adaptée montre que le processus de l'évolution est robuste et est capable de résoudre de nombreuses classes de problèmes et que des systèmes complexes et performants peuvent être élaborés.

Selon Charle Darwin [Darwin, 1859], les mécanismes à l'origine de l'évolution des êtres vivants reposent sur la compétition qui sélectionne les individus les plus adaptés à leur milieu en leur assurant une descendance, ainsi que sur la transmission aux enfants des caractéristiques utiles qui ont permis la survie des parents. Ce mécanisme d'héritage se fonde notamment sur une forme de coopération mise en œuvre par la reproduction.

De plus, nombreuses tentatives de modélisation de l'évolution ont été entreprises. Et par la suite différentes approches et Algorithmes évolutionnaires ont émergé [Siarry, 2014] tel que : La stratégie de l'évolution, La programmation évolutionnaire, Les algorithmes génétiques…etc.

Dans ce chapitre nous développons les grandes lignes de ce qu'est un Algorithme Evolutionnaire avec la détermination de leurs origines historiques, on remontant très brièvement aux racines désormais la théorie de Darwin.

Ensuite nous explorons les Méta-heuristiques Evolutionnaires les plus connues et les plus utilisées, on ce concentrant sur les Algorithmes Génétiques et par la suite nous montrons l'efficacité de ces approches et leurs mise en œuvre par un exemple illustratif classique de voyageur de commerce.

1.2. Origine des Algorithmes Evolutionnaires

Les Algorithmes Evolutionnaires sont des méthodes non déterministes, basés sur la théorie de l'évolution de Darwin par sélection naturelle [Darwin, 1859].

Holland a proposé les algorithmes génétiques écrits dans son livre en 1975 [Holland, 1975]. Il a souligné le rôle de Recombinaison génétique (souvent appelée 'le croisement').

Ingo Rechenberg et Hans-Paul Schwefel ont travaillé sur l'optimisation des formes physiques dans les fluides et, après avoir essayés une Variété de techniques classiques d'optimisation, ils sont découvert que la modification des variables physiques de façon aléatoire (en assurant que les petites modifications étaient plus fréquentes que les plus grandes) se sont révélés être une technique très efficace. Cela a donné lieu à une forme d'évolution algorithmique qu'ils ont qualifiée de stratégie évolutionnaire [Rechenberg, 1971], [Schwefel, 1974].

Lawrence Fogel a étudié l'évolution des machines à état fini par prédire les symboles généré par les processus de Markov et les séries chronologiques non stationnaires [Fogel et al, 1966]. Désormais, Comme c'est souvent le cas en science, divers scientifiques ont envisagé ou suggéré la recherche Algorithmique inspirées par l'évolution darwinienne beaucoup plus tôt. David Fogel Lawrence le fils de Fogel, offre un compte rendu détaillé de ces premiers pionniers dans son livre sur l'histoire du calcul évolutionnaire [Fogel, 1998].

Cependant, il est intéressant de noter que l'idée de l'évolution artificielle a été suggérée par l'un des fondateurs de l'informatique, Alan Turing, en 1948. Turing a écrit un essai tout en travaillant sur la construction d'un ordinateur électronique appelé le moteur de calcul automatique (ACE) au Na-Laboratoire physique national au Royaume-Uni. Son employeur était Sir Charles Darwin, petit-fils de Charles Darwin, l'auteur de 'On the Origin of Species'. Sir Charles a rejeté l'article comme un "essai scolaire"! Il a depuis été reconnu que dans l'article Turing non seulement par les réseaux de neurones artificiels proposés,

mais aussi pour les Champs de l'intelligence artificielle elle-même [Turing, 1992].

1.3. Le principe des Algorithmes Evolutionnaires

La suite de cette section va détailler les principaux algorithmes évolutionnaires, en donnant un exemple illustratif concret. Mais nous allons au préalable définir quelques notions-clés pour la compréhension du fonctionnement de ces algorithmes en pratique.

1.3.1. Le paradigme Darwinien

La théorie de Darwin lors de la conception des Algorithmes Évolutionnaires est basée sur l'idée de l'apparition d'espèces adaptées au milieu qui est la conséquence de la conjonction de deux phénomènes [Allaire, 2006] :

- La sélection naturelle imposée par le milieu (les individus les plus adaptés survivent et se reproduisent).
- Et des variations non dirigées du matériel génétique des espèces.

Ce sont ces deux principes qui sous-tendent les algorithmes évolutionnaires. Précisons tout de suite que le paradigme darwinien qui a inspiré ces algorithmes ne saurait en aucun cas être une justification pour leur emploi.

1.3.2. Les notions principales

La notion principale de l'algorithme, qui est même en fait préalable à toutes les autres, est la **représentation**, ou choix de l'espace de recherche.

Dans de nombreux cas, l'espace de recherche est totalement déterminé par le problème (i.e. c'est l'espace $\boldsymbol{O}$ sur lequel est définie la fonction objectif $\boldsymbol{J}$). Mais il est toujours possible de transporter son problème dans un espace habilement choisi ("changement de variables") dans lequel il sera plus aisé de définir des opérateurs de variations efficaces. Cet espace est alors appelé

espace **génotypique**, et l'espace de recherche initial ***O***, dans lequel est calculé la performance des individus, est alors aussi appelé espace **phénotypique**.

On peut alors répartir les diverses étapes de l'algorithme en deux groupes:

- les étapes relatives au darwinisme (la sélection et le remplacement), qui ne dépendent que des valeurs prises par ***J***, et pas de la représentation choisie, (i.e. pas de l'espace génotypique) et les étapes qui sont intimement liées à la nature de cet espace de recherche.
- l'initialisation et les opérateurs de variation sont spécifiques aux types de génotypes, mais par-contre ne dépendent pas de la fonction objectif ***J*** (c'est le principe Darwinien des variations aveugles, ou non dirigées).

Le terme de diversité génétique désigne la variété des génotypes présents dans la population. Elle devient nulle lorsque tous les individus sont identiques, on parle alors de convergence de l'algorithme. Mais il est important de savoir que lorsque la diversité génétique devient très faible, il y a très peu de chances pour qu'elle augmente à nouveau. Et si cela se produit trop tôt, la convergence a lieu vers un **optimum local**.

On parle alors de convergence prématurée. Il faut donc préserver la diversité génétique, sans pour autant empêcher la convergence. Un autre point de vue sur ce problème est celui du dilemme exploration-exploitation.

A chaque étape de l'algorithme, il faut effectuer le compromis entre **explorer** l'espace de recherche, pour éviter de stagner dans un optimum local et **exploiter** les meilleurs individus obtenus, afin d'atteindre de meilleurs valeurs aux alentours. Trop d'exploitation entraîne une convergence vers un optimum local, alors que trop d'exploration entraîne la non-convergence de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploration (mais de multiples variantes d'algorithmes évolutionnaires s'écartent de ce schéma général). On peut ainsi régler les parts respectives d'exploration et d'exploitation en jouant sur les divers paramètres de

l'algorithme (probabilités d'application des opérateurs, pression de sélection, ...etc).

Cependant, il n'existe pas de règles universelles de réglages et seuls les résultats expérimentaux donnent une idée du comportement de diverses composantes des algorithmes.

1.3.3. Les Composants Evolutionnaires:

Soit à optimiser une fonction ***J*** à valeurs réelles définie sur un espace métrique **Ω**. Le parallèle avec l'évolution naturelle a entraîné l'apparition d'un vocabulaire spécifique (et qui peut paraître légèrement ésotérique) [Siarry, 2014] :

- La fonction objectif ***J*** est appelée fonction de performance, ou la fonction d'adaptation (fitness en anglais) ;
- Les points de l'espace de recherche **Ω** sont appelés des individus ;
- Les tuples d'individus sont appelés des populations ;
- On parlera d'une génération pour la boucle principale de l'algorithme.
- La population de taille fixe ***P***, à la génération ***I***.

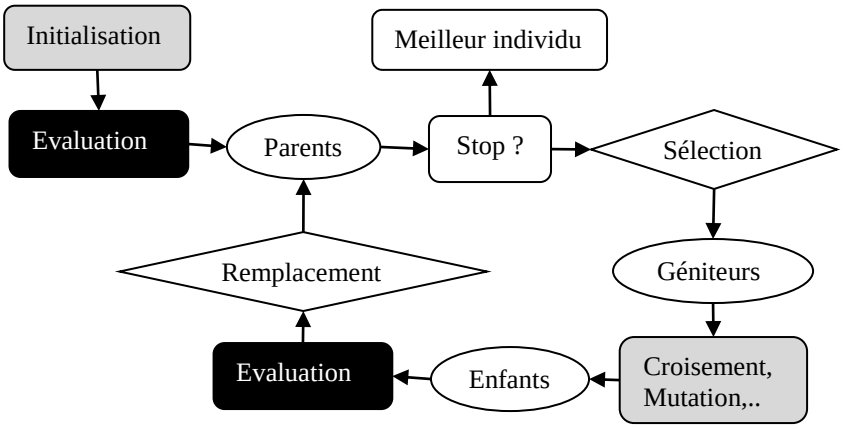


Fig.1.1.Squelette d'un Algorithme Evolutionnaire.

Le temps de l'évolution est supposé discrétisé, et on notera la pression de l'environnement qui est simulée à l'aide de la fonction objective ***J***, et les

principes darwiniens de sélection naturelle et de variation s'aveugles sont implantés dans l'algorithme de la manière suivante dans la Figure.1.1 [Siarry, 2014] montre le squelette d'un algorithme évolutionnaire :

- Initialisation de la population **Π** en choisissant **P** individus dans **Ω**, généralement par tirage aléatoire avec une probabilité uniforme sur Ω ;
- Évaluation des individus de $\mathbf{\Pi}_{00}$ (i.e. calcul des valeurs de J pour tous les individus) ; A partir de la population **Π** ;
- La génération **I** construit la population **Π** ;
- La sélection des individus les plus performants de Π au sens de J (les plus adaptés se reproduisent);
- Application (avec une probabilité donnée) des opérateurs de variation aux parents sélectionnés, ce qui génère de nouveaux individus, les enfants; On parlera de mutation pour les opérateurs unaires, et de croisement pour les opérateurs binaires (ou n-aires); A noter que cette étape est toujours stochastique :
- Évaluation des enfants ;
- Remplacement de la population **Π** par une nouvelle population créée à partir des enfants et/ou des "vieux" parents de la population Π au moyen d'une sélection darwinienne (les plus adaptés survivent).
- L'évolution stoppe quand le niveau de performance souhaité est atteint, ou qu'un nombre fixé de générations s'est écoulé sans améliorer l'individu le plus performant.

Il est important de noter que, dans les applications la plupart des problèmes numériques réels, l'essentiel du coût-calcul de ces algorithmes, provient de l'étape d'évaluation (calcul des performances) : les tailles de populations sont de l'ordre de quelques dizaines et le nombre de générations de quelques centaines, ce qui donne lieu le plus souvent à plusieurs dizaines de milliers de calculs de **J**.

Nous allons maintenant passer en revue les différentes méthodes de sélection possible, en gardant à l'esprit qu'elles sont génériques et quasiment interchangeable quel que soit le problème traité.

1.3.4. La Sélection naturelle

D'un point de vue technique, la différence essentielle entre l'étape de sélection et l'étape de remplacement est qu'un même individu peut être sélectionné plusieurs fois durant l'étape de sélection (ce qui correspond au fait d'avoir plusieurs enfants), alors que durant l'étape de remplacement, chaque individu est sélectionné une fois (et il survit) ou pas du tout (et il disparaît à jamais).

On distingue deux catégories de procédures de sélection ou de remplacement (par abus de langage, nous appellerons sélection les deux types de procédures) [Randy, 2004] :

- Les procédures déterministes.
- Les procédures stochastiques.

1.3.4.1. La Sélection déterministe

On sélectionne les meilleurs individus (au sens de la fonction objective).

Si un nombre d'individus est sélectionné, cela suppose un tri de l'ensemble de la population, Cependant cela implique un problème de temps calcul que pour les grosses tailles de population.

Les individus les moins performants sont totalement éliminés de la population, et le meilleur individu est toujours sélectionné, on dit que cette sélection est **élitiste**.

La Sélection élitiste:

Les membres les plus adaptés de chaque génération doivent être sélectionnés. La plupart des Algorithmes Evolutionnaires n'utilisent l'élitisme pur, à moins qu'ils sont modifiés de chaque génération un reproduites dans la prochaine génération.

1.3.4.2. La Sélection stochastique

En générale, les meilleurs individus sont toujours favoriser, mais de manière stochastique, ce qui laisse une chance aux individus les moins

performants. Par contre, il se peut que le meilleur individu ne soit pas sélectionné, et qu'aucun des enfants n'atteigne une performance aussi bonne que celle du meilleur parent.

Le tirage de roulette :

Elle est la plus célèbre des sélections stochastiques.

Supposant un problème de maximisation avec uniquement des performances positives, elle consiste à donner à chaque individu une probabilité d'être sélectionné proportionnellement à sa performance. Une illustration de la roulette est donnée dans la Figure.1.2. On lance la boule dans la roulette, et on choisit l'individu dans le secteur du quel la boule a fini sa course.

Fig.1.2. Sélection par Tirage de Roulette

Le tirage de roulette présente toutefois de nombreux inconvénients, en particulier reliés à l'échelle de la fonction objective, alors qu'il est théoriquement équivalent d'optimiser **J** et $\boldsymbol{\alpha J + \beta}$, pour tout $\alpha > 0$. Il est clair que le comportement de la sélection par roulette va fortement dépendre de α. dans ce cas. Sachant qu'il existe des procédures ajustant les paramètres α et β pour chaque génération (*mécanismes de mise à l'échelle*).

La sélection par rang :

Elle consiste à faire une sélection en utilisant une roulette dont les secteurs sont proportionnels aux rangs des individus (***P*** pour le meilleur, 1 pour le moins bon, pour une population de taille ***P***). La variante linéaire utilise directement le rang et les variantes polynomiales remplacent ces valeurs par $\mathbf{P}^{\boldsymbol{\alpha}}$, $\alpha > 0$. Le point essentiel de cette procédure de sélection est que les valeurs

de **J** n'interviennent plus. Seuls comptent les positions relatives aux individus entre eux. Optimiser **J** et **αJ + β** sont alors bien totalement équivalents.

La sélection par tournoi :

Elle n'utilise que des comparaisons entre individus et ne nécessite même pas de tri de la population. Elle possède un paramètre ***T*** (taille du tournoi) pour sélectionner un individu, on tire T uniformément dans la population, et on en sélectionne le meilleur de ces individus.

On choisit des sous-groupes d'individus parmi la grande population et les membres de chaque sous-groupe sont en concurrence les uns avec les autres. Un seul de chaque sous-groupe est choisi pour se reproduire.

La progéniture des individus sélectionnés de chaque génération remonte au pool génétique préexistant remplaçant ainsi les membres les moins adaptés de la génération précédente.

1.3.5. La Sélection multi-critères

Toutes les techniques de sélection présentées ci-dessus concernent le cas d'une fonction objective à valeurs réelles. Cependant, la plupart des problèmes réels sont en fait des problèmes multi-critères, que l'on cherche à optimiser simultanément. (Typiquement, maximiser la qualité d'un produit en minimisant son prix de revient).

Or les A**lgorithmes Evolutionnaires** sont une des rares méthodes d'optimisation permettant la prise en compte de telles situations. Il suffit de modifier les étapes Darwiniennes d'un Algorithme Evolutionnaire pour en faire un algorithme d'optimisation multi-critère.

1.4. Le codage binaire versus le codage réel :

La première étape de la mise en œuvre d'un algorithme évolutionnaire consiste à construire une représentation des solutions potentielles aux problèmes. , dans la conception de l'ingénierie, cela peut constituer un défi considérable [Gaffney, 2010]. Cependant, pour le problème d'optimisation

d'une fonction est simple; Utilisant la valeur de la fonction. Ensuite, un procédé de codage des solutions est choisi.

La question à savoir utiliser soit un codage binaire ou réel est peut être litigieux. La représentation traditionnelle est la représentation binaire [Holland, 1975]. Cependant, dans de nombreuses applications, le codage réel est utilisé [Michalewicz, 1996]. Différents arguments sont donnés quant à savoir si une méthode de codage binaire ou réelle devrait être adoptée. Il semble que la discrétisation de l'espace des paramètres joue un rôle dans l'efficacité de l'algorithme.

Nous considérons le cas simple d'un problème d'optimisation d'une fonction variable et la manière dont les solutions à ce problème pourraient être codées. Supposons que l'espace des paramètres soit [0, M]. Le mécanisme de codage donne lieu à des décisions possibles aussi équitables. Nous définissons un espacement d'une grille comme étant la différence entre deux décisions consécutives codées. Pour un code binaire de longueur *L* est fixé, l'espacement entre les grilles est $M / (2^l-1)$. Pour le codage binaire, nous déterminons le nombre de bits que nous devrions dans la chaîne binaire pour donner un espacement de grille. Pour le codage réel, la précision est donnée par la représentation décimale et fixe les nombres réels sur un ordinateur, par exemple 0.0000000001 pour une représentation à dix décimales.

Le code binaire est un code gris d'un système à numération binaire, si deux valeurs successives diffèrent en un seul bit. Le tableau 1.1 donne la représentation de chacun des codes pour une chaîne binaire de trois bits.

TAB.1.1 : exemple de comparaison entre le codage de trois bit

Code Réel	Code Gris	Code Binaire
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Exemple de simulation :

Nous avons simulé la fonction de minimisation d'Ackley figure (1.3), l'une des fonctions de benchmark fournies par [MacNish, 2006], à l'aide d'un algorithme génétique codé en gris. La fonction d'Ackley (1.1) est :

$$\mathbf{f(x) = -20exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2) - exp(\frac{1}{n}\sum_{i=1}^{N} cos(2\pi X_i))}} \qquad \mathbf{(1.1)}$$

Tel que **f** est dans le domaine (-6,6) de R2. Bien que n'étant pas quantitativement comparable, les résultats démontrent un comportement qualitatif similaire lorsque la valeur de la probabilité de mutation p_{Mut} est augmentée de 0,03 à 0,6; Les chemins d'échantillons affichés sont typiques, mais ne sont encore que des chemins d'échantillonnage et varient en fonction de la position de départ initiale et aussi en raison de la nature stochastique des algorithmes.

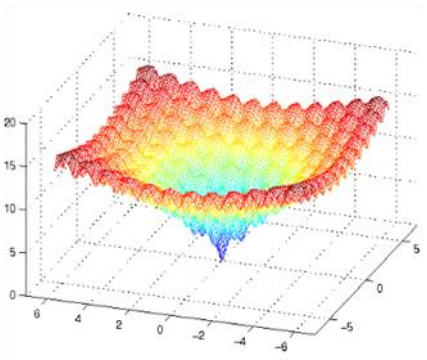


Fig.1.3 Simulation de la fonction Ackely

Par exemple, avec P_{Mut} = 0,03; Les simulations en code réel et en gris trouvent le minimum à dix décimales dans leurs espaces de grille respectifs. Lorsque P_{Mut} est augmenté, dans tous les cas, le résultat devient progressivement pire, l'algorithme codé réel apparaissant mieux au moins avec ces paramètres sur ce problème. Cela souligne essentiellement la situation que, quels que soient les algorithmes que vous employez, ils doivent être adaptés à l'application particulière. Un code binaire gris de 37 bits et un code réel de dix décimales ont été utilisés à des fins de comparaison. Cela permet une comparaison dans un certain sens, car ils ont tous deux le potentiel de réaliser un zéro à dix décimales, on effet que l'espacement de grille pour le code gris est alors que le

code réel est 10-10. La probabilité de croisement P_{Cross} = 0,6 dans toutes les simulations.

1.5. Les méta-heuristiques Evolutionnaires

1.5.1. L'Algorithme d'essaim de particules

L'optimisation par essaim de particules (PSO) (en anglais Particle Swarm Optimization) est l'un des dernières techniques méta-heuristique développées par Kennedy et Eberhart en 1995.

Dans cet algorithme, les solutions candidates d'une population, appelées des particules, coexistent et évoluent simultanément en se basant sur le partage des connaissances avec les particules voisines. Alors qu'il volait à travers l'espace de recherche. Chaque particule génère une solution utilisant son vecteur de vitesse, ainsi les particules modifies leurs vitesse pour trouver une meilleure solution (position) en appliquant leurs propres expérience de vol [Zidani, 2013]

1.5.2. L'Evolution Différentielle

L'Evolution Différentielle ED (en anglais Differential Evolution) est une version améliorée des algorithmes génétiques, proposée par Price et Storn en 1995. ED est un algorithme basé sur une population initiale comme les algorithmes génétiques, il utilise les mêmes opérateurs : croisement, mutation et sélection. La différence principale en construisant de meilleures solutions est que les algorithmes génétiques se fondent sur le croisement tandis que l'ED se fonde sur l'opération de mutation. Cette opération principale est basée sur la différence des paires de solutions aléatoirement tirées dans la population. L'algorithme utilise l'opération de mutation comme un mécanisme de recherche et l'opérateur de sélection pour diriger la convergence vers les régions éventuelles dans l'espace de recherche.

ED utilise également un croisement non uniforme qui peut prendre des paramètres de vecteur d'enfant d'un seul parent.

ED démarre avec une population initiale de NP individus générée aléatoirement. Chaque individu est représenté par un vecteur de D dimension.

A chaque génération, pour chaque individu *x* un vecteur muté est créé selon la formule (1.2) suivante :

$$\boldsymbol{v_{i.g} = x_{r1.g} + F.(x_{r2.g} - x_{r3.g})} \quad \textbf{(1.2)}$$

Où les indices aléatoires r_1, r_2, $r_3 \in \{1, 2, ..., NP\}$ sont mutuellement différents et distincts de l'indice i. $F \in [0.5, 1]$ est le coefficient de mutation qui permet de contrôler l'amplitude des mutations. Un processus de croisement discret est introduit ensuite en vue d'augmenter la diversité de la population. Il combine le vecteur muté nouvellement créé avec le vecteur parent par la règle (1.3) suivante :

$$\boldsymbol{u_{ji,g+1} = \begin{cases} v_{ji,g+1}\ si(rand_j \leq CR) ou\ (i = j)_{rand} \\ x_{ji,g}\ si(rand_j \leq CR) ou\ (i \neq j)_{rand} \end{cases}} \quad \textbf{(1.3)}$$

Où j = 1, 2, ..., D; rand ∈ [0, 1] ; CR est la probabilité de croisement∈ [0, 1] ; et le $rand_j$ est un indice aléatoire ∈ (1, 2, ..., D) qui assure que *u* a au moins un paramètre issue du vecteur v.

Finalement, l'opérateur de sélection compare la valeur de la fonction objective des deux vecteurs $u_{i,g+1}$ et $x_{i,gi,g+1}$; Le meilleur individu est sélectionné pour la population de la génération suivante. Ce processus se poursuit, génération après génération, jusqu'à atteindre le critère d'arrêt (nombre maximal de génération ou précision sur la valeur de la fonction objective).[Hachimi, 2013]

1.5.3. L'algorithme de Chauve-souris

L'algorithme des chauves-souris (en anglais Bat Algorithm) est un algorithme d'optimisation méta-heuristique développé par Xin-She Yang en 2010 [Yang, 2010]. Basé sur :

- Echolocation des chauves-souris

Les chauves-souris sont les seuls mammifères avec des ailes. Il existe 2 types de chauves-souris:

les Mégachiroptères (méga-chauves-souris) utilisent un sonar biologique, appelé écholocation en raison de leur manque de capacité de vision pour identifier et localiser leurs proies, les obstacles...Elles peuvent grâce à son son émis détecter la distance et distinguer la différence entre les aliments et les obstacles environnementaux même dans l'obscurité complète, et le deuxième type : les **Microchiroptères** (micro-chauves-souris) [Rekaby,2013] sont contrairement aux mégachiroptères qui ont une excellente vision.

- Comportement des chauves-souris

Les chauves-souris qui utilisent l'écholocation ont la capacité de déterminer

L'environnement autour d'eux, ils peuvent détecter la distance et l'orientation de la cible (proie) ainsi que l'emplacement des obstacles.

La nature et le rythme, le temps de la réponse, le volume et la différence de temps entre les deux oreilles de la chauve-souris lui permettent de dessiner l'image 3D de l'environnement dans son cerveau.

1.5.4. La Programmation génétique

La programmation génétique (PG) (en anglais Genetic Programming) peut être vue comme l'évolution artificielle de "programmes" [Miller, 2011], ces programmes étant représentés sous forme d'arbres (il existe des variantes de PG qui utilisent une représentation linéaire des programmes). Elle constitue aujourd'hui une des branches les plus actives des algorithmes évolutionnaires.

On suppose que le langage dans lequel on décrit les programmes parmi lesquels on cherche un programme optimal est constitué d'opérateurs et d'opérandes de base. Tout opérateur pouvant opérer sur un nombre fixe d'opérandes, et rendant lui-même un résultat peut à son tour être l'opérande d'un des opérateurs. L'idée de base consiste à représenter de tels programmes sous forme d'arbres. L'ensemble des nœuds de l'arbre N est l'ensemble des

opérateurs, et l'ensemble des terminaux de l'arbre T est l'ensemble des opérandes de bases.

L'intérêt d'une telle représentation, qui permet d'utiliser les principes évolutionnaires sur ce type d'espace de recherche est la fermeture syntaxique de l'opérateur de croisement : en fait, alors qu'il est difficile d'imaginer croiser deux programmes séquentiels écrits dans un langage de haut niveau (comme le langage C ou Java). En obtenant un programme valable comme résultat du croisement, le croisement d'arbres ne pose aucun problème. Il donne toujours un arbre représentant un programme valide, comme représenté dans la figure 1.4.

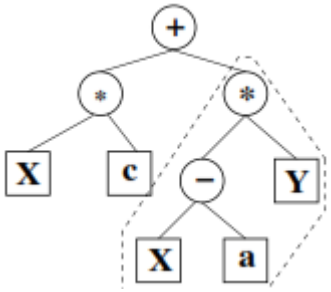


Fig.1.4 Représentation en Arbre de Programmation Génétique.

