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

Université des Sciences et de la Technologie Houari Boumediene
Faculté des Mathématiques

N° d'ordre :36/2012-M/HT

MÉMOIRE
PRÉSENTÉ POUR L'OBTENTION DU DIPLÔME DE MAGISTER EN MATHÉMATIQUES

SPÉCIALITÉ RECHERCHE OPÉRATIONNELLE : MATHÉMATIQUE DE GESTION

Par : Ali ZAIDI

Optimisation sur l'ensemble des solutions efficaces pour quelques problèmes d'optimisation combinatoire

Soutenue publiquement le 26/04/2012 Devant le jury composé de :

M^{r}	MOULAI Mustapha	Professeur	à l'USTHB Président .
M^{r}	CHAABANE Djamel	Professeur	à l'USTHB Directeur de mémoire.
M^{r}	CHERGUI Med El Amine	Maitre de conférence/A	à l'USTHB Membre
M^{me}	MERAZEKA Fatiha	Maitre de conférence/A	à l'USTHB Membre.

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Introduction générale

L'une des principales missions de la recherche opérationnelle est d'aider à prendre des décisions en vue d'une gestion efficace, rationnelle et logique. La modélisation traditionnelle des problèmes de recherche opérationnelle n'a connu que des modèles d'optimisation à objectif unique jusqu'à l'apparition d'une nouvelle vision plus proche de la réalité et qui consiste à modéliser les problèmes de sorte que tous les objectifs pertinents soient pris en considération. Cette philosophie de multiplicité d'objectifs semble très raisonnable et contribue avec une efficacité très forte à l'élaboration de nouvelles méthodes pratiques de gestion plus réalistes que les méthodes classiques.

L'optimisation à objectifs multiples est sans aucun doute un domaine de recherche important pour les scientifiques et les ingénieurs, non seulement en raison de la nature multi-objectifs de la plupart des problèmes réels, mais également parce qu'il reste beaucoup de questions en suspens dans ce secteur. En recherche opérationnelle, plusieurs techniques ont été développées au cours des années en vue de traiter des problèmes à objectifs multiples ; beaucoup d'approches différentes ont été suggérées, allant d'une combinaison naïve des objectifs en un seul 'a l'utilisation de la théorie des jeux pour coordonner l'importance relative de chaque objectif et le comportement de différents intervenants.

Parmi les problèmes de l'optimisation multi-objectifs, les problèmes d'optimisation combinatoire multi-objectifs et cette catégorie regroupe une large classe de problèmes ayant des applications dans de nombreux domaines applicatifs.

Un problème d'optimisation combinatoire multi-objectif est défini par un ensemble fini de solutions discrètes D et plusieurs fonctions objectifs Z (z1, z2,. . .. . .,zp) associant à chaque solution une valeur pour chaque fonction objectif. Ainsi, un problème d'optimisation combinatoire consiste en l'optimisation (minimisation ou maximisation) d'un certain critère sous différentes contraintes permettant de délimiter l'ensemble des solutions réalisables (ou solutions admissibles).

Dans certaines situations , les décideurs n'ont pas besoin de l'ensemble des solutions efficaces d'un problème de programmation multi-objectifs mais uniquement de solutions efficaces qui réalisent l'optimum d'un objectif différent des objectifs déjà fixés.

Ceci nous mène vers la recherche de la solution optimale d'un critère sur l'ensemble des

solutions efficaces du problème multi-objectifs.

Dans cette thèse, nous sommes fixé comme objectif d'étudier l'un des problèmes classique d'optimisation combinatoire qu'est le problème de voyageur de commerce TSP, mais dans cette thèse nous optimisons un objectif principale sur l'ensemble des solutions efficaces de problème de voyageur de commerce multi-objectifs MOTSP. Et suite à la complexité de problème nous développons une heuristique basée sur des Algorithmes génétiques pour résoudre ce problème.

Cette thèse comporte cinq chapitres. Le premier chapitre expose le cadre général du travail. Il contient un rappel des notions fondamentales de la théorie de l'optimisation multi-objectif et de multi-objectifs combinatoire.

Le deuxième chapitre est entièrement consacré aux méthodes de résolution des problèmes multi-objectifs, nous avons présenté quelques méthodes et quelques algorithmes de résolution des problèmes multi-objectifs.

Le troisième chapitre est entièrement consacré aux algorithmes génétiques, nous avons détaillé tous les principes et toutes caractéristiques de ces algorithmes.

Dans le quatrième nous avons présenté quelques méthodes d'optimisation d'un critère sur l'ensemble des solutions efficaces.

Dans le cinquième chapitre, nous avons présenté notre algorithme et notre application. Dans ce chapitre nous avons fait une analyse de résultats et nous avons présenté un exemple illustratif.

Enfin, une brève conclusion résume les principaux acquis de cette thèse et ouvre des perspectives de recherche prolongeant le travail accompli.

1

Panorama sur l'optimisation combinatoire multi-objectifs MOCO.

Contents

Présenté par : Ali ZAIDI *Sous la direction de : CHAABANE Djamel, Professeur à l'USTHB*

1.1 Aide à la décision multi-critères

A maintes reprises dans la vie quotidienne, on se retrouve confronté à des problèmes nécessitant une prise de décision tenant compte du contexte dans lequel on se situe, des informations dont on dispose et de nos préférences. Ces prises de décision peuvent intervenir dans des circonstances plutôt anodines, dont les conséquences sont peu significatives (décider entre l'achat d'un pain complet ou aux céréales dans une boulangerie par exemple), mais elles interviennent aussi très fréquemment dans des circonstances dont l'enjeu est important (le choix d'une maison à acheter par exemple), voire primordial (les décisions prises par les traders peuvent entraîner des pertes ou des gains conséquents, comme on a pu le voir récemment). Ainsi, si dans le premier cas on peut estimer qu'un individu peut prendre sa décision sans aucune aide, il est facile de se convaincre qu'une assistance paraît appropriée, voire nécessaire, lorsque les enjeux sont importants. Notons toutefois que l'on ne considère dans ces travaux que des situations suffisamment complexes pour que la décision ne soit pas triviale.

Le domaine de l'aide à la décision s'est alors naturellement installé pour définir des modèles de décision formels dans l'intention d'aider un individu ou un ensemble d'individus (que l'on appelle décideur) à décider. Cette étude formelle de modèles décisionnels a notamment fait émerger deux difficultés majeures, communes à l'ensemble des travaux menés en aide à la décision :

- la *structuration*, qui consiste à concevoir un modèle représentatif de la réalité
- la *résolution*, qui explore l'espace des possibles afin d'établir une recommandation.

1.2 Problèmes d'optimisation multi-objectif

Dans les 30 dernières années, la plupart des travaux ont porté sur la programmation multi-objectif linéaire en variables continues. Les raisons principales de cet intérêt sont d'une part le développement de la programmation linéaire uni-objectif en recherche opérationnelle, et la facilité relative de traiter de tels problèmes, et d'autre part l'abondance des cas pratiques qui peuvent être formulés sous forme linéaire. Ainsi, un certain nombre de logiciels ont vu le jour depuis le développement de la méthode du simplexe multi-objectif [88].

1.2.1 Applications académiques

La plupart des benchmarks utilisés dans la comparaison d'algorithmes d'optimisation multi-objectif ont été réalisés sur des problèmes académiques. Les problèmes les plus étudiés sont :

✓ optimisation des fonctions continues : dans la communauté "Algorithmes Evolutionnaires", les fonctions de Schaffer sont souvent utilisées pour évaluer et comparer les performances des algorithmes.

✓ optimisation combinatoire : plusieurs problèmes classiques d'optimisation combinatoire ont été étudiés dans une version multi-objectif [14] : sac à dos [2], ordonnancement [51], plus court chemin dans un graphe [81], arbre recouvrant (minimum spanning tree) [89], affectation [76], voyageur de commerce [67], routage de véhicules [56], etc.

1.2.2 Applications industrielles

La plupart des travaux portant sur l'optimisation combinatoire multi-objectif concernent des applications industrielles. Plusieurs problèmes ont été traités dans différents domaines :

- design de systèmes dans les sciences d'ingénieurs (mécanique, aéronautique, chimie, etc.) : ailes d'avions [54], moteurs d'automobiles [29] ;
- finances : investissements financiers, etc ;
- ordonnancement et affectation : ordonnancement en productique [68], localisation d'usines, planification de trajectoires de robots mobiles [28], etc ;
- agronomie : programme de production agricole, etc ;
- transport : gestion de containers [75], design de réseaux de transport [27], tracé autoroutier, etc ;
- environnement : gestion de la qualité de l'air [45], distribution de l'eau [34], etc ;
- télécommunications : design d'antennes [79], affectation de fréquences [15] [83], etc.

1.3 Optimisation combinatoire multi-objectifs MOCO

1.3.1 Formulation et définitions

Un problème d'optimisation combinatoire multi-objectif peut être formulé sous la forme :

$$(MOCO)\begin{cases} & \min f(x) = z = (f_1(x) = z_1, ..., f_p(x) = z_p) \\ x \in C & \\ s.c.C & \text{ensemble fini de } \mathbb{R}^n \end{cases} .$$

- C est appelé espace de décision (ou ensemble réalisable).
- Le vecteur f des fonctions objectifs est appelé vecteur critère.
- L'espace $Z = \{z = f(x) \mid x \in C\}$, est appelé espace objectif.
- Un vecteur $z \in Z$est appelé point de Z.
- Le vecteur

$$\overline{z} = [\min_{x \in C} f_1(x), ..., \min_{x \in C} f_m(x)]$$

est appelé point ideal.

