République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Djilali BOUNAAMA Khemis Miliana

Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Mémoire Présenté
Pour l'obtention de diplôme de
Master en Informatique
Spécialité : Ingénierie Du Logiciel
Titre :

Méthode Hybride Basée Sur Les Algorithmes A Evolution Différentielle.

Réalisé par :
DRIF Asma.
BENSAIFIA Nesrine.
Soutenu publiquement le : 25/06/ 2018
devant le jury composer de :

M. F. MEGHATRIA	Encadrante.
Mr. M. BOUZIANE	Président .
Mr. M. BOUKEDROUN	Examinateur1 .
Mr. R. SAKRI	Examinateur2.

Année Universitaire : 2017/2018.

Remerciements

Nous remercions Allah le tout puissant de nous avoir donné la volonté et le courage pour mener à bien ce travail.

Nous tenons à exprimer notre sincère gratitude envers tous ceux qui nous ont aidé ou ont participé au bon déroulement de ce projet.

Nous sommes reconnaissant également à f.meghatria notre encadreur durant le déroulement du projet pour son aide à la mise en place de ce modeste travail.

Nos vifs remerciements s'adressent à tous les membres de jury qui nous ont fait l'honneur d'examiner ce travail.

Enfin, nous remercions tous nos amis qui nous ont aidé, encouragé et toute personne ayant contribué à l'élaboration de ce travail, par un conseil, ou même par un sourire.

Dédicace

Je dédie ce modeste travail

Mes parents :

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Mes frères et mes soeurs qui n'ont cessé de m'encouragé et de me gardé dans la bonne voie.

Mes amis sans exception qui m'ont été un support psychique et moral pour la continuation de ce mémoire.

Ma binôme BENSAIFIA Nesrine qui a été une amie et une collègue dans les dernières années à l'Université.

ASMA

Dédicace

Je dédie ce modeste travail

A mes très chers parents qui m'ont quidé durant les moments les plus pénibles de ce long chemin, ma mère qui a été à mes côtés et ma soutenu durant tout ma vie,et mon père qui a scrifié toute sa vie afin de me voir devenir ce que je suis,merci infiniment mes parents.

A mes très chers soeurs Amira et Malak.

A mes amis Imene,Asma, Khouloud, Rachida ,Amina et à tous ceux qui me sont chére sans exception .

A toute ma famille.

A ma meilleur amie l'informaticiene Drif Asma .

A toute la promotion de 2018.

NESRINE

الملخص:

تستخدم خوارزمية التطور التفاضلي على نطاق واسع لحل مشكلة التحسين العامة ,و لتحسين قدرة البحث العامة لخوارزمية التطور التفاضلي ، تم اقتراح خوارزمية التفاضلي الهجين .

في هذا العمل ,نقدم طريقة لتحسين خوارزمية التطور التفاضلي من خلال دمجها مع التجميع في مجال متعدد الأهداف,حيث يتمثل الهدف الرئيسي لهذا العمل في تحديد الحل الامثل عن طريق اختيار أفضل الأفراد.

الكلمات الرئيسية: خوارزمية تطور تفاضلي ، طريقة التحسين الهجين ، تجميع.

Résumé :

Algorithme à évolution différentielle est largement utilisée pour résoudre le problème d'optimisation global.
Pour améliorer la capacité de recherche global de l'algorithme à évolution différentielle (DE), on a proposé une algorithme à évolution différentielle hybride (HDE).
Dans ce travail, nous présentons un moyen d'améliorer DE en le combinant avec le clustering dans un domaine multi-objectif, où le but principale est d'identifier la solution optimale en choisissant les meilleurs individus.

Mots clés : algorithme à évolution différentielle(DE),méthode d'optimisation hybride, clustering.

Abstract :

Differential evolution is widely used to solve the global optimization problem.
To improve the global search capability of the differential evolution algorithm (DE), an hybrid differential evolution algorithm (HDE) has been proposed.
In this thesis, we present a way to improve DE by combining it with clustering in a multi-objective domain, where the ultimate goal is to identify the optimal solution by choosing the best individuals.

Keywords : differential evolution algorithm (DE),hybrid optimization method, clustering.

Table des matières

La table des notations

AE	Algorithme évolutionnaire.
DE	Algorithme à évolution différentielle.
AG	Algorithme génétique.
SE	Stratégie d'evolution.
PE	Programmation évolutionnaire.
PG	Programmation génétique.
n	Taille de la population.
D	Dimension de X.
itermax	Nombre maximum d'itérations.
F	Facteur d'echelle.
CR	Taux de croisement.
Lb	Borne inférieure.
Ub	Borne supérieure.
λ	Descendants de dimension
μ	Dimension de l'ensemble des parents

Table des figures

Introduction générale :

L'algorithme à évolution différentille (DE) est une nouvelle approche évolutionnaire qui a été récemment développés par STORN et PRICE en 1995[36]. Cet algorithme est simple mais puissant,surpassant un grand nombre de techniques de recherche stochastique globale existantes.Il est généralement considéré comme un outil d'optimisation précis,rapide et robuste.L'avantage principal du DE est sa simplicité permettant ainsi une implémentation facile pour l'optimisation globale dans des espaces continus.

La famille des métaheuristiques regroupe un ensemble de méthodes performantes en optimisation difficile elles offrent des solutions approchées de bonne qualité pour des applications réelles de grande taille. Des essais d'amélioration sont proposés pour obtenir la méthode la plus efficace possible. L'hybridation des méthodes peut permettre de trouver une amélioration car les avantages et les inconvénients de chaque méthode se compensent.

Les algorithmes de clustering permettent de partitionner les données en sous-groupes, ou clusters, de manière non supervisée. Intuitivement, ces sous-groupes regroupent entre elles des observations similaires. Les algorithmes de clustering dépendent donc fortement de la façon dont on définit cette notion de similarité, qui est souvent spécifique au domaine d'application.

Notre projet de fin d'étude présente l'application du DE avec clustering pour cela nous avons organisé notre travail en trois chapitres :

- Au premier chapitre, nous présenterons les algorithmes évolutionnaires les plus importants et nous présenterons de manière plus détaillée l'algorithme à évolution différentielle.
- Le deuxième chapitre représente Les approches métaheuristiques.
- Le dernier chapitre consiste à présenter la phase de réalisation et de mise en œuvre de l'application.

Chapitre 1

L'algorithme à évolution différentielle

1.1 Algorithmes évolutionnaires

Introduction

Les algorithmes évolutionnaires (Evolutionary Algorithms ou Evolutionary Computation)est un discipline de l'intelligence artificielle, sont des algorithmes d'optimisation stochastique simulant l'évolution naturelle. Ces algorithmes sont inspirés du paradigme de l'évolution darwinienne des espèces, telle qu'elle a été définie par Charles Darwin. Les espèces naturelles sont en compétition pour survivre, et seules les plus aptes survivent à la sélection naturelle. Au cours de leur évolution, ces même individus auront la possibilité de transmettre leurs patrimoine génétique à la génération suivante par reproduction. L'itération de ce principe pendant des générations permet de faire apparaître et de sélectionner dans la population des individus plus adaptés à leur environnement[33].

1.1.1 Les algorithmes génétiques (AG)

Les AG ont été nommés au début « adaptive plan » et ont été popularisés par Goldberg [8]. Les algorithmes génétiques impliquent l'évolution d'une population de vecteurs de longueur fixe composée généralement de bits, même si des versions actuelles travaillent sur les vecteurs de réels ou même sur des

structures plus complexes. Les AG sont utilisés dans le but de découvrir une solution à un problème donné, sans information ou peu d'informations a priori sur l'espace de recherche. Au cours de l'évolution, on utilise des opérateurs inspirés de l'évolution naturelle (la mutation qui modifie un bit du vecteur et le croisement entre deux vecteurs pour en produire un nouveau). Un critère de qualité est nécessaire pour discriminer différentes solutions, cette fonction s'appelle fitness ou fonction objectif ou d'adéquation[33].

1.1.2 Les stratégies d'évolution (SE)

Cette technique a été développée par Ingo Rechenberg et Hans Paul Schwefel [14] [17] [16]. Les populations utilisées par les stratégies d'évolution sont représentées par des vecteurs de nombres réels de dimension fixe, qui représentent les caractéristiques d'une solution potentielle. Les SE sont souvent nommées sous la forme (μ, λ)-ES ou $(\mu + \lambda)$-ES qui désigne deux différentes stratégies, où μ désigne la dimension de l'ensemble des parents pour produire un ensemble de descendants de dimension $\lambda(\geq \mu)$. (μ, λ) -ES indique que μ vecteurs sont choisis pour former la nouvelle génération parmi les meilleurs vecteurs λ. Pour la stratégie $(\mu + \lambda) - ES$, les μ vecteurs de la nouvelle génération sont sélectionnés parmi le meilleurs entre les μ parents et leurs λ descendants[33].

1.1.3 La programmation évolutionnaire (PE)

La programmation évolutionnaire à été introduite par Fogel [22], et a été initialement conçue pour faire évoluer les machines à états finis et a été par la suite étendue aux problèmes d'optimisation de paramètres. À la différence des autres branches de la famille des algorithmes évolutionnaires, la PE n'utilise pas une représentation spécifique. Elle se focalise sur l'opérateur de mutation approprié à un problème spécifique. Pour résoudre un problème, on génère aléatoirement une population de taille μ, et chaque individu i va générer λ descendants suite à l'application de l'opérateur de mutation. Une opération de sélection naturelle permet par la suite de former une nouvelle génération de taille μ à partir des parents et des descendants[28].

1.1.4 Programmation génétique (PG)

La programmation génétique est un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques basées sur les mêmes principes d'évolution que les algorithmes génétiques : des opérations de variation génétique et des opérations de sélection naturelle. La différence entre la programmation génétique et les algorithmes génétiques réside essentiellement dans la représentation des individus. En effet, la programmation génétique consiste à faire évoluer des individus dont la structure est similaire à des programmes informatiques. La programmation génétique représente les individus sous forme d'arbres, c'est-à-dire des graphes non orientés et sans cycle, dans lesquels chaque noeud est associé à une opération élémentaire relative au domaine du problème. Plusieurs autres représentations comme des programmes linéaires et des graphes cycliques.[28]

1.2 Algorithmes à Évolution différentielle

L'évolution différentielle (*Differential Evolution DE*) est une métaheuristique stochastique d'optimisation qui a été inspirée par les algorithmes génétiques et des stratégies évolutionnaires combinées avec une technique géométrique de recherche. Les algorithmes génétiques changent la structure des individus en utilisant la mutation et le croisement, alors que les stratégies évolutionnaires réalisent l'auto-adaptation par une manipulation géométrique des individus. Ces idées ont été mises en œuvre grâce à une opération simple, mais puissante, de mutation de vecteurs, proposée en 1995 par K. Price et R. Storn[35]. Même si, à l'origine, la méthode de l'évolution différentielle était conçue pour les problèmes d'optimisation continus et sans contraintes, ses extensions actuelles peuvent permettre de traiter les problèmes à variables mixtes et gèrent les contraintes non linéaires[20].

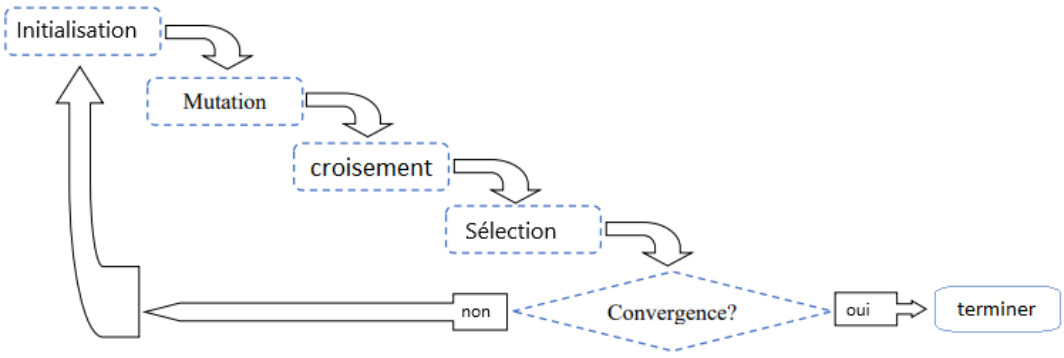

FIGURE 1.1 – Les étapes impliquent un algorithme d'évolution différentielle

Dans la méthode DE, la population initiale est générée par tirage aléatoire uniforme sur l'ensemble des valeurs possibles de chaque variable. Les bornes inférieures et supérieures des variables sont spécifiées par l'utilisateur selon la nature du problème. Après l'initialisation, l'algorithme effectue une série de transformations sur les individus, dans un processus appelé évolution.[1]

La population contient N individus. Chaque individu $x_{i,G}$ est un vecteur de dimension D, où G désigne la génération :

$$x_{i,G} = (x_{1i,G}, x_{2i,G}, \cdots, x_{Di,G}) \quad avec \quad i = \{1, 2, \cdots, N\}$$

Le standard DE utilise trois techniques (mutation, croisement et sélection) comme les algorithmes génétiques. A chaque génération, l'algorithme applique successivement ces trois opérations sur chaque vecteur pour produire un vecteur d'essai (*trial vector*) :

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \cdots, u_{Di,G+1}) \quad avec \quad i = \{1, 2, \cdots, N\}$$

Une opération de sélection permet de choisir les individus à conserver pour la nouvelle génération $(G + 1)$.