Les composantes spécifiques de la Programmation Génétique:

- L'Initialisation :

La manière générale pour initialiser un arbre consiste à donner à chacun des nœuds et des terminaux une probabilité d'être choisi, et à tirer au sort, à chaque étape un symbole parmi l'ensemble des nœuds et des terminaux. (Si c'est un nœud, on renouvelle la procédure pour chacun des arguments dont l'opérateur qu'on a besoin).

- Le Croisement

Définir une procédure de croisement n'offre a priori aucune difficulté : on choisit aléatoirement un sous-arbre dans chacun des parents, et on échange les deux sous-arbres en question. Du fait de l'homogénéité des types, le résultat

est toujours un programme valide. L'ensemble des programmes est fermé pour l'opérateur de croisement.

- La Mutation

Consiste à remplacer un sous-arbre aléatoirement choisi, par un sous-arbre aléatoirement construit à l'aide de la procédure d'initialisation. Signalons les mutations par changement de nœud (on remplace un nœud choisi aléatoirement par un nœud de même parité), par insertion de sous-arbre (plutôt que de remplacer totalement un sous-arbre, on insère à un emplacement donné un opérateur dont l'un des opérandes est le sous-arbre qui était présent à cet endroit), et son inverse (qui supprime un nœud et tous ses arguments sauf un, qui prend sa place). [Allaire, 2006].

1.5.5. La Stratégies d'Evolution

Les stratégies d'évolution (SE) (en anglais evolution strategies) ont été développées dans les années soixante par Rechenberg et Schwefel. Ils ont été conçus comme heuristiques de recherche pour des problèmes d'optimisation dans le domaine de l'ingénierie et ils ont été spécialisés pour optimiser les nombres réels en tant qu'attribut de solution [Beyer et al, 2002]. Contrairement aux AG et aux PG, les SE ont été conçus dès le début comme une méthode d'optimisation axée sur la pratique.

Cette approche, axée sur le phénotype permet une optimisation accrue des opérateurs de modification. Particulièrement pour les attributs de nombres réels, les opérateurs de mutation peuvent adopter automatiquement la taille et la direction de la mutation par rapport à la topologie locale de l'espace de recherche. Les paramètres spécifiant les propriétés de la mutation sont appelés paramètres de stratégie et les attributs de la solution sont appelés paramètres de décision.

Les composantes spécifiques de la Stratégie d'Evolution:

- L'individu

Dans une implémentation SE simple, les paramètres de stratégie sont stockés dans un vecteur supplémentaire de **n** nombres réels:

$$a_{i,s} = (\sigma_1, \sigma_2, \dots, \sigma_n) \; ou \; \sigma_I : nombres \; réel \; positifs$$

Dans ce cas simple, nous utilisons un paramètre de stratégie σ_i pour chaque paramètre de décision x_i.σ_i attribue un écart type à la mutation du phénotype pour chaque paramètre de décision, qui est égal à la taille moyenne d'une étape de mutation. Si le codage est omis, la fonction cible peut être évaluée sans autres calculs.$\varphi(a_i) = F(a_{i,d})$

- La sélection

Dans la plupart des SE, la sélection déterministe est utilisée pour sélectionner les parents possibles pour la prochaine génération. Seuls les λ meilleurs individus sont choisis parmi une population A (s) de taille μ et seuls ces L meilleurs individus sont utilisés pour la population mère B(S). Cette approche s'appelle la stratégie (μ, λ). Si les L meilleurs individus sont traités comme une élite qui entre dans la prochaine génération sans modification, la stratégie s'appelle la stratégie (μ+λ).

- Le croisement

SE utilise deux méthodes de croisement différentes pour la stratégie et les paramètres de décision pour mélanger deux parents sélectionnés (e1 et e2) de B (s): Le croisement discret est utilisé pour les paramètres de décision: Un descendant *c* hérite de l'attribut x_i de l'un ou de l'autre parent. Le parent réel qui fournit un attribut peut être choisi arbitrairement. Le croisement Inter medium : Attribue la valeur de descendante *c* a σ_i qui est la moyenne des σ_i correspondants des deux parents. Cette méthode de croisement a un effet de concentration, puisque les paramètres stratégiques de l'enfant sont une simple interpolation linéaire des paramètres stratégiques des deux parents.

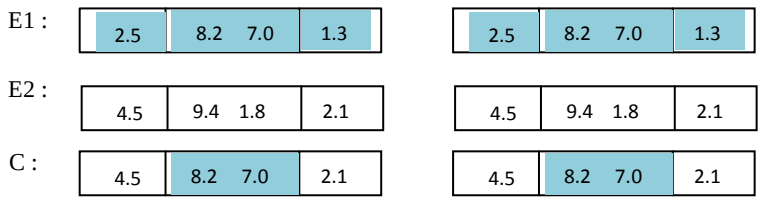


Fig1.5. La technique de croisement dans la Stratégie d'Evolution.

- La mutation

Pour la mutation, deux méthodes différentes peuvent être utilisées pour la stratégie et les paramètres de décision: Premièrement, les paramètres de stratégie sont mutés selon: $\sigma'_j = \sigma_j . \exp(\tau_1 . N(0,1) + \tau_2 . N_j(0,1))$

N (0,1) est un nombre aléatoire uniformément distribué sur la plage [0,1], générée pour chaque mutation par l'individu. La nouvelle taille de la mutation σ'_i est utilisée pour muter les paramètres de décision x'_i:.

$x'_j = x_j . + \sigma'_j . N'_j(0,1)$/ N'j (0,1) est un nombre aléatoire distribué gaussien généré pour chaque xi. Les descendants forment la prochaine génération A (s + 1) après avoir subi ces processus de modification. Encore une fois, le processus générationnel peut être répété jusqu'à ce qu'une solution satisfaisante soit trouvée.

1.5.6. La Programmation Evolutionnaire

La programmation évolutionnaire (PE) en anglais (Evolutionnary programing) a été développée par Fogel dans les années 1960 [Fogel et al, 1966]. Elle a été initialement conçue pour évoluer les machines à états finis et a été par la suite étendue aux problèmes d'optimisation de paramètres. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que de simuler des opérateurs génétiques d'inspiration naturelle. Dans PE, l'évolution adopte le comportement général et non les sous-structures génotypiques. En fait on n'utilise que des opérateurs de modification orientés phénotype. Et puisqu'il n'y a pas d'échange à travers une limite d'espèce, l'opérateur de croisement est omis.

Les composantes spécifiques de la Programmation Evolutionnaires:

- **L'individu**

La représentation des individus du PE (représentant chacun des espèces entières) n'est déterminée que par le problème d'optimisation donné et ne présente pratiquement aucune restriction concernant les types de données pouvant être traitées.

- **La sélection**

Le PE utilise la sélection par tournoi. Pour chaque individu, un groupe de tournoi de *n* individus sélectionnés par hasard de A (s), le nombre d'individus de ce groupe de tournoi inférieur à a_i, en ce qui concerne l'aptitude physique, est déterminé et attribué comme score à a_i. Les individus λ avec le score le plus élevé sont sélectionnés comme parents B (s).

- **La mutation**

Pour Chaque individu l''opérateur de mutation basé sur les différents attributs de types de données. Si, par exemple, un individu EP n'est constitué que d'un vecteur de valeurs réelles comme dans SE, les mêmes opérateurs de mutation de ES peuvent être appliqués. Pour les attributs numériques réels, le comportement de la PE ressemble à celui de SE.

1.5.7. Les algorithmes Génétiques :

Les algorithmes génétiques AG (en anglais genetic algorithm) [Fogel et al, 1966] impliquent l'évolution d'une population de vecteurs de dimension fixe composés de caractères, généralement les bits. L'idée des AG est vaguement inspirée des chaînes d'ADN, qui composent tout organisme vivant. Les AG sont utilisés dans le but de découvrir une solution, généralement numérique, résolvant un problème donné, et ce sans avoir de connaissance a priori sur l'espace de recherche. Seul un critère de qualité est nécessaire pour discriminer déférentes solutions. Dans la plupart des cas, ce critère, l'adéquation, est une mesure objective qui permet de quantifier la capacité de l'individu à résoudre le problème donné. Le processus de recherche s'effectue

en appliquant itérativement à une population de solutions potentielles des opérations de variation génétique (généralement le croisement et la mutation), et des opérations de sélection naturelle biaisées vers les individus les plus forts. En utilisant ce processus Darwinien, la population de solutions potentielles évolue dans le temps jusqu'à ce qu'un certain critère d'arrêt soit atteint.

Dans un contexte d'optimisation de fonctions à paramètres réels, une variante importante des AG consiste à représenter les individus directement par des vecteurs de nombres réels.

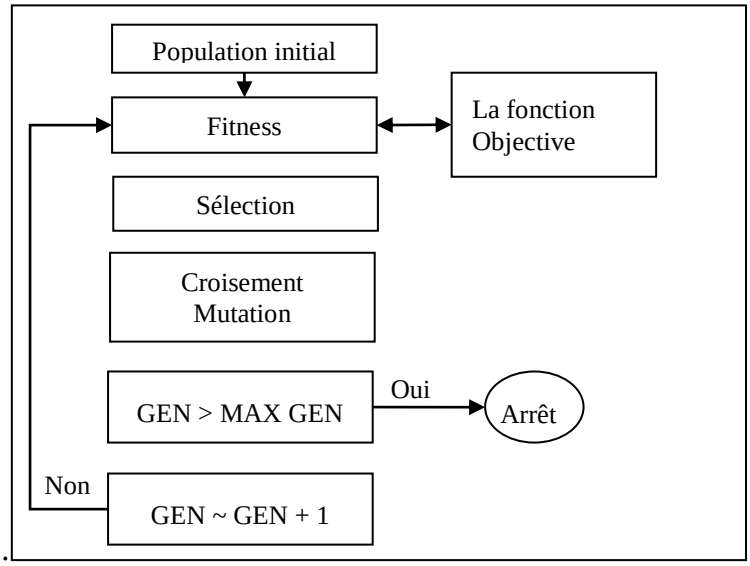


Fig.1.6. Le Squelette d'un Algorithme Génétique

Les composantes spécifiques d'un Algorithme Génétique:

Un algorithme génétique est représenté par un ensemble de chromosomes, une population, un ensemble d'individus, la fonction d'adaptation et la reproduction (le croisement et la mutation). Comme la montre la figure 1.6

1.5.7.1. La représentation de chromosome

Avant la mise en œuvre d'un algorithme génétique sur n'importe quel problème, une méthode est nécessaire pour coder des solutions potentielles à ce problème sous une forme qu'un ordinateur peut le traiter. Utilisant Les approches suivantes :

- En codant des solutions en séquences de chaînes binaires de 1 et 0 où le chiffre de chaque position représente la valeur d'un aspect de la solution.
- En codant des solutions en tant que tableaux d'entiers ou de nombres décimaux, ce qui permet plus de précision et de complexité que la méthode de nombres binaires relativement restreinte.
- En représentant les individus en tant que cordes de lettres où chaque lettre représente encore un aspect spécifique de la solution.

La vertu de toutes ces méthodes consiste à simplifier la définition des opérateurs qui provoquent des changements aléatoires dans la sélection des candidats.

L'AG canonique (Pseudo-Code) est représenté simplement comme suit:

Algorithme1.1 : pseudo code de l'Algorithme Génétique

```
    Choisir la population initiale
    Évaluer la fitness de chaque individu
Répéter
    Sélectionner des individus à reproduire
    S'accoupler au hasard
    Appliquer l'opérateur de croisement
    Appliquer un opérateur de mutation
    Evaluer la fonction objective de l'individu
Jusqu'à la fin de la condition
```

La boucle Répéter peut se terminer soit avec trouvée une solution satisfaisante, soit que le nombre de générations passent une limite prédéfinie, suggérant qu'une solution complète ne sera pas trouvée avec cet ensemble d'individus.

1.5.7.2. L'initialisation

La population initiale est créée à partir d'une sélection aléatoire de solutions (qui sont analogues aux chromosomes). C'est contrairement à une intelligence artificielle symbolique où l'état initial d'un problème est donné à la place.

1.5.7.3. L'Evaluation

La fonction d'adaptation (objectif) est attribuée à chaque solution (chromosome) en fonction de la proximité de la résolution du problème, ce qui arrive à la réponse du problème conçu. Les solutions sont considérées comme des caractéristiques possibles que le système utiliserait pour atteindre la réponse.

1.5.7.4. La Sélection

Selon [Dawkins, 1986], la sélection naturelle signifie la survie différentielle des entités où certaines entités vivent et d'autres meurent. Il existe de nombreuses techniques différentes, qu'un algorithme génétique peut utiliser pour sélectionner les individus à copier dans la prochaine génération. Certaines sont mutuellement exclusives, mais d'autres peuvent et souvent utilisées en combinaison.

1.5.7.5. La Reproduction

La sélection naturelle signifie la survie différentielle des entités [Dawkins, 1986], où certaines vivent et d'autres meurent. Pour que cela se produise, il doit y avoir une population d'entités capables de reproduction.

Le croisement:

Est un opérateur qui implique le choix de deux individus pour échanger des segments de leur code, ce qui génère de nouveaux descendants qui sont la combinaison de leurs parents. Ce processus est destiné à simuler le processus analogue de recombinaison qui se produit aux chromosomes pendant la reproduction sexuelle. En d'autres termes, le croisement génère des génotypes (ensemble de gènes qui forment une forme de vie ou un phénotype) pour être coupés et épissés.

La Mutation:

cet opérateur forme un nouveau chromosome en faisant des modifications (habituellement petites) aux valeurs des gènes dans une copie d'un chromosome monoparentale, tout comme la mutation chez les êtres vivants

modifie un gène par rapport à un autre, donc aussi dans un algorithme génétique.

Pour le codage binaire:

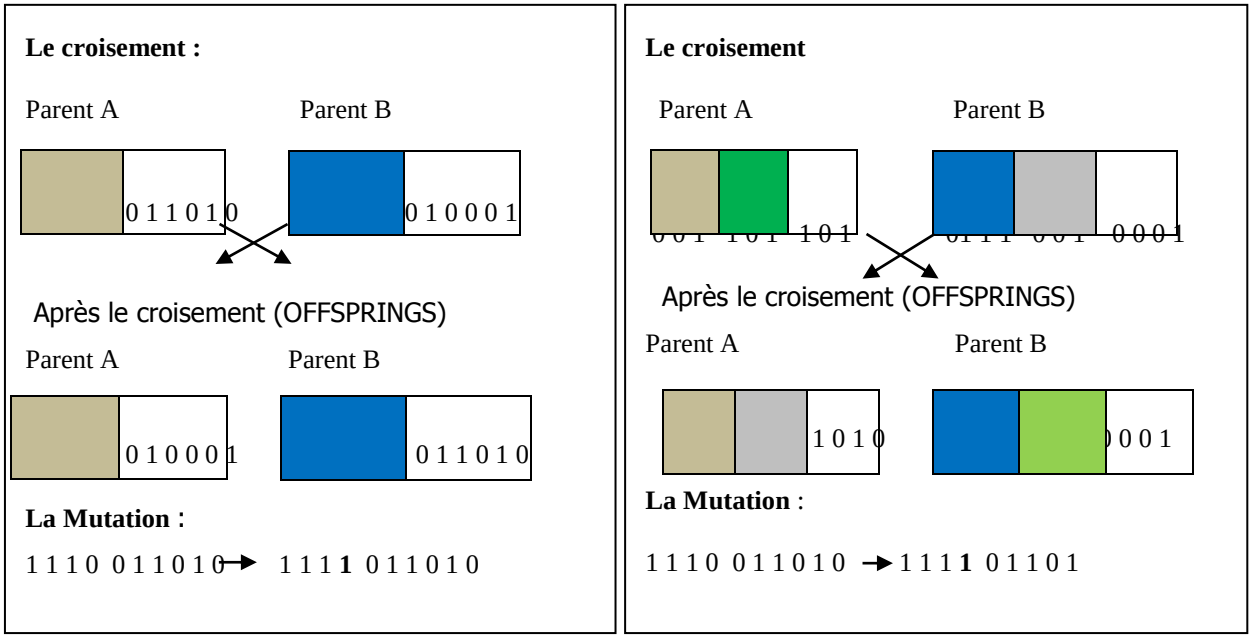


Fig.1.7. La Technique De Croisement et de Mutation en binaire dans un Algorithme Génétique.

Pour le codage réel :

Un point de croisement sélectionné, la permutation est copiée du premier parent jusqu'au point de croisement, puis l'autre parent est scanné et si le nombre n'est pas encore à la progéniture, il est ajouté.

Le **Croisement** et la **Mutation** sont deux opérateurs de base d'AG. La performance de l'AG dépend beaucoup d'eux. Les types et la mise en œuvre de ces opérateurs dépendent du codage et du problème. Voici quelques méthodes très importantes de croisement et de mutation qui sont énumérées dans la figure1.7:

Le Croisement:

Parent A Parent B Offspring

(1 2 3 4 5**6 7 8** 9) + (4 5 3 **6 8 9** 7 2 1) = (1 2 3 4 5**6 8** 9 7)

La Mutation (Deux numéros sont sélectionnés et échangés)

Offspring

(1 2 3 4 5 6 8 9 7) => (1 8 3 4 5 6 2 9 7)

Fig.1.8. La Technique De Croisement et de Mutation en réeldans un Algorithme Génétique.

Dans les AG traditionnelles, la longueur du chromosome est déterminée lorsque le phénotype (une machine de survie construite par un ensemble de gènes (génotypes) est encodée dans un génotype. La longueur est toujours corrigée et ne peut pas changer avec l'évolution. [Week, 2004] ont développés un algorithme génétique efficace qui peut changer la longueur du chromosome en mettant en œuvre la liberté de conception en étendant la longueur du chromosome permettant la réduction du coût de calcul pour les problèmes complexes avec un grand nombre de variables de conception. Exemples de conception dans le phénotype et le chromosome correspondant dans la le génotype est illustré dans la figure 1.9.

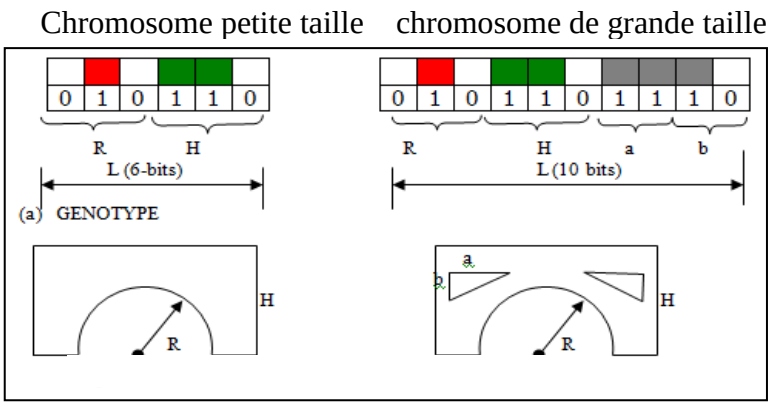


Fig.1.9. La représentation Génotype Vs Phénotype.

1.5.7.6. Les avantages et les limites des Algorithmes génétiques :

Les avantages :

Les principaux avantages des algorithmes évolutionnaires résident dans le fait qu'aucune analyse de sensibilité est nécessaire et qu'on peut obtenue une solution globale optimale.

Les AG fonctionnent bien dans les problèmes où l'espace de recherche est complexe. Ce sont des cas où les fonctions de conditionnement physique sont discontinues et changent avec le temps avec de nombreux optima locaux. Les algorithmes évolutionnaires se sont révélés efficaces pour échapper à l'optimum local et découvrir l'optimum global.

Les API excèdent leur capacité à manipuler plusieurs paramètres simultanément [Forrest, 1993]. Dans une solution Pareto Optimal (non dominée) [Cocilo, 2000], une solution particulière à un problème multi-objectif optimise un paramètre à tel point que le paramètre ne peut plus être amélioré sans provoquer une diminution correspondante de la qualité de Certains autres paramètres.

Les AG ne connaissent pas les problèmes qu'ils sont développés pour résoudre en n'utilisant pas d'informations spécifiques au domaine déjà connues pour guider chaque étape et apporter des modifications à une amélioration spécifique. Les AG apportent des modifications aléatoires à la solution de leur candidat, puis utilisent la fonction de conditionnement physique pour déterminer si ces changements produisent une amélioration.

Les techniques de recherche basées sur l'entropie avec multi-population et la fonction de pénalité quasi-exacte [Wang, 2002] ont été développées pour assurer une convergence rapide et cohérente des résultats.

Les limites:

Lors de la création d'un Algorithme Génétique, la première considération consiste à définir une représentation du problème. La langue requise doit être robuste afin de tolérer des modifications aléatoires pour éviter les erreurs. Cela peut être dépassé: (i) en définissant les individus comme une liste de nombres

(valeur binaire, nombre entier ou valeur réelle) qui représentent certains aspects d'une solution candidate, (ii) cependant, la question de représenter les solutions candidates de manière robuste N'existe pas dans la nature, donc pratiquement tout changement au gène d'un individu produira toujours un résultat intelligent et donc la mutation ou l'échange de sous-arbres a une plus grande chance de produire une amélioration.

Afin d'atteindre une meilleure forme physique pour un problème donné, le problème de l'écriture de la fonction d'adaptation doit être soigneusement pris en compte. La taille de la population, le taux de mutation et de croisement, le type et la force de sélection, tous doivent être soigneusement choisis. Si la taille de la population tombe trop bas, le taux de mutation sera trop élevé, ce qui entraînera une forte pression de sélection (comme dans les changements environnementaux) afin que les espèces disparaissent.

L'AG semble avoir un problème dans le traitement des fonctions de fitness trompeuses [Mitchell, 1996] pour trouver l'emplacement de l'optimum global. Un problème avec les chaînes de 8 bits 00000001 serait moins adapté que 00000011 et la chaîne 11111111 s'avère moins adaptée que la chaine 00000000. Dans un tel problème, un **AG** ne serait plus susceptible de trouver une optimisation globale que la recherche aléatoire d'une convergence prématurée dans l'AG. On réduisant trop rapidement la diversité de la population, si un individu plus adapté que la plupart de ses concurrents émerge tôt au cours de l'exécution, ce qui conduit l'algorithme à converger sur l'optimum local. Cela se produit souvent dans la nature (dérive génétique). La plupart des chercheurs [Holland 1992],[Forrest, 1993]et [Haupt et al, 1998] ont conseillé l'utilisation de l'AG pour résoudre des problèmes analytiquement résolus car les méthodes analytiques traditionnelles prennent beaucoup moins de temps et d'effort de calcul que l'AG.

1.6. Conclusion

Dans ce chapitre nous avons présentés les algorithmes évolutionnaires en toutes généralités. On retenant d'une part leurs robustesses vis-à-vis des optima

locaux et d'autre part leur souplesse d'utilisation. Particulièrement en termes de choix d'application (choix de l'espace de recherche). Les algorithmes Génétiques est un Exemple des Méta-heuristiques le plus connu, dont il est plus détaillé.

CHAPITRE II:

APPLICATION A LA CONCEPTION DES LOGICIELS

2.1. Introduction

La conception de systèmes embarqués spécialisés basés sur la technologie reconfigurable souffre de l'absence de méthodes et d'outils capables d'exploiter efficacement le parallélisme et les possibilités de reconfiguration dynamique qu'offrent ou vont offrir ce type d'architectures. L'une des principales étapes de conception est celle de partitionnement. Dans le cas général, cette étape se subdivise en trois parties principales :

- Une partie qui effectue l'*allocation* en choisissant le type et le nombre de ressources matérielles et logicielles nécessaires.
- Une partie qui effectue le partitionnement spatial en prenant des décisions sur l'affectation (assignation) des tâches sur le matériel et le logiciel.
- Une partie qui effectue l'ordonnancement des exécutions des tâches et des communications.

2.2. Le développement de logiciels

Dès le début des années 70, le développement de larges systèmes logiciels est apparu comme un processus par étapes qui vise à construire un système conforme à ses spécifications. Pour mener à bien cette mission, différents cycles de développement ont été proposés tels que le cycle en cascade, le cycle en « V » et le cycle en spirale. Un des points communs entre ces différents cycles est qu'ils identifient généralement au moins trois phases : la conception, le développement, la vérification et la validation.

2.2.1. La Conception et La modélisation

Depuis le langage Simula-67 [Meyer, 1979] qui a jeté les bases de l'approche orientée-objet, mais plus spécialement depuis le début des années 80, les langages à objets n'ont pas cessé d'évoluer tant sur le plan théorique que pratique. L'approche objet s'est vue appliquée dans de nombreux

domaines : analyse, conception, base de données, etc. En 1997, l'Object Management Group a proposé un langage de modélisation unifié, UML (en anglais Unified Modeling Language [OMG, 1997]) dédié à l'approche objet.

Depuis sa première version, UML n'a cessé d'évoluer et de se complexifier, et intègre maintenant treize diagrammes permettant de capturer les différents aspects d'un système logiciel. L'approche objet, largement adoptée par la communauté industrielle par le biais d'UML, a contribué à répondre au problème de la réutilisation lors du développement, mais pas lors du déploiement ou lors de la maintenance.

De nombreux travaux ont vu le jour pour formaliser les assemblages de composants sous le terme d'architecture logicielle, notamment avec les langages de description d'architecture (ADL). Toutefois, pour être utilisable dans un grand nombre de domaines d'application, la sémantique d'UML n'est pas complètement définie et comporte un certain nombre de points de variation sémantique. UML n'est donc pas un langage, mais plutôt un canevas pour des langages de conception.

2.2.2. Expression du parallélisme

Afin de mettre en évidence le parallélisme de l'application, il est assez usuel de décrire le système sous forme d'un ensemble de tâches communicantes.

Le formalisme KPN (en anglais *Kahn Process Network*) [Kahn, 1974] permet de présenter un modèle d'un graphe orienté. Les nœuds sont les tâches séquentielles et les arcs les canaux de communication(figure 2.1).