Il est à noter que le point ideal est irréalisable en general (i.e. ; il est rare qu'on trouve $\overline{x} \in C : \overline{z} = f(\overline{x})$. Ceci est du, en partie, au fait que les objectifs a optimiser sont souvent conflictuels. De ce fait, la relation d'ordre totale, utilisée dans le cas mono-objectif

pour mesurer la qualité des solutions, n'est plus satisfaisante dans le cas multi-critères. D'ou l'introduction de la relation de dominance (voir la sous-section suivante) qui est une relation d'ordre partielle visant a distinguer les solutions réalisant le meilleur compromis possible entre tous les objectifs du problème.

1.4 Relations d'ordre et de Dominance

Comme la solution optimale est une multitude de points de R^p, il est vital pour identifer ces meilleurs compromis de définir une relation d'ordre entre ces éléments. Dans le cas des problèmes d'optimisation multi-objectifs, ces relations d'ordre sont appelées relations de dominance. Plusieurs relations de dominance ont déjà été présentées : la a-dominance Othmani,1998, la dominance au sens de Géoffrion [22], la cone-dominance (Collette and Siarry, 2002), . . .

Mais la plus célèbre et la plus utilisée, c' est la dominance au sens de Pareto.(Au XIX éme siècle, Vilfredo Pareto, un mathématicien italien, formule le concept : $\ll$ dans un problème multi-objectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorera au moins un des autres critères$\gg$. C'est cette relation de dominance que nous allons définir et utiliser dans ce travail. De maniéré à définir clairement et formellement cette notion, les relations $=; \leq et <$ usuelles sont entendues aux vecteurs.

1.4.1 Vocabulaire et définitions

Définition 1.1 Soient u et v, deux vecteurs de même dimension :

$$\begin{cases} u = v, & ssi \quad \forall i \in \{1, 2, ..., m\}; \quad u_i = v_i; \\ u \leq v, & ssi \quad \forall i \in \{1, 2, ..., m\}; \quad u_i \leq v_i\,; \\ u < v, & ssi \quad u \leq v \bigwedge u \neq v. \end{cases}$$

Les relations $\geq$ et $>$, sont définies de manière analogue.

Les relations définies précédemment ne couvrent pas tous les cas possibles. En effet, il est impossible de classer les points a = (1 ; 2) et b = (2 ; 1) à l'aide d'une de ces relations. Contrairement aux problèmes à un seul objectif, où les relations usuelles $<;\leq;$..... suffisent pour comparer les points, elles sont insuffisantes pour comparer des points issus des problèmes multi-objectif. Nous définissons donc maintenant la relation de dominance au sens de Pareto, permettant de prendre en compte tous les cas de figures rencontrés lors de la comparaison de deux points (ici des vecteurs).

Définition 1.2 La dominance au sens de Pareto : Considérerons un problème

de minimisation. Soient u et v deux vecteurs de décision :

$$\begin{cases} u < v, & (u \quad domine & v) \quad ssi \quad f(u) < f(v); \\ u \leq v, & (u \quad dominefaiblement & v) \quad ssi \quad f(u) \leq f(v); \\ u \sim v, & (u \quad estincomparable(ounon - domine)avec & v) \quad ssi \quad f(u) \nleq f(v) \bigwedge f(v) \nleq f(u) \end{cases}$$

Définition 1.3 Une solution $x \in \chi_f$ est dite non dominée par rapport à un ensemble $\chi_a \subseteq \chi_f$ si et seulement si :

$$\nexists X_a \subseteq \chi_a, \quad X_a \prec X;$$

Illustrons maintenant cette relation par un exemple en dimension 2 (cf. figure 1.1). Sur cette figure, $\mathcal{F}$ (l'espace réalisable dans l'espace des objectifs) est l'image de χ. Ainsi, chaque point y^i est l'image de x^i par $f : y^i = f(x^i)$. Prenons le point y^1 comme point de référence. Nous pouvons distinguer trois zones :

- La zone de préférence : est la zone contenant les points dominés par y^1,
- La zone de dominance : est la zone contenant les points dominant y^1,
- La zone d'incompatibilité : contient les points incomparables avec y^1.

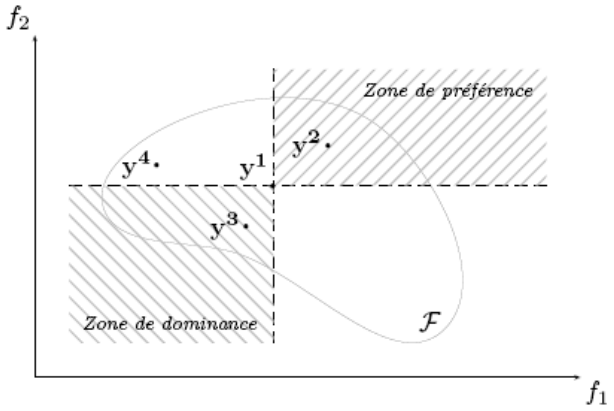

FIG. 1.1 – Exemple de dominance .

Ainsi, il est clair que y^2 est dominé par y^1 (y^1 est préféré à y^2), que y^3 domine y^1 (y^3 est préféré à y^1), et que y^4 est non dominé (incomparable) avec y^1.

Forts de ce nouvel outil, nous pouvons maintenant définir l'optimalité dans le cas de problèmes multi-objectif. Nous pouvons définir les concepts d'optimalité globale et locale au sens de Pareto (c.à.d utilisant la dominance au sens de Pareto) :

Définition 1.4
Un vecteur de décision $x \in \chi$ est dit Pareto globalement optimal si et seulement si : $\nexists y \in \chi, y \prec x$. Un vecteur de décision $x \in \chi$ est dit Pareto optimal

Définition 1.5

Un vecteur de décision $x \in \chi$ est dit localement optimal si et seulement si, pour un $\delta > 0$ fixé : $\nexists y \in \chi; f(y) \in B(f(x); \delta)$ et $y \prec x$, où $B(f(x); \delta)$ représente une boule de centre $f(x)$ et de rayon δ.

La figure 5.4 donne un exemple en dimension 2 d'optimalité locale. Le point $f(x)$ est localement optimal, car il n'y a pas de point compris dans la boule B le dominant.

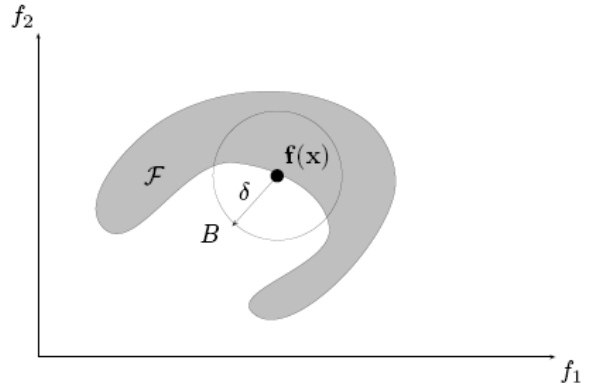

FIG. 1.2 – Exemple d'optimalité locale.

Définition 1.6
$\widetilde{x} \in C$ est une solution efficace ou optimale au sens de Pareto si et seulement s'il n'existe pas de $x \in C$ tel que $z_k(x) \leq z_k(\widetilde{x}), \forall k = 1, ..., p$ avec au moins une inégalité stricte (c-à-d. $\nexists x \in C : f(x) < f(\widetilde{x})$).

1.5 Classification des méthodes et approches de résolutions

Dans la littérature, une attention particulière a porté sur les problèmes à deux critères en utilisant les méthodes exactes tel que le "branch and bound" [66], l'algorithme A* (Stewart and White, 1991), et la programmation dynamique [84]. Ces méthodes sont efficaces pour des problèmes de petites tailles. Pour des problèmes à plus de deux critères ou de grandes tailles, il n'existe pas de procédures exactes efficaces, étant donné les difficultés simultanées de la complexité NP-complet, et le cadre multi-critères des problèmes.