A- Mutation

Pour chaque vecteur courant $x_{i,G}$, on génère un vecteur mutant $v_{i,G+1}$ qui peut être créé en utilisant une des stratégies de mutation suivantes :

- Rand/1 :

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G})$$

- Best/1 :

$$v_{i,G+1} = x_{best,G} + F(x_{r1,G} - x_{r2,G})$$

- Current to best/1 :

$$v_{i,G+1} = x_{i,G} + F(x_{r1,G} - x_{r2,G}) + F(x_{best,G} - x_{i,G})$$

- Best/2 :

$$v_{i,G+1} = x_{best,G} + F(x_{r1,G} - x_{r2,G}) + F(x_{3,G} - x_{4,G})$$

- Rand/2 :

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) + F(x_{r4,G} - x_{r5,G})$$

Les indices $r_1, r_2, r_3, r_4 \cdots r_N \in \{1, 2, \cdots, N\}$ sont des entiers aléatoires et tous différents. Ils sont également choisis différents de l'indice courant i.$x_{best,G}$ est le meilleur individu à la G^{me} génération.$F \in [0, 2]$ est une valeur constante, appelée differential weight, qui contrôle l'amplification de la variation différentielle de $(x_{ri,G} - x_{rj,G})$.

B- Croisement

Après la mutation, une opération de croisement binaire forme le vecteur d'essai final $u_{i,G+1}$, selon le vecteur $x_{i,G}$ et le vecteur mutant correspondant $v_{i,G+1}$. L'opération de croisementest introduite pour augmenter la diversité des vecteurs de paramètres perturbés. Le nouveau vecteur $u_{i,G+1}$ est donné par la formule suivante :

$$u_{ji,G+1} = \begin{cases} v_{1i,G+1} & si \quad (randb(j) \leq CR) \quad ou \quad j = rnbr(i) \\ & pour \quad toutj \in 1, 2, \cdots, D \\ x_{ji,G} & si \quad (randb(j) > CR) \quad et \quad j \neq rnbr(i) \end{cases}$$

où randb(j) est la j^{me} valeur procurée un générateur de nombre aléatoire uniforme appartenant à l'intervalle [0,1]. CR est le coefficient de croisement qui appartient à l'intervalle [0,1] et est déterminé par l'utilisateur. rnbr(i) est un indice choisi au hasard dans l'ensemble $\{1, 2, \cdots, N\}$.

C- Sélection

Pour décider quel vecteur, parmi $u_{i,G+1}$ ou $x_{i,G}$, doit être choisi dans la génération $G + 1$, on doit comparer les valeurs de fonction du cout de ces deux

vecteurs. En effet, on garde le vecteur ayant la plus petite valeur de fonction du cout en cas de minimisation. Le nouveau vecteur $x_{i,G+1}$ est choisi selon l'expression suivante :

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & si \quad f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & sinon \end{cases}$$

Il est clair qu'un bon choix des principaux paramètres de l'algorithme (taille de la population N, facteur de mutation F et facteur de croisement CR) contribue de façon importante à l'efficacité de la méthode. L'auto-adaptation de ces paramètres paraît donc intéressante pour l'amélioration de l'algorithme.

1.3 DE pour les problèmes d'optimisation mono-objectif

1.3.1 Optimisation mono-objectif

Définition 1.3.1 Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est mono-objectif. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal). De manière formelle, à chaque instance d'un tel problème est associé un ensemble Ω des solutions potentielles respectant certaines contraintes et une fonction d'objectif $f : \Omega \rightarrow \Psi$ qui associe à chaque solution admissible $s \in \Omega$une valeur $f(s)$. Résoudre l' instance (Ω, f) du problème d'optimisation consiste à trouver la solution optimale $s* \in \Omega$qui optimise (minimise ou maximise) la valeur de la fonction objectif f. Pour le cas de la minimisation : le but est de trouver $s* \in \Omega$ tel que $f(s*) \leq f(s)$ pour tout élément $s \in \Omega$. Un problème de maximisation peut être défini de manière similaire.

Algorithm 1 Algorithme DE

```
1: fonction EvolutionDifférentielle(f : fonction objectif, NP : taille de la
   population, W : facteur d'amplitude, CR : taux de croisement)
2: P ← population initiale alatoire
3: initialize all the vectors of the population randomly
4: repeat
5:    P' ← ∅ {population temporaire }
6:    for x ∈ P do
7:       (u, v, w) ∈ ChoixParents(x, P)
8:       y ← Croisement(x, u, v, w, W, CR) { génère un nouvel individu }
9:       if f(y) < f(x) then
10:         P' ← P' ∪ y {y remplace x}
11:      else
12:         P' ← P' ∪ x { x est conservé}
13:      end if
14:   end forp ← p' {la population temporaire remplace la population }
15: until critère d'arrêt vérifié
16: return meilleur individu de P
17: end fonction
```

1.4 DE pour les problèmes d'optimisation multi-objectifs

1.4.1 Optimisation multi-objectif

Problèmes d'optimisation multi-objectifs

Définition 1.5.1 Un problème d'optimisation multi-objectifs consiste à rechercher les meilleures solutions qui minimisent un nombre M de fonctions, appelées fonctions coût, $f_m[X], m = 1 \cdots M$ par rapport à un vecteur X, qui est le vecteur des n variables de contrôle (ou paramètres) :
$X = [x1, x2 \cdots xn]^T$, en satisfaisant un certain nombre de contraintes, explicites comme les contraintes de borne $(x_i^I \leq x_i \leq x_i^8, i = 1, \cdots, n)$ ou implicites, d'égalité $(h_k(x) = 0, k = 1, \cdots, K)$ou d'inégalité $(g_j(x) \geq 0, j = 1, \cdots, J)$. On se limite aux problèmes de minimisation, puisque la maximisation d'une fonction $f(X)$ peut facilement être transformée en un problème de minimisation :[5]

$$max(f(X)) = -min(-f(X))$$

Un problème d'optimisation peut s'écrire sous la forme générale suivante :

$$(PM)\begin{cases} \text{Opt} \quad f_m(X), m \geq 1 \\ s.c \\ g_j(x) \geq 0, j = 1, \cdots, J \\ xi \geq 0 \end{cases}$$

Dans notre travail, on considère le cas de minimisation.

Quand le problème d'optimisation contient une seule fonction coût ($M = 1$), il est appelé mono-objectif. Ce problème, utilise un seul espace de recherche, connu aussi sous le nom d'espace de décision S. Il s'agit de l'ensemble des combinaisons possibles des paramètres qui satisfont toutes les contraintes. En revanche, si le problème d'optimisation contient plusieurs fonctions coût ($M > 1$) , et si les fonctions coût sont antagonistes, il est appelé multiobjectifs. Un problème d'optimisation multi-objectifs porte sur deux espaces, que sont l'espace de décision S et l'espace fonctionnel (l'espace des fonctions coût)F.[5]

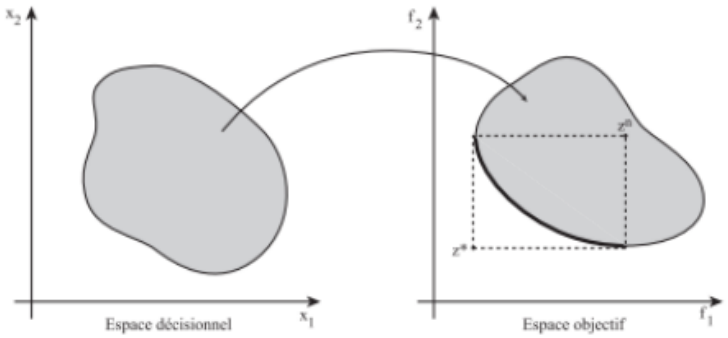

FIGURE 1.2 – Espace décisionnel et espace objectif d'un problème d'optimisation multiobjectif(exemple avec deux variables de décisions et deux fonctions objectifs)

Algorithm 2 Algorithme MODE

1: Evaluer Les NP individus aléatoires de la population initiale P.
2: **while** le critère d'arrêt n'est pas atteint **do**
3: Classer les individus dans des fronts selon la non-dominance.
4: calculer la distance d'encombrement des individus de chaque front.
5: appliquer les opérations DE sur les individus de la population triée (P') pour créer de nouveaux individus de la population (P").
6: effectuer un Elitisme :
- former une population(PT) de taille 2NP contenant les individus parents et les nouveaux individus crées.
- trier la population (PT) selon la non-dominance (PT').
- choisir les meilleurs NP individus de(PT') et les mettre dans (P"').
7: **end while**

conclusion

Dans ce chapitre, nous avons présenté l'algorithme à évolution différentielle pour résoudre les problèmes mono-objectif et multi-objectif, reste à étudier l'amélioration de DE multi-objectif, pour cela nous aborderons dans le chapitre suivant l'étude des approches métaheuriqtiques.

Chapitre 2

Approches métaheuristique

Introduction

Les métaheuristiques forment un ensemble de méthodes utilisées en recherche opérationnelle et en intelligence artificielle pour résoudre des problèmes d'optimisation réputés difficiles. Résoudre un problème d'optimisation combinatoire, c'est trouver l'optimum d'une fonction, parmi un nombre fini de choix, souvent très grand. Les applications concrètes sont nombreuses, que ce soit dans le domaine de la production industrielle, de transports ou de l'économie ... etc.
De nombreuses définitions ont été faites dans la littérature, dans ce chapitre nous retiendrons que deux définitions "Un métaheurisic est un ensemble de concepts qui peuvent être utilisés pour définir des méthodes heuristiques qui peuvent être appliquées à un large éventail de problèmes différents. En d'autres termes, une métaheuristique peut être considérée comme un cadre algorithmique général qui peut être appliqué à différents problèmes d'optimisation avec relativement peu de modifications pour les adapter à un problème spécifique."[9]

"Une métaheuristique est un processus maître itératif qui guide et modifie les opérations de l'heuristique subordonnée pour produire efficacement des solutions de haute qualité. Il peut manipuler une solution unique complète (ou incomplète) ou un ensemble de solutions à chaque itération. Les heuristiques subordonnées peuvent être des procédures de niveau élevé (ou bas), ou une simple recherche locale, ou simplement une méthode constructive." (S. Voß, et al. 1999).
Alors une métaheuristique est une méthode algorithmique capable de guider

et d'orienter le processus de recherche dans un espace de solution, souvent très grand à des régions riches en solutions optimales. Le fait de rendre cette méthode abstraite et plus générique conduite à un vaste utilisation pour des champs d'applications différents.
A ces applications, les métaheuristiques permettent, de trouver des solutions, peut-être pas toujours optimales, en tout cas très proches de l'optimum et en un temps raisonnable. Elles se distinguent en cela des méthodes dites exactes, qui garantissent certes la résolution d'un problème, mais au prix de temps de calcul prohibitifs.[9]

2.1 Les métaheuristiques

2.1.1 Définition

Heuristique :
Le mot heuristique, dérivé du Grec "*heuriskein*", signifie découvrir. Une heuristique est "une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème."[30]. Comme les heuristiques ne garantissent pas l'atteinte d'un optimum global, leur utilisation est justifiée dans certains cas, soit :

1. Les données sont inexactes ou limitées ;
2. Un modèle simplifié du problème est utilisé ;
3. Une méthode exacte existe, mais n'est pas intéressante au niveau informatique parce que trop vorace en temps de calcul ou en espace mémoire ;
4. une solution heuristique est acceptable.