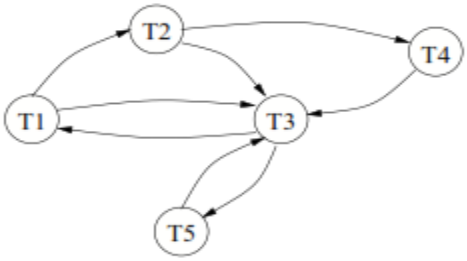


Fig.2.1. Un graphe de tâches de communicantes

Dans la théorie de ce modèle, chaque tâche exécute un programme séquentiel et communique en utilisant des canaux unidirectionnels de type FIFO de taille infinie. Chaque Fifo possède une unique tâche productrice et une unique tâche consommatrice. Une tâche peut lire dans ses canaux d'entrée et écrire dans ses canaux de sortie, mais ne peut pas vérifier la présence de données dans un canal d'entrée. Si une tâche essaye de lire dans un canal vide, elle est bloquée jusqu'à ce que le canal ne le soit plus. Les Fifo étant de profondeur infinie, les écritures ne sont jamais bloquantes. Kahn démontre que ces contraintes (producteur et consommateur unique, tâches séquentielles et lecture bloquante) font que les tâches sont monotones, ce qui rend le graphe déterministe. Un tel graphe garantit que l'ordre des données est toujours le même et ne dépend pas de l'ordre d'évaluation des tâches. Bien évidemment, la profondeur infinie des Fifo n'est pas réaliste pour une réalisation matérielle. Ajouter une taille de mémorisation limitée aux canaux dans le modèle de Kahn implique qu'une tâche peut aussi être bloquée.

2.2.3. Le Partitionnement

Le graphe est partitionné en fonction de sa faisabilité et des contraintes de débit de données de l'application. Le résultat du partitionnement est un ensemble de modèles matériels et logiciels. Ces modèles sont représentés par des graphes hiérarchiques acycliques. Les opérations d'entrées /sorties sont réalisées par passage de messages. Les boucles dépendant des données sont considérées comme des opérations dont le délai n'est pas borné.

L'utilisation de la hiérarchie permet de transformer le problème d'exécution conditionnelle en délais incertain d'exécution. L'incertitude de la boucle est le nombre de fois où elle est exécutée.

En l'absence de point de synchronisation multiple, un graphe peut être implémenté comme une simple fonction. Un système hiérarchique sera implémenté comme un ensemble de fonctions où chaque graphe acyclique est une fonction. Une application est transformée en un ensemble de graphes. Chaque graphe devient un *thread*. Le parallélisme de l'exécution des *threads* est obtenu en entrelaçant l'exécution des *threads* sur le processeur.

2.2.4. Partie logicielle

La partie logicielle de l'application est découpée en *threads* de telle façon qu'un *thread* est définie comme un ensemble d'opérations linéaires (*basic block*) Seule la première opération peut être une opération dont le délai n'est pas borné.

Les *threads* sont construits de façon à ce que cet ordre partiel soit respecté. Deux threads peuvent être soit concurrentes, soit il existe une relation de hiérarchie entre elles. Dans ce dernier cas les threads sont crées avec une opération d'autorisation, leur permettant d'attendre que leur prédécesseur soit terminé.

La synchronisation logicielle est nécessaire pour assurer l'ordre d'exécution des opérations d'un thread et entre les threads.

La Synchronisation matériel/logiciel

A cause de l'exécution séquentielle du logiciel, un transfert de données entre matériel et logiciel doit être explicitement synchronisé. Pour tenir compte des différentes vitesses d'exécution des composants matériels et logiciels, et des opérations dont le délai n'est pas connu, un ordonnancement dynamique des *threads* est utilisé. Cet ordonnancement est basé sur la disponibilité des données. Il est obtenu par un Fifo de contrôle. L'interface matérielle/logicielle consiste à gérer les identificateurs des *threads,* on contrôlant ainsi l'ordre des données. La taille de la Fifo de contrôle dépend du nombre de *threads*.

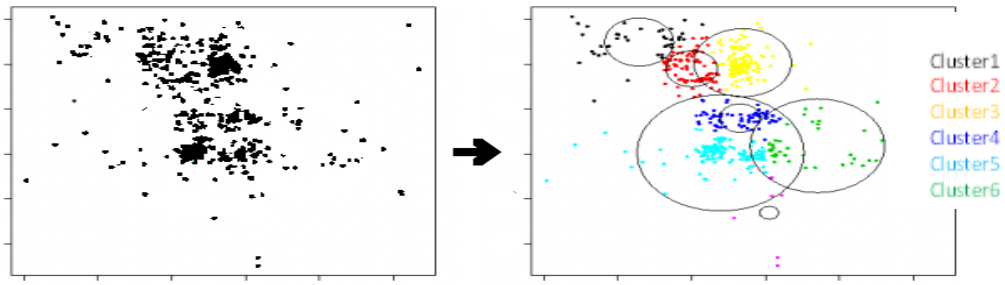
2.3. Le partitionnement de donnée

La classification automatique est une méthode de réduction des données. Il s'agit d'une démarche très courante qui permet de mieux comprendre l'ensemble analysé. Ces applications sont nombreuses en statistique, en traitement d'image, intelligence artificiel, la reconnaissance des formes et/ou encore la compression de données. Pour toutes ces raisons, elle constitue une étape importante dans le processus de Fouille des Données.

On peut distinguer deux grandes familles de classification : par partitionnement et par hiérarchie. Dans ce chapitre, nous nous concentrons sur méthodes de partitionnement couramment utilisées et disponibles dans la plupart des logiciels. Comme nous décrivons les différentes techniques et les mesures de partitionnement. En effet, cette approche est un outil puissant qui permet entre autres de mieux comprendre la plupart des méthodes des deux grandes familles citées. Dans ce qui suit nous nous intéressons à l'approche non supervisée de la classification, où encore appelée le Clustering.

Le Clustering aussi connu sous le nom (Segmentation ou regroupement) des classes homogènes qui consistent à représenter un nuage des points d'un espace quelconque en un ensemble de groupes appelé *Cluster*. C'est un traitement sur un ensemble d'objets qui n'ont pas été étiquetés par un superviseur. Ce type de méthodes vise à répondre au problème de diminution de la dimension de l'espace d'entrée, ou pour le groupement des objets en plusieurs catégories (clusters) non assemblés à l'avance.

Un «Cluster» est donc une collection d'objets qui sont «similaires» entre eux et qui sont dissemblables par rapport aux objets appartenant à d'autres groupes. On peut voir cette définition clairement dans l'exemple suivant (figure 2.2) :



a) données brutes b) données regroupées

Fig.2.2.Iillustration de regroupement en cluster.

2.3.1. Le processus de Clustering

Le but principal du clustering est la recherche des structures similaires dans l'espace de données R^d. Ce problème à été abordé dans plusieurs

ouvrages [Jain et al, 1988] [Anderberg, 1973] [Hartigan, 1975] [Spath, 1980] [Duran et al, 1974] [Everitt, 1993] [Backer, 1995]. A travers cette littérature, on constate que toutes les techniques de Clustering suivent le même principe général qui consiste à maximiser la similarité des objets à l'intérieur d'un cluster, et minimiser la similarité des objets ente les clusters. Chaque cluster issu de ce processus doit vérifier les deux propriétés suivantes :

1. La cohésion interne (les objets appartenant à ce cluster soient les plus similaires possibles).

2. L'isolation externe (les objets appartenant aux autres clusters soient les plus distincts possibles).

Le Clustering repose sur une mesure précise de la similarité / disimilarité des objets que l'on veut regrouper. Cette mesure est appelé distance ou métrique. [Murty et al, 1995].

Etant donné un ensemble d'objets X={x1,…xn} dans l'espace d'attributs R^d avec d : dimension de l'espace, n : le nombre d'objets. $x_i=(x_{i1},x_{i2}…x_{id})$ représente le $i^{ième}$ objet ; et x_{ij} correspond à la valeur du $j^{ième}$ attribut pour le $i^{ième}$ objet. La figure.2.3 est un exemple illustrtif de différentes étapes de la tâche de Clustering

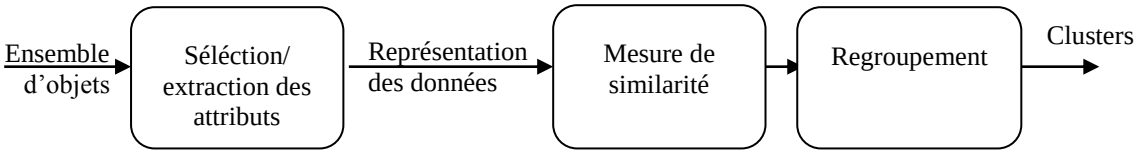


Fig.2.3. les étapes de la tâche de Clustering

Les différentes étapes de Clustering:

1. la séléction / extraction des attributs correspond à l'utilisation d'une ou plusieurs transformations des attributs fournis en entrée afin de séléctionner le sous ensemble le plus efficace à utiliser pour le Clustering. Plusieurs méthodes qui traitent ce problème ont été proposées dans la littérateure;

2. La représentation des données se refère à la spécification du nombre de données, ainsi que la dimension et le type des attributs disponible pour l'algorithme de Clustering.

3. La mesure de similarité consiste à définir une métrique appropriée au domaine des données. Différentes mesures ont été utilisées dans le Clustering.

4. Le regroupement consiste en la construction des groupes similaires qui représentent le résultat du processus de Clustering, qui est peut être "Hard ", partition des objets en groupes distincts ou flous "Fuzzy" (chaque objets a un degré variable d'appartenance à chacun des groupes formés). La section présente une varation de technique de clustering.

2.3.2. Mesure de rapprochement

La majorité des algorithmes de clustering sont basés sur une notion de mesure pour effectuer le regroupement (ou la séparation) des objets. On distingue principalement deux mesures pour dégager un rapprochement entre deux objets. La plus immédiate est la notion de distance, cette notion utilise un vecteur d'attribut de dimension constante *d,* le coefficient de corrélation sera lui aussi utiliser avec cette approche de vecteur d'attributs. La deuxième mesure est celle des coefficients de similarité qui fait intervenir des notions de présences ou d'absence communes. Ces coefficients peuvent être transformés en semi métrique.

Nous indiquons quelques rappels de topologie pour définir une distance.

Une distance d_{ij} dans R^p entre xi et xj est une application : i x j$\rightarrow d_{ij}$ qui doit vérifier :

- la positivité : $d_{ij}= d(xi,xj)\geq 0$.

- La réflexivité :$d_{ij} =0 \Leftrightarrow$ xi =xj.

- L'identité : $d_{ij}=d_{j\ i}$

- La symétrie : $d_{ij}=d_{j\ i}$

Dans certains cas, on utilise la similarité qui permet de mesurer la ressemblance entre objets, cette mesure est plus souple que la notion de distance.

Une similarité S_{ij} dans R^p entre xi et xj est une application : i x j$\rightarrow S_{ij}$ qui doit vérifier :

- la positivité : $S_{ij} = S(xi,xj) \geq 0$.

- La symétrie : $S_{ij}=S_{j\,i}$.

Plus les objets sont similaires, plus la similarité est importante.

2.3.3. Domaines d'application du Clustering

Le Clustering est très utilisé dans plusieurs domaines, comme l'intelligence artificielle, la biologie, le web, l'analyse de données et beaucoup d'autres domaines. Dans ce contexte, nous allons citer quelques exemples d'application:

- L'analyse de données qui peuvent provenir des images satellites, équipement médical, système d'information géographique. On y extrait les caractéristiques nécessaires pour accélérer le processus d'exploitation de ces données [Ray et al, 1999] [Natalia et al, 2001].
- La génération des hypothèses afin d'inférer des règles pour caractériser les données et suggérer des modèles, par exemple : l'aide à l'établissement des diagnostiques médicaux sur des bases de données de patients [Fink et al, 2004].
- La réduction de la dimension des bases de données afin de conserver le maximum de l'information utile dans un espace de dimension inférieure [Eschrich et al, 2003].
- La prospection du Web (Web Mining) et l'analyse des données textuelles (text Mining) pour la recherche d'information à partir de certains mots clés [Chen et al, 2002].

2.4. Les techniques de Partitionnement

Il y a un grand nombre d'algorithmes de regroupement dans la littérature. Le choix de l'algorithme de cluster dépend à la fois du type de données disponibles et du but de l'application en particuliers. Si l'analyse par clustering est utilisée comme outil descriptif ou exploratoire, il est possible d'essayer plusieurs algorithmes sur les mêmes données pour voir ce que les données peuvent révéler [Hinneburg, 1998]. En général, on distingue deux grandes catégories de techniques de Clustering. (Figure 2.4).

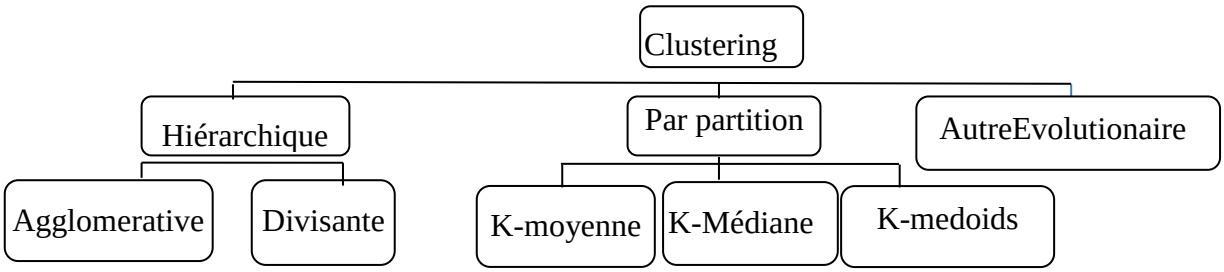


Fig.2.4. Les techniques de Clustering.

2.4.1. Le Clustering Hiérarchique :

Il s'agit d'un arbre dans lequel chaque niveau correspond à une partition et chaque nœud correspond à une partie de cette partition. Cette structure dont le but est de former une hiérarchie de clusters appelée dendrogramme, de telle sorte que plus on descend dans la hiérarchie, plus les clusters sont spécifiques à un certain nombre d'objets considérés comme similaires. Ce type se décompose en deux familles :

- Les approches Agglomératives qui construisent le dendrogramme par la base, en regroupant à chaque étape les amas les plus similaires.
- Les approches divisantes qui construisent le dendrogramme par le haut, en partitionnant à chaque étape un amas en sous amas.

2.4.2. Le Clustering par partition:

Le but est de former plusieurs partions dans l'espace des objets, de tel sort que chaque partition représente un cluster. Dans ce qui suit nous allons

décrire les techniques de Clustering par partition existantes en faisons la distinction entre ces différentes méthodes. Un accent particulier est mis sur l'approche basée sur l'optimisation d'une fonction objective, en particulier la famille K moyenne qui constitue une des bases de notre travail. Selon la figure.2.5. Il existe quatre grandes techniques de partitionnement qui peuvent être utilisées pour regrouper les éléments d'un ensemble de donnée autour d'un centre de gravité (la moyenne empirique): Les k-moyennes [MacQueen, 1967] ; d'une médiane géométrique : les k-médianes [Bradley et al, 1997] ; d'un centre contenant les modes les plus fréquents : les k-modes [Huang, 1998] ;d'un médoïde (l'élément d'un ensemble qui minimise la somme des distances entre lui et chacun des autres éléments de cet ensemble) : les k-medoïdes [Kaufman et al, 1990].

2.4.2.1. Les K-moyenne :

Ce type de méthodes repose généralement sur des algorithmes simples, et permet de traiter rapidement des ensembles d'effectif assez élevé en optimisant localement un critère. Cette catégorie a un bon sens géométrique et statistique pour les attributs numériques. La somme des distances entre un point et son centroïd exprimée par une distance appropriée est utilisée comme fonction objective.

Le plus célèbre de ces algorithmes est incontestablement l'algorithme des centres mobiles (**k-means**) [Forgy, 1965]; [MacQueen, 1967]. Cet algorithme se déroule de la façon suivante :

1. Initialisation : *s* points tirés au hasard pour les centres de gravité de chaque classe ;
2. Affectation : On affecte les points à la classe la plus proche,
3. Représentation : On recalcule les nouveaux centres de gravité,
4. répéter les étapes 2 et de 3 jusqu'à la convergence de l'algorithme (i. e. plus de changement de partition).

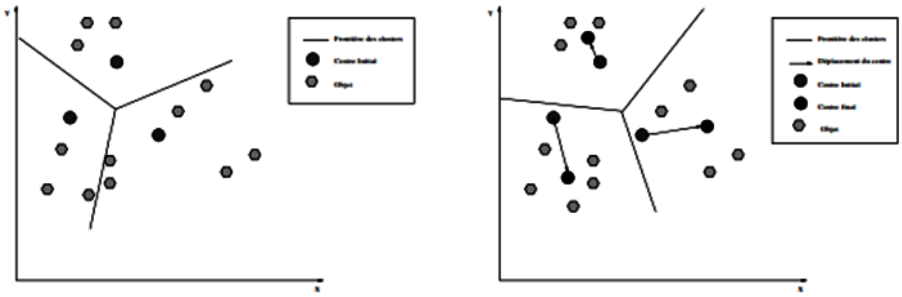


Fig.2.5Illustration de Kmeans dans le déplacement des centres et des frontières des clusters

2.4.2.2. Les k-Medians

Dans l'algorithme des k- médianes, on impose aux centroïdes d'être à la position d'un des exemples d'entrainement [Bradley et al, 1997]. Ainsi, le centroïde d'un ensemble d'exemples correspond à l'exemple le plus central de cet ensemble. De manière générale, l'algorithme des k-médianes est moins sensible aux données aberrantes (outliers) qui peuvent affecter grandement la position des centroïdes dans l'algorithme des k-moyennes.

2.4.2.3. Les k-Medoïdes

Dans les méthodes k-médoïdes une classe est représentée par un de ses points, c'est un algorithme efficace puisqu'il traite n'importe quel type d'attributs et que les médoïdes ont une résistance intégrée contre les données atypiques parce que les points périphériques des clusters ne sont pas pris en compte. Quand les medoïdes sont sélectionnés, les classes sont définies comme des sous ensembles de points proches des medoïdes respectifs, et la fonction objective est définie comme la distance moyenne ou une autre mesure de dissimilarité entre un point et son médoïde. Parmi les premières versions des méthodes de K-médoïdes nous citons l'algorithme PAM (Partioning Around Medoids) et l'algorithme CLARA (Clustering LARge Application). [Kaufman et al, 1990]

2.4.3. Le Clustering Evolutionnaire :

Les approches évolutionnistes, motivées par l'évolution naturelle, utilisent des opérateurs évolutionnaires et une population de solutions pour obtenir la partition globalement optimale des données. Les solutions candidates au problème de la classification sont codées sous la forme de chromosomes. Les opérateurs évolutionnaires les plus couramment utilisés sont:

La sélection, le croisement et la mutation.

Chacun transforme un ou plusieurs chromosomes d'entrée en un ou plusieurs chromosomes de sortie. Une fonction de fitness évaluée sur un chromosome détermine la probabilité de survie d'un chromosome dans la génération suivante. Nous donnons ci-dessous une description de haut niveau d'un algorithme évolutionnaire appliqué au clustering :

(1) Choisir une population aléatoire de solutions, où chaque solution correspond à une k-partition valide des données. Associer une valeur de remise en forme à chaque solution. Typiquement, la fonction objective est inversement proportionnelle à la valeur d'erreur au carré. Une solution avec une petite erreur au carré aura une plus grande valeur de forme physique.

(2) Utiliser la sélection des opérateurs évolutionnaires, la recombinaison et la mutation pour générer la prochaine population de solutions et évaluer les valeurs de fitness de ces solutions.

(3) Répéter l'étape 2 jusqu'à ce qu'une condition de terminaison soit satisfaite. [Jain et al, 1999]

2.5. Les méthodes de clustering

2.5.1. Méthodes basées sur la densité

Le but, ici, est de chercher à former des clusters denses, de telle sorte que chaque cluster représente une région homogène de haute densité, entourée par des régions de faible densité. Pour cela, deux paramètres qui contrôlent la densité sont utilisés :

Eps: Rayon maximum du voisinage.

MinPts : Nombre minimum de points qui doivent être contenus dans ce voisinage.

Le principe général est défini comme suit :

1. Sélectionner aléatoirement un objet Xi selon une loi uniforme sur les objets.
2. Vérifier si son voisinage respecte le critère de densité ; i.e. s'il y a au moins MinPts points dans la sphère de centre Xi et de rayon Eps.
3. Si le critère de densité est respecté, intégrer les objets correspondants dans le cluster, et répéter le procédé avec ces objets.

Sinon, aller à 1 (la sélection aléatoire se fait sur les objets non encore classés).

DENCULE [Hinneburg et al, 1998] et DBSCAN [Ester et al, 1996] sont des exemples d'algorithmes appartenant à cette catégorie. La figure 2.6 montre un exemple de clustering basé sur la densité.

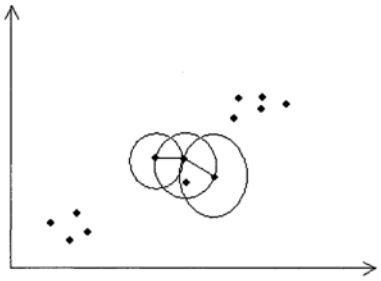


Fig.2.6. Exemple de Clustering basé sur la densité

2.5.2. Les méthodes basées sur une grille

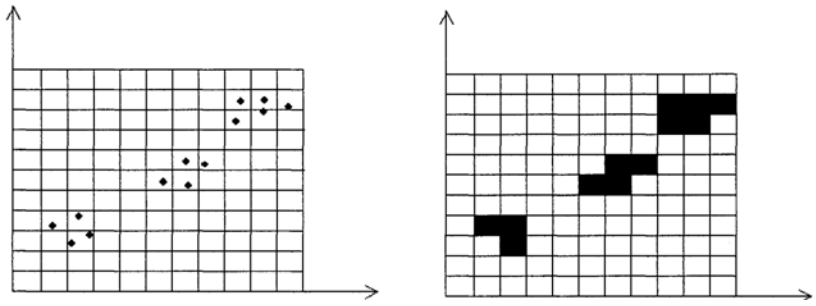
Le principe de base de ces méthodes est d'utiliser une grille pour diviser l'espace en un ensemble de cellules. Un cluster est vu comme un ensemble de cellules denses et connectées. Il existe deux types de méthodes pour identifier un cluster :

1. Les méthodes qui calculent la densité de chaque cellule, puis fusionne les cellules et ensuite identifier les ensembles denses de cellules

connectées pour former les clusters. (Le résultat soit suffisamment uniforme).

2. Les méthodes qui se basent sur la détection des limites des clusters. Le principe de base ici est la détection des limites entre les zones de haute densité et les zones de faible densité, ensuite la reconstitution des clusters à partir de ces limites. La majorité des algorithmes appartenant à cette catégorie souffrent d'une problématique importante, qui est le choix de la taille des cellules. Les cellules de petite taille, amènent à une estimation bruitée de la densité (problème du "sur-partitionnement"). À l'inverse, les cellules de taille importante, amènent à une estimation trop faible de la densité (problème du "sous-partitionnement"). La figure.2.8 est une illustration graphique de ces méthodes.

STING " Statistical Information Grid-based method " [Wang et al, 1997] et WaveCluster [Sheikholeslami et al, 1998] sont des exemples d'algorithmes appartenant à cette catégorie.



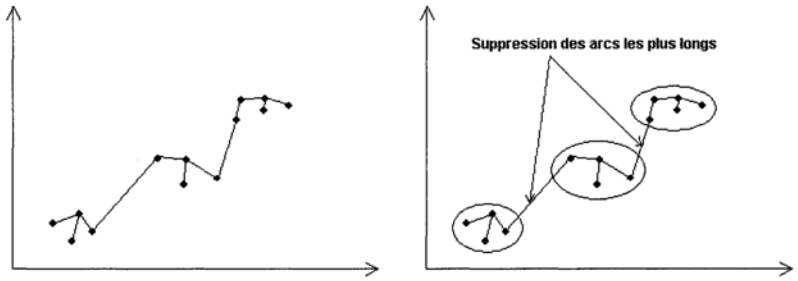
a) division de l'espace à base d'une grille b) Le Clustering associé

Fig.2.7. Exemple de Clustering basé sur une grille

2.5.3 Les méthodes basées sur la théorie des graphes

Le principe de ces méthodes est la recherche des arcs à conserver, dans un graphe qui connecte les différents objets entre eux afin de former des clusters. Ici un cluster est défini comme un ensemble de nœuds connectés dans un graphe. La figure.2.8 est une illustration graphique d'un exemple de clustering

basé sur la théorie des graphes, dont le principe général est décrit comme suit [Zhang, 1971]:



a) Création du MST b) Le clustering associé

Fig.2.8. Exemple de Clustering sur théorie des graphes

- Construction d'un MST " Minimal Spanning Tree " de données, cela revient à définir un graphe connexe en joignant tous les objets de la base, dont la somme des valeurs des étiquettes associées aux arcs est minimale.
- Suppression des arcs les plus longs pour la création des clusters.

Une autre possibilité proposée dans [Hinneburg, 1999] consiste à conserver les liens entre les objets séparés par une distance inférieure à un certain seuil. Les clusters étant alors l'ensemble des objets connectés.

D'autres algorithmes ont été proposés dans la littérature. Comme exemple on peut citer l'approche proposée dans [Toussaint, 1980] basé sur " Relative Neighborhood Graph " (RNG). Alors que dans [Ozawa, 1885] une autre méthode est présentée, et, spécialement dédiée à des partitions avec recouvrement entre les clusters.

2.5.4. Les méthodes basées sur l'approche probabiliste

L'approche probabiliste considère que les objets à grouper sont les réalisations indépendantes d'une variable aléatoire suivant une certaine distribution. La connaissance de la loi induit un partitionnement des données. Le but est d'estimer les paramètres de cette distribution (moyenne, variance)

ayant généré le groupe d'objets en question. L'hypothèse de mélange gaussien est alors classiquement utilisée dans ce contexte [Bock, 1981].