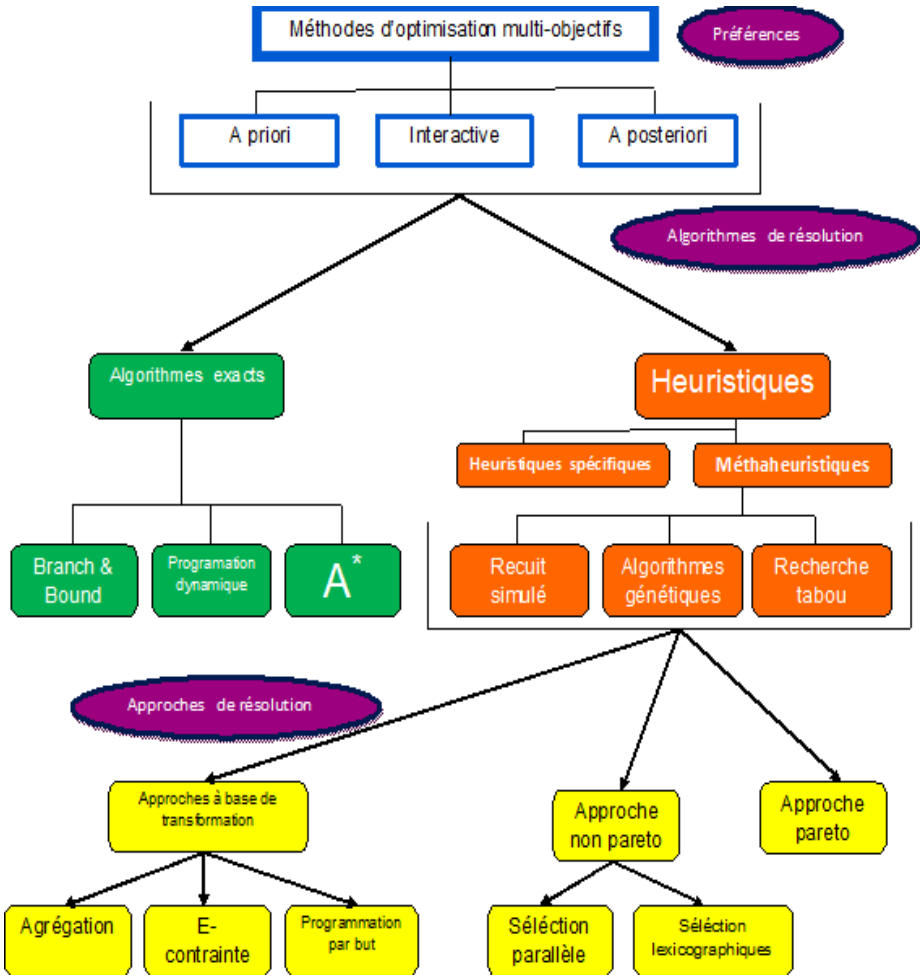

FIG. 1.3 – Classification des méthodes d"optimisation multi-objectif.

Les approches utilisées pour la résolution de PMO peuvent être classées en trois catégories :

1.5.1 Approches transformant le problème en un ou plusieurs problème(s) mono-objectif(s) :

ces approches transforment le problème initial afin de se ramener à la resolution de un ou plusieurs problèmes mono-objectif. Parmi ces méthodes, on peut citer les méthodes d'agrégation [49, 71], les méthodes . $\varepsilon - contrainte$ [1]ou encore les méthodes utilisant un vecteur cible [11, 85]. En général ces méthodes nécessitent une connaissance du problème et ne fournissent qu'une seule solution. Elles peuvent alors . être classées dans la famille des méthodes d'optimisation a priori.

1.5.1.1 Méthode d'agrégation :

C'est l'une des premières méthodes utilisée pour la génération de solutions Pareto optimales. Elle consiste à transformer le problème (PMO)en un problème (PMO_λ)qui revient à combiner les différentes fonctions coût $f_i$du problème en une seule fonction objectif F généralement de façon linéaire [36]

$$F(x) = \sum_{i-1}^{n} \lambda_i f_i(x)$$

où les poids $\lambda_i \in [0..1]$et $\sum_{i-1}^{n} \lambda_i = 1$. Différents poids fournissent différentes solutions supportées.

1.5.1.2 Méthode ε-contrainte :

Dans cette approche, le problème consiste à optimiser une seule fonction objectif f_k sujette à des contraintes sur les autres fonctions objectif.

$$(PMO_k(\varepsilon)) \begin{cases} \min f_k(x) \\ x \in C \\ s.c. f_j(x) \leq \varepsilon_j \quad j = 1..n, \quad j \neq k. \end{cases}$$

où $\varepsilon = (\varepsilon_1, ..., \varepsilon_{k-1}, \varepsilon_{k+1}, .., \varepsilon_n)$. Ainsi, un problème uni-objectif (objectif f_k) sujet à des contraintes sur les autres objectifs est résolu. Différentes valeurs de $\varepsilon_i$peuvent être données pour pouvoir générer différentes solutions Pareto optimales. La connaissance a priori des intervalles appropriés pour les valeurs $\varepsilon_i$est requise pour tous les objectifs. Pour pouvoir définir les valeurs adéquates pour ε_i, le vecteur idéal doit être calculé pour déterminer les bornes inférieures. On aura donc :

$$\varepsilon_i \geq f(x^*), i = 1, 2, k-1, k+1, n$$

théorème 1 :[Chankong et Haimes 1983.]Supposons que C et f sont convexes. Si, pour un certain k, x^* est solution de $PMO_k(\varepsilon)$, il existe alors λtel que x^* est solution de PMO_λ,

et inversement.

Plusieurs hybridations de ces deux modèles ont été proposées. Ci-dessous un exemple de modèle combinant le modèle PMO_λet le modèle $PMO_k(\varepsilon)$:

$$(PMO_k(\varepsilon)) \begin{cases} \min F(x) = \sum_{i-1}^{n} \lambda_i f_i(x) \\ s.c.x \in C \\ f_j(x) \leq \varepsilon_j \qquad\qquad\qquad , j = 1..n. \end{cases}$$

1.5.1.3 Programmation par but :

Dans cette méthode, le décideur doit définir des buts ou références qu'il désire atteindre pour chaque objectif. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème uni-objectif. Par exemple, la fonction coût peut intégrer une norme pondérée qui minimise les déviations par rapports aux buts. Le problème peut être formulé de la manière suivante :

$$(PMO_k(\varepsilon)) \begin{cases} \min(\sum_{j-1}^{n} \lambda_j \mid f_j(x) - z_j \mid^p)^{\frac{1}{p}} \\ s.c.x \in C \end{cases}$$

où $1 \leq p \leq \infty$, et z est le vecteur de référence (but) ou le vecteur idéal. La norme utilisée est la métrique de Tchebycheff (Lp-métrique). Généralement p est égal à 2 ; dans ce cas on a une métrique euclidienne. Si p=1, l'équation revient à une fonction Min-Max. Une sélection arbitraire du vecteur de référence peut ne pas être désirable, étant donné qu'un mauvais choix du vecteur de référence peut aboutir à une solution qui n'est pas Pareto optimale.

1.5.1.4 Analyse de la première classe de méthodes :

la transformation du problème multi-objectif en un problème uni-objectif nécessite des connaissances a priori du problème traité. L'optimisation d'un problème uni-objectif peut garantir l'optimalité Pareto de la solution trouvée mais trouve une seule solution. Dans les situations réelles, les décideurs ont généralement besoin de plusieurs alternatives. En effet, si certains objectifs sont bruités ou des données sont incertaines, ces méthodes ne sont pas efficaces. Ils sont aussi sensibles au paysage de la frontière Pareto (convexité, discontinuité, etc.). L'autre inconvénient de ces méthodes est leur sensibilité par rapport aux poids, aux contraintes ou aux vecteurs de référence. Les solutions obtenues dépendent fortement de ces paramètres. Pour différentes situations, différents paramètres sont utilisés, et le problème doit être résolu plusieurs fois, d'où le coût associé à cette classe d'algorithmes, qui nécessitent parfois l'optimisation indépendante de chacun des objectifs. Dans l'exécution multiple des algorithmes, on espère trouver différentes solutions Pareto optimales.

Dans les deux classes suivantes des méthodes,ou ces difficultés peuvent être éliminées.

Une seule résolution du problème permet de trouver un ensemble de solutions Pareto optimales. Le décideur peut ainsi choisir une solution suivant la situation courante. La connaissance de plusieurs solutions Pareto optimales est utile aussi pour une utilisation future, particulièrement quand la situation change et une nouvelle solution est requise.

1.5.2 Approches Non Pareto :

ces approches transforment le problème d'origine. Elles effectuent leur recherche en traitant indépendamment chacun des objectifs. L'exemple le plus classique est l'algorithme VEGA *(Vector Evaluated Genetic Algorithm)*. Ces méthodes ont souvent du mal à trouver les solutions de compromis puisqu'elles se focalisent sur les portions extrêmes du front. Nous pouvons classer dans cette catégorie les méthodes lexicographiques qui donnent un ordre de priorité sur les objectifs à traiter.

1.5.2.1 Sélection parallèle dans les algorithmes évolutionnaires :

Le premier travail consistant à utiliser les AGs pour résoudre des PMO est celui de Schaffer (Schaffer, 1985). L'algorithme développé VEGA (Vector Evaluated Genetic Algorithm) sélectionne les individus de la population courante suivant chaque objectif, indépendamment des autres (sélection parallèle). A chaque génération, la population est donc divisée en un nombre de sous-populations qui est égal au nombre d'objectifs de la fonction coût. Chaque sous-population i est sélectionnée suivant l'objectif f_i. L'algorithme VEGA compose la population complète, et applique les opérateurs génétiques (mutation, crossover).

Comme l'algorithme affecte aléatoirement les individus aux sous-populations à chaque génération, un même individu peut être évalué différemment suivant l'objectif qui lui est affecté, d'une génération à l'autre. L'analyse de l'algorithme VEGA a montré que son comportement est le même qu'un algorithme réalisant une agrégation linéaire [62]. Malgré cet effet, Schaffer a obtenu de bons résultats dans la recherche des solutions Pareto optimales dans l'optimisation de fonctions réelles. Cependant, il y a une certaine tendance à ignorer les solutions "compromis".

L'algorithme VEGA a été utilisé et amélioré par plusieurs auteurs. Surry et Radcliffe l'ont utilisé dans la modélisation des contraintes d'un problème uni-objectif pour éviter l'utilisation des pénalités dans la fonction coût [39]. Leur algorithme traite les contraintes comme des objectifs. La procédure standard de VEGA a été modifiée pour introduire une forme de "ranking" basée sur le nombre de contraintes violées par un individu. D'autres opérateurs de sélection parallèle ont aussi été proposés comme le "n-branch tournament" [72].