Il existe plusieurs classifications pour les heuristiques. Parmi les catégories proposées, on retrouve, entre autres, les heuristiques de construction, d'amélioration, de programmation mathématique et les métaheuristiques.
Méta-heuristique :
Le terme méta-heuristique a été introduit par Glover en 1986, pour différencier la recherche avec tabous des autres heuristiques[30]. Les métaheuristiques sont apparues avec une ambition commune : résoudre au mieux les problèmes d'optimisation NP-difficiles. Le mot métaheuristique est dérivé de la composition de deux mots grecs :

- Heuristique qui vient du verbe heuriskein (euriskein) et qui signifie "trouver" ;
- Meta qui est un suffixe signifiant "au-delà", et indique un changement de niveau, un passage à un niveau "supérieur" pour étudier ou manipuler des informations de niveau inférieur.

Plusieurs définitions ont été proposées pour expliquer clairement ce qu'est une métaheuristique. Aucune de ces définitions n'est universellement reconnue. Une définition est donnée dans :
Une métaheuristique généralement, n'est pas propre à un problème précis, mais adaptable et applicable à une large classe de problèmes. Elles sacrifient la garantie d'optimalité ou d'approximation avec en contrepartie l'espoir de trouver très rapidement de bonnes solutions. Pour adapter une métaheuristique à un problème d'optimisation particulier, il est nécessaire de modéliser adéquatement un certain nombre d'ingrédients :

- L'ensemble S des solutions ; cet ensemble définit le domaine dans lequel la recherche d'un optimum va s'effectuer.
- La fonction objectif f à minimiser (ou à maximiser) ; cette fonction induit une topologie sur l'espace de recherche, avec des montagnes (mauvaises solutions), des vallées (régions autour d'un minimum local), etc
- Le voisinage d'une solution, ce concept indique les déplacements possibles dans S. L'exploration de S consiste à parcourir un chemin qui se dirige à chaque pas d'une solution vers une solution voisine.
- L'opérateur de combinaison ; utilisé dans le cadre des méthodes évolutives, cet opérateur permet de générer de nouvelles solutions à partir des solutions présentes dans la population courante

2.1.2 Classification des Métaheuristiques

Les problèmes d'optimisation combinatoire sont souvent des problèmes très difficiles dont la résolution par des méthodes exactes peut s'avérer très longue ou peu réaliste. L'utilisation de méthodes heuristiques permet d'obtenir des solutions de bonne qualité en un temps de résolution raisonnable. Les heuristiques sont aussi très utiles pour le développement de méthodes exactes fondées sur des techniques d'évaluation et de séparation (Branch and Bound).
Une heuristique est un algorithme qui a pour but de trouver une solution réalisable, sans garantie d'optimalité, contrairement aux méthodes exactes qui garantissent des solutions exactes. Comme les algorithmes de résolution exacte sont de complexité exponentielle pour les problèmes difficiles, il peut

être plus judicieux de faire appel aux heuristiques pour calculer une solution approchée d'un problème ou aussi pour accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, mais les approches peuvent contenir des principes plus généraux. On parle de métaheuristique.
Une manière de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthodes de recherche locale ou méthodes de trajectoire.[9]

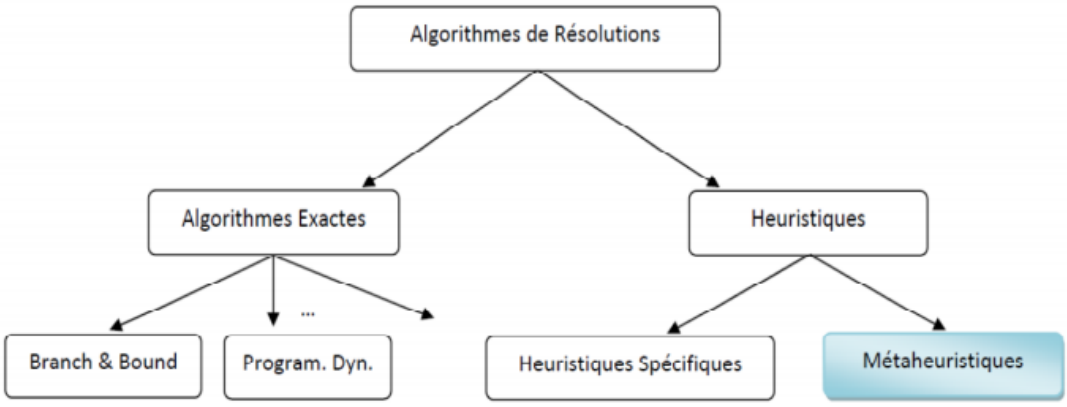

FIGURE 2.1 – Classes des méthodes de résolutions.

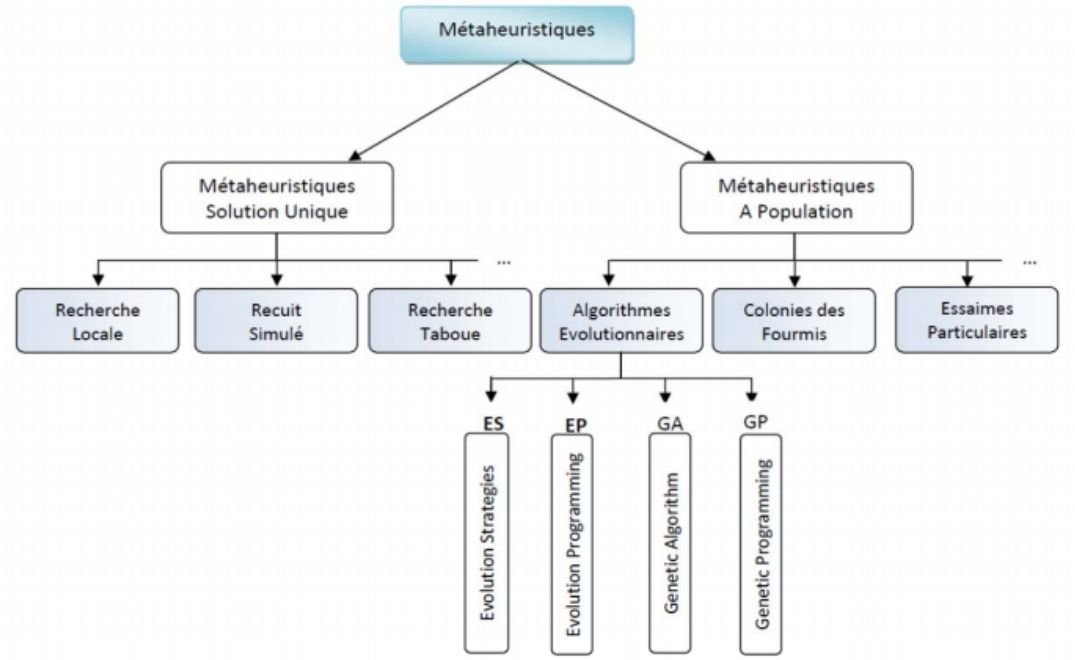

FIGURE 2.2 – Classes des méthaheuristiques

2.2 Méthodes hybrides

Définition

Une méthode hybride est une méthode de recherche constituée d'au moins de deux méthodes de recherche distinctes. Elle consiste à exploiter les avantages respectifs de plusieurs méthodes en combinant leurs algorithmes suivant une approche synergétique. Une méthode hybride peut être mauvaise ou bonne selon le choix et les rôles de ses composants. Pour définir une méthode hybride efficace, il faut savoir caractériser les avantages et les limites de chaque méthode. Par exemple, les algorithmes génétiques sont très performants lorsqu'il s'agit d'explorer l'espace de recherche, mais ils s'avèrent ensuite incapable d'exploiter efficacement la zone vers laquelle la population des solutions converge. Il est alors plus intéressant d'utiliser dans ce stade une autre méthode permettant une bonne exploitation comme par exemple le recuit simulé ou une autre heuristique d'amélioration. Il faut souligner qu'il faut être prudent sur le choix des méthodes à hybrider ainsi sur le problème de multiplication des paramètres[30].
Les origines des algorithmes hybrides remontent probablement aux travaux

décrits dans [32]. Les auteurs ont clairement démontré l'effet bénéfique de l'intégration d'une méthode de descente à l'intérieur d'une méthode évolutive. D'autres travaux ont montrés que lors de la résolution du problème de coloration des graphes, les meilleurs résultats sont obtenus en hybridant une méthode évolutionnaire avec une méthode de recherche locale. Plusieurs publications ont été réalisées dans ce domaine, qui devient de plus en plus intéressant[30].

Actuellement, les approches hybrides gagnent en popularité car ce type d'algorithme produit généralement les meilleurs résultats pour plusieurs problèmes d'optimisation combinatoire. En effet, selon[39], les approches hybrides ont permis d'obtenir de bons résultats dans une grande variété de problèmes théoriques d'optimisation combinatoire tels que le problème du voyageur de commerce, le problème de coloration de graphe, le problème d'affectation quadratique, le problème de tournée de véhicules, le séquençage d'ADN ou encore le calcul des trajectoires des satellites. Les premières approches hybrides proposées ont combiné des métaheuristiques à base de populations (algorithmes génétiques) avec des métaheuristiques à solution unique (recuit simulé ou recherche tabou).

Plusieurs approches hybrides combinant les métaheuristiques et les méthodes exactes ont été proposées dans la littérature [39]. Etant donné que les méthodes exactes se limitent généralement à de petites instances pour les problèmes d'optimisation combinatoire difficiles, tel que décrit précédemment, l'hybridation de métaheuristiques avec les méthodes exactes peut devenir une alternative très intéressantecar les deux méthodes ont des particularités bien différentes qui peuvent être associées pour produire de meilleurs résultats[30].

2.3 Classification des stratégies d'hybridation

L'hybridation peut prendre place à deux niveaux : soit plusieurs méthodes de recherche coopèrent au sein d'une méthode hybride, soit une heuristique (ou une méthode de recherche est insérée à l'intérieur d'une autre méthode de recherche. Dans ce dernier cas, il est nécessaire de définir à quel niveau, c'est-à-dire dans quel composant de la méthode de recherche va prendre place l'hybridation. Plusieurs classifications ont été proposées dans la littérature, les plus importantes sont décrites ci-dessus[30].

2.3.1 Classification I

Dans [Duvidier 00], l'auteur propose une classification des techniques d'hybridation en s'appuyant sur les travaux du groupe PERFORM (Parallel gEnetic algoRithms FOR optiMization) du laboratoire LIFL de Lille. Les approches hybrides sont classifiées en trois catégories, selon leurs architectures[30] :

- **Hybridation séquentielle.**
- **Hybridation parallèle synchrone.**
- **Hybridation parallèle asynchrone.**

Hybridation séquentielle : C'est le type d'hybridation le plus populaire. Elle consiste à appliquer plusieurs méthodes de telle manière que le (ou les) résultat(s) d'une méthode serve(nt) de solution(s) initiale(s) à la suivante. Par exemple utiliser l'algorithme de recuit simulé pour améliorer les solutions obtenues par un algorithme génétique. C'est la technique d'hybridation la plus simple (figure 2.3)

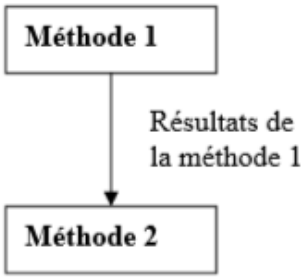

FIGURE 2.3 – Hybridation séquentielle [30]

Hybridation parallèle synchrone : Cette hybridation est réalisée par incorporation d'une méthode de recherche particulière dans un opérateur. Elle est plus complexe à mettre en œuvre que la précédente. L'objectif est de combiner une recherche locale avec une recherche globale dans le but d'améliorer la convergence. Par exemple une recherche tabou est utilisée pour générer des solutions initiales pour un algorithme d'optimisation par colonies de fourmi figure 2.4

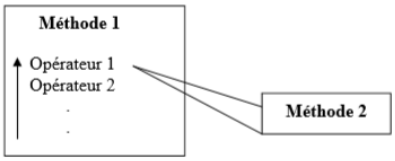

FIGURE 2.4 – Hybridation parallèle synchrone[30]

Hybridation parallèle asynchrone (coopérative) : Les méthodes hybrides appartenant à cette classe sont caractérisées par une architecture telle que deux algorithmes A et B sont impliqués simultanément et chacun ajuste l'autre. Les Algorithmes A et B partagent et échangent de l'information tout au long du processus de recherche figure 2.5

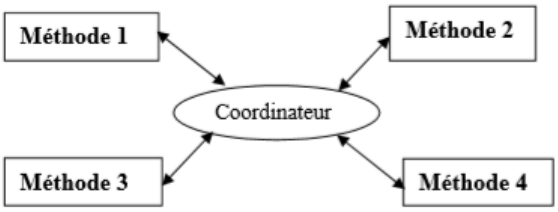

FIGURE 2.5 – Hybridation parallèle asynchrone (coopérative)[30]

2.3.2 Classifications II

Une autre classification des méthodes hybrides a été proposée dans [Raidl], elle est inspirée des travaux de [Talbi 02] et [Cotta et al 05]. Cette classification est illustrée par la figure(2.6). Quatre types de classification ont été proposés selon différents critères :
La première classification est basée sur la nature de la méthode à hybrider, d'où on peut tirer plusieurs combinaisons [30] :

- **Métaheuristique avec métaheuristique :** par exemple un algorithme génétique est hybridé avec le recuit simulé.
- **Métaheuristique avec un algorithme spécifique au problème.**
- **Métaheuristique avec différentes techniques de la recherche opérationnelle et de l'intelligence artificielle :** programmation dynamique, logique floue, réseaux de neurones

La deuxième classification est basée sur le niveau d'hybridation. On distingue deux niveaux : hybridation de haut niveau et hybridation de bas niveau.