2.6. Les Types de résultat de Clustering

Comme toutes les techniques de Clustering, le but principal des algorithmes est de diviser une partition en un ensemble de groupes, dont les objets appartenant à ces groupes partagent les mêmes caractéristiques. Ainsi leur résultat final peut se présenter sous différentes formes selon qu'il est possible ou non que deux clusters se chevauchent, (un objet puisse appartenir ou non à plusieurs clusters au même temps). On parle alors d'une forme de partition dure "hard ", doux"soft" et floue "fuzzy".figure 2.9.

	C1	*C2*	*C3*
x_1	1	0	0
x_2	0	1	0
x_3	0	0	1
x_4	0	0	1

résultat Dur

	C1	*C2*	*C3*
x_1	1	1	0
x_2	0	1	1
x_3	0	1	1
x_4	0	0	1

résultat Doux

	C1	*C2*	*C3*
x_1	0.9	0.1	0
x_2	0	0.8	0.2
x_3	0	0.3	0.7
x_4	0	0	1.0

résultat Flou

Fig.2.9. Exemple des degrés d'appartenance des objets aux clusters pour un résultat Dur, doux et flou

Le concept de partition floue et dure étant précisé, la question suivante émerge: comment trouver une partition optimale d'un ensemble de données, lorsque la ressemblance entre deux individus est évaluée par une mesure de proximité?

2.6.1. La partition dure "Hard"

Le résultat le plus simple, souvent rencontré est le clustering Dur. Dans ce cas, chaque élément appartient à un et un seul cluster. L'ensemble des données *X* est divisé en un ensemble de *K* Clusters formant une partition de *C.*

$$\mathbf{U}_{k=1}^{K} \boldsymbol{C_k} = \boldsymbol{X} \qquad \textbf{(2.1)}$$

X est une partition dure si :

Dans un ensemble X = {x_1,... ,x_n} groupé en c clusters, chaque élément de l'ensemble appartient à un et un seul cluster. Une manière pratique de décrire l'ensemble *C* consiste à utiliser une notion matricielle U.

$$U = \begin{pmatrix} u_{11} & \cdots & u_{1c} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nc} \end{pmatrix}$$

Où :

- U la matrice caractéristique de la partition *X*.

- U_{ik} = 1 si et seulement si $x_i \in C_k$, sinon u_{ik}= 0. Remarquons que la somme de la i^{éme} ligne est égale à 1 (un élément appartient à un seul cluster) et la somme des valeurs de la $k^{léme}$ colonne vaut n_k le nombre d'éléments du cluster C_k. On a donc l'équation (2.2) :

$$\sum_{k=1}^{c} n_k = n \qquad \textbf{(2.2)}$$

L'algorithme connu aussi sous le nom Kmeans (l'algorithme des centres mobiles) est un algorithme très répandu pour résoudre ce problème. Cet algorithme basé sur des considérations géométriques doit certainement son succès à sa simplicité et son efficacité, dont on peut les résumer en deux étapes: La première étape affecte chaque élément X_i, au prototype le plus proche, et la seconde recalcule la position des prototypes en considérant la partition dure partielle.

Il peut arriver que certains objets se distinguent de manière trop significative par rapport à d'autres. Leurs affecter un cluster peut perturber le processus de clustering. Il arrive que ces objets soient rejetés et qu'aucun cluster ne leur soit affecté dans le résultat final.

2.6.2. La partition Doux "soft"

On peut avoir des difficultés à définir la frontière entre les clusters peut être difficile à définir, il arrive que certains objets soient à la frontière des plusieurs clusters. Pour pouvoir refléter ce type d'appartenance, le clustering doux (soft Clustering) permet à chaque objet d'appartenir à un ou plusieurs

clusters. On peut alors parler de clustering doux partiel si dans le résultat un élément peut appartenir à aucun, un ou plusieurs clusters.

2.6.3. La partition floue"Fuzzy "

La partition floue, généralise une approche classique de classification en élargissant la notion d'appartenance à un ensemble. En effet l'**appartenance** d'un élément à un ensemble n'est plus une valeur vraie ou fausse, mais elle est caractérisée par un réel compris entre 0 et 1 appelé degré d'appartenance u_{ik}. Ainsi un élément peut appartenir à plusieurs ensembles avec différents degrés d'appartenance. Le fonctionnel est définit dans l'équation (2.3) :

$$\boldsymbol{j_m(U,V;w) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^{m} d_{ik}^{2}} \qquad \textbf{(2.3)}$$

Où :

U est la matrice de partition floue d'éléments u_{ik} respectant les contraintes relatives à l'algorithme utilisé.

$V = \{v_i,\dots ,v_c\}$ est l'ensemble des centres (prototypes) des clusters C_k dont la distance aux X_i est noté d_{ik}.

$w = \{w_i,\dots^2 ,w_c\}$ est l'ensemble des termes de pénalité de PCM des données atypiques associés à chacun des cluster (égale à zéro dans le cas de HCM et FCM).

m, appelé coefficient de flou, est un paramètre de l'algorithme contrôlant la quantité de flou dans la partition (m> 1).

Le principe de cette famille de méthodes est de minimiser itérativement la fonctionnelle J_m en alternant une mise à jour de U et V. La table (2.1) récapitule les équations de mise à jour de U et V. La mise à jour des centres des clusters dépend de la forme des clusters recherchés.

L'algorithme populaire de cette catégorie est l'algorithme présenté dans ce qui suit.

Algorithme 2.1. Fuzzy C-Means "FCM "

Entrée : X = {x_1,... ,x_n} ensemble de données, *c* le nombre de clusters,$v^{(0)}$ l'ensembles des centres
des clusters initiaux, *d* une métrique, *m* le coéfficient de flou, ε le seuil pour la convergence de l'algorithme.

Sortie : La matrice de degré d'appartenance U, et les centres de clusters V.

1. t ← 0 ;
2. Mise à jour des degrés d'appartenance :
3. Mise à jour des centres :
4. Test de convergence :

Si $\|v^{(t+1)} - v^{(t)}\| < \varepsilon$ Alors fin de l'algorithm.
Sinon $v^{(t)} \leftarrow v^{(t+1)}$; t ← t+1 ; aller à 2.

L'algorithme des C-moyennes floues " Fuzzy C-Means FCM " :

Introduit par [Dunn, 1973] et généralisé par [Bezdek, 1981], l'algorithme FCM est l'algorithme le plus utilisé en pratique. Dans ce contexte, un seul point suffit à représenter un cluster C_k. Il n'y a donc aucune modélisation intrinsèque des clusters. L'algorithme 2.1 décrit le fonctionnement de FCM.

2.7. Les mesures d'évaluation d'un cluster.

L'évaluation de la qualité d'un résultat de Clustering est un domaine de recherche actif où de nombreuses méthodes continuent d'être proposées régulièrement. Ceci est dû au fait que l'évaluation d'un clustering contient toujours une part de subjectivité et qu'il est impossible de définir un critère universel qui permettrait une évaluation sans biais de tous les résultats produits par toutes les méthodes de clustering existantes. Cependant, un certains nombres de critères existants sont utilisés de manière récurrente par de nombreux chercheurs pour comparer les résultats obtenus. Comme il existe un nombre important de résultats de Clustering possibles pour un même jeu de données, l'objectif est d'évaluer si un de ces résultats est meilleur qu'un autre.

Les critères [pakhira et al,04] se basent sur des informations internes au clustering, par exemple la distance entre les objets d'un cluster et le centroïde de celui-ci, des mesures de distances sont calculées entre les représentants de clusters et les objets de résultats. Elles permettent d'évaluer la compacité et la séparabilité des clusters. La définition de la qualité d'un cluster n'étant pas

définie formellement, il existe de nombreux critères évaluant de manière différente les résultats. Certains peuvent être directement utilisés comme fonction objective et être optimisé par un algorithme de clustering. D'autres sont cependant trop couteux à évaluer pour être calculés au cours de l'exécution d'un algorithme et sont par conséquent destinés à être calculés à l'issue de l'application de celui-ci. Nous présentons les mesures d'évaluation les plus connues.

- **La somme des erreurs au carré (SSE)**

La somme des erreurs au carré est la façon la plus simple d'évaluer la qualité d'un résultat. Elle est définie comme suit :

$$\boldsymbol{SSE(C) = \sum_{i=1}^{K} \sum_{x \in C_i} d(x - \mu_i)^2} \qquad \textbf{(2.4)}$$

Avec μ_i le centroïde du cluster C_i, et d'une mesure de distance entre les objets. Plus la valeur est petite plus les clusters sont compact.

- **Le coefficient silhouette (CS) :**

Le coefficient silhouette [kaufman et al, 1990] permet d'évaluer la compacité des clusters ainsi la séparabilité de ceux-ci. Il peut être calculé pour chaque objet, pour chaque cluster et pour le clustering entier. Pour un objet x *CS* est défini comme suit :

$$\boldsymbol{CS(x) = \frac{b_x - a_x}{\max(a_x - b_x)}} \qquad \textbf{(2.5)}$$

Avec a_x est la distance moyenne entre l'objet *x* et tous les autres objets appartenant au même cluster que *x*. b_x est la distance moyenne entre l'objet *x* et tous les autres objets n'appartenant pas à ce même cluster. Le $CS_{(x)}$ varie antre -1 et 1, la valeur positive ($a_x < b_x$) signifie que les objets appartenant au même cluster que x sont plus proches de x que des objets des autres groupes. Donc pour un cluster la silhouette est la moyenne des coefficients des objets appartenant à ce cluster.

$$\boldsymbol{CS(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} (CS(x))} \qquad \textbf{(2.6)}$$

- **L'indice de Wemmert et Gançarski (WG) :**

L'indice de compacité Wemmert et Gançarski [wemmert2000] évalue la séparabilité et la compacité des clusters. Il est défini comme suit :

$$\boldsymbol{WG(C_i)} = \begin{cases} \boldsymbol{0\ si\ \frac{1}{|c_i|}\sum_{x\in C_i}\frac{d(x,\mu_i)}{d(x,\mu_j)} > 1} \\ \boldsymbol{1 - \frac{1}{|c_i|}\sum_{x\in C_i}\frac{d(x,\mu_i)}{d(x,\mu_j)}\ sinon} \end{cases} \quad \textbf{(2.7)}$$

Où j $= arg\min_{k\neq i}(dist(x,\mu_k))$

L'indice WG prend ses valeurs entre 0 et 1, 1 pour une très bonne compacité et séparabilité des clusters, l'équation suivante présente l'indice WG pour le clustering.

$$\boldsymbol{WG(C) = \frac{1}{n}\sum_{i=1}^{K}|C_i|WG(C_i)} \quad \textbf{(2.8)}$$

2.8. Application à l'optimisation de graphe: Cas du voyageur de commerce

Le voyageur de commerce est un exemple de problème NP-Complet, qu'est connu sous le nom du TSP (en anglais Travelling Salesman Problem).

Un voyageur de commerce désire visiter un certain nombre de villes, débutant et finissant son parcours dans la même ville en visitant chacune des autres villes une et une seule fois. Il désire sélectionner la tournée qui minimise la distance totale parcourue.

La complexité en temps des algorithmes exacts proposés croît exponentiellement avec *n* (la taille du problème ou le nombre de villes). Plusieurs méthodes d'approximation (heuristiques) ont été proposées qui approchent en temps raisonnable la solution optimale. [Bernard et al, 2006].

Formellement, à partir d'une matrice C = (c_{ij}) où c représente le coût du déplacement de la ville i à la ville j (1≤ i, j≤ n), il faut trouver une permutation $\begin{pmatrix} 1 & 2 \dots & n \\ \delta(1) & (2) \dots & \delta(n) \end{pmatrix}$ qui minimise la somme$\sum_{i=1}^{n-1} c_{\delta(i)\delta(i+1)} + c_{\delta(n)\delta(1)}$.

(Le coût d'une tournée est égal à la somme des coûts de tous les arcs appartenant à la tournée). Autrement dit, il faut trouver un circuit (respectivement cycle) Hamiltonien de longueur minimale dans un graphe orienté (respectivement non orienté) value complet. Si c pour tout i ∈ {1,...,n} le problème est dit symétrique sinon il est asymétrique.$c_{ij} = c_{ji}$.

2.8.1. Les Méta-heuristiques de résolution

Pour résoudre ce problème, nous allons nous intéresser à une famille d'algorithmes particulière: les algorithmes méta-heuristiques. Ceux-ci se basent sur la réunion d'un grand nombre de phénomènes plus ou moins aléatoires pour trouver un minimum global à un problème d'optimisation. L'utilisation de procédés stochastiques sert à faire sortir de minimum locaux vers lesquels pourraient converger des algorithmes plus classiques. Ces procédés viennent souvent de l'observation de phénomènes naturels comme le brassage génétique, l'organisation des atomes en fonction de la température ou même les colonies de fourmis. Tous ces exemples sont particulièrement intéressants :

- les fourmis, dont l'intelligence est limitée individuellement, optimisent néanmoins leurs trajets grâce à leur système de communication ;
- la génétique, bien que croisant des gènes aléatoirement, conduit à une amélioration globale du génome ;
- la technique du recuit simulé, en alternant phase de réchauffement puis de refroidissement d'un métal, permet d'obtenir une édifice atomique très solide.

2.8.2. La résolution par Algorithme Génétique :

L'utilisation d'un algorithme génétique est particulièrement adaptée à cet exemple vu l'indépendance des solutions potentielles et la simplicité de création et formalisation de solutions potentielles. L'algorithme s'inspire grandement de la théorie de l'évolution et des principes génétiques. Utiliserons un langage spécifique à la génétique lors de cette étude :

- **L'individu** : un trajet possible

- **La population** : un ensemble de trajets
- **La mutation** : modification aléatoire dans le trajet d'un individu
- **L'adaptation** : plus le trajet est court, plus il est adapté au problème
- **La sélection naturelle** : élimination des individus les moins adaptés

Le Principe :

1. On génère aléatoirement une première population d'individu

2. On fait la sélection de la moitié des individus (sélection naturelle)

3. On arrange les individus restant en couple et on crée pour chaque couple un couple d'enfant qui représente des trajets ayant des caractéristiques de leurs deux parents

4. les enfants peuvent avoir une chance sur q d'avoir une mutation (paramètre q définie par l'utilisateur)

5. On retourne à l'étape 2.

La Première génération : On génère aléatoirement *n* trajets possibles. Le nombre *n* est fixé par l'utilisateur. Plus il sera grand plus l'algorithme sera efficace et long.

Algorithme 2.2: Exemple de Voyageur de commerce par Algorithme Génétique

Données *: n; i; q; L ; % ;n* :
n est le nombre d'individu par population, *i* le nombre d'itération voulue, *q* l'inverse le nombre de chances qu'un individu soit un mutant à la naissance, et *L* contient les villes à parcourir avec des informations sur les distances entre les villes
Résultat : La liste % contient la liste du parcours de villes à effectuer
Début
P'←Générer-population-initial S
Pour z = 1 à i **faire**
si z = 0 mod 2 **alors**
P←Sélection-élitiste P
fin
si z = 1 mod 2 **alors**
P Sélection-par-tournois P
fin
P Passage-génération-suivante P
fin
Élément-le-mieux-adapté P
fin

La Sélection : On ne souhaite garder que la moitié des individus. Il existe deux méthodes pour faire cette sélection.

- méthode élitiste : On ne conserve que la meilleure moitié.
- méthode par tournois : on regroupe les individus par couple et on les laisse se battre. Le moins adapté est supprimé et l'autre est conservé.

On appliquera chacune des deux méthodes une fois sur deux.

Le Passage à la génération suivante : On regroupe les individus par couple et chaque couple passe par l'opérateur croisement.

L'opérateur de croisement on applique un individu sur q.

L'opérateur mutation qui modifie de façon aléatoire le trajet. Par exemple inter changer la position de deux points dans le trajet.

Le Critère d'arrêt : cet algorithme ne ce termine pas à priori. Il faut donc demander à l'utilisateur de fixer un nombre *m* fini d'itération à faire.

2.9. Conclusion

Malgré l'existence d'un grand nombre de méthodes de clustering ainsi que leur utilisation avec succès dans de nombreux domaines, le Clustering pose encore de nombreux problèmes. Ces problèmes sont liés d'une part, au manque de précision dans la définition de ce qu'est réellement un cluster mais également dans la difficulté de définir une mesure de similarité entre objets au encore dans la définition d'une fonction objective pour un problème donnée, d'autre part.

[Jain et al, 1988] ont listé un ensemble de questions qu'il est nécessaire de se poser lorsqu'on entreprend d'effectuer une tâche de clustering. Cette liste met en avant la multiplicité et la nature différente des paramètres à prendre en compte dans ce type d'approche :

Qu'est ce qu'un cluster ? Quels attributs doivent être utilisés ? Les données doivent elles être normalisées et, est ce qu'elles contiennent des objets atypiques et des clusters? Comment est définie la similarité entre deux objets ?

Combien de clusters sont présents ? Quelle méthode de clustering doit-on utiliser ? Et, est ce que la partition découverte est valide ?

Nous allons étudier dans le chapitre quatre certains des problèmes et limites récurrents en clustering qui sont soulevés par ces questions.

PARTIEII:

CONTRIBUTION

CHAPITREIII:

HYBRIDATION DE L'ALGORIRHME CHAUVE-SOURIS AVEC LE GENETIQUE POUR UNE MEILLEURE QUALITE DE SOLUTION

3.1. Introduction

Ces dernières années, de nouvelles approches de résolution de problèmes d'optimisation ont été développées, permettant d'obtenir des avantages par rapport aux techniques plus traditionnelles. La nécessité de rechercher de nouvelles techniques d'optimisation découle du fait que les techniques traditionnelles sont devenues inefficaces pour résoudre des problèmes pratiques [Hegazy et al, 2017].

La complexité des problèmes d'optimisation à résoudre pour des systèmes de plus en plus complexes, les nombreuses variables de conception ainsi que leurs exigences pratiques rendent ces problèmes difficiles à gérer en utilisant des méthodes d'optimisation traditionnelles [Karaboga et al, 2007].

Récemment, les techniques méta-heuristiques ont connu un engouement fort, tant au niveau académique qu'industriel, à la recherche de solutions robustes à des problèmes complexes. S'inspirant des phénomènes naturels, ces techniques permettent d'obtenir des solutions convenables à des problèmes d'optimisation NP difficiles [Mridul et al, 2015].

Cependant, ces algorithmes ont tendance à produire des solutions différentes même lorsque leurs conditions initiales restent constantes à chaque exécution et ce, en raison de leur nature aléatoire. Ils sont préférés pour ces fonctions qui ont plusieurs optimaux locaux, car ils peuvent échapper facilement aux minimums locaux en dépit de leur lenteur de convergence [Sayadi et al, 2013].

L'idée fondamentale derrière ces techniques est de simuler des systèmes biologiques et physiques dans la nature, tels que l'évolution naturelle, le système immunitaire, l'intelligence d'essaim, le processus de recuit, etc., dans un algorithme numérique. Ces algorithmes diffèrent dans la façon avec laquelle ils évoluent dans l'espace de recherche, en se basant sur une stratégie associée, souvent inspirée de la nature [Paril, 2008].

L'une des méthodes méta-heuristiques récemment proposées et des plus prometteuses, est l'algorithme de chauve-souris dont l'appellation d'origine est Bat Algorithm. C'est une méta-heuristique très récente qui se base sur la simulation du comportement d'écholocation des chauves-souris [Yang, 2010]. La capacité de l'écholocation est fascinante car ces chauves-souris peuvent trouver leurs proies et discriminer différents types d'insectes, même dans l'obscurité complète. Elles y parviennent en émettant des signaux sonores vers l'environnement et en écoutant les échos qui leur renvoient. Elles peuvent identifier, localiser d'autres objets et mesurer instinctivement la distance à laquelle ils sont éloignés en retardant le retour du son figure 3.1.

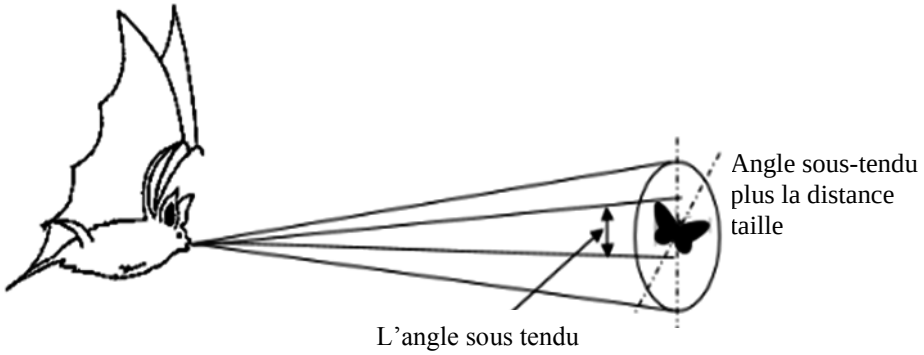


Fig 3.1 signal sonar d'une chauve-souris

En plus de ces modifications, des chercheurs ont étendu l'algorithme de chauve-souris afin de pouvoir gérer un ensemble d'applications de nature à obtenir de meilleurs résultats, tels que les problèmes multi-objectifs, les problèmes de clustering des données, et d'autres applications d'optimisation [Yang et al, 2013] [Ramesh, 2013].

Dans ce chapitre nous présentons une amélioration de l'algorithme de chauve-souris standard, modifiant certaines règles d'évolution et en introduisant la règle génétique de croisement. Ces modifications visent à améliorer la qualité des résultats, la consommation en temps d'exécution, ainsi que les caractéristiques d'exploration et de convergence de l'algorithme de chauve-souris.

Dans ce Chapitre nous présentons l'algorithme standard de chauve-souris. Nous introduisons notre algorithme modifié qui est présenté dans la

section 3. Par la suite une illustration des expérimentations et une comparaison des résultats. À la fin, une conclusion est présentée à la section 5.

3.2. L'Algorithme de Chauve-souris standard

Parmi plusieurs méta-heuristiques biologiquement inspirées, l'algorithme des chauves souris (Bat Algorithm: BA) proposé par Xin She Yang [Yang,2010] ont attiré l'attention des chercheurs qui travaillent dans le domaine des algorithmes d'optimisation à inspiration biologique.

Cet algorithme est basé sur le comportement de l'écholocation de microchiroptères [Suga, 1990] figure3.1. L'écholocation est un sonar biologique (à cause de manque de vision) qui permet détecter la distance, et ils ont aussi la capacité de faire la différence entre la nourriture/proie et les obstacles. Il s'appuyé sur une technique de régulation de fréquence pour augmenter la diversité des solutions dans la population, et par là même, il tente d'équilibrer l'exploration et l'exploitation pendant le processus de recherche, en imitant les variations des taux d'émission d'impulsions et la vitesse des chauves-souris lors de la recherche de proies [Yang et al, 2013].

L'algorithme standard de chauve-souris présente de nombreux avantages; L'un d'entre eux est qu'il peut obtenir une convergence rapide aux étapes initiales en passant de l'exploration à l'exploitation. Cela en fait un algorithme efficace lorsqu'une solution rapide est nécessaire. Afin d'améliorer les performances, de nombreuses modifications ont été ajoutées pour augmenter la diversité de la solution et pour améliorer les performances de l'algorithme de chauve-souris standard [Yang et al, 2013] [Ramesh, 2013].

3.2.1 Initialisation de l'Algorithme de chauve-souris

La population initiale est générée de façon aléatoire pour *n* nombre de chauve-souris. Chaque individu de la population est décrit par un vecteur à valeurs réelles avec une dimension D. L'équation 3.1 est utilisée pour générer la population initiale.

$$\mathbf{X_{ij} = X_{min\,j} + rand(0,1)(X_{max\,j} - X_{min\,j})} \quad \textbf{(3.1)}$$

Où i = 1,2, ..., n; j = 1,2, ..., d; X_{maxj} et X_{minj} sont les limites supérieures et inférieures pour la dimension j.

3.2.2 Solution, fréquence et vitesse

Dans les simulations, nous utilisons naturellement des chauves-souris virtuelles. Nous devons définir les règles, comment mettre à jour leurs positions x_i et les vitesses v_i, dans un espace de recherche bidimensionnel, à chaque itération *t*. Parmi toutes les chauves-souris, il existe une meilleure solution courante x*. Les règles précédentes peuvent être traduites pour obtenir les nouvelles solutions x_i^t et vitesses v_i^t à l'étape *t*, dans les équations suivantes de mise à jour.

$$\mathbf{f_i = f_{min} + (f_{max} - f_{min})\beta}, \quad \textbf{(3.2)}$$

$$\mathbf{V_i^t = V_i^{t-1} + (X_i^t - X^*)f_i}, \quad \textbf{(3.3)}$$

$$\mathbf{X_i^t = X_i^{t-1} + V_i^t} \quad \textbf{(3.4)}$$

Où $\beta \in [0,1]$ est un vecteur aléatoire issu d'une distribution uniforme. X* est la meilleure solution globale courante qui est déterminée en comparant toutes les solutions parmi tous les *n* chauves-souris. Alors que $\lambda_i f_i$ est l'augmentation de vitesse, nous utilisons f_i pour régler la vitesse tout en fixant l'autre facteur λ_i. La plage de valeurs de f_i diffère d'un problème à l'autre en fonction du domaine, de la taille, etc. Initialement, chaque chauve-souris reçoit de manière aléatoire une fréquence qui est dérivée uniformément de $[f_{min}, f_{max}]$. Lorsqu'une solution est sélectionnée parmi les meilleures solutions courantes, une nouvelle solution pour chaque chauve-souris est générée localement à l'aide d'une transformation intégrant un facteur aléatoire.

$$\mathbf{X_{new} = X_{old} + \varepsilon A^t}, \quad \textbf{(3.5)}$$

Où $\varepsilon \in [-1,1]$ est un nombre aléatoire, alors que A^t est le volume moyen de toutes les chauves-souris à cette étape de traitement. La mise à jour des vitesses et des positions des chauves-souris est similaire à la procédure d'optimisation standard des essaims de particules [Induja et al, 2016] [Ramesh,

2013]. Étant donné que le contrôle de la portée du mouvement des particules envahissantes, l'algorithme de chauve-souris peut être considéré comme étant une combinaison équilibrée de l'optimisation standard des essaims de particules et de la recherche locale intensive, contrôlée par le volume et le taux de pulsation.