1.5.2.2 Sélection lexicographique :

Dans cette approche, la sélection est réalisée suivant un ordre définie par le décideur. Cet ordre définit l'ordre d'importance des objectifs. Plusieurs métaheuristiques ont été utilisées pour résoudre des PMO en se basant sur la sélection lexicographique :

- **algorithmes génétiques :** Fourman a proposé une méthode de sélection dans un AG qui est basée sur l'ordre lexicographique. La sélection est réalisée en comparant des paires d'individus ; chaque paire d'individus est comparée suivant l'objectif avec la plus grande priorité. Si le résultat est le même pour cet objectif, alors l'objectif avec la deuxième priorité est utilisé, etc. Comme dans l'algorithme VEGA, ceci correspond à faire une agrégation implicite avec des poids correspondant aux probabilités de choix de chaque objectif ;
- **stratégies evolutionists :** une autre approche basée sur une sélection lexicographique a été proposée par Kursawe en utilisant les stratégies evolutionists[42]. Les individus sont comparés suivant un objectif, qui est choisi de façon aléatoire suivant une probabilité pré-déterminée.

Les méthodes non-Pareto de sélection parallèle et lexicographique reviennent à considérer à chaque génération la moyenne des valeurs associées aux différents objectifs. Dans la stratégie de Schaffer, le résultat attendu correspond, en effet, à une combinaison linéaire des objectifs avec des poids variables suivant la distribution des individus de la population courante (Richardson et al., 1989). La stratégie de Fourman, quant à elle, correspond à faire une moyenne des rangs et non pas des valeurs des fonctions objectifs. Dans les deux cas, des valeurs différentes peuvent être affectées aux individus non-dominés.

1.5.2.3 Reproduction multi-sexuelle :

L'utilisation de plusieurs classes d'individus dans un algorithme à base de populations, chaque classe étant associée à un objectif à l'aide d'une fonction bijective, a été proposée dans (Allenson, 1992). Chaque classe d'individus est identifiée par un sexe différent ; un individu d'un sexe donné est évalué par rapport à la fonction objectif associée à son sexe. Dans la résolution d'un problème bi-critères, deux sexes différents, male et femelle, sont associées aux deux objectifs. La reproduction est permise seulement entre un individu male et un individu femelle, le sexe étant affecté aléatoirement à la création d'un individu. A l'initialisation, le nombre d'individus par sexe est le même ; ce qui n'est pas le cas au cours des générations suivantes. L'auteur utilise les algorithmes évolutionnaires pour implementer une forme d'attracteurs sexuels.

Lis et Eiben [44] ont généralisé cette approche dans les AGs. L'opérateur de crossover a été modifié pour porter sur plusieurs individus et non pas sur deux individus seulement comme dans l'AG standard. La sélection d'un parent de chaque sexe (objectif) permet ainsi à tous les sexes de contribuer à la génération d'un nouvel individu. L'individu créé aura le sexe du parent dont il hérite le plus grand nombre de gènes, et réalise un compromis entre les différents critères. Les individus sont évalués suivant la fonction objectif qui est associé à leur sexe. L'opérateur de mutation est restreint pour éviter à un individu de changer de

sexe.

L'utilisation des reproductions multi-sexuelles est une manière de définir des sous-populations séparées par rapport à chaque objectif. La différence avec la sélection parallèle et lexicographique est que la reproduction impose une restriction de voisinage, évitant la reproduction aléatoire entre individus.

L'inconvénient majeur de ces approches est qu'elles tendent à générer des solutions qui sont largement optimisées pour certains objectifs, et l'inverse pour les autres objectifs. Les solutions "compromis" sont négligées.

1.5.3 Approches Pareto :

ces approches utilisent la notion de dominance pour comparer les solutions entre elles. Une seule résolution permet d'approximer l'ensemble de la frontière Pareto. L'un des premiers à discuter de l'intérêt de l'utilisation de la notion de dominance pour la recherche de solutions a été Goldberg .

1.6 Structure du front Pareto

L'objectif est donc de fournir aux décideurs un ensemble (le plus complet possible) de solutions Pareto, afin qu'ils puissent ensuite choisir les solutions qui les intéressent le plus. Une question se pose donc sur la nature de ces solutions Pareto et la nécessité de les obtenir toutes. Une étude de la frontière Pareto doit donc être réalisée.

1.6.1 Front minimal / front maximal complet

La définition de front se réfère à l'espace des objectifs. Une solution appartient au front si elle n'est pas dominée par aucune autre solution réalisable.

Lorsque deux solutions ont exactement les mêmes valeurs pour l'ensemble des objectifs, elles sont équivalentes dans l'espace objectif, mais peuvent correspondre à deux solutions différentes dans l'espace décisionnel. Une question importante est de savoir s'il est intéressant de garder ces deux différentes solutions.

La réponse peut dépendre du contexte (type de problème étudié) en plus de la volonté des décideurs :

- Lors de la résolution d'un problème comportant énormément de solutions Pareto, il est peut être préférable de privilégier une bonne approximation de l'ensemble de la frontière et donc favoriser la diversité (du côté objectif) des solutions retenues.
- Au contraire, lorsque la frontière Pareto comporte peu de solutions, afin d'avoir une bonne représentation de l'ensemble des solutions non dominées, il sera intéressant de

rechercher les solutions de même valeur.
Nous parlerons alors de recherche du **front minimal**, dans le premier cas, et du **front maximal complet**, dans le second.

1.6.2 Solutions supportées / non supportées

Sur le front Pareto, deux types de solutions peuvent être différenciées : les solutions supportées et les solutions non supportées. Les premières sont celles situées sur l'enveloppe convexe de l'ensemble des solutions (voir Fig 1.4) et peuvent donc être trouvées à l'aide d'une agrégation linéaire des objectifs [3]. Elles sont donc plus simples à obtenir que les solutions non supportées. D'ailleurs, les premiers travaux en optimisation combinatoire multi-objectif se sont pour la plupart focalisés sur la recherche de ces solutions supportées en optimisant des combinaisons linéaires des objectifs utilisant différents vecteurs de poids.

Alors pourquoi ne pas se satisfaire des solutions supportées ? Tout d'abord parce que ces solutions peuvent ne représenter qu'un petit sous-ensemble des solutions efficaces. De plus, ces solutions supportées ne sont pas forcément bien réparties le long du front et ne représentent pas toujours un bon compromis. La figure 1.5 nous montre l'exemple d'un problème de flowshop bi-objectif où les deux seules solutions supportées sont des solutions extrêmes. Donc, si l'on veut obtenir des solutions de bon compromis entre les objectifs, il est nécessaire de considérer les solutions Pareto non supportées.

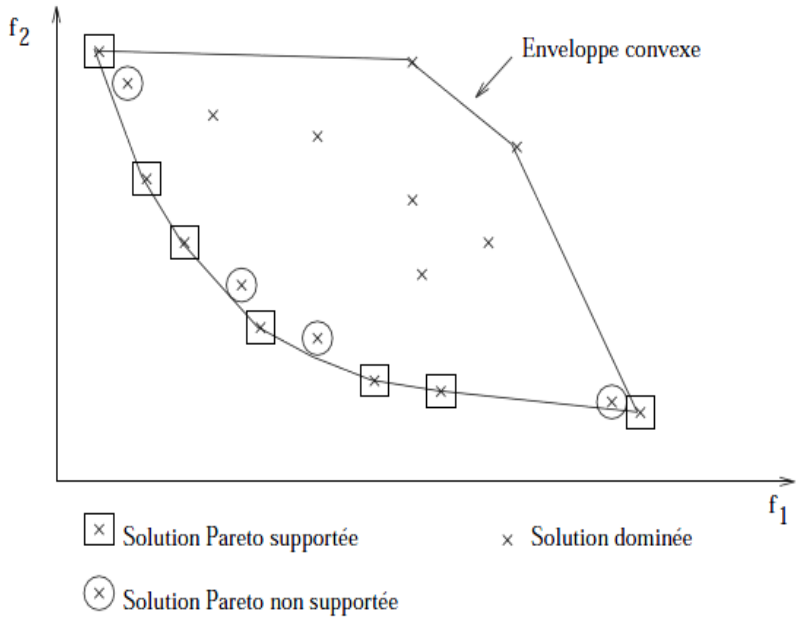

FIG. 1.4 – Représentation des différents types de solutions en bi-objectif.

1.6.3 Stratégie à adopter

Ainsi, en fonction du problème sous étude, et de la connaissance de la structure de sa frontière Pareto, différentes stratégies peuvent être adoptées.

1. **La structure du problème est bien connue et sa frontière a été montrée convexe :** dans ce cas, toutes les solutions Pareto sont supportées. Il est donc possible de rechercher ces solutions en utilisant des agrégation linéaires d'objectifs.
2. **La structure du problème est bien connue et sans être convexe, il est montré que les solutions supportées sont bien réparties :** si l'objectif est d'obtenir un ensemble de solutions représentatif du front Pareto (sans nécessairement obtenir toutes les solutions), alors il est encore possible de ne rechercher que les solutions supportées, à l'aide d'aggregations d'objectifs.
3. **La structure du problème est pas ou mal connue, ou les solutions supportées sont mal réparties sur le front :** afin de ne prendre aucune hypothèse sur la répartition des solutions sur le front, il est nécessaire de rechercher à la fois les solutions supportées et non supportées. De plus, dans un environnement non stable, la connaissance de plusieurs solutions Pareto optimales permet de mieux appréhender les aléas en permettant parfois de changer le choix de la solution en fonction d'un changement de condition sans avoir à réaliser une nouvelle recherche de solution.