— **Hybridation de haut niveau :** où, on préserve les caractéristiques individuelles des algorithmes originaux. La coopération entre les différents algorithmes est réalisée via une interface bien définie.
— **Hybridation de bas niveau :** les algorithmes combinés dépendent chacun à l'autre et leurs composants sont échangées et sans que leur fonctionnement interne ne soit en relation

Pour la troisième classification, le critère considéré est celui de l'ordre d'exécution : séquentiel ou parallèle. Les caractéristiques des algorithmes parallèles ont permet de les distinguer selon l'architecture SIMD (Single Instruction Multiple Data streams) et MIMD (Multiple Instruction Multiple Data streams), la mémoire (partagée ou distribuée), la l'allocation des ressources (statique ou dynamique), et le type de synchronisation (synchrone ou asynchrone).
Le dernier critère utilisé pour différentier entre les différentes approches hybrides basées sur les métaheuristiques est celui de la stratégie de contrôle : collaborative ou intégrative [30] :

— **Stratégie intégrative :** c'est la plus populaire, les composants d'un algorithme dit subordonné sont incorporés dans un autre algorithme. L'exemple le plus proche est celui des algorithmes mémétiques où plusieurs techniques de recherche locale sont introduites dans un algorithme évolutionnaire[30].

— **Stratégie coopérative :** les algorithmes échangent de l'information, mais aucun ne fait partie de l'autre

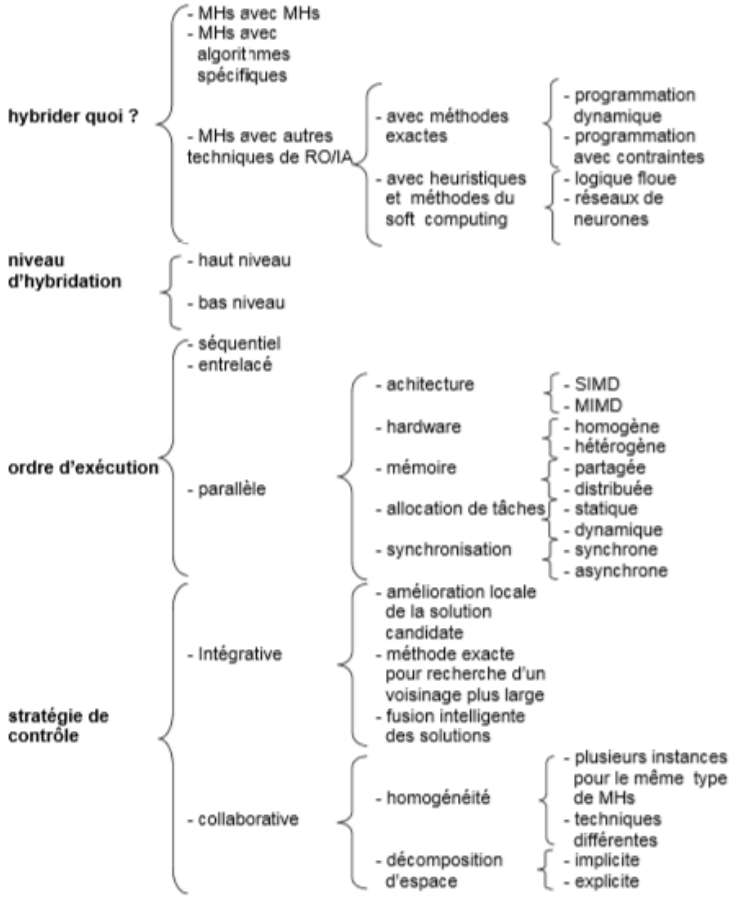

FIGURE 2.6 – Classification II des métaheuristiques hybrides[30]

2.3.3 Classification III

Une taxonomie a été proposée par [Talbi 02], permettant de classifier les différentes hybridations et de les comparer de façon qualitative. Cette taxonomie comporte deux aspects principaux. Une classification hiérarchique permettant d'identifier la structure générale de l'hybridation et une classification générale spécifiant les détails des algorithmes impliqués dans l'hybridation[30].
La classification hiérarchique : Se subdivise en deux classes : une hybridation de bas niveau et une autre de haut niveau. On parle d'une hybridation de bas niveau lorsqu'une fonction d'une métaheuristique est remplacée

par une autre métaheuristique. Par contre, une hybridation de haut niveau est obtenue lorsque deux métaheuristiques sont hybridées sans que leur fonctionnement interne ne soit modifié. Chacune des deux classes d'hybridation précédentes se subdivise en deux sous classes : à relais et co-évolutionnaire. Lorsque les métaheuristiques sont exécutées de façon séquentielle, l'une utilisant le résultat de la précédente comme entrée, on a une hybridation à relais. L'hybridation coévolutionnaire se fait lorsque des agents coopèrent en parallèle pour explorer l'espace de solutions. La combinaison des classes citées précédemment permet d'avoir quatre classes différentes d'hybridation[30] :

1. **Hybridation de bas niveau à relais :** représente les algorithmes dans lesquels une métaheuristique est incorporée dans une autre métaheuristique à solution unique, par exemple, une recherche locale est incorporée dans un algorithme de recuit simulé pour résoudre le problème du voyageur de commerce[30].
2. **Hybridation de bas niveau co-évolutionnaire :** consiste à incorporer un algorithme de recherche locale basé sur l'exploitation avec une métaheuristique à population de solution axée sur l'exploration. Par exemple, le remplacement de l'opérateur de mutation d'un algorithme génétique par une recherche locale pour résoudre le problème de partition des graphes (figure 2.7)
3. **Hybridation de haut niveau à relais :** dans ce cas, des métaheuristiques complètes sont exécutées séquentiellement. Par exemple prendre les meilleures solutions trouvées par un algorithme d'optimisation par colonie de fourmis comme solution initiale d'une recherche tabou (figure 2.8)
4. **Hybridation de haut niveau co-évolutionnaire :** implique un ensemble de métaheuristiques complètes qui travaillent en parallèle et coopèrent pour trouver la solution optimale du problème. Par exemple coopérer entre une recherche locale et une recherche tabou, de sorte que, à intervalles réguliers, les deux méthodes échangent des information (figure 2.9)[30].

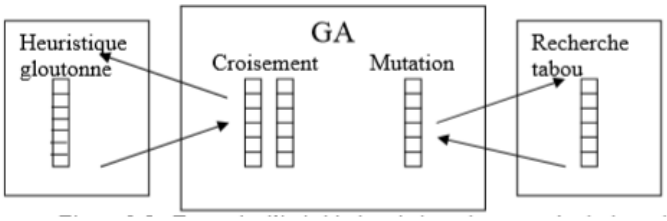

FIGURE 2.7 – Exemple d'hybridation de bas niveau co-évolutionnaire [30]

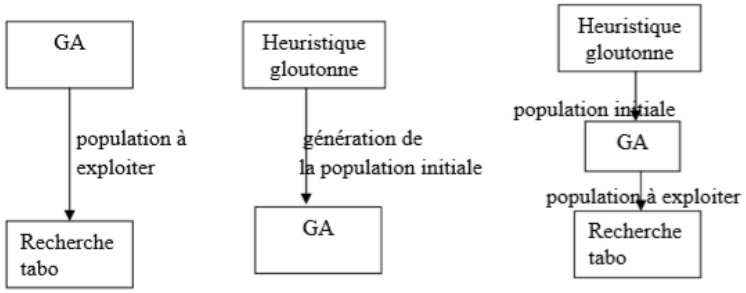

FIGURE 2.8 – Exemple d'hybridation de haut niveau à relais [30]

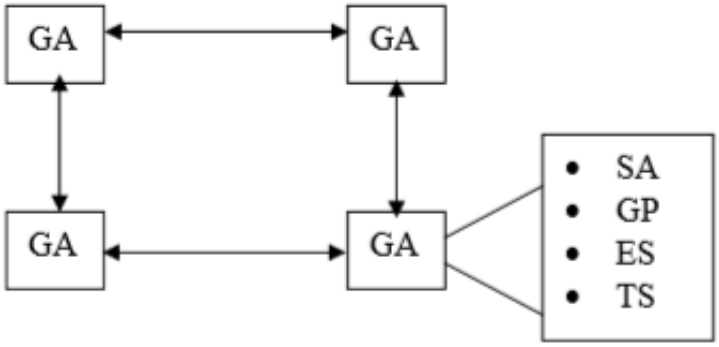

FIGURE 2.9 – Exemple d'hybridation de haut niveau co-évolutionnaire [30]

La classification générale : Une approche hybride peut être homogène ou hétérogène, globale ou partielle et spécialisée ou générale. Une hybridation est dite hétérogène lorsque les métaheuristiques combinées sont différentes. Une hybridation globale fait en sorte que toutes les métaheuristiques explorent l'ensemble de l'espace des solutions. Par contre une hybridation est dite partielle lorsqu'elle décompose un problème en sous-problèmes ayant leur propre espace de solutions, et que chaque sous-problème est résolu par un algorithme. Les hybridations générales sont celles où tous les algorithmes hybridés résolvent le même problème d'optimisation. Les hybridations spé-

cialisées sont celles où chaque algorithme résout un problème d'optimisation différent(voir figure2.10)

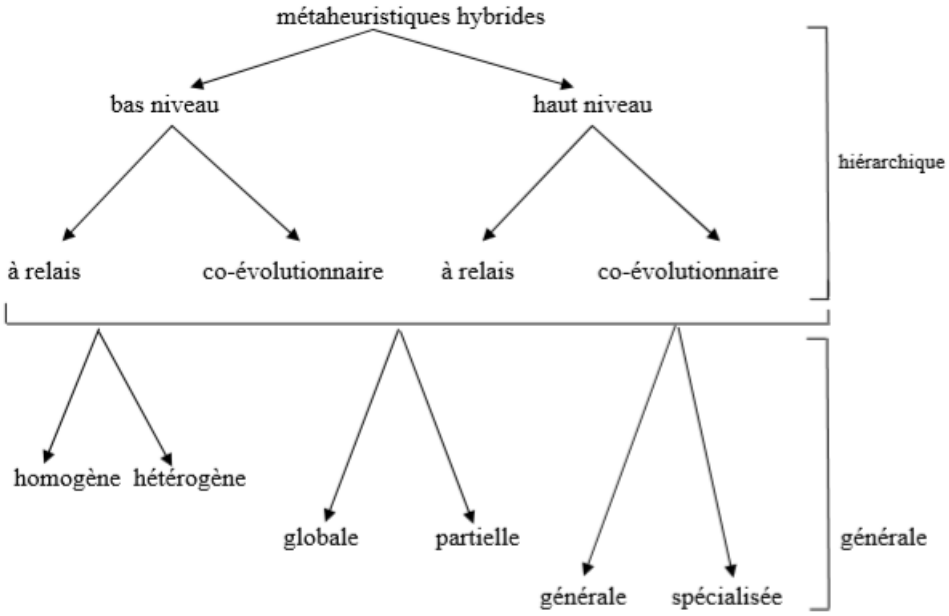

FIGURE 2.10 – Classification III des métaheuristiques hybrides [30]