3.2.3 La Mise à jour du volume et du taux de pulsation

Le volume A_i et le taux de pulsation r_i doivent être mis à jour à chaque itération. Au fur et à mesure que l'intensité diminue une fois que la chauve-souris a trouvé sa proie, alors que le taux de pulsation r_i augmente, le volume peut être choisi comme valeur de commodité. Lorsque le volume atteint le minimum A_{min}, cela signifie que la chauve-souris a trouvé la proie et a cessé d'émettre un son. Les équations suivantes montrent comment le volume Ai et le rythme r sont mis à jour pendant les itérations.

$$\mathbf{A_i^{t+1} = \alpha A_i^t,} \tag{3.6}$$

$$\mathbf{r^{t+1} = r_i^0[1 - exp(-\gamma t)],} \tag{3.7}$$

Où α et γ sont des constantes. L'α constant est similaire au facteur de refroidissement dans le recuit simulé. Pour tout $0 < \alpha < 1$ et $\gamma > 0$ tel que

$$\mathbf{A_i^t \to 0, r_i^t \to r_i^0, ast \to \infty} \tag{3.8}$$

Le choix des paramètres nécessite une certaine expérimentation. Chaque chauve-souris devrait avoir différentes valeurs de sonorité et taux d'émission d'impulsion. Le volume et les taux d'émission ne sont mis à jour que si les nouvelles solutions sont améliorées, ce qui signifie que ces chauves-souris se déplacent vers la solution optimale.

Algorithme 3.1. Pseudo code de l'algorithme de chauve-souris standard.

1. Fonction objectif: f(x), x = (x_1, ..., x_d)
2. Initialiser la population de chauve-souris x_i et la vitesse v_i, i = 1,2, ..., n
3. Définir la fréquence d'impulsions $f_i$de chaque position x_i
4. Initialiser le taux de pulsation r_i et le volume A_i
5. Tant que (t <nombre maximum d'itérations)
   - 5.1. Générer de nouvelles solutions en ajustant la fréquence et en actualisant les vitesses et les positions / solutions. (équations 3.2, 3.3; 3.4).
   - 5.2. Si (rand> r_i)
     - 5.2.1. Sélectionnez une solution parmi les meilleures solutions
     - 5.2.2. Générer une solution locale autour de la meilleure solution sélectionnée x* (équation 3.5)
   - 5.3. Fin si
   - 5.4. Si (rand <A_i et f(x_i) <f(x*))
     - 5.4.1. Accepter de nouvelles solutions
     - 5.4.2. Augmenter ri et réduire Ai (équations 3.6, 3.7)
   - 5.5. Fin si
   - 5.6. Trouver la meilleure solution x*
6. Fin tant que
7. Afficher les résultats donnés par la meilleure solution x*
8. Fin de l'algorithme

3.3. Hybridation Chauve-souris avec l'algorithme génétique

Dans cette section nous développons notre proposition de modification de l'algorithme de chauve-souris standard, que nous appelons BATA (Bat Accéléré). En effet, des études montrent que les chauves-souris utilisent le retard de l'émission et la détection de l'écho, la différence de temps entre leurs deux oreilles et les variations de volume des échos pour construire un scénario tridimensionnel de l'environnement. Ils peuvent détecter la distance et l'orientation de la cible, le type de proie et même la vitesse de déplacement de la proie, comme les petits insectes.

Pour simplifier, on suppose f $\in$ [0, fmax]. On sait que les fréquences plus élevées ont des longueurs d'ondes courtes et parcourent une distance plus courte. Pour les chauves-souris, les plages typiques sont de quelques mètres. La fréquence est donc un facteur efficace important qui peut affecter la recherche globale de l'algorithme global. Afin d'améliorer l'algorithme de chauve-souris, une modification est appliquée pour améliorer les capacités d'exploitation de son exploration. La nouvelle étape de chaque chauve-souris est contrôlée par le vecteur de position, la meilleure position globale et la fréquence. L'équation de

la vitesse est supprimée, ce qui réduit le temps d'exécution. Cette modification accélère la convergence de l'algorithme global. L'équation 3.3 est annulée et l'équation de la nouvelle étape (équation 3.4) est modifiée comme suit

$$\mathbf{x_i^t = f_i x_i^{t-1} + (1 - f_i) r_i x^*} \quad \textbf{(3.9)}$$

L'équation (3.9) est modifiée de façon à intégrer un effet multiplicatif de la solution courante par la fréquence au premier terme et un deuxième effet multiplicatif de la meilleure solution globale au deuxième terme.

De plus, si la nouvelle solution obtenue x_i^t n'est pas meilleure que la solution x_i^{t-1}, un croisement est effectué entre les deux solutions. Si le résultat obtenu de ce croisement (S(t)) est meilleur que x_i^{t-1}, il le remplacera. La règle de croisement applique les équations (3.10) et (3.11) et sélectionne la meilleure des deux qui va devenir la nouvelle solution à intégrer à la population.

$$\mathbf{S^{(t1)} = x_i^t + \alpha x_i^{t-1}} \quad \textbf{(3.10)}$$

$$\mathbf{S^{(t2)} = x_i^t + (1 - \alpha) x_i^{t-1}} \quad \textbf{(3.11)}$$

Où $\alpha \in [0, 1]$.

L'effet de cette modification est illustré par les résultats de test sur un ensemble de benchmarks présentés dans la section suivante.

3.3.1. RÉSULTATS EXPÉRIMENTAUX

3.3.1.1 Fonctions de référence

Afin de vérifier l'efficacité de modification proposée, notre algorithme est testé en utilisant cinq fonctions de test de référence de minimisation avec zéro comme solution optimale en différentes dimensions comme on le voit dans le tableau 3.1. Les valeurs de "Meilleure, +mauvaise, ainsi que l'écart-type" sont montrées dans le tableau 3.2.

TAB 3.1. Les Fonctions Benchmark de Test

nom de la fonction	La dimension	La plage	La Fonction
Sphère	10, 50	[-5, 5]	$\mathbf{f_1(x) = \sum_{i=1}^{n} x_i^2}$
Griewangk	10, 50	[-600,600]	$\mathbf{f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1}$
Ackley Function	10, 50	[-100,100]	$\mathbf{f_3(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2) - \exp(\frac{1}{n}\sum_{i=1}^{N} \cos(2\pi X_i))}}$
Rastrigin	10, 50	[-15,15]	$\mathbf{f_4(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]}$
Rosenbrock	10, 50	[-15,15]	$\mathbf{f_5(x) = \sum_{i-1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2}$

3.3.1.2 Les Paramètres

Notre algorithme, appelé BATA (BAT Accéléré), l'algorithme chauve-souris standard (BA) et les autres modifications les plus notables sont testés avec 25 exécutions indépendantes pour chaque fonction de test. La taille de la population (nombre de chauves-souris) est fixée à 30. Le nombre de générations est fixé à 1000, et 2000 pour la dimension = 10 et 50 respectivement pour chaque exécution. La valeur de la fréquence minimale est fixée à 0 alors que la valeur maximale est fixée à 2. La mise en œuvre s'effectue à l'aide du langage Java sur un PC à processeur Intel® I3, CPU 3 GHz et RAM 4Go.

3.3.1.3 Comparaison

En comparant les résultats avec BA et les modifications précédentes comme Hybrid Bat Algorithm (HBA) [Fister et al, 2013], Modified Bat Algorithm (MBA) [Zhen et al, 2013], Novel Adaptive Bat Algorithm (NABA) et "Bat

Algorithm with Self-Adaptive Mutation" (BA-SAM) [Wasi et al,2014], nous montrons que notre algorithme est meilleure en terme de fonction objectif(fitness) et déviation standard (qualité du résultat), comme illustré dans le tableau 3.2.

*TAB 3.2 Comparaison entre BA, HBA, MBA, NABA, BA-SAM et **BATA**.*

Fonction	nom de la fonction	Dimension	Algorithme	La meilleure fitness	+ mauvaise fitness	Déviation standard
F1	Sphère	10	BA	1.42	1.03e+0.1	2.11
			HBA	4.31e-09	2.24e-03	1.41e-07
			MBA	3.73e -03	1.42e -02	3.31e- 03
			NABA	1.42e-04	1.32	2.81e-01
			BA-SAM	1.74e-06	3.41e-01	1.15e-01
			BATA	**2.44e-13**	**5.40e-11**	**2.42e-11**
		50	BA	7.17e+02	1.27e+03	9.42e+01
			HBA	4.17e-09	2.89e-03	1.46e-07
			MBA	1.87e+02	5.51e+02	1.14e+02
			NABA	2.24e+02	5.20e+02	8.21e+01
			BA-SAM	2.01e+02	5.49e+02	8.77e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
F2	Griewangk	10	BA	5.81	4.15e+01	9.19
			HBA	2.25e-09	3.97e-05	1.14e-07
			MBA	2.45	2.46e+01	5.19
			NABA	2.68	5.16e+01	1.21e+01
			BA-SAM	1.31e-01	1.14e+01	2.91
			BATA	0	1.24e-12	4.43e-13
		50	BA	7.15e+02	1.11e+03	9.91e+01
			HBA	4.80e-09	2.82e-03	1.61e-07
			MBA	1.84e+02	5.51e+02	1.21e+02
			NABA	2.26e+02	5.21e+02	8.27e+01
			BA-SAM	2.22e+02	5.19e+02	8.17e+01
			BATA	**0**	**8.19e-07**	**3.80e-07**
F3	Ackley	10	BA	1.17e+01	1.68e+01	1.40
			HBA	6.31e-04	2.00e+01	1.78e+01
			MBA	3.21e-02	1.29	3.68e-01
			NABA	1.21e-01	1.25e+01	4.26
			BA-SAM	2.09e-02	6.44	1.54
			BATA	**7.46e-12**	**1.46e-08**	**4.96e-09**
		50	BA	7.05e+02	1.13e+03	9.97e+01

			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	1.85e+02	5.58e+02	1.21e+02
			NABA	2.29e+02	5.20e+02	8.47e+01
			BA-SAM	2.02e+02	5.49e+02	8.47e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
F4	Rastrigi	10	BA	8.59e+01	1.94e+02	2.53e+01
			HBA	5.12	2.38e+01	1.69e+01
			MBA	1.46e+01	3.48e+01	4.35
			NABA	1.34e+01	5.08e+01	9.33
			BA-SAM	6.73	3.83e+01	8.85
			BATA	**1.98e-11**	**3.03e-08**	**1.09e-08**
		50	BA	7.05e+02	1.13e+03	9.97e+01
			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	1.85e+02	5.58e+02	1.21e+02
			NABA	2.29e+02	5.20e+02	8.47e+01
			BA-SAM	2.02e+02	5.49e+02	8.47e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
F5	Rosenbrock	10	BA	1.50e+03	1.22e+05	3.66e+04
			HBA	6.34e-02	5.10e+02	7.73
			MBA	4.83e-09	2.89e-03	1.66e-07
			NABA	5.57e-01	4.19e+04	1.02e+04
			BA-SAM	2.72	4.70e+02	8.75e+01
			BATA	**2.63**	**8.03**	**1.86**
		50	BA	1.96e+05	2.06e+06	4.48e+05
			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	4.23e-09	2.85e-03	1.26e-07
			NABA	2.11e+02	4.76e+05	1.18e+05
			BA-SAM	3.14e+02	1.08e+06	1.96e+05
			BATA	**1.89e+00**	**4.39e+01**	**1.23e+01**

Le tableau 3.2 illustre l'application de notre algorithme BATA sur les cinq fonctions benchmarks et montre qu'il réalise de meilleurs résultats que ceux des autres algorithmes et ce, sur toutes ces fonctions benchmark ainsi que les différentes dimensions utilisées.

3.4. Travaux similaires

Dans la littérature, des travaux de recherche ont proposé un ensemble de modifications ou d'hybridations sur l'algorithme de chauve-souris dans le but d'améliorer ses performances [Mridul et al, 2015].

Wasi M. a présenté un algorithme de chauve-souris auto-adaptatif et amélioré pour le problème de l'optimisation numérique globale dans les domaines continus. Il a introduit deux équations de recherche de solution

améliorées. Il a également utilisé une probabilité de sélection pour contrôler la fréquence d'emploi qui conduit à un nouveau mécanisme de recherche auto-adaptable pour l'algorithme de chauve-souris [Wasi et al, 2014].

Yilmaz S. a amélioré les mécanismes d'exploration et d'exploitation de l'algorithme de chauve-souris par trois modifications, la première a analysé la structure de la vitesse avec le processus de mise à jour pondérée par l'inertie. Le second prend en compte la différence entre la solution courante et la meilleure solution globale pour obtenir la plus proche et la plus grande dimension de la solution. La troisième modification consiste à faire l'hybridation entre la colonie d'abeilles artificielles et l'algorithme de chauve-souris pour améliorer la capacité d'exploration de l'algorithme de chauve-souris [Yilmaz et al, 2013].

Yilmaz S. a amélioré le mécanisme d'exploration en égalisant le volume et le taux d'émission d'impulsions à la dimension du problème en les attribuant à chaque dimension de la solution séparément, ce qui permet de réaliser simultanément différentes capacités d'exploration et d'exploitation [Yilmaz et al, 2014].

Amir H. a introduit le concept du chaos dans l'algorithme de chauve-souris pour accroître la mobilité de recherche globale pour une optimisation globale robuste. Cette méthode utilise des cartes chaotiques pour remplacer les variables aléatoires, améliorant l'efficacité de la recherche de l'algorithme de chauve-souris standard [Gandomi et al, 2014].

Zhen a supprimé le paramètre de vitesse et a ajouté le poids d'inertie de l'emplacement dans l'algorithme de chauve-souris qui est déterminé à l'aide de la distribution normale, puis la fréquence des impulsions émises est ajoutée au changement de position aléatoire et à la position optimale des chauve-souris [Zhen et al, 2013].

Fister a fait une hybridation entre les stratégies d'évolution différentielle et l'algorithme de chauve-souris standard aboutissant à un algorithme de chauve-souris hybride. Il a montré que cet algorithme améliore significativement la version originale de cet algorithme [Fister et al, 2013].

Ali A. a accéléré le processus de recherche en invoquant la méthode Nelder-Mead comme méthode de recherche locale afin d'affiner la solution la mieux obtenue à chaque itération [Ali, 2015].

3.5. Conclusion

Dans ce chapitre, nous avons introduit une amélioration de l'algorithme de chauve-souris, modifiant certaines règles d'évolution et introduisant la règle génétique de croisement. Ces modifications ont induit moins de temps d'exécution et les caractéristiques d'exploration et de convergence de l'algorithme proposé sont beaucoup mieux que dans l'algorithme standard.

Nous avons mis en œuvre, testé et évalué notre approche en l'appliquant sur un certain nombre de problèmes de référence, portant sur l'optimisation numérique. Les résultats obtenus (i.e. la qualité finale de la solution et les caractéristiques de convergence) démontrent clairement que l'algorithme proposé est supérieur, non seulement à l'algorithme de chauve-souris standard, mais aussi à d'autres algorithmes modifiant l'algorithme standard. Cette supériorité est démontrée et montrée à travers l'utilisation de benchmarks bien connues dans la littérature spécialisée.

CHAPITRE IV:

UN ALGORITHME GENETIQUE MODIFIE POUR LE PARTITIONNEMENT DE DONNEES

4.1. Introduction

L'exploration de données ajoute au regroupement des complications de très grandes séries de données avec de nombreux attributs de différents types. Cela impose des exigences informatiques uniques sur les algorithmes de clustering pertinents. Plusieurs d'algorithmes ont récemment émergé qui répondant à ces exigences et ont été appliqués avec succès à des problèmes d'exploration de données réelles. Dans la littérature, de nombreuses méthodes de regroupement ont été proposées là où les K-means et les algorithmes génétiques sont les plus remarquables [Hartano, 2015], [Han, 2001], [Scott, 1992], [Abdul Nazeer, 2009], [Ettaouil, 2013], [Romany, 2012] [Delavar, 2014]. Cependant, la plupart d'entre eux souffrent des inconvénients du fait du nombre de clusters et leurs centres initiaux soient fournis par l'utilisateur. Ainsi que de la mauvaise qualité de clustering [Sivanandam, 2008].

Dans ce chapitre, nous présentons un algorithme génétique pour générer non seulement les paramètres mentionnés ci-dessus, mais aussi pour améliorer la vitesse et la précision du processus de Clustering. Il est basé sur une structure et un processus de données appropriées qui gèrent les populations hétérogènes dans un algorithme génétique modifié. Généralement, l'utilisation de paramètres tels que les probabilités de croisement et de mutation qui s'adaptent à l'évolution de l'algorithme est un bon choix, car une plus grande diversibilité dans la population peut être obtenue, ce qui empêche l'algorithme de diminuer les minimums locaux. De plus, pour accélérer le processus d'algorithme génétique et augmenter la diversification individuelle de la population initiale, nous générons cette population initiale de deux façons: la première sous-population est obtenue de façon déterministe et la seconde avec une forme aléatoire. L'augmentation de la diversité de la population permettra d'obtenir une meilleure qualité. Le reste du chapitre est structuré comme suit:

La section 2 donne une analyse des deux principales techniques de regroupement de données; k-means et algorithmes génétiques. La section 3 détaille notre approche de regroupement de données génétiques en décrivant la structure proposée des chromosomes et des opérateurs génétiques. La section 4

donne les résultats de l'application de notre approche et discute de leurs avantages.

4.2. L'algorithme de Kmeans

Kmeans est formellement décrit dans l'algorithme 4.1. Le fonctionnement de Kmeans est illustré dans la figure 4.1, qui montre comment, à partir de trois centroïdes, les clusters finals sont trouvés dans quatre étapes de mise à jour d'assignation. Chaque sous-figure montre

(1) Les centroïdes au début de l'itération, et

(2) L'assignation des points à ces centroïdes.

Les centroïdes sont indiqués par des symboles «+», tous les points appartenant au même cluster ont la même forme de marqueur.

Algorithme 4.1. Kmeans

1. Sélection K
2. Affecte les points à la classe la plus proche,
3. Recalcule les nouveaux centres de gravité,
4. Répéter les étapes 2 et de 3 jusqu'à la convergence de l'algorithme (i. e. plus de changement de le partition).

(a) Itération1. (b) Itération 2. (c) Itéartion 3. (d) Iteration 4.

Fig 4.1 Exemple de Kmeans pour trouver 3 clusters.

Dans la première étape de la figure 4.1 (a), les points sont attribués aux centroïdes initiaux, qui sont dans le plus grand groupe de points. Pour cet

exemple, nous utilisons la moyenne comme centroïde. Une fois les points attribués à un centroïde, le centroïde est mis à jour. Encore une fois, la figure 4.1 de chaque étape montre le centroïde au début de l'étape et l'affectation de points à ces centroïdes. Dans la deuxième étape, les points sont attribués aux centroïdes qui sont à nouveau mis à jour. Dans les étapes 2, 3 et 4, qui sont représentées respectivement sur les figures b, c, d, deux centroïdes se déplacent vers les deux petits groupes de points au bas de la figure. Lorsque l'algorithme Kmeans se termine par la figure 4.1 (d) (parce qu'il n'y a plus de changement), les centroïdes ont identifié les groupements naturels de points. Pour certaines combinaisons de fonctions de proximité et de types de centroïdes, Kmeans converge toujours vers une solution; le Kmeans atteint un état dans lequel aucun point ne se déplace d'un groupe à l'autre et donc les centroïdes ne changent pas. Comme la plus grande partie de la couverture se produit dans les premières étapes, cependant, la condition sur la ligne 4 de l'algorithme 4.1 est souvent remplacée par une condition plus faible, e.i. répète jusqu'à ce que 1% des points modifient les groupes.

Nous considérons chacune des étapes de l'algorithme de base en plus un détail et fournissons ensuite une analyse de la complexité de l'espace et du temps de l'algorithme.

Assigner des points au Centroïde le plus proche

Pour assigner un point au centroïde le plus proche, nous avons besoin d'une mesure de proximité qui quantifie la notion du plus proche des données spécifiques considérées. La distance euclidienne [Tan et al, 2005]est souvent utilisée pour les points de données dans l'espace euclidien. Cependant, plusieurs types de mesures de proximité peuvent convenir à un type de données donné. Par exemple, la distance de Manhattan [tan el al 2005] peut être utilisée pour les données euclidiennes, alors que la mesure de Jaccard est souvent employée. Habituellement, les mesures de similarité utilisées pour les Kmeans sont relativement simples puisque l'algorithme calcule à plusieurs reprises la similarité de chaque point à chaque centroïde. Cependant, certain cas de données sont de faible dimension dans l'espace euclidien, il est possible d'éviter

de calculer beaucoup de similitudes, accélérant ainsi considérablement l'algorithme de Kmeans.

TAB 4.1 tableau de notion Kmeans

Symbole	Description
X	Un objet
C_i	Le i ^{ième} cluster
c_i	Le centroid de C_i cluster
C	Le centre de tout les points
m_i	Le nombre d'objets dans le i^{ième} cluster
M	Le nombre d'objets dans l'ensemble de données
K	Le nombre de clusters

Centroids et fonctions objectives

La quatrième étape de l'algorithme de Kmeans a été lancée de manière générale en recalculant le centroïde de chaque cluster, puisque le centroïde peut varier en fonction de la mesure de proximité pour les données et de l'objectif du regroupement. Le but de la classification est typiquement exprimé par une fonction objective qui dépend des proximités des points entre eux ou des centroïdes du cluster. Par exemple minimiser la distance au carré de chaque point au centroïde le plus proche. Nous illustrons cela avec deux exemples. Cependant, le point clé est le suivant: une fois que nous avons spécifié la mesure de proximité et une fonction objective, le centroïde que nous devrions choisir peut souvent être déterminé mathématiquement, dans l'espace euclidien. Considérons les données dont la mesure de proximité est la distance euclidienne. Pour notre fonction objective, qui mesure la qualité d'un **clustering**, nous utilisons la somme de l'erreur quadratique **(SSE)** equation 4.1**,** également appelée dispersion. En d'autres termes, nous calculons l'erreur de chaque point de données (par distance euclidienne le centroïde le plus proche), puis calculer la somme totale des erreurs au carré. Étant donné deux ensembles différents de clusters produits par deux séries différentes de K-means, nous préférons celui qui a la plus petite erreur. Cela signifie que les prototypes (centroïdes) de cette classification sont une meilleure représentation des points

de leur cluster. En utilisant la notation du tableau 8.1, le SSE est formellement défini comme suit (4.1):

$$\boldsymbol{SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist(c_i, x)^2} \quad \textbf{(4.1)}$$

Où *dist* est la distance Euclidien(L2) standard entre deux objets de l'espace euclidien.

Compte tenu de ces hypothèses, on peut montrer que le centroïde minimisant SSE du cluster est la moyenne. Utilisant la notion du tableau 7, le centroïde (moyenne) du $i^{ème}$ cluster est défini par l'équation (4.2) :

$$\boldsymbol{c_i = \frac{1}{m_i} \sum_{x \in C_i} x} \quad \textbf{(4.2)}$$

Le choix des Centres initiaux

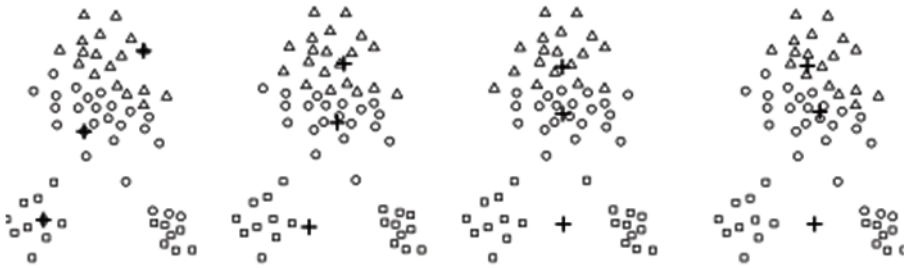
Lorsque l'initialisation aléatoire des centroïdes est utilisée, différentes exécutions de K-means produisent typiquement plusieurs **SSE**. Nous illustrons ceci avec l'ensemble des points bidimensionnels montrés dans la Figure 4.2, qui a trois groupes naturels de points. La figure 4.2 (a) montre une solution de clustering qui est le minimum global de SSE pour trois clusters, alors que la figure 4.2 (b) montre un clustering sous-optimal qui n'est qu'un minimum local. Le choix des centroïdes initiaux appropriés est l'étape clé de la procédure de base de Kmeans. Une approche commune consiste à choisir les centroïdes initiaux au hasard, mais les clusters résultants sont souvent pauvres.

(a) Clustering Optimal (b) Clustering Sous-Optimal

Fig 4.2. Un optimal et le non optimal clustering.