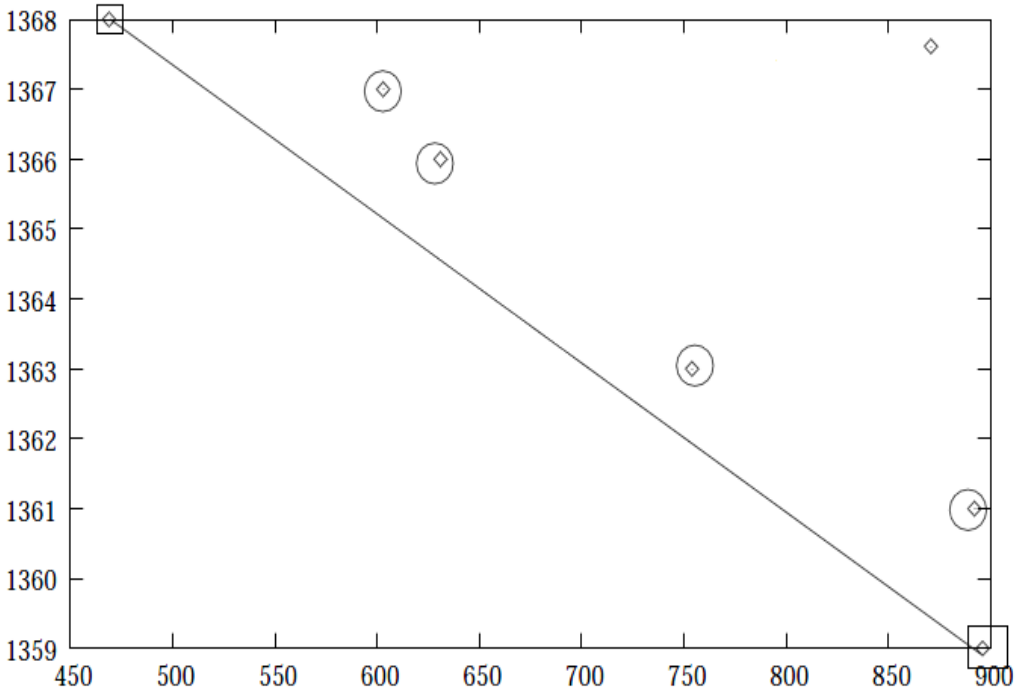

FIG. 1.5 – Exemple de l'importance des solutions non supportées.

Les deux premiers cas cit´es sont bien évidemment les plus favorables. Malheureusement ils nécessitent une bonne connaissance du problème sous étude et en particulier une connaissance de la caractérisation de sa frontière Pareto. Cette caractérisation est très difficile à réaliser car il ne faut pas se focaliser sur quelques instances étudiées, mais bien sur des propriétés générales vérifiées par l'ensemble des instances d'un problème.

Ainsi, pour éviter de nous restreindre à quelques types de problèmes, nous nous placerons dans le cas où la structure du problème est mal connue ou non favorable. Notre objectif sera alors de rechercher l'ensemble des solutions supportées et non supportées.

1.7 Algorithmes de résolution

1.7.1 Algorithmes exacts

Il existe peu de méthodes exactes destinées à la recherche de l'ensemble Pareto-optimal complet pour un problème de type MOCO. De plus, la plupart de ces méthodes ne résolvent pratiquement que des problèmes bi-critères de petite taille. Vu que la majorité de telles méthodes ont été conçues spécifiquement pour des problèmes particuliers, nous nous bornons ici à décrire deux algorithmes ayant des schémas plus génériques. Plus de détails sur les fondements théoriques (cf., outils mathématiques, preuves de convergence) des algo-

rithmes exacts peuvent être trouvés dans la référence [24].

1.7.1.1 L'algorithme à deux phases :

Cet algorithme a été proposé par Ulungu et Teghem [77]pour résoudre des problèmes bi-critères de type MOCO. Leur algorithme comporte deux phases de résolution. La première phase consiste à chercher toutes les solutions Pareto-optimales supportées alors que la deuxième phase cherche, de façon indépendante, les solutions Pareto-optimales non-supportées. Ainsi la première phase est relativement facile puisque, pour trouver les solutions supportées, il suffit d'optimiser une combinaison linéaire des deux objectifs du problème. Quant à la deuxième phase, elle nécessite plus de travail car elle concerne les solutions non-supportées qui ne peuvent pas être obtenues par une technique d'agrégation linéaire. Ulungu et Teghem proposent alors d'utiliser les solutions supportées, trouvées à l'issue de la première phase, pour réduire l'espace de recherche. Pour ce faire, ils se sont basés sur un argument géométrique qu'ils ont établi théoriquement et dont le principe est le suivant : les solutions non-supportées sont forcément dans les triangles rectangulaires basés sur deux solutions supportées consécutives (voir Figure 1.6). Ainsi, une recherche de type deuxième phase est exécutée entre chaque couple de solutions supportées adjacentes. La méthode de recherche adoptée durant la deuxième phase dépend du problème étudié. En général, la deuxième phase utilise un schéma de recherche arborescente par séparation et évaluation (Branch and Bound) [23] en exploitant la structure particulière du problème à résoudre pour rendre la recherche plus efficace. La méthode à deux phases semble intéressante

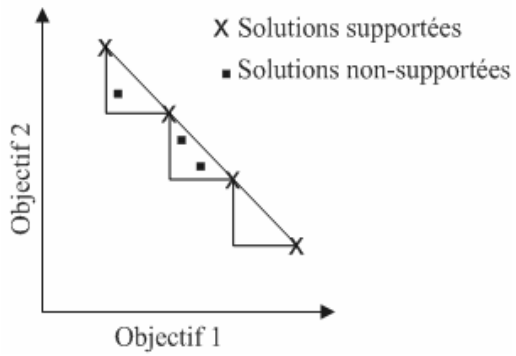

FIG. 1.6 – Solutions supportées et non-supportées pour un problème combinatoire bi-objectif.

dans le sens où elle propose un schéma général pour la résolution exacte des problèmes bi-objectifs de type MOCO. Toutefois, l'application de cette méthode à un problème bi-objectif donné nécessite l'existence d'un algorithme efficace pour la version mono-objectif de ce problème11. Il est à noter que la méthode à deux phases a été appliquée avec succès au problème d'affectation bi-critère [77]. Des améliorations de cette méthode ont été proposées et appliquées sur le problème d'affectation bi-critère [60] et celui de flowshop bi-critère [43].

Néanmoins, ces améliorations restent moins générales que la méthode originale puisqu'elles tiennent amplement compte des spécificités des problèmes étudiés.

1.7.1.2 L'algorithme de recherche dichotomique :

L'approche de recherche dichotomique dans un contexte MOCO bi-objectif propose un schéma évolué d'application de la technique d'agrégation linéaire tout en permettant d'obtenir des solutions non-supportées [17]. Le principe sous-jacent de cette approche consiste à explorer de façon dichotomique des intervalles du front Pareto de plus en plus petit. L'algorithme commence d'abord par chercher les solutions extrêmes12. Ensuite, étant données deux solutions extrêmes r et s, une recherche est menée entre r et s suivant une direction perpendiculaire à la droite (r, s). En interdisant de ré-explorer les solutions r et s et en éliminant les solutions dominées par ces deux solutions, la méthode trouve la meilleure solution relativement à cette direction. Ce qui rend la méthode plus intéressante c'est que la solution trouvée pourra être non-supportée. Cette nouvelle solution génère deux nouveaux intervalles qui seront explorés de la même façon que précédemment.

Cette méthode reste intéressante, surtout pour les problèmes bi-objectifs dont le nombre de solutions est raisonnable. Toutefois, elle peut devenir fastidieuse pour les problèmes bi-objectifs dont la taille de l'ensemble Pareto-optimal est large. Ceci est dû au fait qu'elle nécessite, à peu près, 2^n recherches si la taille du front Pareto est n.

1.7.2 Heuristiques et métaheuristiques

Du fait que tous les algorithmes exacts sous-entendent une énumération explicite ou implicite de toutes les solutions, de tels algorithmes deviennent alors inefficaces dès que la taille des problèmes (exemple., nombre de variables, nombre de fonctions objectifs, nombre de contraintes) augmente de façon considérable. En outre, une grande partie des problèmes de type MOCO sont NP-difficiles [55] [24] dans le sens où il n'existe pas (jusqu'à présent !) d'algorithme(s) exact(s) capable(s) de les résoudre en un temps polynomial en fonction de leur taille. Pour illustrer de manière pratique cette problématique, considérons le fameux problème de voyageur de commerce (Traveling Salesman Problem ou TSP). Rappelons que le TSP consiste à trouver le plus court chemin reliant n villes en supposant que chaque ville doit être visitée une et une seule fois. Le nombre de solutions pour ce problème est égal à $n!$ solutions. Rien que pour un nombre de villes $n = 25$, l'énumération de toutes les solutions possibles ne demande pas moins de 6 siècles comme temps de calcul ! (en considérant la capacité des ordinateurs actuels). Ainsi, afin de procurer rapidement des solutions de bonne qualité mais qui ne sont pas nécessairement Pareto-optimales, des méthodes approchées (i.e. ; heuristiques et métaheuristiques) ont été mises au point. Dans l'absence de définitions précises et normalisées pour les concepts d'heuristique et de métaheuristique, on peut se contenter des définitions suivantes :

- Une *heuristique* est une méthode approchée conçue pour un problème particulier dans le but de fournir des solutions de bonne qualité durant un temps de calcul raisonnable.
- Une *métaheuristique* est un schéma de calcul heuristique, général et adaptable à un ensemble de problèmes différents.