2.3.4 Classification IV

Une classification des méthodes hybrides incluant les métaheuristiques et les méthodes exactes a été proposée dans [Dimistrescu et Stutzle 03]. Le critère considéré est celui de l'objectif de la méthode exacte utilisée. Dimistrescu et Stùtzle proposent la classification suivante [30] :

1. Utilisation d'un algorithme exact dans une recherche locale pour une exploration intensive de voisinage ;
2. Utilisation d'une heuristique pour réduire l'espace de recherche d'une méthode exacte ;
3. Utilisation une méthode exacte pour définir les bornes des solutions pour une heuristique constructive ;
4. Utilisation des informations fournies par une méthode exacte pour orienter un algorithme de recherche locale ou constructif ;

5. Utilisation d'une méthode exacte pour remplacer une fonction spécifique de la métaheuristique.

2.3.5 Classification V

Une autre classification axée sur l'hybridation de méthodes exactes et approximatives a été proposée dans [Puchinger et Raidi 05]. Les méthodes hybrides sont divisées en deux catégories : les hybridations collaboratives et intégratives. Les algorithmes qui échangent des informations de façon séquentielle, parallèle ou entrelacée font partie de la catégorie des hybridations collaboratives. Les algorithmes à hybridation intégrative font en sorte qu'une technique est une composante incorporée à une autre technique[30]

Les combinaisons d'une méthode exacte et d'une métaheuristique de façon intégrative se font de manière à ce qu'un des deux algorithmes soit une composante intégrée à l'autre algorithme. On peut donc intégrer une méthode exacte à une métaheuristique, et inversement. Par exemple, utiliser Branch and Bound pour avoir le meilleur croisement possible entre individus dans un algorithme évolutionnaire [Cotta et Troya 03]. D'autre part, on peut utiliser une métaheuristique pour calculer les bornes des solutions sortantes pour les approches Branch and Bound [40].

2.4 Hybridation : Etat de l'art

2.4.1 Hybridation basée sur EA

Les algorithmes évolutionnaires sont devenus très utilisés pour résoudre les problèmes complexes. Plusieurs techniques sont regroupées dans cette catégorie : Les algorithmes génétiques, la programmation génétique et les stratégies évolutionnaires, etc. Elles sont basées tous sur le même concept en simulant l'évolution d'un individu par sélection, mutation et reproduction. Les avantages des algorithmes évolutionnaires sont multiples, à savoir la simplicité de l'approche, sa robustesse face aux changements de l'environnement et enfin sa flexibilité. Les algorithmes évolutionnaires sont faciles à implémenter par rapport à d'autres métaheuristiques. Par contre, pour plusieurs problèmes complexes, les algorithmes évolutionnaires ne réussissent pas à trouver la solution optimale, d'où la nécessité de les hybrider avec d'autres méthodes

bio-inspirées, techniques d'apprentissage, heuristiques, etc[30].

L'idée essentielle d'hybridation des algorithmes évolutionnaires consiste essentiellement à exploiter pleinement la puissance de recherche de méthodes de voisinage et de recombinaison des algorithmes évolutifs sur une population de solutions. Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itérations ou jusqu'à la découverte d'un ensemble d'optimums locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus. La recombinaison doit impérativement être adaptée au problème traité. En effet, cette approche a permis de produire d'excellents voire les meilleurs résultats sur des benchmarks réputés de problèmes de référence. C'est le cas par exemple du problème du voyageur de commerce, de la coloration des graphes, de la clique maximale et de l'affectation quadratique. Malgré la puissance de l'approche hybride, le temps de calcul nécessaire peut devenir prohibitif à cause du nombre d'individus manipulés dans la population. Une voie pour résoudre ce problème est la parallélisation de ces algorithmes sur des machines parallèles ou sur des systèmes distribués.

Algorithme génétique hybride

Ci-dessous un exemple d'un algorithme génétique hybride :

Algorithme génétique hybride
étape 1(initialisation)
a) générer une population de configurations P
b) appliquer une recherche locale à chacune des configurations de P
étape 2(évolution et terminaison)
a) choisir p1 et p2 dans P
b) générer une configuration e par une recombinaison de p1 et p2
c) améliorer e par l'application de la recherche locale
d) Insérer la nouvelle configuration e dans la population
e) terminer et retourner la meilleure solution trouvée si la condition d'arrêt vérifiée
étape 3(mise à jour)
f) re-organisation de la population (ex, élimination des configurations non performantes de la population
étape 4
aller à (étape 2) jusqu'à critère d'arrêt satisfait.

Il existe plusieurs manières pour hybrider les algorithmes évolutionnaires, qu'on peut les résumer comme suit :

- la population initiale peut être créée avec une heuristique spécifique au problème
- quelques solutions obtenues par un algorithme évolutionnaire peuvent être améliorés par une recherche locale (algorithme mémétique)
- des opérateurs de croisement et de mutation peuvent exploiter les connaissances préalables du problème
- une heuristique est utilisée pour le décodage des solutions en exploitant des connaissances propres du problème (voir figure 2.11)

Plusieurs hybridations sont proposées dans la littérature [Grosan et Abraham 07] hybridant un algorithme évolutionnaire avec :

1. Un autre algorithme évolutionnaire
2. Un réseau de neurones
3. La logique floue
4. Un algorithme d'optimisation par essaim de particule
5. Un algorithme optimisation par colonies de fourmis
6. D'autres heuristiques telles que la recherche locale, la recherche tabou, le recuit simulé, la programmation dynamique, etc.

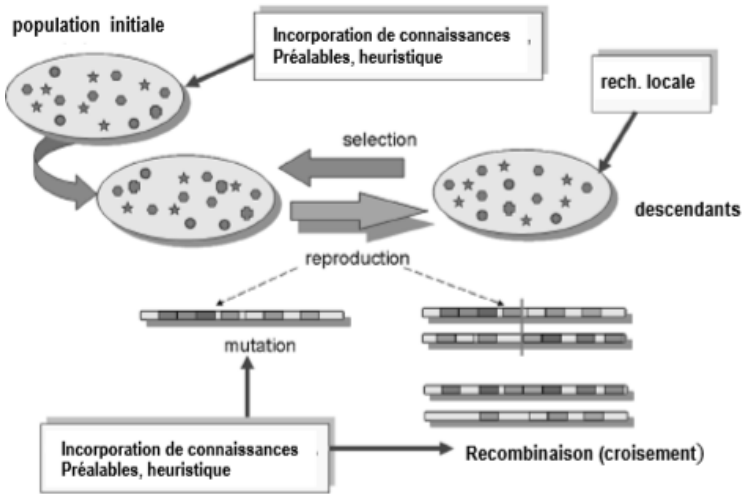

FIGURE 2.11 – hybridation des algorithmes évolutionnaires[?]

2.4.2 Hybridation basée sur DE

L'hybridation avec d'autres algorithmes différents est une direction intéressante pour l'amélioration de l'évolution différentielle (DE).Cette approche est basée sur l'hybridation d'un algorithme de clustering à une étape k-means avec un algorithme d'évolution différentielle(CDE).Celle ci est présentée pour les problèmes d'optimisation globale sans contraintes. proposant une approche hybride pour résoudre les problèmes d'optimisation globale nommée CDE qui consiste à exécuter les deux phases DE et clustering.Le regroupement en une seule étape k-means agit comme plusieurs opérateurs multiparents croisés pour utiliser efficacement l'information de la population, et peut donc améliorer la performance de DE.Les résultats expérimentaux indiquent que cette approche est efficace et efficiente Le pseudo code de l'algorithme est présenté ci-dessous

Algorithm 3 Algorithme CDE

```
Initialize a population of N individuals with K centroid positions each
while stopping condition has not been reached do
  for each individual x_i do
    for each data pattern Z_P do
      Calculate the Eucalidean distance d(z_p, x_ij) to all clusters
      Assign pattern z_p to cluster C_ij such that d(z_p, x_ij) = min_∀k = 1 ··· N_k d(z_p, x_ij)
    end for
    calculate the fitness
    select a target entity and two difference entities
    Generate a trial entity u_i(t)
    Generate the offspring x'_i
    if f(x'_i)is better thenf(x_i) then
      add x'_i to the next population
    else
      add x_i to the next population
    end if
  end for
end while
```

Conclusion

Dans ce chapitre, nous avons donné les approches métaheuristiques avec ses classes ce qui nécessite une mention des méthodes hybrides avec la classification des stratégies en illustrant cela avec des exemples. Enfin, le travail sera clôturé par L'application du DE avec le clustering.

Chapitre 3

Conception

3.1 Apprentissage automatique

Introduction

L'apprentissage automatique est un sous-domaine de l'intelligence artifcielle (IA). En général, l'objectif de l'apprentissage automatique est de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par les tout le monde.[24]
Bien que l'apprentissage automatique soit un domaine de l'informatique, il diffère des approches informatique traditionnelles. En effet dans cette derni ère, les algorithmes sont des ensembles d'instructions explicitement programm ées utilisées par les ordinateurs pour calculer ou résoudre des problèmes. Les algorithmes d'apprentissage automatique permettent aux ordinateurs de s'entraîner sur les entrées de données et utilisent l'analyse statistique pour produire des valeurs qui se situent dans une plage spécifque. Pour cette raison, l'apprentissage automatique facilite l'utilisation des ordinateurs dans la construction du modèles à partir des données d'échantillonnage afn d'automatiser les processus de prise de décision en fonction des données saisies.
Tout utilisateur des plus récentes technologie bénéfcies de l'apprentissage automatique[24].

3.2 Définition de l'Apprentissage Automatique

L'Apprentissage Automatique est une discipline consacrée à l'analyse des données. Le but de cette discipline est de créer de la connaissance de manière automatique à partir de données brutes. Cette connaissance (ou modèle) peut alors être exploitée pour prendre des décisions. On parle parfois de stratégie pilotée par les données (data-driven strategy) pour une entreprise[21].

Comme le modèle est construit à partir des données, il est clair que plus on dispose de données, plus le modèle construit est précis et permettra ainsi de prendre de bonne décisions.

Comme le volume des données nécessaires aux algorithmes de Machine Learning peut être très grand, on associe souvent Machine Learning avec BigData.

Un premier schéma de principe nous permet de fixer les idées :
En résumé :

- Les données disponibles construisent le modèle
- Le modèle permet de prendre des décisions
- On se sert du modèle sur de nouvelles données afin de prendre des décisions

3.2.1 Les données

En Machine Learning, les données sont appelées **échantillons**. Les échantillons sont souvent notés sous forme de vecteur [21] :

$$x = (x_1, x_2, \cdots, x_p)$$

où p est le nombre de **coordonnées** aussi appelé nombre d'**attributs/dimensions/caractéristiques**.

On distingue deux grandes familles de données :

1. **Les données labélisées**. Les données sont accompagnées d'un **label y** qui identifie la décision à prendre pour chaque échantillon. Il est souvent très coûteux (en temps et en argent) d'avoir de grands volumes de données labélisés.
2. **Les données non-labélisées**. Les données ne sont pas accompagnées de labels. Même si les données non labélisées sont plus difficiles à exploiter, elles sont beaucoup plus accessibles. (Un exemple intéressant est la récupération de millions d'images sur le web pour faire de la reconnaissance de visages).

Une remarque importante à ce stade : Les données brutes sont souvent inexploitables. Dans la plupart des cas, il faut procéder à un prétraitement des données afin d'extraire les caractéristiques des données pertinentes pour la prise de décision autrement appelées **caractéristiques** (features).

Cette **Extraction de caractéristiques** (feature selection) fait souvent appel au bon sens, à des facteurs de corrélation statistiques ou à des itérations successives (choix empirique).