Exemple 1. (Pauvres centroïdes initiaux): Les centroïdes initiaux sélectionnés

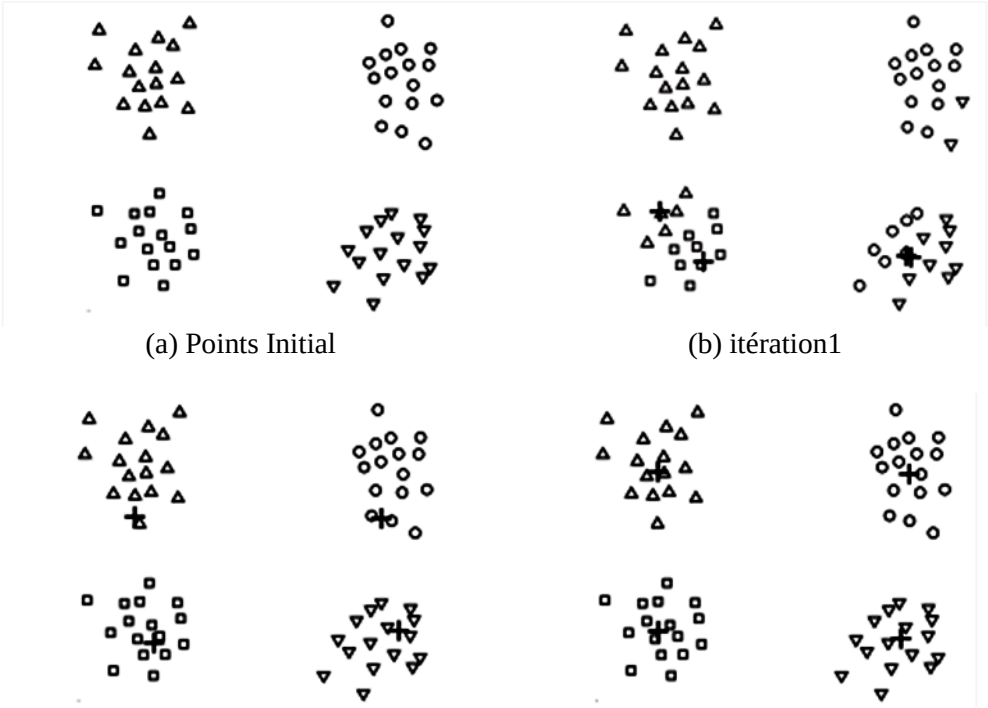
au hasard peuvent être médiocres. Nous en donnons un exemple en utilisant le même ensemble de données utilisé dans la figure 4.1 et la figure 4.2. Les figures4.1 et 4.3 montrent les clusters résultants de deux choix particuliers de centroïdes initiaux. (Pour les deux figures, les positions des centroïdes de cluster dans les différentes itérations sont indiquées par des croix), sur la figure 4.1, même si tous les centroïdes initiaux proviennent d'un cluster naturel, le clustering SSE minimal est toujours trouvé. Cependant dans la figure 4.3, nous obtenons un regroupement sous-optimal, avec une erreur quadratique plus élevée, .même si les centroïdes initiaux semblent être mieux distribués.



(a) itération1 (b) itération1 (c) itération1 (d) itération1

Fig 4.3. Pauvres centroides initiaux

Exemple 2. (Limite de l'initialisation aléatoire): Une technique couramment utilisée pour résoudre le problème du choix des centroïdes initiaux consiste à exécuter plusieurs séries, chacune avec un ensemble différent de centroïdes initiaux choisis au hasard. Et puis sélectionner un ensemble de clusters avec le minimum SSE. Bien que simple, cette stratégie peut ne pas fonctionner très bien, selon l'ensemble de données et le nombre de clusters recherchés. Nous le démontrons en utilisant l'ensemble de données d'échantillon illustré à la figure 4.4. Les données sont constituées de deux paires de clusters, où chaque paire (supérieure-inférieure) qui s’approche l’une de l’autre. La figure 4.4 (b-d) montre que si nous commençons avec deux centroïdes initiaux par paire de clusters, les centroïdes se redistribueront pour que les vrais clusters soient trouvés.



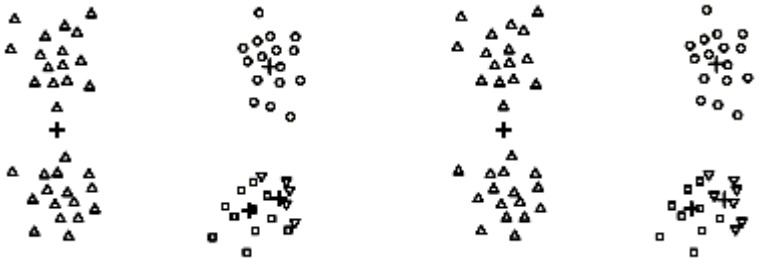
(a) Points Initial (b) itération1

(c) Itération2. (d) itération3

Fig 4.4. Deux paires de clusters avec une paire de centroïdes initiaux

Cependant, la figure 4.5 montre que si une paire de clusters n'a qu'un centroïde initial et que l'autre paire en a trois. Deux des véritables clustres seront combinées et un cluster vrai sera divisé. Notons qu'un **clustering optimal** sera obtenu aussi longtemps que deux centroïdes initiaux tomberont n'importe où dans une paire de clusters, puisque les centroïdes se redistribueront, un pour chaque cluster. Malheureusement, au fur et à mesure que le nombre de clustres augmente, il est de plus en plus probable qu'au moins un cluster ait un seul centroïde initial. Dans ce cas, comme les paires de clusters sont plus éloignées les unes des autres que les clusters d'une paire, l'algorithme de Kmeans ne redistribuera pas les centroïdes entre les paires de clusters, et donc seulement un minimum local sera atteint.

(a) itération2 (b) itération1



(a) itération3 (b) itération4

Fig 4.5. Deux paires de clusters avec plus ou moins deux centroïdes initiaux

En raison des problèmes rencontrés lors de l'utilisation de centroïdes initiaux choisis au hasard (qui sont souvent répétés), beaucoup d'autres techniques sont souvent utilisées pour l'initialisation. Une approche efficace consiste à prendre un échantillon de points et à les regrouper en utilisant une technique de classification hiérarchique. Les clusters K sont extraits de la classification hiérarchique et les centroïdes de ces clusters sont utilisés comme les centroïdes initiaux. Cette approche fonctionne souvent bien, mais elle n'est pratique que si (1) l'échantillon est relativement petit, de quelques centaines à quelques milliers (le regroupement hiérarchique est coûteux) et ;

(2) K est relativement faible par rapport à la taille de l'échantillon.

La procédure suivante est une autre approche de la sélection des centroïdes initiaux. Sélectionner le premier point au hasard ou prendre le centroïde de tous les points. Ensuite, pour chaque centroïde initial successif, sélectionner le point le plus éloigné de l'un des centroïdes initiaux déjà sélectionnés. De cette façon, nous

obtenons un ensemble de Centroides initiales qui sont garantis pour être non seulement choisis au hasard mais paraissent également bien séparés.

Malheureusement, une telle approche peut sélectionner des valeurs aberrantes, plutôt que des points dans des régions denses (clusters). En outre, il est coûteux de calculer le point le plus éloigné de l'ensemble actuel de centroïdes initiaux. Pour surmonter ces problèmes, cette approche est souvent appliquée à un échantillon de points. Les valeurs aberrantes étant rares, elles ont tendance à ne pas apparaître dans un échantillon aléatoire. En revanche, les points de toutes les régions denses sont susceptibles d'être inclus à moins que la taille de l'échantillon ne soit très petite. En outre, le calcul impliqué dans la recherche des centroïdes initiaux est grandement réduit car la taille de l'échantillon est généralement beaucoup plus petite que le nombre de points. Par la suite, nous discuterons d'autres approches utiles pour produire un clustering de meilleure qualité (SSE inférieur): utiliser une variante de kmeans moins sensible aux problèmes d'initialisation (en utilisant kmeans) et utiliser un algorithme génétique modifié pour corriger l'ensemble des clusters produit.

Kmeans et la gestion des clusters vides

Un des problèmes avec l'algorithme de base de Kmeans donné plus tôt, est que les clusters vides peuvent être obtenus si aucun point n'est alloué à un cluster pendant l'étape d'affectation. Si cela se produit, une stratégie est nécessaire pour choisir un centroïde de remplacement, sinon l'erreur quadratique sera plus grande que nécessaire. Une approche consiste à choisir le point le plus éloigné de tout centroïde actuel. Sinon, cela élimine le point qui contribue actuellement le plus à l'erreur quadratique totale. Une autre approche consiste à choisir le centroïde de remplacement dans le cluster qui a le plus fort SSE. Cela divisera généralement le cluster et réduira le SSE global du clustering. S'il existe plusieurs clusters vides, ce processus peut être répété plusieurs fois.

Complexité du temps et de l'espace de recherche:

L'espace requis pour kmeans est modeste car seuls les points de données et les centroïdes sont stockés. Particulièrement, le stockage requis est :

O ((m + K) n), où *m* est le nombre de points et *n* est le nombre d'attributs. Les contraintes de temps pour les Kmeans sont également minimes et linéaires dans le nombre de points de données. En particulier, le temps requis est O*(I*K* m* n)*, où I est le nombre d'itérations requises pour la convergence. Par conséquent la plupart des changements se produisent généralement dans les premières itérations. Kmeans est linéaire en m, le nombre de points est efficace aussi bien que simple à condition que K, le nombre de clusters, soit significativement inférieur à *m*.

Le nombre de façons de classer *n* objets dans k groupes est donné par [Liu,1968] dans l'équation(4.3) :

$$\boldsymbol{N(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{i}\binom{k}{i}(k-i)^{n}} \qquad \textbf{(4.3)}$$

Par exemple, si nous prenons n= 25 et K=5, il y a 2 435454 façons de repartir les 25 objets dans K groupes. Si le nombre de clusters est inconnu (ce qui est souvent le cas des applications de cas réel), les objets peuvent être repartis de $\sum_{i=1}^{n} N(i,k)$ façons. Soit pour l'exemple précédant $4* 10^{18,}$ il est donc évidant qu'un parcours exhaustif des solutions est impossible.

Les méthodes traditionnelles ne travaillent que sur un petit sous ensemble de l'espace de recherche (l'espace de recherche étant défini par le nombre de clusters). Elles obtiennent en général des optimas locaux et rarement globaux.

Les Méta-heuristiques, ayant déjà fait leurs preuves pour la résolution des problèmes combinatoires de grandes tailles, peuvent donc être intéressantes pour se dégager de ces optimas locaux et trouver de façon plus fréquente les optima globaux. [Jourdan, 2003].

4.3. Solution par approches évolutionnaires

Les approches évolutionnaires, motivées par l'évolution naturelle, utilisent des opérateurs évolutifs et une population de solutions pour obtenir la partition globalement optimale des données. Les solutions candidates au problème de la classification sont codées sous la forme de chromosomes. Les

opérateurs évolutionnaires les plus couramment utilisés sont: sélection, croisement et mutation. [Jourdan, 2003] Chacun transforme un ou plusieurs chromosomes d'entrée en un ou plusieurs chromosomes de sortie. Une fonction objective évaluée sur un chromosome détermine la probabilité de survie d'un chromosome dans la génération suivante. Nous donnons ci-dessous une description de haut niveau d'un algorithme évolutif appliqué au clustering.

Les Algorithmes Génétiques (AGs) effectuent une recherche globalisée de solutions alors que la plupart des autres procédures de clusters effectuent une recherche localisée. Dans une recherche localisée, la solution obtenue à l'itération suivante de la procédure se trouve au voisinage de la solution actuelle. Dans ce sens, l'algorithme k-means, les algorithmes de clustering flou, les algorithmess utilisés pour le clustering, divers schémas de recuit simulé et la recherche tabou sont tous des techniques de recherche localisées. Dans le cas des AGs, les opérateurs de croisement et de mutation peuvent produire de nouvelles solutions totalement différentes des solutions actuelles. Nous illustrons ce fait sur la figure 4.6. Supposons que le scalaire X soit codé en utilisant une représentation binaire de 5 bits, et que S_1 et S_2 soient deux points dans l'espace de recherche unidimensionnel. Les valeurs décimales de S_1et S_2 sont respectivement 8 et 31.

Leurs représentations binaires sont S_1= 01000 et S_2= 11111. Appliquons le croisement à point unique à ces chaînes, entre le deuxième et le troisième bit comme indiqué ci-dessous :

01!000

11!111

Cela produira une nouvelle paire de chromosomes S_3=01111 et S4=11000, ce qui correspond respectivement à 15 et 24 en décimale. De même, en mutant le plus significatif bit dans la chaîne binaire 01111 en 11111. Ainsi, comme le montre la figure4.6, des sauts, ou des écarts entre les points dans les générations successives sont beaucoup plus grandes que celles produites par d'autres approches.

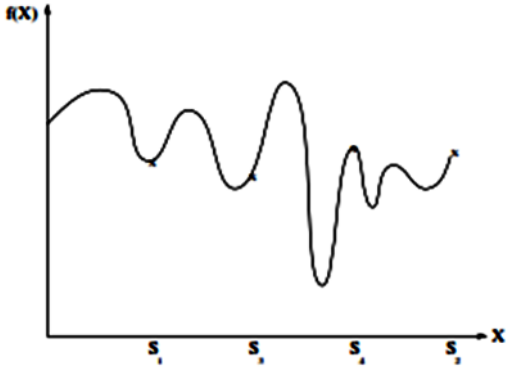


Fig.4.6. Une recherche globale effectue par l'Algorithme Génétique

Le premier article qui a utilisé les AG pour le regroupement est [Raghavan et al, 1979], du fait pour minimiser l'erreur quadratique d'un regroupement. Chaque point ou chromosome représente une partition de N objets en K clusters et est représenté par une chaîne K-aire de longueur N.

4.3.1. Algorithme Génétique modifié pour le partitionnement

L'objectif de notre approche est d'améliorer les résultats du clustering en minimisant les erreurs et le temps d'exécution. C'est une forme modifiée de l'algorithme génétique classique. En effet, l'entrée d'un tel processus d'algorithme génétique modifié est un ensemble de données, qui est stocké dans un vecteur, et génère automatiquement le nombre de clusters avec leurs centres centroides initiaux, tout en définissant des opérateurs de croisement et de mutation appropriés. En outre, il est basé sur une structure de chromosomes qui est différente en nombre de gènes constitutifs.

4.3.1.1. La Représentation de chromosome

La représentation chromosome est un problème d'encodage effectué aux étapes les plus importantes dans l'utilisation de l'algorithme génétique pour résoudre un problème [Najmah, 2016]. Pour coder à la fois le nombre de clusters et leurs centres, nous proposons une structure chromosomique qui applique un véritable algorithme génétique codé au problème de clustering, où les opérateurs de croisement et de mutation sont appliqués directement aux valeurs de paramètres réels. L'utilisation de valeurs de paramètres réels dans la

représentation GA présente plusieurs avantages par rapport au codage binaire. L'efficacité de l'AG est augmentée car il n'est pas nécessaire de convertir les variables de la solution en type binaire, moins de mémoire requise, il n'y a pas de perte de précision par la discrétisation à des valeurs binaires ou autres, et il existe une plus grande liberté d'utilisation de différentes sources génétiques des opérateurs.

Dans ce travail, la population existante est traitée sous la forme de plusieurs sous-populations de taille différente. En d'autres termes, le nombre de gènes dans chaque cluster diffère. Un chromosome dans une sous-population peut contenir un nombre K de gènes variant de 2 à kmax. Kmax est le nombre maximal de cluters. Ces gènes contiennent des informations correspondantes sur les centres de cluster. Ici, chaque gène du chromosome représente le centre de cluster, et le nombre K de gènes dans un chromosome représente le nombre de clusters. On suppose que la valeur de K se situe dans l'intervalle [2; Kmax]. Ainsi, chaque solution i est une chaîne de longueur fixe représentée par les centres de cluster cij; j = 2, ..., Kmax. Ensuite, la solution i de la population est représentée par un vecteur comme suit (figure 4.7):

c_{i1}	. . .	c_{ij}	. . .	c_{iKmax}

Fig.4.7. Représentation de chromosome par notre approche

4.3.1.2 La Fonction objectif

La fonction de conditionnement physique est une mesure de profit dont nous avons besoin lors de l'optimisation. C'est une fonction objectif qui permet de résumer la proximité d'une solution donnée à la réalisation des objectifs et a un effet important sur la réussite d'un algorithme génétique. La condition physique est proportionnelle à l'utilité ou à la capacité de l'individu que représente le chromosome. La mesure de la condition physique contribue à l'évolution de bonnes solutions et à la mise en œuvre de la sélection naturelle. Dans ce travail, la forme physique d'un chromosome est calculée à l'aide de Mean Square Error (MSE) [Allaire, 2000]. La MSE calcule l'erreur entre les clusters équation 4.4.

$$\textbf{MSE}= \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \quad ; \qquad \textbf{Fitness} = \frac{1}{MSE} \tag{4.4}$$

Après les définitions de la fonction objectif, les différents paramètres des opérateurs d'algorithmes génétiques sont fixés.

4.3.1.3. La production de la population initiale

Pour accélérer le processus d'algorithme génétique et augmenter la diversité individuelle de la population initiale, nous générons cette population initiale en deux phases. La première consiste en une production déterministe et la seconde consiste en une génération aléatoire. Nous procédons à une telle génération afin d'obtenir 20% de la population initiale de la première phase et 80% de la deuxième phase.

TAB. 4.2. Listes des paramètres initiaux pour l'algorithme génétiques.

Paramètre	Valeur
Le nombre maximal de clusters, Kmax	15
La taille de la Population, P	80
Croisement probabilité	0.6
Mutation probabilité	0.3
Le nombre maximal d'itérations,	100

Dans cette phase, nous générons de manière déterministe la première sous-population qui représente 20% de la population totale. À partir de l'ensemble de données d'entrée, nous produisons un ensemble de données trié. Ensuite, l'ensemble de données trié est divisé en k segments égaux et les moyens de chaque segment considéré comme un centre. Cela se fera selon trois manières:

1) Les populations Kmax-1 sont créées à l'aide de la valeur modulaire du segment,
2) Les populations Kmax-1 sont créées en utilisant la valeur moyenne du segment,
3) Les populations Kmax-1 sont créées à l'aide du segment min- valeur max. dont un exemple, la première façon décrite dans l'algorithme.

Algorithme 4.2 l'initialisation de l'opérateur K-means "KMO"

1- Trier les données dans un vecteur en ordre croissant;
2- Pour (k=2, k<=Kmax-1, k++)
{
diviser ce vecteur en k segments égaux
Sélectionner de chaque segment la valeur modale, qui y considérée comme centre;
3- Créer la $k^{ième}$ population
}

4.3.1.4. La Reproduction

Les Algorithmes génétiques aident à rechercher la meilleure solution parmi un certain nombre de solutions possibles liées à la reproduction. La reproduction peut être mise en œuvre sous une forme algorithmique, basée sur les opérateurs de reproduction, adaptée au schéma de codage. L'objectif des opérateurs de reproduction est d'assurer la diversité de la population, de sorte que les solutions les plus adaptées puissent être dérivées dans le processus évolutif. Un tel processus d'évolution se compose d'opérations de croisement et de mutation. Au cours d'un tel processus, des chromosomes invalides peuvent être produits. Dans la représentation chromosomique proposée, la répétition des gènes produit des chromosomes invalides car un point de données ne peut pas être un point central de plus d'un cluster. Donc, pour pouvoir détecter la production de chromosomes invalides, nous classons les gènes chromosomiques dans l'ordre croissant. Ce faisant, nous visons à obtenir de petites longueurs, une détection rapide de chromosomes invalides et des opérations de croisement et de mutation plus rapides. Figure.4.8. illustre notre processus de croisement à l'aide d'un masque.

Le croisement

C'est un processus de reproduction que les chromosomes des enfants sont générés selon les valeurs de la fonction de condition physique de leurs parents. Nous définissons un opérateur de croisement de manière à pouvoir accepter les chromosomes parentaux avec différents nombres de gènes. Il se produit avec $cross_p$ de probabilité. Pour chacun des deux parents sélectionnés, un croisement

est appliqué, produisant deux descendants. À cette fin, un masque binaire est créé.

$$C^{(1)}_j = C^{(1)}_j + \alpha\, C^{(2)}_j \qquad (4.5)$$

$C^{(1)}_j(2)$

Sa longueur est égale à celle du parent le plus court. Dans ce masque, Le chiffre "1" se produit avec probabilité p_{one}.

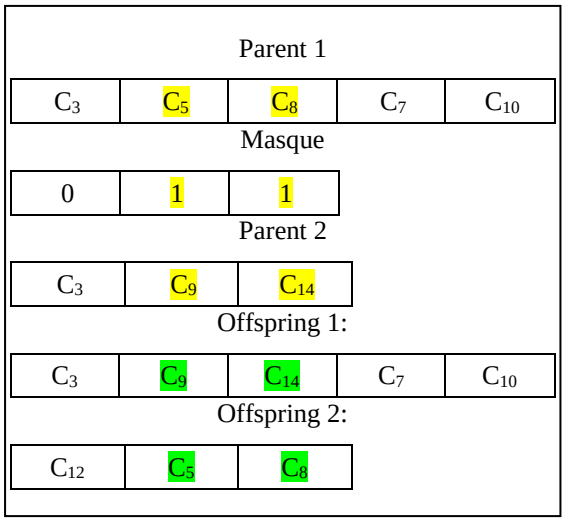


Fig.4.8. Exemple de croisement à l'aide de masque

Notre processus génétique global organise les opérations telles que représentées par la figure 5. Pour trouver le cluster le plus proche de chaque point, nous appliquons un opérateur k-means (KMO) [Najmah, 2016], qui est une étape de l'algorithme k-means classique. En d'autres termes, nous assignons des points de données à leurs clusters pour chaque nouveau chromosome à l'aide de l'opérateur KMO, afin de calculer sa forme physique. Il convient de noter que nous devrions vérifier la validité des chromosomes résultants pour éviter les centres de redondant. Dans le cas d'une redondance, nous reprenons la valeur chromosomique précédente. L'équation 4.5 représente les formules de croisement avec α comme un nombre aléatoire uniformément réparti tel que $\alpha \in [-1,1]$.

Algorithme 4.3.L'opérateur Kmeans "KMO"

Entrée : L'ensemble de points de données, kmax
Sortie : L'ensemble de points de données partitionnés en clusters K
Procédure:
1. Générer les populations initiales de chromosomes hétérogènes P;
2. Évaluer les valeurs de fitness de tous les chromosomes dans la population initiale;
3. Sélectionner les trois chromosomes parents, les meilleures de la population, en les désignant par (P1, P2, P3);
4. Appliquer les opérateurs de croisement et de mutation sur les deux chromosomes, les meilleures en (P1, P2, P3) pour générer deux descendants (O1, O2);
5. Évaluer les valeurs fitness de (O1, O2)
6. Sélectionnez les trois chromosomes, les meilleurs à partir de (P1, P2, P3, O1, O2) Et les insérer dans la population;
7. Dans le cas où un fils O1 ou O2 n'a pas été sélectionné à l'étape 6, remplacer le plus mauvais chromosome de la population par un fils de meilleurs fitness.
8. Mettre à jour les probabilités de croisement et de mutation à l'aide d'une fonction d'ajustement;
9. Répéter à partir de l'étape (3), n temps d'itération;
10. Extraire le chromosome le plus apte,
11. Extraire les clusters et leurs centres du chromosome le plus apte et appliquer l'opérateur KMO.

La Mutation

La mutation est destinée à empêcher la chute de toutes les solutions dans la population en une opportunité locale du problème résolu ou à améliorer le chromosome obtenu. La mutation s'effectue avec une probabilité inférieure à celle du croisement. Nous définissons deux types de mutation à prendre après le croisement. La mutation topologique est destinée à ajouter ou à supprimer des gènes provenant d'une progéniture. La mutation génétique est destinée à modifier un gène sélectionné d'une progéniture, d'une part et à modifier un gène sélectionné d'autre part. Ce processus est appliqué en sélectionnant de façon aléatoire un autre gène provenant d'une progéniture et en le remplaçant par un extrait aléatoire d'un point à partir d'un jeu de données. La procédure de mutation topologique est prise en évidence dans l'algorithme 4.5

Une vérification de validité est effectuée pour éviter les centres redondants et pour garder le numéro de cluster dans [2, Kmax].

Algorithme 4.5. La mutation de l'opérateur Kmeans

1. Pour chaque offspring, générez aléatoirement α comme un nombre aléatoire uniformément distribué tel que α ∈ [0, 1].
2. Si α ∈ [0, 0.5] choisit aléatoirement un centre de l'offspring le plus longue et le supprime.
3. Si α ∈] 0.5, 1] sélectionne aléatoirement une donnée du jeu de données et l'ajoute à l'
   offspring le plus courte.

4.4. Résultats expérimentaux

Pour évaluer notre approche et tester son efficacité, nous la comparons avec les K-means et le travail le plus proche de la nôtre de [Sivanandam, 2008] à l'aide de deux ensembles de données sélectionnés du repositionnement UCI de la base de données d'apprentissage machine, qui sont des jeux de données Iris et Lymphome. Il convient de noter que la première différence de notre approche et les deux méthodes mentionnées ci-dessus est que nous déterminons automatiquement le nombre k de clusters et leurs centres de base initiaux. Dans ce qui suit, nous montrons que notre approche surpasse au moins ces deux méthodes en considérant deux paramètres importants, i.e. l'erreur moyenne et le temps d'exécution moyen. Dans l'ensemble de données Iris, chaque point de données possède quatre valeurs caractéristiques, qui représentent la longueur et la largeur des sépales et des pétales, en centimètres. Il a trois classes avec 50 échantillons par classe. La valeur de k est donc choisie comme étant trois (clusters). Dans l'ensemble de données sur le lymphome, nous trouvons 62 échantillons composés de 4026 gènes couvrant trois classes, dont 42 échantillons diffusés de lymphome b-cellulaire large, neuf échantillons de lymphome folliculaire et 11 échantillons de leucémie lymphocytaire chronique à cellules B. Cet ensemble de données doit être divisé en trois clusters. Chaque ensemble de données a été utilisé pour chaque méthode pendant 10 fois, puis nous avons déterminé le temps et l'erreur moyens mentionnés dans le Tableau 4.3.

TAB. 4.3. L'utilisation de deux ensembles de données.

Dataset	Paramètres	k-means	MGAIK	L'approche proposée
Iris	Moy erreur	36.53	29.59	19.59
	Moy temps	18.34	10.56	9.34
Lymphoma	Moy erreur	44.12	38.49	22.15
	Moy temps	16.15	10.25	8.28

TAB 4.3 montre les expériences efficaces de la méthode proposée par rapport à K-means et le travail de [Sivanandam, 2008], le plus proche de la nôtre. Les expériences ont été menées pour mesurer l'erreur moyenne et les paramètres de temps d'exécution moyens pour les trois méthodes. Comme indiqué dans TAB 4.3, pour chaque jeu de données, l'erreur moyenne et le temps moyen sont répertoriés pour indiquer le compromis entre eux. Dans la table, on constate que notre méthode dépasse les deux autres, car elle a amélioré l'erreur moyenne et le temps d'exécution moyen.