Ainsi, les métaheuristiques sont des stratégies heuristiques génériques dont l'adaptation à un problème donné génère un ensemble d'heuristiques spécifiques pour ce problème. La plupart des méthodes métaheuristiques ont été inspirées de phénomènes naturels (i.e. ; physiques et surtout biologiques) [19]. D'autre part, on peut partager les métaheuristiques en deux grandes classes : les métaheuristiques à solution unique (i.e. ; évoluant avec une seule solution) et celles à solution multiple (i.e. ; évoluant avec plusieurs solutions).

2
Méthodes de résolution

Introduction

Qu'ils comportent un ou plusieurs objectifs, les problèmes d'optimisation sont en général difficiles à résoudre. De plus, le temps de calcul nécessaire à leurs résolutions peut devenir si important que l'algorithme développé devient inutilisable en pratique. Il n'existe pas d'algorithme générique capable de résoudre efficacement toutes les instances de tous les problèmes. De nombreuses de méthodes ont été développées pour tenter d'apporter une réponse satisfaisante à ces problèmes. Parmi celles-ci, nous distinguons les méthodes dédiées à un problème précis et les méthodes plus génériques pouvant s'appliquer à une catégorie de problèmes.

Contents

Dans ce chapitre, nous essayons de faire un état d'art sur les différentes méthodes d'optimisation multi-objectifs, on distingue deux grandes classes à savoir les méthodes exactes et les méthodes approchées.

Les méthodes exactes examinent, souvent de manière implicite, la totalité de l'espace de recherche. Ainsi, elles ont l'avantage de produire une solution optimale lorsqu'aucune contrainte de temps n'est donnée. Néanmoins, le temps de calcul nécessaire pour atteindre une solution optimale peut devenir vite prohibitif, et ce malgré les diverses techniques et heuristiques qui ont été développées pour accélérer l'énumération des solutions.

Dans de telles situations (et dans beaucoup d'autres), les méthodes approchées constituent une alternative indispensable et complémentaire. Le but d'une telle méthode n'est plus de fournir une solution optimale au problème donné. Elle cherche avant tout à produire une solution sous-optimale de meilleure qualité possible avec un temps de calcul raisonnable. En général, une méthode approchée examine seulement une partie de l'espace de recherche.

A partir de deux méthodes de résolution différentes, il est possible de concevoir des méthodes hybrides dans le but de fournir des résultats supérieurs à ceux des deux méthodes qui les composent.

2.1 Schéma de méthodes

Il existe un nombre important de méthodes, que certains auteurs les classifient en cinq groupes :[13]

- Méthodes scalaires ;
- Méthodes interactives ;
- Méthodes floues ;
- Méthodes exploitant une métaheuristique ;
- Méthodes d'aide à la décision.

Les méthodes de ces cinq groupes peuvent aussi être rangées en trois familles de méthodes d'optimisation multi-objectifs[80] :

- **Méthodes à préférence a priori**
  Dans ces méthodes, l'utilisateur définit le compromis qu'il désire réaliser (il fait part de ses préférences) avant de lancer la méthode d'optimisation. On retrouve dans cette famille la plupart des méthodes par agrégation (où les fonctions objectif sont fusionnées en une seule).
- **Méthodes à préférence progressive**
  Dans ces méthodes, l'utilisateur affine son choix de compromis au fur et à mesure du

déroulement de l'optimisation. On retrouve dans cette famille les méthodes interactives.

- **Méthodes à préférence a posteriori**
  Dans ces méthodes, l'utilisateur choisit une solution de compromis en examinant toutes les solutions extraites par la méthode d'optimisation. Les méthodes de cette famille fournissent, à la fin de l'optimisation, une surface de compromis.

Il existe des méthodes d'optimisation multi-objectifs qui n'entrent pas exclusivement dans une de ces familles. Par exemple, on peut utiliser une méthode à préférence a priori en lui fournissant des préférences choisies au hasard. Le résultat sera alors un grand nombre de solutions qui seront présentées à l'utilisateur pour qu'il décide de la solution de compromis. Cette combinaison forme alors une méthode à préférence a posteriori. [13] D'autres types de classement classifient les méthodes d'optimisation multi-objectifs selon les principes mathématiques, (utilisation de les dérivées des fonctions objectif ou non).

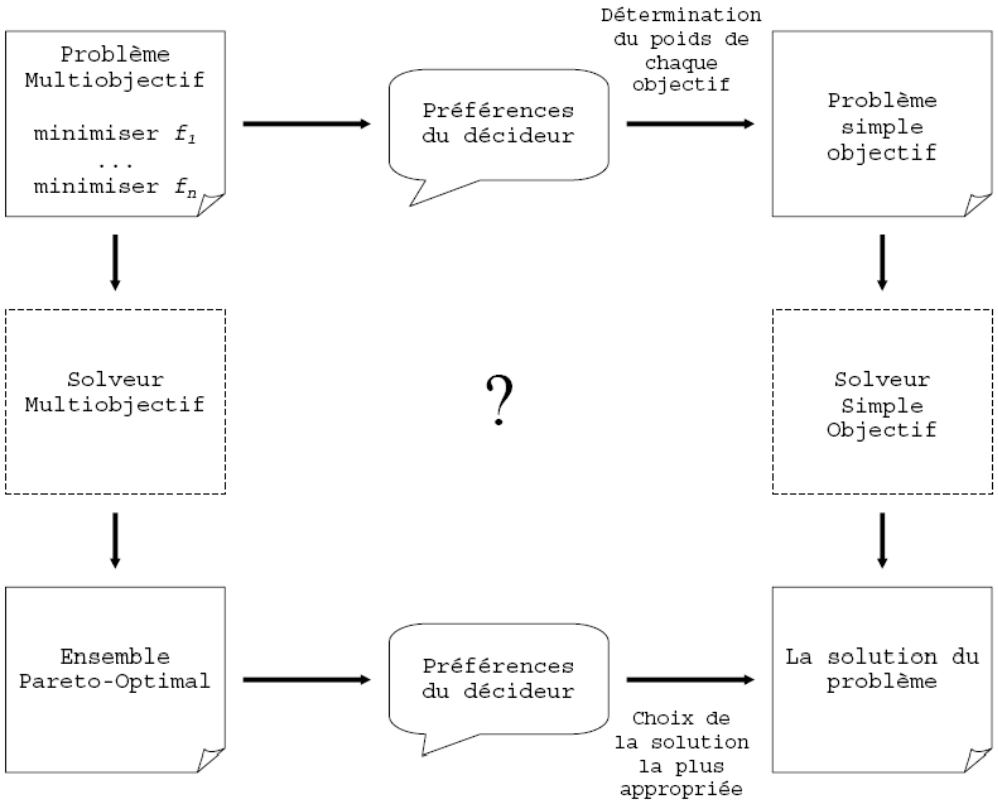

FIG. 2.1 – Schéma de résolution

2.2 Méthodes scalaires

2.2.1 La méthode de pondération de fonctions objectif

Cette approche de la résolution d'un problème d'optimisation multi-objectifs est la plus évidente. D'ailleurs, on appelle aussi cette méthode "approche naïve" de l'optimisation multi-objectifs [Coello [33]]. Le but, ici, est de revenir à un problème d'optimisation mono-objectif, dont il existe de nombreuses méthodes de résolution. La manière la plus simple de procéder consiste à prendre chacune des fonctions objectif, à leur appliquer un coefficient de pondération et à faire la somme des fonctions objectif pondérées. On obtient alors une nouvelle fonction objectif [13].

$$(p) \left\| \begin{array}{ll} \text{Minimiser} & f_{eq}(x) = \sum_{i=1}^{k} w_i.f_i(x) \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & h(x) = 0 \end{array} \right.$$

Souvent, les coefficients de pondération respectent la relation suivante : $w_i \geq 0$ pour tous $i \in \{1, \ldots, k\}$ et $\sum_{i=1}^{k} w_i = 1$.

2.2.2 La méthode de Keeney-Raiffa

Cette méthode utilise le produit des fonctions objectif pour se ramener à un problème d'optimisation monobjectif. L'approche utilisée est semblable à celle utilisée dans la méthode de pondération des fonctions objectif. La fonction objectif ainsi obtenue s'appelle la fonction d'utilité de Keeney-Raiffa.

$$\left\| \begin{array}{ll} \text{Minimiser} & f_{eq}(x) = \prod_{i=1}^{k} w_i.f_i(x) \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & h(x) = 0 \end{array} \right.$$

On retrouve cette fonction dans la théorie de la fonction d'utilité multi-attribut exposée dans [Keeney et al. [41]] (M.A.U.T, Multi Attribute Utility Theory). Cette théorie, issue des sciences d'aide à la décision, traite des propriétés et de la manière de créer une "fonction d'utilité".[13]

2.2.3 La méthode de la distance à un objectif de référence

Cette méthode permet de transformer un problème d'optimisation multi-objectifs en un problème d'optimisation monobjectif.

La somme que l'on va utiliser ici prendra la forme d'une distance ($\sqrt[k]{()^k}$) où la racine $k^{\text{ème}}$ de la somme d'éléments élevée à une certaine puissance k). De plus, dans certains cas on normalisera les éléments par rapport à une valeur avant d'en faire la somme [Azarm [5], Miettinen [48]].