3.2.2 La décision

Le choix des features impose de penser aux sorties. On distingue deux types de **sortie y** :

- **Sorties catégorisées ou discrètes**. Le nombre de valeurs que peut prendre y est fini. Dans ce cas, les sorties possibles sont appelées **classes**. Sans perte de généralité, pour C classes, on peut dire que :

  $$y \in \{1, 2, \cdots, C\}$$

  où y est un entier qui varie entre 1 et C)
- **Sorties non-catégorisées ou continues**. Le nombre de valeurs que peut prendre y est infini. Sans perte de généralité, on peut dire que :

  $$y \in R$$

  où y est un élément de l'ensemble mathématique des réels

3.2.3 Le modèle

On peut voir le modèle comme une **fonction notée f** qui prend un échantillon en entrée, et qui renvoie une décision.

La détermination de f est le cœur d'un problème de Machine Learning. Celle-

ci se passe en deux étapes :

1. Choix d'une fonction générique.
2. Détermination des paramètres de la fonction à l'aide des échantillons. Cette phase s'appelle **entraînement** ou **apprentissage** du modèle.

3.3 Domaines de l'Apprentissage Automatique

Les principaux domaines d'applications de l'apprentissage automatique (AA) sont les fouilles de données et l'intelligence artificielle.[29]

- La fouille de données (Data Mining, en anglais) est le processus d'extraction de la connaissance : il consiste à sélectionner les données à étudier à partir de bases de données (BDs) (hétérogènes ou homogènes), à épurer ces données et enfin à les utiliser en apprentissage pour construire un modèle.

Exemples :

- Trouver une prescription pour un malade (patient) à travers des fichiers médicaux antérieurs.
- Apprentissage de la reconnaissance de transactions frauduleuses par carte de crédit, par examen des transactions passées avérées frauduleuses.
- L'intelligence artificielle, la vision par ordinateur, la robotique, l'analyse et la compréhension des images, la reconnaissance de formes, reconnaître des objets dans les vidéo et extraire des contenus sémantiques des images sont autant d'applications qui requièrent la construction de modèles par apprentissage automatique[29].

3.4 L'analyse de l'Apprentissage Automatique

Il convient de retenir quelques étapes simples à effectuer dans l'ordre :

- **Récupération des données à analyser**. Cette étape semble triviale mais est souvent celle qui prend plus de temps... Si vous souhaitez faire de la reconnaissance de visages, vous allez devoir prendre/disposer de photos de milliers de visages pour (prétendre) avoir un classifieur performant.
- **Sélection des caractéristiques**. Comme nous l'avons vu dans l'exemple, il est souvent difficile de faire le choix des features à utiliser pour décrire les données.
- **Choix du modèle**. Comme pour la sélection des features, il n'y a pas de méthodes automatiques. Cela dépend en grande partie des données à analyser et de facteurs empiriques.
- **Entraînement du modèle**. Pour chaque modèle, il existe un algorithme d'entraînement. Cette étape prend souvent beaucoup de temps et augmente avec la taille des données d'entrée.
- **Evaluation du modèle**. En général, on procède par **validation croisée**. Une des déclinaisons de cette méthode consiste à découper l'ensemble de données en deux, d'entraîner le modèle avec la première moitié et de tester le modèle sur la seconde moitié. On calcule ensuite plusieurs indicateurs pour évaluer le modèle, en particulier sa **précision** (nombre de bonnes prévisions divisées par le nombre total de prévisions)[21].

3.5 Types d'algorithmes d'apprentissage

Les algorithmes d'apprentissage peuvent se catégoriser selon le type d'apprentissage qu'ils emploient :

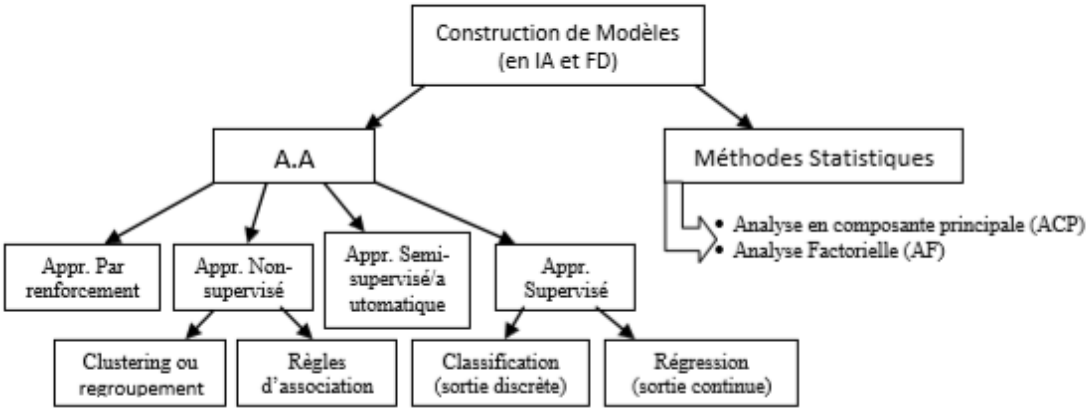

FIGURE 3.1 – Schéma des différentes techniques de l'IA [29]

3.5.1 L'Apprentissage supervisé

L'apprentissage supervisé est le concept derrière plusieurs applications sympas de nos jours : **reconnaissance faciale** de nos photos par les smartphones, **filtres anti-spam** des emails, etc.

Plus formellement, étant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver **une fonction de mapping entre les variables prédictives en entrée X et la variable à prédire Y**. la fonction de mapping décrivant la relation entre X et Y s'appelle un modèle de prédiction.

$$f(X) = Y$$

Les caractéristiques (features en anglais) X peuvent être des valeurs numériques, alphanumériques, des images... Quant à la variable prédite Y, elle peut être de deux catégories :

- **Variable discrète :** La variable à prédire peut prendre une valeur d'un ensemble fini de valeurs (qu'on appelle des classes). Par exemple, pour prédire si un mail est SPAM ou non, la variable Y peut prendre deux valeurs possible :$Y \in \{SPAM, NONSPAM\}$.
- **Variable continue :** La variable Y peut prendre n'importe quelle valeur. Pour illustrer cette notion, on peut penser à un algorithme qui prend en entrée des caractéristiques d'un véhicule, et tentera de prédire le prix du véhicule (la variable Y).

FIGURE 3.2 – Schéma d'un modèle supervisé.

3.5.2 L'Apprentissage non-supervisé

Définition

Dans l'apprentissage non supervisé, les données sont non étiquetées, de sorte que l'algorithme d'apprentissage trouve tout seul des points communs parmi ses données d'entrée. Les données non étiquetées étant plus abondantes que les données étiquetées, les méthodes d'apprentissage automatique qui facilitent l'apprentissage non supervisé sont particulièrement utiles.

L'objectif de l'apprentissage non supervisé peut être aussi simple que de découvrir des modèles cachés dans un ensemble de données, mais il peut aussi avoir un objectif d'apprentissage des caractéristiques, qui permet à la machine intelligente de découvrir automatiquement les représentations nécessaires pour classer les données brutes.
- Les formes principales d'apprentissage non supervisé sont les suivantes :
    - La d'écouverte de classes naturelles, ou clustering (e.g., l'algorithme K-moyennes), qui cherche à d'ecouvrir les modes principaux de la distribution, les "prototypes", les catégories principales, etc... Cela donne une forme de réduction de dimensionalité qui associe un entier à chaque exemple.

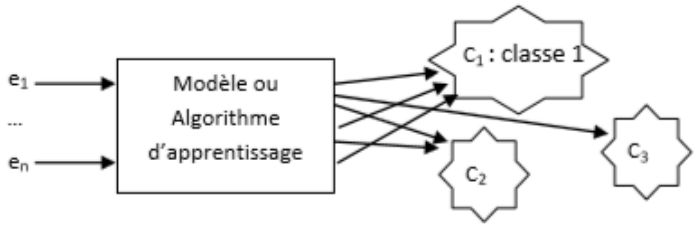

FIGURE 3.3 – Schéma d'un modèle non supervisé.[29]

A.1 La classifcation non-supervisée

Dans une classification non supervisée, les classes sont encore inexistantes.Pour cette démarche, on dispose donc au départ d'un ensemble d'objets.L'idée consiste à découper l'ensemble des objets en groupes (clusters) de telle sorte que les caractéristiques des objets dans des clusters différents soient distinctes.[25]

A.2 Définition de clustering

Le clustering est une méthode d'apprentissage non supervisé (unsupervised learning). Ainsi, on n'essaie pas d'apprendre une relation de corrélation entre un ensemble de features X d'une observation et une valeur à prédire Y, comme c'est le cas pour l'apprentissage supervisé. L'apprentissage non supervisé va plutôt trouver des patterns dans les données. Notamment, en regroupant les choses qui se ressemblent.
En apprentissage non supervisé, les données sont représentées comme suit :

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,\dots} & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,\dots} & x_{2,n} \\ \dots & \dots & \dots & \dots \\ x_{m,1} & x_{m,2} & x_{m,\dots} & x_{m,n} \end{pmatrix}$$

Chaque ligne représente un individu (une observation). A l'issu de l'application du clustering, on retrouvera ces données regroupées par ressemblance. Le clustering va regrouper en plusieurs familles (clusters) les individus/objets en fonction de leurs caractéristiques. Ainsi, les individus se trouvant dans un même cluster sont similaires et les données se trouvant dans un autre cluster ne le sont pas.
Il existe deux types de clustering :

- Le clustering hiérarchique
- Le clustering non-hiérarchique (partitionnement)

A.3 Clustering K-Means

K-means est un algorithme non supervisé de clustering **non hiérarchique**. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents[31].

A.3.1 Notion de similarité

our pouvoir regrouper un jeu de données en K cluster distincts, l'algorithme K-Means a besoin d'un moyen de comparer le degré de similarité entre les différentes observations. Ainsi, deux données qui se ressemblent, auront une distance de dissimilarité réduite, alors que deux objets différents auront une distance de séparation plus grande[31].

Les littératures mathématiques et statistiques regorgent de définitions de distance, les plus connues pour les cas de clustering sont :

- **La distance Euclidienne :** C'est la distance géométrique qu'on apprend au collège. Soit une matrice X à n variables quantitatives. Dans l'espace vectoriel E^n. La distance euclidienne d entre deux observations x_1 et x_2 se calcule comme suit :

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{n} (x_{1n} - x_{2n})^2}$$

- **La distance de Manhattan (taxi-distance) :** est la distance entre deux points parcourue par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou un quadrillage. Un taxi-chemin est le trajet fait par un taxi lorsqu'il se déplace d'un nœud du réseau à un autre en utilisant les déplacements horizontaux et verticaux du réseau.

A.3.2 Choisir K : le nombre de clusters

Choisir un nombre de cluster K n'est pas forcément intuitif. Spécialement quand le jeu de données est grand et qu'on n'ait pas un a priori ou des hypothèses sur les données. Un nombre K grand peut conduire à un partitionnement trop fragmenté des données. Ce qui empêchera de découvrir des patterns intéressants dans les données. Par contre, un nombre de clusters trop petit, conduira à avoir, potentiellement, des cluster trop généralistes contenant beaucoup de données. Dans ce cas, on n'aura pas de patterns "fins" à découvrir.
Pour un même jeu de données, il n'existe pas un unique clustering possible. La difficulté résidera donc à choisir un nombre de cluster K qui permettra de mettre en lumière des patterns intéressants entre les données. Malheureusement il n'existe pas de procédé automatisé pour trouver le bon nombre de clusters.
La méthode la plus usuelle pour choisir le nombre de clusters est de lancer K-Means avec différentes valeurs de K et de calculer la variance des différents clusters. La variance est la somme des distances entre chaque centroid d'un cluster et les différentes observations inclues dans le même cluster. Ainsi, on cherche à trouver un nombre de clusters K de telle sorte que les clusters retenus minimisent la distance entre leurs centres (centroids) et les observations dans le même cluster. On parle de minimisation de la distance intra-classe[31].
La variance des clusters se calcule comme suit :

$$V = \sum_{j} \sum_{x_i \to c_j} D(c_j - x_i)^2$$

Généralement, en mettant dans un graphique les différents nombres de clusters K en fonction de la variance, on retrouve un graphique similaire à celui-ci :

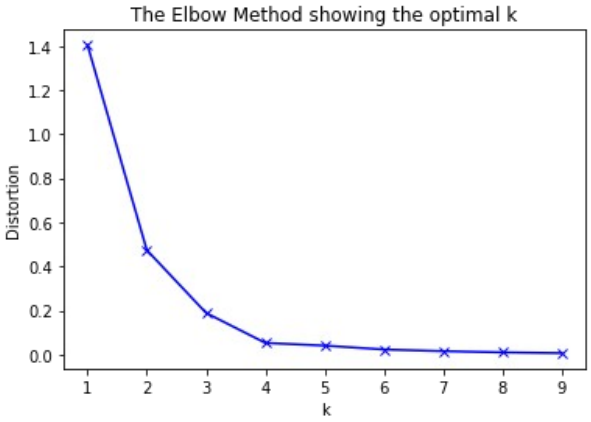

FIGURE 3.4 – The Elbow method showing the optimal K

On remarque sur ce graphique, la forme d'un bras où le point le plus haut représente l'épaule et le point où K vaut 9 représente l'autre extrémité : la main. Le nombre optimal de clusters est le point représentant le coude. Ici le coude peut être représenté par K valant 2 ou 3. C'est le nombre optimal de clusters. Généralement, le point du coude est celui du nombre de clusters à partir duquel la variance ne se réduit plus significativement. En effet, la "chute" de la courbe de variance (distortion) entre 1 et 3 clusters est significativement plus grande que celle entre 5 clusters et 9 clusters.
Le fait de chercher le point représentant le coude, a donné nom à cette méthode : La méthode Elbow (coude en anglais).