4.5. Conclusion

Dans ce chapitre, nous avons introduit un algorithme génétique modifié, qui repose sur des structures différentes de chromosomes dans une population, afin d'être appliqué sur le problème du regroupement de données. Il permet l'utilisation de la détermination automatique des valeurs initiales des centres de cluster, fournissant de meilleurs résultats que l'utilisation de nombres aléatoires. L'approche a permis d'accélérer le processus de recherche en réduisant le temps d'exécution moyen et en obtenant les meilleures partitions de données. À l'avenir, nous prévoyons améliorer la fonction de conditionnement physique en enquêtant sur d'autres opérateurs de reproduction pour obtenir de meilleures performances.

CHAPITRE V:

UN ALGORITHME D'EVOLUTION DIFFERENTIELLE POUR L'OPTIMISATION DES REQUETES DE BASE DE DONNEES

5.1. Introduction

Dans les systèmes informatiques, les utilisateurs ont vraiment besoin d'utiliser des requêtes de jointure très volumineuse pour les prendre en charge dans leurs décisions commerciales [Swami et al, 1988]. Avec ces scénarios courants, les systèmes de gestion de bases de données (SGBD) commerciaux actuels deviennent incapables de retrouver des résultats satisfaisants vérifiant les exigences de performances minimales [Jarke et al, 1984]. De plus, les techniques de programmation dynamique appliquées à l'optimisation des requêtes présentent des limites en temps et en mémoire. L'espace de recherche croit exponentiellement avec l'augmentation linéaire de nombre de relations impliquées dans la requête, ces techniques doivent sauvegarder un nombre croissant de plans partiels en mémoire. Ce processus prend beaucoup de temps et se termine généralement sans solution lorsque l'optimiseur manque de mémoire.

Différentes approches ont été proposées pour remédier à cette situation.

Dans ce chapitre nous présentons un algorithme d'Evolution Différentielle (ED) modifié dans l'un des problèmes d'optimisation les plus importants en informatique: L'optimisation des requêtes de base de données pour une jointure large.

L'algorithme ED est considéré comme l'un des outils pratiques et efficaces pour les problèmes d'optimisation non-linéaire. Il réserve la stratégie de recherche globale basée sur la population et utilise l'opération de mutation pour la compétition différentielle et individuelle. En même temps, la capacité de mémoire spécifique d'ED lui permet d'ajuster sa stratégie de recherche avec une forte convergence et robustesse globales.

Dans ce qui suit, nous menons une étude et une comparaison en termes de durée d'estimation entre une simple requête sql ayant cinq jointures et une même requête utilisant notre algorithme d'Evolution différentielle. A la suite nous donnons une description générale sur notre algorithme d'optimisation basé sur la l'Evolution Différentielle.

5.2. Optimisation des requêtes dans les Bases de données

Les systèmes de gestion de bases de données (SGBD) sont des logiciels très complexes qui sont conçus pour définir, manipuler, récupérer et gérer les données stockées dans une base de données. Les SGBD relationnelles, dépendent d'une requête d'une base de donnée [Gonçalves et al, 2014] et, cette requête peut être formée par de nombreuses relations, qui peuvent être filtrées et / ou connectées par opérateurs relationnels [Molina et al, 2008].

Dans les systèmes de bases de données relationnelles, la jointure est l'un des opérateurs algébriques relationnels et son coût d'exécution est très élevé [Dong et al, 2007]. L'optimisation de requête transforme une expression de requête déclarative en plans d'exécution de requête (PER) procéduraux et détermine le plan d'exécution de requête le plus bas. Cependant, lorsqu'une requête comprend plus de 10 relations, on l'appelle une « grande requête de jointure ».

5.2.1. Processus d'optimisation des requêtes

Dans une base de données relationnelle, toutes les informations peuvent être trouvées dans une série de tables. Une requête est composée d'opérations sur ces tables [Ghanemi et al, 2008] ; Notant que les requêtes les plus courantes sont : *Select-Project-Join*. Les permutations de l'ordre de jointure ont l'effet le plus important sur la performance des requêtes relationnelles [Özsu et al, 1999]. Le processus d'optimisation des requêtes illustré dans la figure 5.1 consiste à obtenir une requête sur ***n*** relations et à générer un meilleur Plan d'exécution d'une requête.

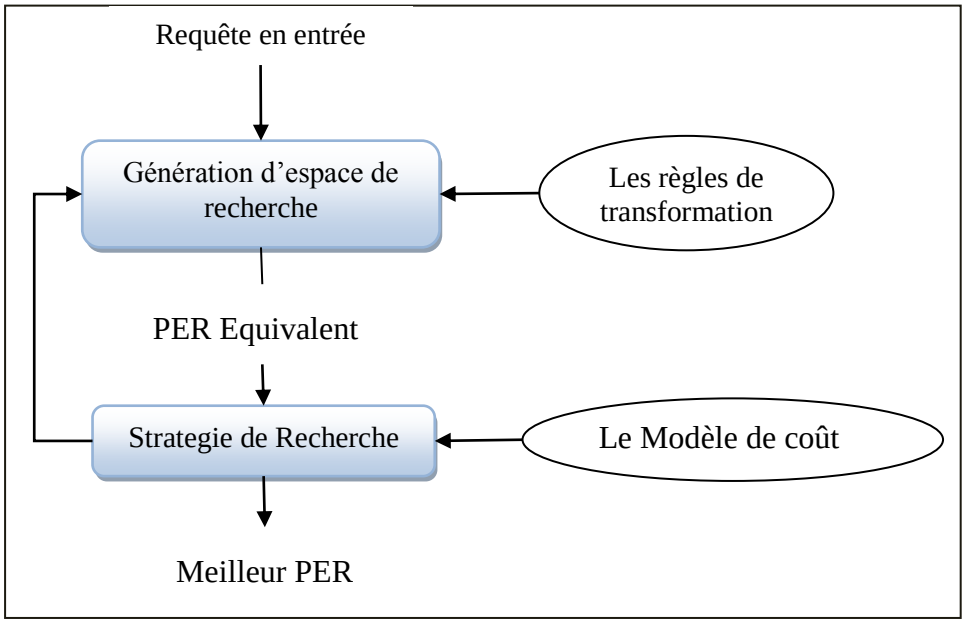
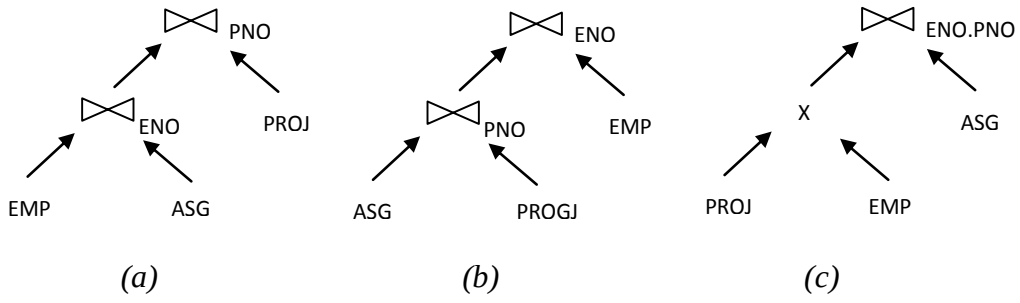


Fig 5.1.Le processus d'optimisation de requête

Pour une requête donnée, l'espace de recherche peut être défini comme l'ensemble des arbres d'opérateurs équivalents pouvant être produits à l'aide de règles de transformation. L'exemple ci-dessous illustre trois arbres de jointure équivalents (figure 5.2), obtenus en exploitant la propriété associative des opérateurs binaires. Rejoindre l'arbre (c) qui commence par un produit cartésien peut avoir un coût beaucoup plus élevé que les autres arbres de jointure.

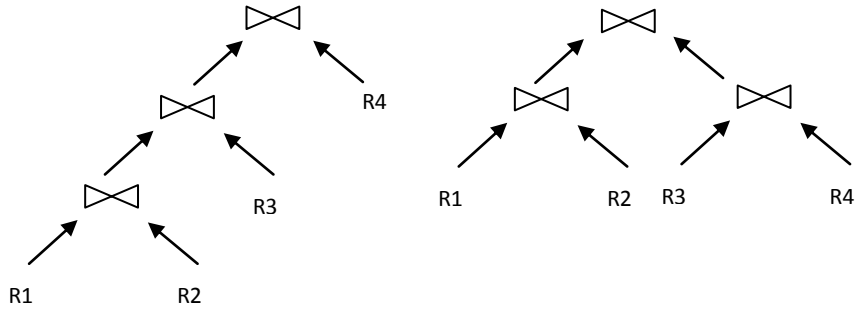
```
SELECT ENAME, RESP
FROM  EMP, ASG, PROJ
WHERE EMP.ENO=ASG.ENO
AND   ASG.PNO=PROJ.PNO
```



(a) *(b)* *(c)*

Fig 5.2. Arbre d'équivalence

En ce qui concerne les différents espaces de recherche, il y aurait une forme différente de l'arbre de jointure. Dans un arbre linéaire, au moins un opérande de chaque nœud est une relation de base. Cependant, un arbre ramifié peut avoir des opérateurs dont les deux opérandes sont des opérateurs intermédiaires. Dans un environnement distribué, les arbres ramifiés sont utiles pour montrer le parallélisme [Özsu et al, 1999].



(a) Arbre de jointure linéaire (b) Arbre de jointure ramifié

Fig 5.3. Arbre de jointure linéaire VS Arbre de jointure ramifié

5.2.3. Estimation de coût

Le processus d'optimisation est destiné à choisir l'ordre de jointure optimal pour les relations dans un graphique de requête, puis à déterminer l'algorithme de jointure optimal pour chaque opérateur de jointure. L'espace de solution d'une requête est l'ensemble de tous les plans d'exécutions de requête (PER). L'objectif de l'optimisation des requêtes de jointure volumineuse est de trouver le PER optimal dans l'espace de solution. Une optimisation souhaitable résoudra trois sous-problèmes:

(1) choisir un espace de recherche qui est le sous-ensemble de l'espace de solution et comprend le PER au coût le plus bas;

(2) choisir un modèle de coût qui peut estimer avec précision le coût des PER.

Un modèle de coût est un ensemble de formules utilisées pour estimer le coût d'un PER avant son exécution. Généralement, le coût inclut le coût du processeur et le coût d'accès au disque. Dans un système de base de données volumineux, le coût d'accès au disque est beaucoup plus élevé que le coût du

processeur. Nous ne prenons donc en compte que le coût d'accès au disque [Lahiri, 1999].

L'Algorithme d'Evolution Différentielle que nous avons conçu peut facilement être modifié en d'autres modèles de coûts.

5. 3. Les algorithmes d'optimisation d'une requête de jointure large

Compte tenu des nouvelles applications de base de données à grande échelle tels que les systèmes de base de données déductives et la bioinformatique ; Il est nécessaire de pouvoir traiter des requêtes de plus grandes tailles. La complexité de la recherche augmente constamment et nécessite davantage de meilleurs algorithmes que nos requêtes de bases de données relationnelles traditionnelles.

Différentes approches ont été proposées pour remédier à cette situation [Melanie, 2004].

Algorithmes heuristiques: l'espace de recherche est réduit en utilisant des estimations. Ils sont généralement très rapides, mais trouvent rarement la solution optimale

Algorithmes aléatoires: Une marche aléatoire à travers l'espace de recherche est effectuée afin de trouver une solution quasi optimale. Différentes stratégies conduisent à différents algorithmes, à savoir amélioration itérative, recuit simulé, algorithmes hybrides, etc.

Algorithmes génétiques: Les algorithmes génétiques tentent de trouver un optimum parmi une population. Cette population souffre de transformations constantes, réalisées en utilisant trois types d'opérations majeurs: la sélection, le croisement et la mutation.

La Programmation génétique : Dans la programmation génétique chaque plan d'exécution de requête peut être considéré comme une solution possible (ou un programme) pour trouver un bon chemin d'accès et de récupérer les données requises par la requête. [Ioannidis et al, 1990] [Swami et al, 1988].

5.3.1. Exemple de requête sql ayant 5 relations.

Considérant les tableaux relationnels suivants: *Personne, Commande, Produit, Departement,* et *Quantité* avec un ensemble de données approprié *init.Following* est une requête de jointure simple pour récupérer les données à partir de certaines conditions, on exécutant la requête dans SQL Server (figure 5.4) :

```
Select
p.Nom, p.Prenom, c.CommandeNo, c.p_id, pd.nomProd, p.p_id, v.vente_id,
dept.nomDepartement, qte.quantité
    from Personne p
    inner join Commande Produit on p.p_id = c.p_id
    inner join Produit pd on o.prd_id = pd.prd_id
    inner join Vente s on v.prd_id = pd.prd_id,
    inner join Departement dpt on v.vente_id = dpt.vente_id
    inner join Quentite qte on dpt.dept_id = qty.dept_id
    and qty.prd_id = qte.prd_id
    order by p.Prenom
    group by dpt.dept_id;
```

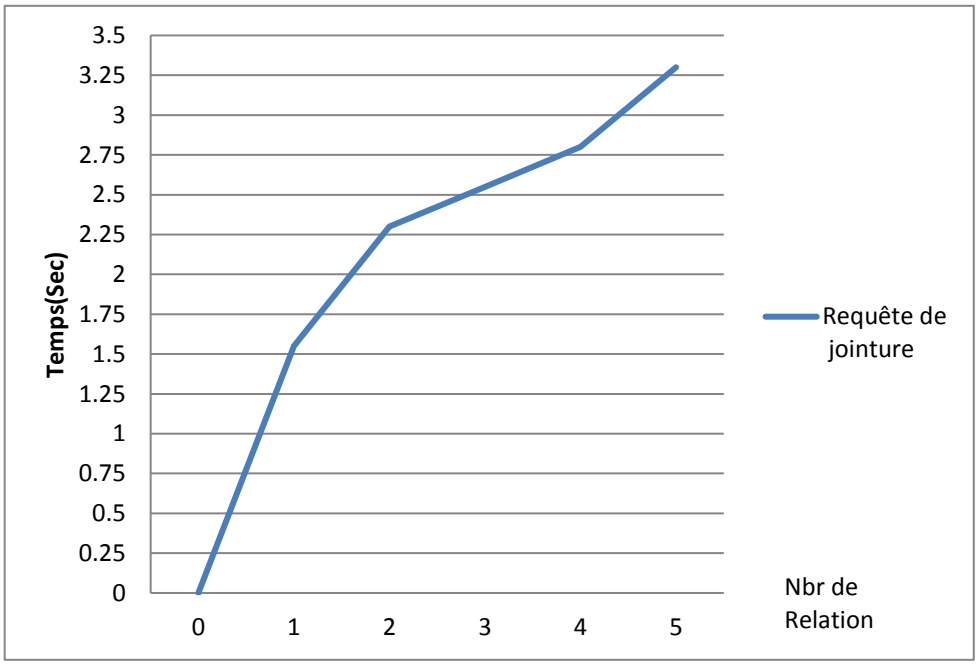


Fig 5.4. Le temps requis pour exécuter une simple requête de jointure.

5.3.2. Application de l'algorithme d'Evolution Différentielle:

Considérant une requête avec N relations où (N = 5). Le principe de fonctionnement de l'algorithme **d'Evolution Différentielle** est :

1. Création de la population initiale.

   a) Etant donné que le temps d'exécution de la requête impliquant une jointure dépend de l'ordre des tables utilisées, on crée une population initiale avec un ordre de combinaisons possibles jusqu'à un nombre d'individus fixé.

   b) Chaque individu est structuré comme suit :

Table $_1$	Table $_2$	……	Table $_k$

   c) On calcule la fonction «objectif» de chaque individu et on trie les individus par ordre décroissant.

   d) La fonction objectif: f (x) = x2.

2. Fixer le nombre de générations et le nombre de descendants qui constituent la base de la population.

3. Sur un vecteur V, calculer l'aptitude de chaque individu et sélectionner le mieux adapté.

4. Appliquer la mutation à l'individu le plus mauvais.

5. Appliquer le croisement.

Le temps nécessaire pour exécuter la requête SQL avec l'algorithme d'Evolution Différentielle est représenté dans la figure 5.5.

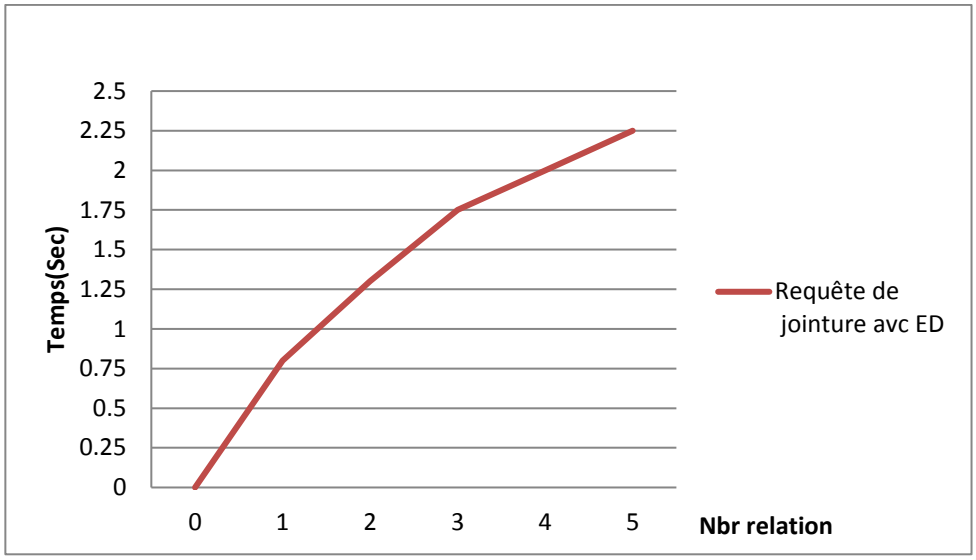
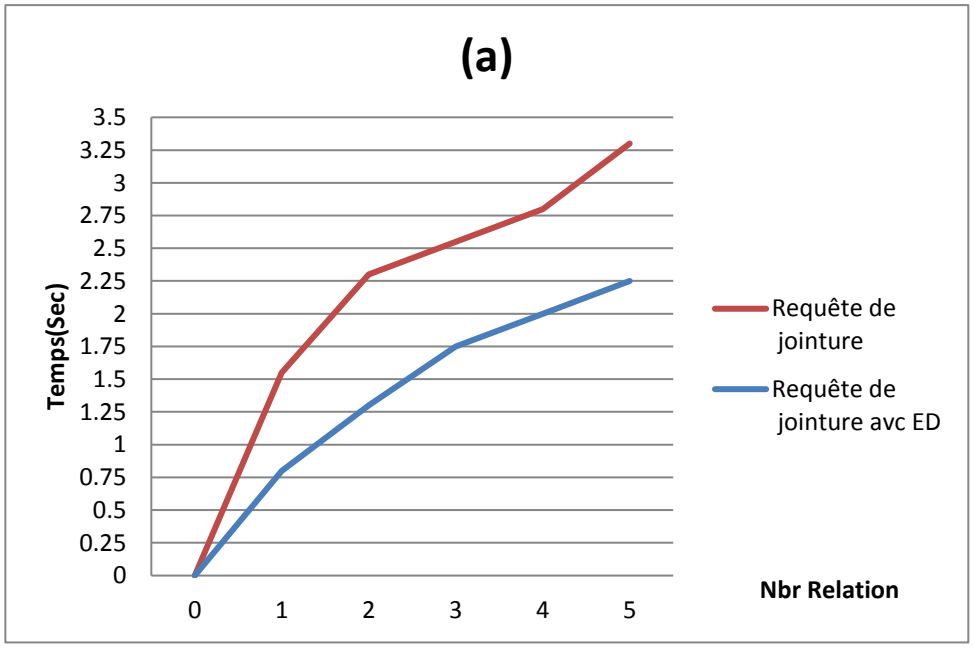


Fig 5.5. Le temps requis pour exécuter une simple requête de jointure avec Evolution Différentielle

Comparaison :

La figure 5.6 représente une comparaison du temps requis pour l'exécution d'une simple requête avant et après application d'EDM.



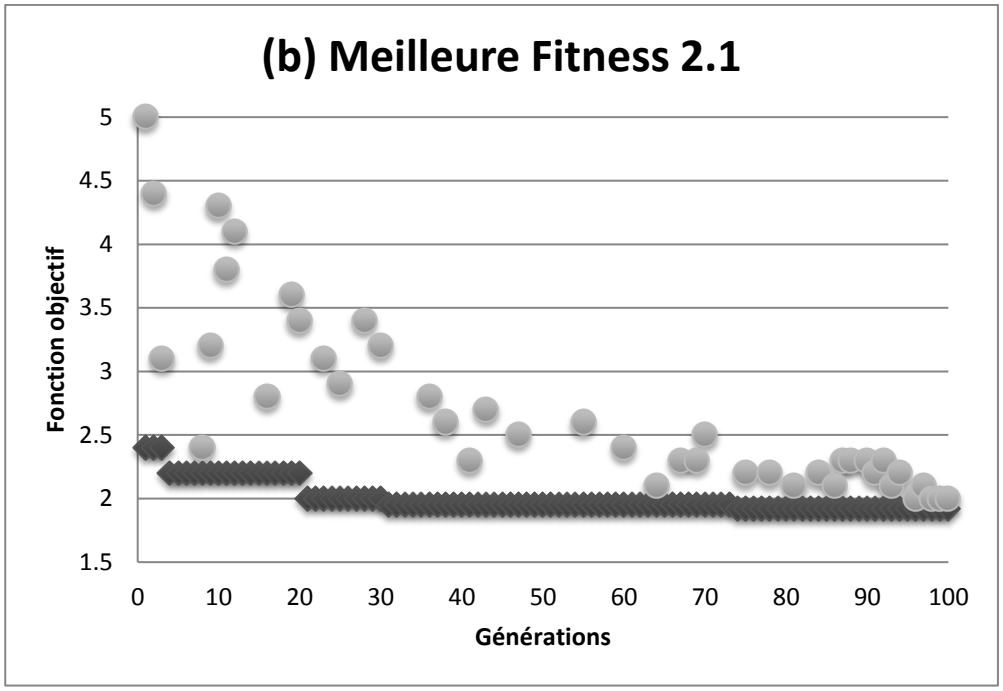


Fig 5.6. Comparaison de temps requis pour exécuter une simple requête avant et après application d'algorithme d'Evolution Différentielle

1. Dans la figure 5.6 (b) les points noirs en bas montrent les meilleures valeurs de la fonction objectif, et les points au-dessus en gris mesurent sa valeur moyenne dans chaque génération.

2. Dans la figure 5.6 (b) les valeurs de la fonction objectif présentent une bonne évolution d'individus pour chaque génération.

3. La figure 5.6 (b) montre aussi une meilleure valeur de la fonction objectif égale à 2.1 à la 100 ème génération.

4. la figure montre aussi que le progrès est peu dans les valeurs de la fonction «objectif» et que la distance moyenne entre les individus pour chaque génération donne une bonne mesure de la diversité d'une population.

5. Dans la plage initial de [1; 100] il ya une diversité de l'algorithme pour avoir un bon progrès.

6. Les valeurs de fitness sont calculées par la fonction "objectif" F (x) = x^2.

Où x est l'aptitude de l'individu choisi pour chaque génération.

7. Les individus ayant les meilleures valeurs de fonction objectif sont sélectionnés et cela permet à l'algorithme d'Evolution Différentielle de rechercher d'autres domaines dans l'espace de solution en se mutant les individus faibles.

8. Les individus avec meilleures valeurs de la fonction objectif dans la génération actuelle garantie leurs reproductions à la prochaine génération. L'algorithme d'Evolution Différentielle utilise les individus dans la génération actuelle pour créer les enfants qui composent la prochaine génération.

9. Augmenter la taille de la population permet à l'algorithme d'Evolution Différentielle de rechercher plus de points et ainsi obtenir un meilleur résultat.

10. Les valeurs de fonction objectif s'améliorent rapidement dans les premières générations, où les individus sont plus loin de l'optimum. Et ces valeurs sont améliorées de plus en plus lentement dans les dernières générations, dont les populations sont plus proches du point optimal.

1. Description opérationnelle d'ED

L'implémentation d'un nouvel optimiseur différentiel basé sur l'Evolution Différentielle permet de présenter de nouvelles idées pour améliorer l'optimisation des requêtes de jointure de grande taille.

L'algorithme 5.1 présente une description simple de la procédure de base exécutée par ED.

Algorithme 5.1 pseudo code de base de l'approche d'Evolution Différentielle "ED"

La procédure ED: La fonction principale
1: Création de la population initiale;
2: Choisir trois individus de façon aléatoire (**a, b, c**);
3: On crée le vecteur mutant **V**
V= a+ f(b-c).
4: Prendre un individu **X** au hasard;
5: **Si** Valeur aléatoire θ générée dans [0,1] vérifie: $\theta < \alpha$; $\alpha \in [0,1]$:
Faire un croisement entre X et V et mettre le résulta dans **R**;
Sinon: mettre V dans R

6 : Si **R** est supérieur à X, R le remplace ;
7: Retour à 2 si le nombre d''itérations est inférieur à *Iter ;*
8: fin procédure:
le meilleur individu de la population donne la **meilleure solution.**

5.4. Conclusion

Cette approche d'Evolution Différentielle ED convient aux bases de données qui ont un grand volume d'informations qui effectuent l'exécution des requêtes complexes. L'utilisation de ces opérateurs différentiels tel que: la méthode de sélection, la fonction «objectif» pour l'évaluation des individus et le processus de mutation diminue le temps et le coût CPU en fonction du nombre de relations. La fonction de sélection attribue une probabilité de sélection plus élevée aux individus ayant des valeurs mises à l'échelle. La plage des valeurs mises à l'échelle affecte la performance de l'algorithme d'Evolution différentielle. Cette méthode peut être utilisée pour optimiser le temps et le coût requis pour exécution. L'optimiseur ED basé sur d'Evolution Différentielle est capable de gérer le problème de requête de jointure large.