On part encore du problème P. Dans les definitions suivantes, le vecteur F (de coordonnées $F_i, i \in \{1, \ldots, k\}$)) est un vecteur dont les coordonnées correspondent à celles d'un objectif idéal (ou de reference) au sens défini par l'utilisateur (cela peut être le point ideal, ou un autre point qui représente mieux les préférences de l'utilisateur). Le vecteur F est choisi par l'utilisateur. On peut, par exemple, utiliser la distance absolue. Voici la definition de cette distance :

$$L_r\big(f(x)\big) = \big[\sum_{i=1}^{k} |F_i - f_i(x)|^r\big]^{\frac{1}{r}}$$

avec $0 \leq r \leq \infty$

2.2.4 Méthode de compromis (approche par ϵ-contrainte)

Une autre façon de transformer un problème d'optimisation multi-objectifs en un problème monobjectif est de convertir $m-1$ des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant [13].

La démarche est la suivante :

- On choisit un objectif à optimiser prioritairement ;
- On choisit un vecteur de contraintes initial ;
- On transforme le problème conservant l'objectif prioritaire et on transforme les autres objectifs en contraintes d'inégalité.

On appelle aussi cette méthode la méthode de la ϵ-contrainte [Miettinen [48]]. Le problème peut être reformulé de la manière suivante [6] :

$$\left\|\begin{array}{ll} \text{Minimiser} & f_i(x) \\ \text{tel que,} & f_1(x) \leq \epsilon_1 \\ & \vdots \\ & f_{i-1}(x) \leq \epsilon_{i-1} \\ & f_{i+1}(x) \leq \epsilon_{i+1} \\ & \vdots \\ & f_m(x) \leq \epsilon_m \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & x \in \mathbb{R}^n, f(x) \in \mathbb{R}^m, g(x) \in \mathbb{R}^q \end{array}\right.$$

L'approche par ϵ-contrainte doit aussi être appliquée plusieurs fois en faisant varier le vecteur ϵ pour trouver un ensemble de points Pareto optimaux.

Cette approche a l'avantage par rapport aux autres de ne pas être trompée par les problèmes non convexes. Ainsi la figure 2.2 illustre, en dimension 2, le cas où un point $(\epsilon; f_{1_{min}})$, de la partie non convexe, est trouvé. La figure 2.2 montre aussi comment cette approche procède. En transformant des fonctions objectifs en contraintes, elle diminue la zone réalisable par paliers. Ensuite, le processus d'optimisation trouve le point optimal sur l'objectif restant.

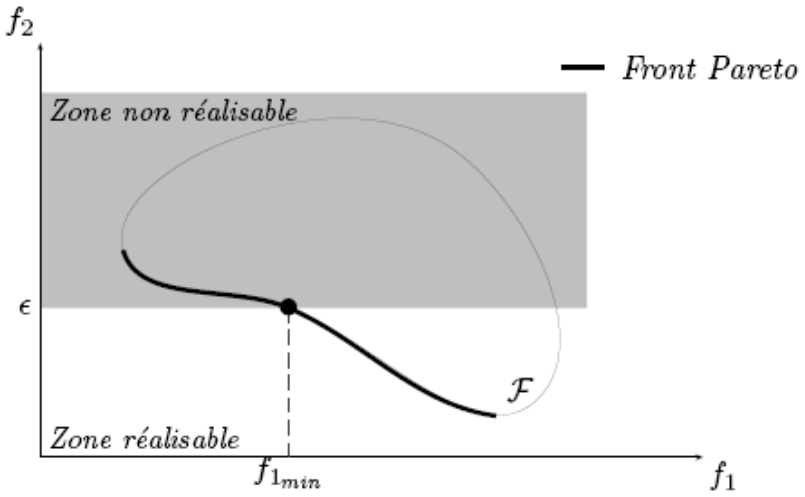

FIG. 2.2 – Interprétation graphique de l'approche par ϵ-contrainte.

L'inconvénient de cette approche réside dans le fait qu'il faille lancer un grand nombre de fois le processus de résolution. De plus, pour obtenir des points intéressants et bien répartis sur la surface de compromis, le vecteur ϵ doit être choisi judicieusement. Il est clair qu'une bonne connaissance du problème a priori est requise.

2.2.5 Méthode du but à atteindre (Goal attainment method)

Cette approche, comme celle de min-max, utilise un point de référence pour guider la recherche. Mais elle introduit aussi une direction de recherche, si bien que le processus de résolution devra suivre cette direction. À la différence de l'approche min-max, qui utilise des normes pour formaliser la distance au point de référence, l'approche du *but à atteindre* utilise des contraintes, à l'instar de l'approche ϵ-contrainte, pour déterminer la position du point de référence (aussi appelé le but). L'écart par rapport à ce but est contrôlé grâce à la variable λ introduite à cet effet :

$$\left\| \begin{array}{ll} \text{Minimiser} & \lambda \\ \text{tel que,} & f_1(x) - w_1.\lambda \leq B_1 \\ & \vdots \\ & f_m(x) - w_m.\lambda \leq B_m \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & x \in \mathbb{R}^n, f(x) \in \mathbb{R}^m, g(x) \in \mathbb{R}^q \end{array} \right.$$

Ainsi en minimisant λ et en vérifiant toutes les contraintes, la recherche va s'orienter vers le but B et s'arrêter sur le point A faisant partie de la surface de compromis (voir l'illustration en 2 dimensions à la figure 2.3. Cette approche permet, comme l'approche par ϵ-contrainte et l'approche min-max, de trouver les parties non convexes des fronts Pareto.

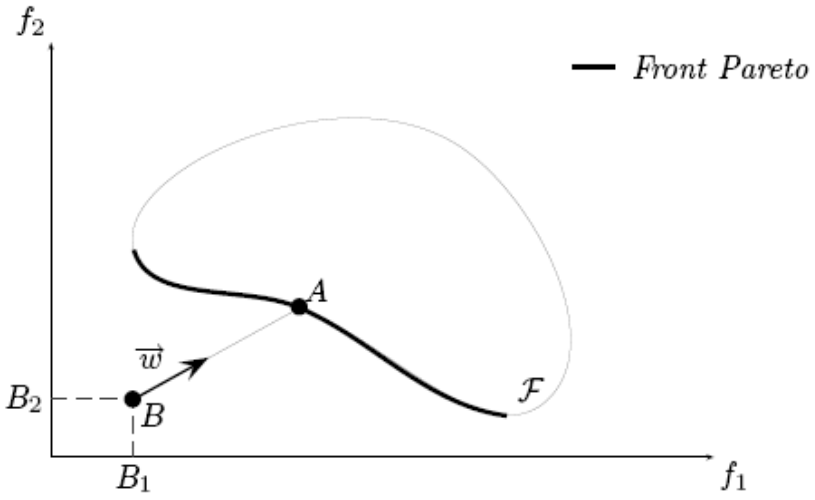

FIG. 2.3 – Interprétation graphique de l'approche par "but à atteindre".

Cependant, cette approche, comme les précédentes, doit être itérée plusieurs fois dans le but d'obtenir un ensemble de points Pareto optimaux. Les paramètres w et B doivent être bien choisis par l'utilisateur. Bien que ces paramètres permettent une grande flexibilité de la recherche (orientation et but), s'ils sont mal choisis, ils peuvent, dans certains cas extrêmes, donner des résultats non cohérents.

Il y a lieu noter que cette approche est très similaire à celle de la "goal programming" [Charnes and Cooper, 1961 ; Romero, 1991 ; Coello, 1998], où les contraintes deviennent des égalités. Des variables d'écart sont alors introduites [6].

2.2.6 Programmation par but (Goal programming method)

Cette méthode est proche de la méthode de but à atteinte. La différence principale est que, après avoir transformé le problème d'optimisation, nous avons des contraintes d'égalité au lieu des contraintes d'inégalité. Cette approche est la suivante [13] :

- Choisi un vecteur initial des fonctions objectif F,
- Associons à chaque objectif deux nouvelles variables "déviations" liées au vecteur initial des fonctions objectif choisies.
- minimiser après une des deux variables ; le choix de la variable se fait en fonction du type de dépassement que l'on veut (au-dessus ou au-dessous de l'objectif que l'on s'est fixé).

Présentation de la méthode On part du problème P. On choisit un vecteur de fonctions objectif initial $F \in \mathbb{R}^k$. On associe aussi un ensemble de variables d_i^+ et d_i^- à chaque fonction objectif $f_i(x), i \in \{1, \ldots, k\}$.

On obtient alors le problème suivant :

$$\left\|\begin{array}{ll} \text{Minimiser} & (d_1^+ \ \ ou \ \ d_1^-, \ldots, d_k^+ \ \ ou \ \ d_k^-) \\ \text{tel que,} & f_1(x) = F_1 + d_1^+ - d_1^- \\ & \vdots \\ & f_k(x) = F_k + d_1^+ - d_k^- \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & h(x) = 0 \end{array}\right.$$

Les variables de déviation que l'on cherche à minimiser doivent respecter certaines contraintes :
$d_1^+ \geq 0$ et $d_k^- \geq 0$ et $d_1^+ \cdot d_k^- = 0$ avec $i \in \{1, \ldots, k\}$

En fonction de la façon dont on désire atteindre le but F, on cherchera à minimiser différentes combinaisons des coefficients d_i^+ et d_i^-.

2.2.7 Méthode hybride

Comme nous allons le voir, il est possible d'exploiter plusieurs méthodes pour en former une nouvelle. Dans ce cas, nous obtenons une "méthode hybride".