Finalement, le choix (au vu de ce graphique), entre 2 ou 3 clusters, reste un peu flou et à votre discrétion. Le choix se fera en fonction de votre jeu de données et ce que vous cherchez à accomplir. Finalement, Il n'y a pas de solution unique à un problème de clustering[31].

A.3.3 Cas d'utilisation K-means

K-Means en particulier et les algorithmes de clustering de façon générale ont tous un objectif commun : Regrouper des éléments similaires dans des clusters. Ces éléments peuvent être tous et n'importe quoi, du moment qu'ils sont encodés dans une matrice de données.

Les champs d'application de K-Means sont nombreux, il est notamment uti-

lisé en :

- La segmentation de la clientèle en fonction d'un certain critère (démographique, habitude d'achat etc....)
- Utilisation du clustering en Data Mining lors de l'exploration de données pour déceler des individus similaires. Généralement, une fois ces populations détectées, d'autres techniques peuvent être employées en fonction du besoin.
- Clustering de documents[31].

A.3.4 Fonctionnement de l'algorithme K-Means

k-means est un algorithme itératif qui minimise la somme des distances entre chaque individu et le centroid. Le choix initial des centroïdes conditionne le résultat final.
Admettant un nuage d'un ensemble de points, K-Means change les points de chaque cluster jusqu'à ce que la somme ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, sous réserve de choisir la bonne valeur K du nombre de clusters[31] .
Principe algorithmique

Algorithm 4 Algorithme K-means

Entrée :
  K le nombre de cluster à former
  Le Training Set (matrice de données)
Begin
  Choisir aléatoirement K points (une ligne de la matrice de données).Ces points sont les centres des clusters (nommé centroId).
repeat
    Affecter chaque point (élément de la matrice de donnée) au groupe dont il est le plus proche au son centre
    Recalculer le centre de chaque cluster et modifier le centroide
until Convergence
  OU (stabilisation de l'inertie totale de la population)
End

Remarque 1 : Lors de la définition de l'algorithme, quand je parle de "point", c'est un point au sens "donnée/data" qui se trouve dans un espace vectoriel de dimension n. Avec n : le nombre de colonnes de la matrice de données.

Remarque 2 : La convergence de l'algorithme K-Means peut être l'une des conditions suivantes :

- Un nombre d'itérations fixé à l'avance, dans ce cas, K-means effectuera les itérations et s'arrêtera peu importe la forme de clusters composés.
- Stabilisation des centres de clusters (les centroids ne bougent plus lors des itérations).

L'affectation d'un point λ à un cluster se fait en fonction de la distance de ce point par rapport aux différents K centroides. Par ailleurs, ce point λ se fera affecté à un cluster i s'il est plus proche de son centroid (distance minimale). Finalement, la distance entre deux points dans le cas de K-Means se calcule par les méthodes évoquées dans le paragraphe "notion de similarité".

A.3.5 Remarques sur le K-Means

Optimums locaux
En analysant la façon de procéder de l'algorithme de K-means, on remarque que pour un même jeu de données, on peut avoir des partitionnements différents. En effet, L'initialisation des tous premiers K centroids est complétement aléatoire. Par conséquent l'algorithme trouvera des clusters différents en fonction de cette première initialisation aléatoire. De ce fait, la configuration des clusters trouvés par K-Means peut ne pas être la plus optimale. On parle d'optimum local.

Afin de palier aux problèmes des optimums locaux, il suffit de lancer K-means plusieurs fois sur le jeu de données (avec le même nombre K de clusters et des initialisations initiales des centroids différentes) et voir la composition des clusters qui se forment[31].

3.5.3 L'application du DE avec le clustering

Après avoir choisi les centres de clusters initiaux par algorithme d'évolution différentielle, on appliquera une sorte de processus de changement de centres de clusters, basé sur la recherche des échantillons les plus éloignés de chaque cluster puis calculant la moyenne pour chaque nième échantillon le plus éloigné. La moyenne calculée pour chaque nième groupe d'échantillons le plus éloigné. Avec ce travail approche, le problème du choix des clusters initiaux

sera résolu. Après avoir choisi les meilleures graphes initiales, le principal processus de recherche et de polarisation des centres de graphes et d'allocation des échantillons au cluster le plus proche a lieu. Le processus principal aura lieu entre 3 et 5 itérations.

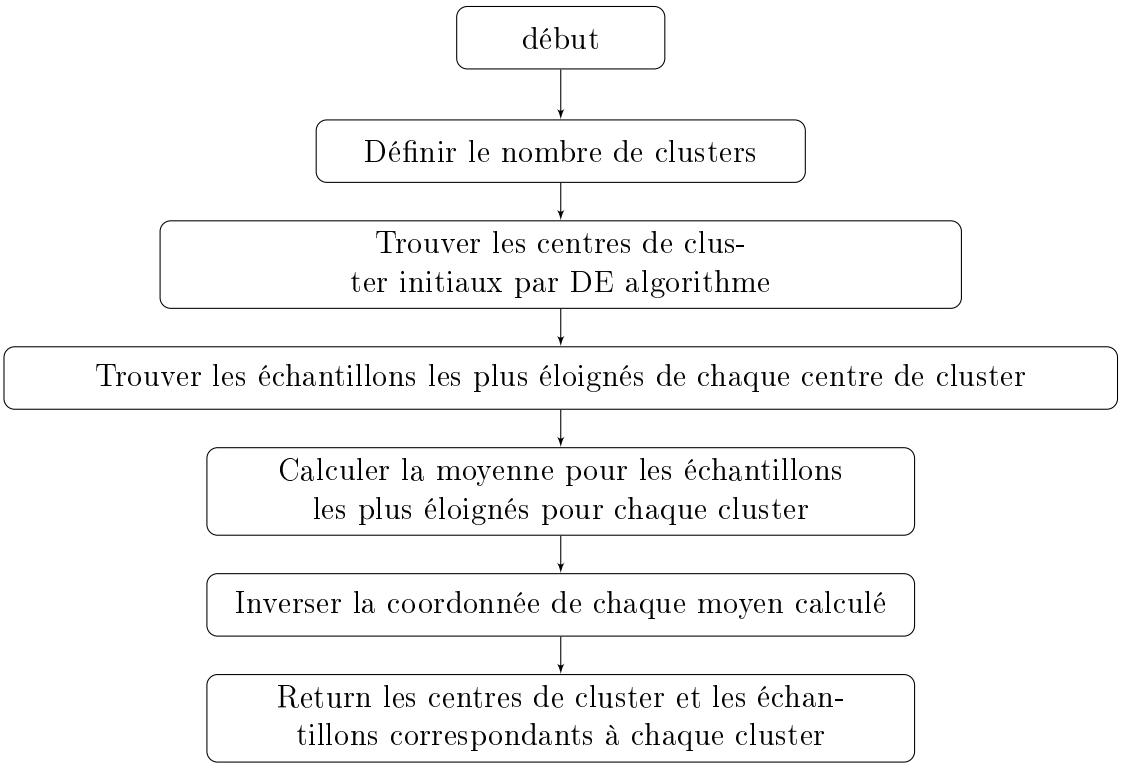

FIGURE 3.5 – L'organigramme de DE avec Clustering

3.5.4 Exemple d'application

Fisher iris :
Les iris de Fisher" sont des données proposées en 1933 par le statisticien Ronald Aylmer Fisher comme données de référence pour l'analyse discriminante et la classification.
Pour tester notre application on à prand 251 échantillons qui a represent par :
x :c'est un vecteur.
y :lable.

Data_User_Modeling_Dataset

251x6 double

	1	2	3	4	5	6
1	0	0	0	0	0	1
2	0.0800	0.0800	0.1000	0.2400	0.9000	4
3	0.0600	0.0600	0.0500	0.2500	0.3300	2
4	0.1000	0.1000	0.1500	0.6500	0.3000	3
5	0.0800	0.0800	0.0800	0.9800	0.2400	2
6	0.0900	0.1500	0.4000	0.1000	0.6600	3
7	0.1000	0.1000	0.4300	0.2900	0.5600	3
8	0.1500	0.0200	0.3400	0.4000	0.0100	1
9	0.2000	0.1400	0.3500	0.7200	0.2500	2
10	0	0	0.5000	0.2000	0.8500	4
11	0.1800	0.1800	0.5500	0.3000	0.8100	4
12	0.0600	0.0600	0.5100	0.4100	0.3000	2
13	0.1000	0.1000	0.5200	0.7800	0.3400	3

FIGURE 3.6 – Les données

3.6 Résultats de l'application

Le résultat du DE avec clustering

CR	N.Iteration				
		[2,8]	[10,15]	[17,28]	[36,50]
0.2	DEclust	40.6391	39.5779	38.8599	38.4451