CONCLUSION GENERALE ET PERSPECTIVES

Dans cette thèse nous avons étudié les principaux algorithmes évolutionnaires proposés dans la littérature et nous avons développé des approches pour leur utilisation dans la conception de logiciels, en particulier, dans le domaine du clustering des données et des traitements.

En examinant quelques heuristiques inspirées de la nature et en les comparant sur un même codage, on doit noter, en point commun, la grande difficulté à régler les paramètres de ces heuristiques. En dehors des paramètres numériques, il y a le choix des règles de mise à jour de la population lors du passage des générations. Il est plus facile lorsque les opérateurs sont explicites, car ils peuvent être mis dans une famille d'opérateurs non figée (mutations diverses, croisements divers ...) et se pose alors le problème de leur mode d'application (en suivant leur ordre, au hasard, en tentant de noter leur score...). Le renouvellement des générations est aussi une question (élitisme ou non, renouvellement régulier...). En fait, l'observation pas à pas des fonctions diverses, mais aussi des problèmes classiques (route royale, reines de Gauss, reconnaissance d'une figure...) montre très souvent que l'homogénéité de la population doit être à tout prix évitée, et que peu d'individus peuvent suffire alors que plusieurs opérateurs, vus comme des moyens d'exploration de l'espace de recherche, sont une bonne chose. Il s'avère que beaucoup de ces méthodes aboutissent trop tôt à une convergence, c'est-à-dire un nuage de points non nécessairement autour de l'optimum global.

Une des principales contributions de cette thèse porte sur l'introduction d'une amélioration de l'algorithme de chauve-souris, modifiant certaines règles d'évolution et introduisant la règle génétique de croisement. Ces modifications ont induit moins de temps d'exécution et les caractéristiques d'exploration et de convergence de l'algorithme proposé sont beaucoup mieux que dans l'algorithme standard.

Nous avons mis en œuvre, testé et évalué notre approche en l'appliquant sur un certain nombre de problèmes de référence, portant sur l'optimisation numérique. Les résultats obtenus (la qualité finale de la solution et les caractéristiques de convergence) démontrent clairement que l'algorithme proposé est supérieur, non seulement à l'algorithme de chauve-souris standard, mais aussi à d'autres algorithmes modifiant l'algorithme standard. Cette supériorité est démontrée et prouvée à travers l'utilisation de benchmarks (bien connu dans la littérature spécialisée). Comme travail futur, nous comptons appliquer nos algorithmes sur des problèmes de data mining et sur des applications temps réels où le respect des contraintes de temps est primordial.

Une autre contribution sert à étudier les techniques et le problème de partitionnement de données (clustering) , telles que Kmeans. Particulièrement, l'initialisation de nombre de partitions (cluster) et ses centres. Nous avons introduit la technique évolutionnaire qui est l'algorithme génétique avec une structure modifiée, qui repose sur :

- l'hétérogénéité de la population.
- les opérateurs de croisement et de mutation modifiés.

Cette technique permet la détermination automatique des valeurs de centres initiaux, fournissant de meilleurs résultats que l'utilisation de valeurs aléatoires. on réduisant le temps d'exécution et obtenant des meilleures partitions de données.

Comme perspectives nous comptons

1. Améliorer la fonction fitness.
2. explorer d'autres opérateurs de reproduction pour meilleures performances

Une troisième contribution importante est notre application de l'algorithme de l'évolution différentielle ED à l'optimisation des requêtes de base de données. Les résultats obtenus sont largement encourageants. Sur le plan des perspectives nous comptons de continuer ce travail dus ou moins trois directions :

1. Utilisation de l'algorithme ED pour l'optimisation Web.
2. L'approfondissement des chercheurs d'information par combinaison et apprentissage en profondeur.
3. L'utilisation de l'algorithme ED dans les applications Cloud.

RÉSUMÉ

La conception est la phase créative d'un projet d'ingénierie. Le but premier de la conception est de permettre de créer un système ou un processus répondant à un besoin en tenant compte des contraintes. Le système doit être suffisamment défini pour pouvoir être installé, fabriqué, construit et être fonctionnel, et pour répondre aux besoins du client. Des paramètres essentiels doivent être optimisés. Ils portent généralement sur les coûts et les délais mais aussi sur la qualité et la sécurité. Nous avons montré dans la présente thèse que les algorithmes évolutionnaires sont d'un apport certain à ces activités de conception, permettant de réaliser des gains divers dans le cas où ils sont utilisés à bon escient. Cela s'est traduit à travers trois principales contributions; à savoir, un algorithme génétique modifié pour le clustering des données, un algorithme chauve-souris modifié pour optimisation de meilleure qualité, et enfin, une optimisation de requêtes de base de données utilisant une modification de l'algorithme à évolution différentielle.

MOTS CLES: Algorithme Evolutionnaire, Conception du Logiciel, Clustering, optimisation de requêtes.

ABSTRACT

Design is the creative phase of an engineering project. The first purpose of the design is to create a system or process that meets a need, taking into account constraints. The system must be sufficiently defined to be able to be installed, manufactured, built and functional, and to meet the needs of the customer. Essential parameters need to be optimized. They generally relate to costs and deadlines but also to quality and safety. We have shown in the present thesis that evolutionary algorithms are of a certain contribution to these design activities, making it possible to achieve various gains in the case where they are used wisely. This resulted in three main contributions; namely, a modified genetic algorithm for data clustering, a modified bat algorithm for better quality optimization, and finally, database query optimization using a modification of the differential evolution algorithm.

KEY WORDS: Evolutionary Algorithm, Software Design, Clustering, Query Optimization.

ملخص

التصميم هو المرحلة التصورية والإبداعية لأي مشروع هندسي حيث يتمثل الهدف الأول من التصميم في إنشاء نظام أو عملية تلبي الحاجة، مع مراعاة القيود . يجب أن يكون النظام محددًا بشكل كافٍ ليتم تثبيته وتصنيعه وبنائه وتشغيله، ولتلبية احتياجات العميل .و وضع إعدادات مثلى تتعلق عموما بالتكاليف, الوقت, الجودة والسلامة .لقد أظهرنا في هذه الرسالة أن استعمال الخوارزميات التطورية يساهم بشكل كبير في أنشطة التصميم، مما يجعل من الممكن تحقيق مكاسب مختلفة في حالة استخدامها بحكمة .وقد نتج عن ذلك ثلاث مساهمات رئيسية ؛ وهي أولا خوارزمية جينية معدلة بهدف تجميع البيانات ، ثانيا خوارزمية الخفاش معدّلة لتحسين جودة النتيجة، وأخيرًا تحسين استعلام قاعدة البيانات باستخدام خوارزمية التطور التفاضلي.

الكلمات المرشدة: خوارزمية تطورية ، تصميم برمجي ، تجميع البيانات ، تحسين الاستعلام.

REFERENCES

[Ali, 2015] A. F. Ali, « Accelerated Bat Algorithm for Solving Integer Programming Problems », Egypt. Comput. Sci. J., vol. 39, no. 1, p. 25, (2015).

[Allaire, 2006] Grégoire Allaire « Optimisation optimal de structure »ISBN-10 3-540-36710-1 Springer Berlin Heidelberg New York, paris (2006).

[Anderberg, 1973] M. R. Anderberg, « Cluster Analysis for Applications ». Academic Press, Inc, New York, NY, (1973).

[Backer, 1995] E. Backer, « Computer-Assisted Reasoning in Cluster Analysis ». Prentice Hall International (UK) Ltd., Hertfordshire, UK, (1995).

[Bernard et al, 2006] T.T. Bernard, F.L. Paulin, « Heuristiques du problème du voyageur de commerce », CARI06 – Volume 1 – (2006).

[Bernard, 2005] W. Bernard, « Méthode combinatoire : Initiation progressive à la lecture », avril (2005).

[Beyer et al, 2002] H.G. Beyer et H.P. Schwefel, « Evolution strategies : a comprehensive introduction », Nat. Comput, (2002).

[Bezdek, 1981] J. C. Bezdek , « Pattern Recognition with Fuzzy Objective Function Algoritms », Plenum Press, New York, (1981).

[Bock, 1981] H. Bock, « Statistical Testing and Evaluation Methods in cluster analysis », on proceeding of the indian Statistical Institute golden jubilee international conference on statistics, pages116-146, Calcutta India, (1981).

[Bradley et al, 1997] P. S. Bradley, O. L. Mangasarian, et W. N. Street, « Clustering via concave minimization », In Advances in Neural Information Processing Systems -9, pp. 368–374. MIT Press, (1997).

[Bradley et al, 1997] P. S., Bradley, O. L. Mangasarian, et W. N. Street, « Clustering via concave minimization», In Advances in Neural Information Processing Systems -9, pp. 368–374. MIT Press. (1997).

[Cesare, 2015] N. Di. Cesare, « Validation de la méthode d'optimisation par essaim particulaire basée sur un processus de Markov à temps discret (PageRank PSO) : Application à la mécanique », CSMA Colloque National en Calcul des Structures, 18-22, Presqu'île de Giens (Var), (2015).

[Chen et al, 2002] Y. Chen, W. Chen, L. Nguyen et R. Katz, « Clustering Web Content for efficien Replication », in proceedings of the 10th IEEE international Conference on Network Protocols (ICNP02), pages 165-174, Paris- France, (2002).

[Darwin, 1859] C. Darwin, « On the Origin of Species by Means of Natural Selection, or the Preservation Of Favoured Races in the Struggle for Life ». John Murray (1859).

[Dawkins, 1896] R. Dawkins, « The blind watchmaker: Why the evidence of evolution reveals a universe without design ». (1986)

[Delfour, 2012] M. C. Delfour, « Introduction à l'optimisation et au calcul semi-différentiel », Dunod, 354 pages, (2012).

[Dong et al, 2007] H. Dong, Y. Liang, « Genetic Algorithms for Large Join Query Optimization », L ondon, E ngland, United Kingdom GECCO'07,ACM 978-1-59593-697,(2007).

[Dunn, 1973] J. C. Dunn, « A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters », Journal of Cybernetics 3: 32-57, (1973).

[Duran et al, 1974] B.S. Duran, et P. L. Odell, « Cluster Analysis: A Survey », Springer-Verlag, New York, NY, (1974).

[Eschrich et al, 2003] S. Eschrich, J. Ke, L.O. Hall et D.B. Goldgof, « Fast Accurate Fyzzy Clustering Through Data Reduction », IEEE Transaction on Fuzzy Systems, 11(2) : 262-270, (2003).

[Ester et al, 1996] M. Ester, H.P. Kreigel, J.sander et X. Xu, « A Density-Based Algorithm for Discovering clusters in Large Spatial Databases with noise », in proceeding of the second international Conference on Knowledge Discovery and Data Mining, pages 226-231, Portland, Oregon, (1996).

[Everitt, 1993] B. S. Everitt, « Cluster Analysis ». Edward Arnold, Ltd., London, UK. (1993).

[Fink et al, 2004] E. Fink, P.K. Kokku, S. Nikiforou, L.O. Hall, D.B. Goldgof et J.P. Krischer, « Selection of Patients for Clinical Trials : An Interactive Web- Based System », Artifical Intelligence in Medicine, 31(3) :241-254, (2004).

[Fister et al, 2013] I. Fister, D. Fister, et X. S. Yang, « A hybrid bat algorithm », Elektroteh. Vestnik/Electrotechnical Rev., vol. 80, no. 1–2, pp. 1–7, (2013).

[Fogel , 1998] D.B. Fogel, « Evolutionary Computation: The Fossil Record », Wiley-IEEE Press (1998)

[Fogel et al, 1966] D.B. Fogel, A.J. Owens, M.J. Walsh, « Artificial Intelligence through Simulated Evolution », John Wiley (1966)

[Forgy,1965] E. Forgy, «Cluster analysis of multivariate data: efficiency vs », interpretability of classification, Biometrics, 21, 768. (1965)

[Forrest, 1993] S. Forrest, « Genetic algorithms principles of natural selection applied to computation », Science, Vol.261, pp.872 – 678. (1993).

[Gaffney, 2010] J. Gaffney, D.A. Green, « C.E.M Binary versus real coding for genetic algorithm », (2010).

[Gandomi et al, 2014] A. H. Gandomi et X. S. Yang, « Chaotic bat algorithm » J. Comput. Sci., vol. 5, no. 2, pp. 224–232, (2014).

[Ghaemi et al,2008] R. Ghaemi, A. M. Fard, H.Tabatabaee, et M. Sadeghizadeh « Evolutionary Query Optimization for Heterogeneous Distributed Database Systems », World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol: 2, No:7, (2008).

[Gonçalves et al,2014] A.C.A. Gonçalves, G. Guimarães, J.F. Souza, « Query join ordering optimization with evolutionary multi-agent systems »,Elsevier Ltd, Expert Systems with Applications 41 6934–6944(2014).

[Hachimi, 2013] H. Hachimi, « Hybridations d'algorithmes metaheuristiques en optimisation globale et leurs applications », (2013).

[Hartigan, 1975] J. A. Hartigan, « Clustering Algorithms », John Wiley and Sons, Inc, New York, NY. (1975).

[Haupt et al, 1998] Haupt, R. and Haupt, S. E., Practical genetic algorithms, John Wiley & Sons. (1998)

[Hegazy et el, 2017] Z. Hegazy, An Improved Approach for Bat Algorithm, International Journal of Advanced Research in Computer Science and Software Engineering 7(2), , pp. 134-140, (2017)

[Hinneburg et al, 1998] A. Hinneburg, D.A. keim, « an Efficient approche to Clustering in large multi-media DataBases with noise », on proceeding international conference on Knowledge Discovery in databases (KDD'98), page 58-65, New Yourk USA, (1998).

[Hinneburg et al, 1999] A. Hinneburg et D.A. keim, « Clustering Techniques For large Sets: from the past to the future », on proceeding international conference on Knowledge Discovery in databases (KDD'99), San Diego, CA, USA, (1999).

[Holland, 1975] J.H. Holland, « Adaptation in natural and artificial systems: An introductory analysis with applicationsto biology, control, and artificial intelligence », University of Michigan Press, Ann Arbor, MI, 19 (1975).

[Holland, 1992] J.H. Holland, « Genetic algorithms, Scientific American », July 1992, pp.66-72, (1992)

[Huang, 1998] Z. Huang, « Extensions to the k-means algorithm for clustering large data sets with categorical values Data Min». Knowl. Discov. 2, 283–304, (1998).

[Induja et al, 2016] S. Induja, V.P. Eswaramurthy, « Bat Algorithm: An Overview and its Applications », International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 1, January (2016).

[Ioannidis et al, 1990] Y. E. Ioannidis et Y. Kang , « Randomized algorithms for optimizing large join queries », In Proc. of the 1990 ACM-SIGMOD Conference on the Management of Data, pages 312-321, Atlantic City, NJ, May (1990).

[Ioannidis et al, 1991] Y. E. Ioannidis et Y. Kang , « Left-deep vs. bushy trees: an analysis of strategy spaces and its implications for query optimization », ACM SIGMOD Record , Volume 20 (2) – Apr 1, (1991).

[Jain et al, 1988] A.K. Jain et R.C. Dubes, « Algorithms for Clustering Data» . Prentice-Hall advanced reference series. Prentice-Hall, Inc., Upper Saddle River, NJ. (1988).

[Jain et al, 1991] A.K.Jain et F. Farrokhnia, « Unsupervised texture segmentation using Gabor filters », Pattern Recogn. 24, 12, 1167±1186, (1991).

[Jain et al, 1999] A.K. Jain, M.N. Murty et P.J. Flynn, « Data Clustering: A Review », ACM Computing Surveys, Vol. 31, No. 3, September (1999).

[Jarke et al, 1984] M. Jarke et J. Koch, « Query optimization in database systems », ACM Computing Surveys, 16(2):111,152, (1984).

[Jourdan, 2003] L. Jourdan, « Méthaheuristique pour L'extraction De Connaissances : Application à la Génomique », Lille,1, (2003).

[Kahn, 1974] G. Kahn, « The semantics of a simple language for parallel programming », Proceedings of IFIP Congress, vol. 6, p. 471-475,(1974).

[Karaboga et al, 2007] D. Karaboga, B. Basturk, « Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems », Lecture Notes in Artificial Intelligence, vol. 4529, pp. 789-798, (2007).

[Kaufman et al, 1990] L. Kaufmanet, P. J. Rousseeuw «Finding Groups in Data : An Introduction to Cluster Analysis». John Wiley. (1990)

[Lahiri, 1999] T. Lahiri, « Genetic Optimization Techniques for LargeJoin Queries ». In Proceedings of the Third Genetic Programming Conference (Univ. of Wisconsin,Madison, 535-540. Morgan Kaufmann, (1998).

[Liu, 1968] G. L. Liu, « Introduction to Combinatorial Mathematics » McGraw Hill, New York. , (1968).

[MacNish, 2006] C. MacNish, « Benchmarking Evolutionary and Hybrid Algorithms using Randomized Self-Similar Landscapes» Proc. 6th International Conference on Simulated Evolution and Learning (SEAL'06), LNCS 4247, pp. 361-368, Springer,(2006).

[MacQueen, 1967] J. MacQueen, « Some methods for classification and analysis of multivariate observations », In 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1, pp.281–297. (1967).

[Melanie, 2004] M. Melanie, « An introduction to Genetic Algorithms» , Prentice Hall of India, (2004).

[Meyer, 1979] B. Meyer, « Quelques concepts importants des langages de programmation modernes, et leur expression en SIMULA 67 », Conférence sur Panorama des Langages d'aujourd'hui, 8-9, GROPLAN, (1979).

[Michalewicz, 1996] Z. Michalewicz, Genetic Algorithm et data structures= evolution programs. 3rd edition, New York : springer-Verlag.(1996).

[Michalski et al, 1983]. R. Michalski, R.E. Stepp et E.Diday, « A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts », In Progress in Pattern Recognition, Vol. 1, L, (1983).

[Miller, 2011] Julian F. Miller, **«** cartesian genetic programming **»**,Springer-Verlag Berlin Heidelberg (2011).

[Mitchell, 1996] M. Mitchell, « An Introduction to Genetic Algorithms » , Institute of Technology , London, England, First MIT Press paperback edition, Massachusetts, (1996).

[Molina et al, 2008] H.G. Molina, J. D. Ullman, et J. Widom, " Database systems: The complete book (2nd ed.)", Upper Saddle River, NJ, USA: Prentice Hall Press. (2008).

[Mridul et al, 2015] MridulChawla, ManojDuhan, Bat Algorithm: A Survey of the State-of-the-Art, International Journal of Applied Artificial Intelligence, Taylor & Francis, Volume 29, - Issue 6, (2015)

[Murty et al, 1995] M.N. Murty et A. K. Jain, « Knowledge based clustering scheme for collection management and retrieval of library books », Pattern Recogn, 28, 949±964. (1995).

[Natalia et al, 2001] V. Natalia, J.H. Brian et L.S. Steven, « A Clustering Method for Repeat Analysis in DNA Sequences », Genome Biology 2(8), (2001).

[Oehler et 1995] K. L. Oehler, et R. M. GRAY, « Combining image compression and classification using vector quantization ». IEEE Trans.Pattern Anal. Mach. Intell. 17, 461±473, (1995).

[OMG, 1997] Object Management Group, « UML semantics, version 1.1. », ftp://ftp.omg.org/pub/docs/ad/97-08-04.pdf,

September (1997).

[Ozawa, 1885] K. Ozawa, « A Stratificational Overlapping Cluster Scheme », pattern recognition, 81(3-4):279-268, (1985).

[Özsu et al, 1999] M. Tamer Özsu, P.Valduriez, « Principles of Distributed Database Systems, Second Edition », Prentice Hall, ISBN 0-13-659707-6, (1999).

[Paril, 2008] A. L. Parrill, « Introduction to Evolutionary Algorithms », Evolutionary Algorithms in Molecular Design, vol. 8, pp. 1–13, (2008).

[Qin, 2009] A. K. Qin, V. L. Huang, et P. N, « Suganthan, Differential Evolution Algorithm With Strategy Adaptation For Global Numerical Optimization », Ieee Transactions On Evolutionary Computation, Vol. 13, No. 2, April (2009).

[Raghavan et al, 1979] V. Raghavan et K. Birchand, « A clustering strategy based on a formalism of the reproductive process in a natural syst em.In Proceedings of the Second International Conference on Information Storage and Retrieval,10±22. (1979).

[Ramesh, 2013] B. Ramesh, V. C. J.Mohan, V. C. V. Reddy, « Application of bat algorithm for combined economic load and emission dispatch », Int. J. of Electrical Engineering and Telecommunications, Vol. 2, No. 1, pp. 1–9. (2013).

[Ray et al, 1999] S. Ray et R.H. Turi, « Determination of Clusters in k-means Clustering and Application in Color Image Segmentation », In proceeding of the 4th international conference on advances in pattern Recognition and Techniques, pages 137-143, Calcutta, India, (1999).

[Rechenberg, 1971] I. Rechenberg, « Evolutions strategie Optimie run gtechnischer System enach Prinzipien derbiologischen Evolution ». Ph.D. thesis, Technical University of Berlin, Germany (1971)

[Rekaby 2013] A. Rekaby, « Directed Artificial Bat Algorithm (DABA) », A new Bio-Inspired Algorithm, Egyptian Research and Scientific Innovation Lab (ERSIL), Cairo, Egypt, (2013).

[Ripley, 1988] B. D. Ripley, « Statistical Inference for Spatial Processes ». Cambridge University Press, New York, NY, Ed (1989).

[Sam et al, 2000] H. Sam et M. James, « An introduction to Genetic Algorithms », (2000).

[Sayadi et al, 2013] Sayadi, « Firefly-inspired algorithm for discrete optimization problems : an application to manufacturing cell formation ». Journal of Manufacturing Systems, 32(1), 78-84. (2013).

[Schwefel, 1974] H.P. Schwefel, « NumerischeOptimierung von Computer-Modellen ». Ph.D. thesis, TechnicalUniversity of Berlin (1974).

[Sheikholeslami et al,1998] G. Sheikholeslami, S. Chatterjee et A. Zhang, « WaveCluster : A multi-Resolution Clustering Approch for Very large Spatial Databases », in proceeding of the 24 international Conference on very large data bases (VLDB), pages 428-439, New York USA, (1998).

[Siarry, 2014] P. Siarry, « Métaheuristiques: Recuits simulé, recherche avec tabous, recherche à voisinages variables, méthodes GRASP, algorithmes évolutionnaires, fourmis artificielles, essaims particulaires et autres méthodes d'optimisation », Eyrolles, (2014).

[Sneath et al, 1973] P.H.A Sneath, et R. R. Sokal, « Numerical Taxonomy », Freeman, London,UK, (1973).

[Spath,1980] H. Spath, « Cluster Analysis Algorithms for Data Reduction and Classification », Ellis Horwood, Upper Saddle River, NJ, (1980).

[Suga,1990] N. Suga, « Biosonar and neural computation in bats ». Scientific American, 262(6):60–68. (1990).

[Swami et al, 1988] A. Swami et A. Gupta, « Optimization of large join queries », In Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data, pages 8-17, Chicago, IL, (1988).

[Tan et al, 2005] P.N. Tan , V. Kumar, M.Steinbach , « Cluster Analysis: Basic Concepts and Algorithms »,(2005).

[Toussaint, 1980] G.T. Toussaint, « the relative Neighborhood Graph of a finite Planar Set », pattern recognition, 12(4):261-268, (1980);

[Turing 1992] A. M. Turing, « Intelligent Machinery » In: D. Ince (ed.) Collected Works of A. M. Turing: MechanicalIntelligence. Elsevier Science (1992)

[Wang et al, 1997] W. Wang, J. Yang, et R. Muntz, « STING : A Statistical Informarmation Grid Approch to Spatial Data Mining », in proceeding of the 24 international Conference on very large data bases (VLDB), pages 186-195, Athens, Greece, (1997).

[Wasi et al,2014] M. Wasi, Ul. Kabir, N. Sakib, S. Mustafizur, M. Shafiul « A Novel Adaptive Bat Algorithm to Control Explorations and Exploitations for Continuous Optimization Problems », International Journal of Computer Applications (0975 – 8887) Volume 94 – No 13, May (2014).

[Yang et al, 2013] X.S. Yang et X. He, « Bat algorithm: literature review and applications », Int. J. Bio-Inspired Comput., vol. 5, no. 3, pp. 141–149, (2013).

[Yang, 2010] X.S. Yang, « A New Metaheuristic Bat-Inspired Algorithm », Nat. Inspired Coop. Strateg. Optim. (NICSO 2010), pp. 65–74, (2010).

[Yilmaz et al, 2013] S. Yilmaz et E. U. Kucuksille, « Improved Bat Algorithm (IBA) on Continuous Optimization Problems », Lect. Notes Softw. Eng., vol. 1, no. 3, pp. 279–283, (2013).

[Yilmaz et al, 2014] S. Yilmaz, E. U. Kucuksille, et Y. Cengiz, "Modified bat algorithm," Elektron. irElektrotechnika, vol. 20, no. 2, pp. 71–78, (2014).

[Yu et al , 2008] T. Yu, L. Davis, C. Baydar, et R. Roy, « Evolutionary Computation in Practice » Springer Science et Business Media, (2008).

[Zhang et al, 1996] T. Zhang, R. Ramakrishnan et M. Livny, « BIRCH : an efficient Data Clustering Method for Very Large Data Bases », in proceeding of the ACM SIGMOD international conference on management of Data, pages 103-114, montréal Québec Canada, (1996).

[Zhen et al, 2013] C. Zhen, Z. Yongquan, et L. U. Mindi, "A simplified Adaptive Bat Algorithm Based on Frequency," J. Comput. Inf. Syst., vol. 9: 16 (201, pp. 6451–6458, (2013).

[Zidani, 2013] M. Zidani, « Représentation de solution en optimisation continue,multi-objectif et applications Other », INSA de Rouen; Universite Mohammed V-Agdal (Rabat, Maroc),(2013).