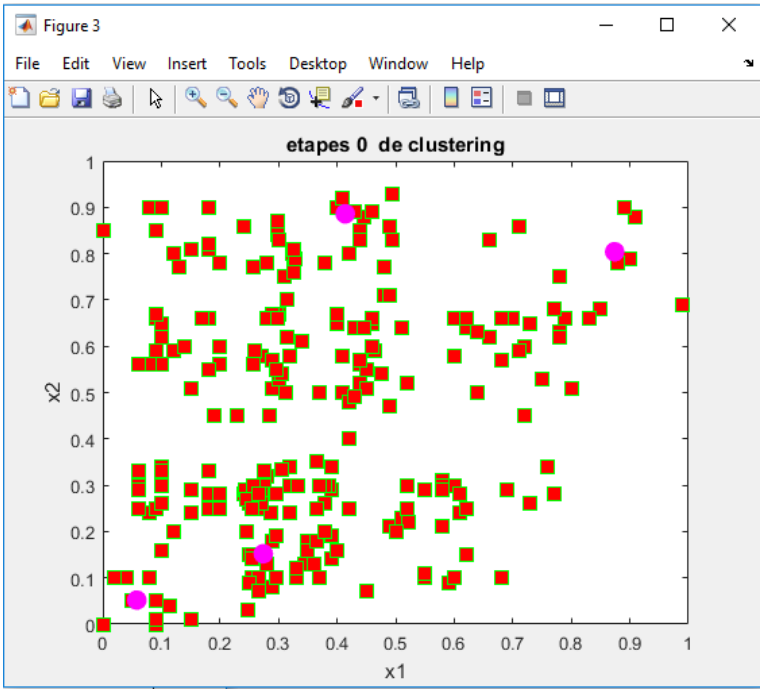

FIGURE 3.7 – étape 0 de clustering

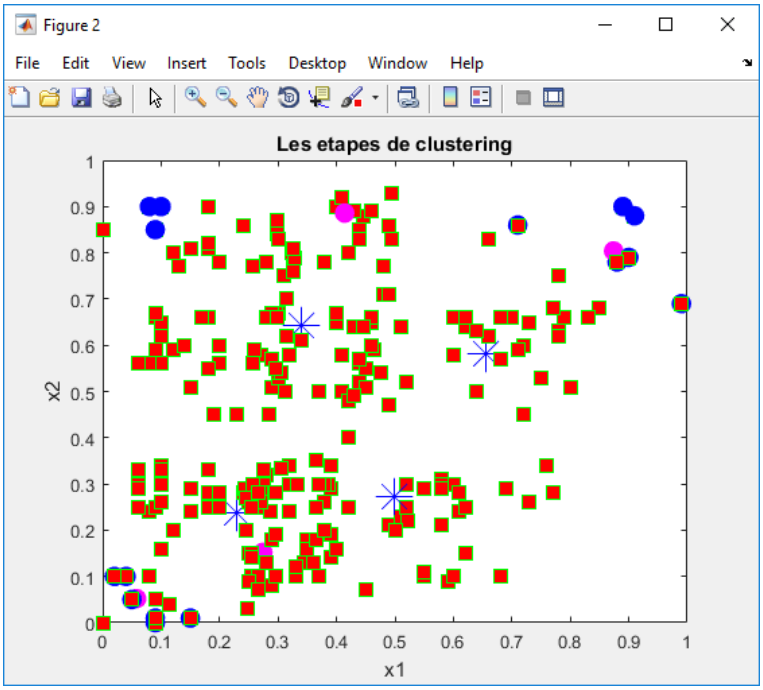

FIGURE 3.8 – les étapes de clustering

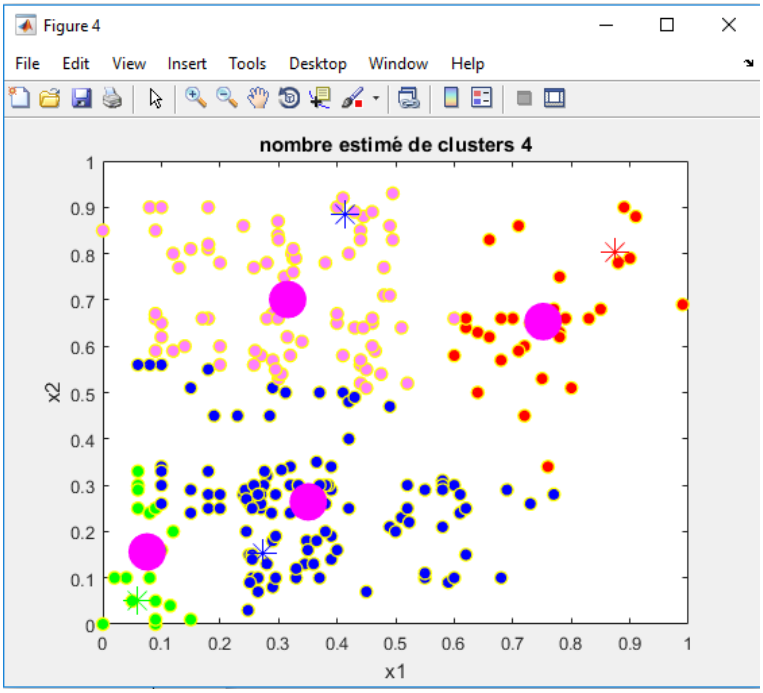

FIGURE 3.9 – nombre estimé de clusters 4

3.7 Comparaison entre DE multi-objectif et DE avec clustering

Dans le cadre de validation des résultats obtenus en combinant de DE avec clustering, on va appliquer la méthode de base DE multi-objectif et comparer les derniers résultats de DE avec clustering avec le DE multi-objectif. D'après l'application de DE multi-objectif donnée les résultats suivants :

Résultat DE multi-objectif :

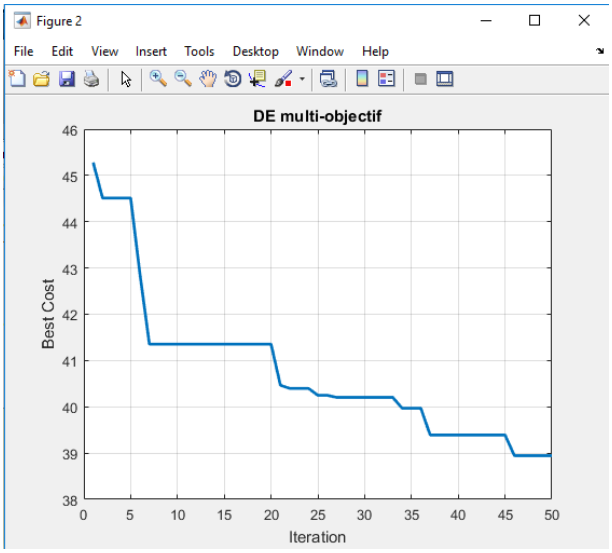

FIGURE 3.10 – DE multi-objectif

Conclusion générale et perspectives :

Dans ce travail, nous avons étudié une méthode hybride basée sur l'algorithme à évolution differentielle (DE) qui consiste à donner l'ensemble des solutions initiales puis ajouter la méthode de clustering (k-means) pour avoir l'ensemble des solutions finales. Cette dernière méthode appelée DE avec clustering, et à travers l'application sur un problème bi-objectifs non linéaire (apprentissage automatique) on teste les résultats obtenus.

perspectives :

Dans le futur, nous proposons ;

- Utiliser autre méthode métaheuristiques.

Bibliographie

[1] Abbas EL DOR,*Perfectionnement des algorithmes d'Optimisation par Essaim Particulaire.Applications en segmentation d'images et en électronique*,p.18,5/12/2012.

[2] Apprentissage automatique (Machine Learning)
`https://blogs.msdn.microsoft.com/big_data_france/2013/04/24/apprentissage-automatique-machine-learning/`

[3] D. Applegate, R. Bixby, V. Chvàtal et W. Cook, On the Solution of Travelling Salesman Problems, Documenta Mathematica, Extra Volume ICM III, p. 645-656, 1998.

[4] L. Chang, C. Liao, W. B. Lin, L.-L. Chen, et X. Zheng , A Hybrid method based on differential evolution and continuous ant colony optimization and its application on wideband antenna design, Progress In Electromagnetics Research, Vol. 122, p. 105-118, 2012.

[5] CHELOUTI Sara, KAIDI Karima,*Résolution d'un problème d'optimisation multi-objectif fractionnaire Linéaire flou en nombres entiers*,Université M'hamed Bougara Boumerdes,p.16-17,2015/2016.

[6] C. Cotta, A study of hybridisation techniques and their application to the design of evolutionary algorithms. AI Communications 11(3–4),p. 223–224, 1998.

[7] C. Cotta et E.G. Talbi, Parallel hybrid metaheuristics, In Alba, E., ed. : Parallel Metaheuristics, a New Class of Algorithms. John Wiley p. 347-370, 2005.

[8] D.E. Goldberg, Genetic Programming in Search, optimization and machine learning, Addison Wesley, 1989.

[9] Dr. LEMOUARI Ali,*Introduction aux Métaheuristiques*,Université de Jijel,p.1-5,2014.

[10] D. Duvidier, Etude de l'hybridation des méta-heuristiques, application à un problème d'ordonnancement de type jobshop, Thèse de Doctorat, université du littoral France, décembre 2000.

[11] H. Feltl et G.R. RaidI, An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem, in Proceedings of the ACM Symposium on Applied Computing, Nicosia, Cyprus, ACM Press, p. 990-995, 2004.

[12] J.J. Grefenstette, Incorporating problem specific knowledge into genetic algorithms, L.Davis (Ed.) Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, p.42-60, 1987.

[13] C. Grosan et A. Abraham, Hybrid Evolutionary Algorithms : Methodologies, Architectures, and Reviews, Studies in Computational Intelligence (SCI) 75, p. 1–17, 2007.

[14] I. Rechenberg, Evolutions strategies, Friedrich Frommann Verlag (Gunther Holzboog KG), Stuttgart, 1973.

[15] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[16] J. H. Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnick. Master's thesis, Technical University of Berlin, 1965.

[17] J. H. Schwefel. Numerical Optimization of Computer Model. Wiley 1981.

[18] Jingfeng Yan, Chaofeng Guo, Wenyin Gong, *Hybrid Differential Evolution with Convex Mutation*,p.2323-2324, 2011.

[19] J. Koza, Genetic Programming, on the Programming of Composants by Means of Natural Selection, MIT press, Cambridge, MA, 1992.

[20] J. Lampinen and I. Zelinka.*"Mixed Integer-Discrete-Continuous Optimization By Differential Evolution - Part 1 : the optimization method"*. In : Proceedings of MENDEL'99 -5th International Mendel Conference on Soft Computing.

[21] L'apprentissage automatique (Machine Learning), comment ça marche ? `https://blogs.msdn.microsoft.com/big_data_france/2014/06/05/lapprentissage-automatique-machine-learning-comment-a-marche/`

[22] L.J. Fogel, A.J. Owens, and M. J. Walash. Artificial Intelligence through Simulated Evolution, New York : John Wiley, 1966.

[23] Mast. Mandar Pandurang Ganbavale , *Differentia Evolution algorithm for structural optimization using Matlab*,2013- 2014.

[24] Machine Learning :Introduction à l'apprentissageautomatique-Supinfo, `https://www.supinfo.com/articles/single/6041machine-learningintroductionapprentissageautomatique`

[25] Makhtar MBAO,*Distance Sémantique et Carte Conceptuelle*,p.19-23,2007.

[26] Mast. Mandar Pandurang Ganbavale , *Differentia Evolution algorithm for structural optimization using Matlab*,2013- 2014.

[27] P. Millie et T. Radha, DE-PSO : A New hybrid meta-heuristic for solving global optimization problems, New Mathematics and Natural Computation vol. 7, no. 3, p. 363-381, 2011.

[28] Mohamed Oulmahdi,*Algorithmes Evolutionnaires dans les Systèmes de Parole*,p.26,Master recherche informatique 2011,

[29] Mokhtar TAFFAR ,*INITIATION A L'APPRENTISSAGE AUTOMATIQUE* ,Université de Jijel,2010.

[30] Mr LABED SAID, *Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes*,p.56-70 , 2013.

[31] Mr Mint,*Tout ce que vous voulez savoir sur l'algorithme K-Means*,`https://mrmint.fr/algorithmekmeans`

[32] H. Muhlenbein, M. Gorges-Schleuter et O. Kramer, Evolution algorithms in combinatorial optimization. Parallel Computing 7 :p. 65-85, 1988.

[33] Oussama EL GERARI,*Contributions à l'Amélioration des Techniques de la Programmation Génétique et de la Programmation Évolutionnair*,p.8-11,2011.

[34] J. Puchinger et G.R. Raidl, Combining metaheuristics and exact algorithms in combinatorial optimization : A survey and classification. In : Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, Part II. Vol. 3562 of LNCS., p. 41-53, Springer 2005.

[35] R. Storn and K. Price,*"Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces"*. Journal of Global Optimization, Vol. 11, No. 4, p. 341–359, 1997.

[36] R. Storn,K. Price,*"Differential Evolution - A Simple and Efficient adaptive scheme for global Optimization over Continuous Spaces"*.Technical Report TR 95-012,Berkley,USA :International Computer Science Institute,1995.

[37] D. Swagatam, A. Ajith et A. Konar, Particle Swarm Optimization and Differential Evolution Algorithms : Technical Analysis, Applications and Hybridization Perspectives, Advances of Computational Intelligence in Industrial Systems, Ying Liu et al. (Eds.), Springer Verlag, Germany, 2008.

[38] Talbi E.G., A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), p.541-565, 2002.

[39] Talbi, E.-G. Metaheuristics : From Design to Implementation, Wiley 2009.

[40] D. L. Woodruff, A Chunking Based Selection Strategy for Integrating Metaheuristics with Branch and Bound, in Metaheuristics : Advances

and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers,p.499-511,1999.

[41] Z. Xiangyin, D. Haibin et J. Jiqiang DEACO : Hybrid Ant Colony Optimization with Differential Evolution. IEEE Congress on Evolutionary Computation, p. 921-927, 2008.

[42] Y. Bengio, J-F. Paiement, and P. Vincent. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. Technical Report 1238, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.

[43] Y. Bengio, P. Vincent, J-F. Paiement, O. Delalleau, M. Ouimet, and N. Le Roux. Spectral Clustering and Kernel PCA are Learning Eigenfunctions. Technical Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.