Présentation de la méthode La méthode hybride la plus connue est la méthode de Corley. Cette méthode utilise la méthode de pondération des fonctions objectif et la méthode du compromis. On part du problème P. On le transforme de la manière suivante :

$$\left\|\begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i.f_i(x) \\ \text{tel que,} & f_j(x) \leq \varepsilon_j, j = 1, \ldots, k \\ \text{et que} & g(x) \leq 0 \\ \text{avec,} & h(x) = 0 \end{array}\right.$$

2.2.8 Méthode lexicographique

Cette méthode est très intuitive. En effet, elle consiste à considérer les fonctions objectif les unes après les autres et à minimiser un problème d'optimisation monobjectif, en complétant au fur et à mesure l'ensemble des contraintes [12].

Présentation de la méthode On part du problème P. On procède ensuite en k étapes (autant d'étapes qu'il y a de fonctions objectif). Commençons avec la première fonction objectif. On résout :

$$\left\|\begin{array}{ll} \text{Minimiser} & f_1(x) \\ \text{tel que,} & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array}\right.$$

On note f_1^* la solution de ce problème.

Ensuite, on transforme la première fonction objectif en contrainte d'égalité puis on

prend la seconde fonction objectif et on résout le problème :
On répète cette démarche jusqu'à la fonction objectif k. Et, pour finir, on aura :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_k(x) \\ \text{tel que} & f_1(x) = f_1^*, \ldots, f_{k-1}^* \\ & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array} \right.$$

La dernière valeur de x est celle qui minimise tous les objectifs.

D'autres méthodes existent appartenant à cette classe dans la littérature nous pouvons citer :

- La méthode des contraintes d'égalité propres ou Proper-equality-constraints-method ;
- La méthode des contraintes d'inégalité propres ou Proper-inequality-constraints-method ;
- L'algorithme de Lin-Tabak ;
- L'algorithme de Lin-Giesy ;
- etc.

2.3 Méthodes interactives

Les méthodes interactives permettent de chercher une et une seule solution. Elles forment la famille des méthodes progressives et permettent à l'utilisateur de déterminer ses préférences vis-à-vis d'un compromis entre objectifs au cours de l'optimisation. Ces méthodes sont à comparer aux méthodes :

▷ à préférence a priori, où l'utilisateur choisit le compromis à réaliser entre objectifs avant de lancer l'optimisation ;

▷ à préférence a posteriori, où l'utilisateur n'a pas à choisir de compromis avant de lancer l'optimisation. La méthode d'optimisation calcule toutes les solutions efficaces et l'utilisateur peut ainsi les comparer et en choisir une.[13]

Nous allons maintenant présenter quelques méthodes interactives (ou plutôt quelques principes d'interaction).

2.3.1 Méthode de compromis par substitution

Cette méthode (Surrogate Worth Tradeoff (SWT) ou "méthode du compromis par substitution") a été très utilisée pour l'optimisation de ressources en eau [Haimes et al. [33]]. Elle est basée sur la méthode du compromis, à laquelle on a ajouté un processus interactif, pour que la méthode converge vers la solution la plus susceptible de satisfaire l'utilisateur [13].

Algorithme 1 Surrogate Worth Tradeoff (SWT)-algorithm

Étape 1 Trouver le minimum de la fonction f_j en résolvant

$$\left\| \begin{array}{ll} \text{Minimiser} & f_j(x) \\ \text{sachant que} & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array} \right.$$

La solution de ce problème devient la $j^{\text{ème}}$ composante du vecteur f_{min}, qui regroupe les $k-1$ valeurs minimales de f_j, $j = 1, \ldots, k$. Si possible, on peut chercher à cette étape les composantes du vecteur f_{max}, qui regroupe les $k-1$ valeurs maximales de f_j, $j = 2, \ldots, k$.

Étape 2 Choisir la valeur de départ de $\epsilon \geq f_{j_{min}}$, $j = 2, \ldots, k$.

Étape 3 Résoudre le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_1(x) \\ \text{Minimiser} & f_2(x) \geq \epsilon_2, \ldots, f_k(x) \geq \epsilon_k \\ \text{sachant que} & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array} \right.$$

Si certaines des contraintes de ce problème ne peuvent pas être respectées, on pose $\epsilon_j = f_j(x)$ pour j correspondant aux indice des contraintes non respectées (on relâche les contraintes). Il faut ensuite recommencer cette étape. Si les contraintes sont respectées, on appelle x^* le vecteur solution et on passe à l'étape suivante.

Étape 4 Si les contraintes sont maintenant suffisamment relâchées, on passe à l'étape 5 sinon, on choisit une nouvelle valeur de $\epsilon_j \geq f_{j_{min}}$, pour tout $j = 2, \ldots, k$ et on retourne à l'étape 3.
Une méthode pour sélectionner une nouvelle valeur de ϵ consiste à commencer par une grande valeur de ϵ puis à diminuer chaque ϵ_j, pour tout $j = 2, \ldots, k$ d'un certain $\Delta_j > 0$, chaque fois que les contraintes sont respectées.

Si les contraintes ne sont pas respectées, on pose $\epsilon_j = f_j(x)$ avec j indice des contraintes non respectées (ici, x désigne la valeur courante de la solution).

Étape 5 On demande maintenant à l'utilisateur de choisir les coefficients W_{1j}, pour tout $j = 2, \ldots, k$. Ces coefficients représentent l'avis de l'utilisateur vis-à-vis d'un accroissement de la fonction f_j de λ_j unités (le mode de calcul de λ_{1j} est précisé plus loin). Ce coût est mesuré sur une échelle de -10 à +10, où -10 représente un avis défavorable de l'utilisateur vis-à-vis de cet accroissement et +10 un avis favorable. 0 représente l'indifférence.

Cette opération est répétée pour j variant de 2 à k.

Étape 6 On répète l'étape 5 jusqu'à ce que l'on trouve pour les coefficients W_{1j}, $j = 2, \ldots, k$ des valeurs égales à zéro. On note f_j^*, pour tout $j = 2, \ldots, k$ les valeurs des fonctions objectif correspondant à ces coefficients.

Étape 7 Le vecteur solution préféré x^*. est déterminé en résolvant le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & f_1(x) \\ \text{sachant que} & f_2(x) = f_2^*, \ldots, f_k(x) = f_k^* \\ \text{et} & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array} \right.$$

Présentation de la méthode Cette méthode se décompose en sept étapes : (voir algorithme 2) :
Le plus difficile dans cette méthode est de bien comprendre la signification du coefficient W_{1j}.

2.3.2 Méthode de Fandel

Le but de cette méthode est d'aider l'utilisateur dans le choix des coefficients de pondération [25]

Présentation de la méthode On part du problème P. On modifie le problème de la manière suivante :

$$P = \left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i.f_i(x) \\ \text{tel que} & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \end{array} \right.$$

On a aussi $w_i \geq 0$ et $\sum_{i=1}^{k} w_i = 1$. On retrouve cette condition dans la méthode de pondération des fonctions objectif.

Comme pour la méthode de pondération des fonctions objectif, la variation des coefficients w_i permet de déterminer les points de la surface de compromis. En revanche, dans la méthode de Fandel, on suppose que ces coefficients ne sont pas connus. On suppose aussi que l'utilisateur cherche une solution qui soit proche de la solution idéale.

2.3.3 Méthode STEP

Cette méthode est similaire à la méthode de Fandel. Ici aussi, les informations sur la préférence de l'utilisateur permettent de restreindre l'espace de recherche étape par étape [25].

Présentation de la méthode On part du problème P et on le transforme de la manière suivante :

$$\left\| \begin{array}{ll} \text{Minimiser} & \beta \\ \text{tel que} & \left[f_i(x) - \overline{Y}\right]^t \cdot w_i < \beta \\ & g(x) \leq 0 \\ \text{avec} & h(x) = 0 \\ & f(x) \leq \overline{Y} \end{array} \right.$$

ou le vecteur $\overline{Y}$ correspond a un vecteur de bornes supérieures servant à restreindre l'espace de recherche. Le vecteur w_j est défini dans la suite de cette section.

De la même manière que l'approche précédente, on forme la matrice B.
Les coefficients de pondération w_j des différentes fonctions objectif f_j sont nécessaires pour la résolution du problème. On va déterminer ces coefficients. On va donner un poids plus important aux fonctions f_j dont l'excursion est importante (f_j prend ses valeurs dans un domaine étendu). Ces coefficients ont la forme suivante :

$$w_j = \frac{v_j}{\sum_{i=1}^{k} v_j}$$

avec

$$v_j = \frac{\max_i\{f_j^{*i}\} - f_j^{*j}}{\min_i\{f_j^{*i}\}} \cdot \frac{1}{f_j^{*j}}$$

avec $i = 1, \ldots, k$; Par conséquent les poids respectent les conditions $w > 0$ et $\sum_{j=1}^{k} w_j = 1$.

2.3.4 Méthode de Jahn

Cette méthode est complètement différente des deux autres. On commence par choisir un point de départ, puis le problème est résolu pas à pas à travers l'espace de recherche en direction de la solution optimale au sens de Pareto [25].

Cette méthode est fondée sur une méthode de minimisation cyclique (aussi appelée "méthode d'optimisation scalaire dans les directions réalisables" de Zoutendjik).

présentation de la méthode On transforme le problème P de la manière suivante :

$$\left\| \begin{array}{ll} \text{Minimiser} & \beta \\ \text{tel que} & w_j.\left[\nabla_x f_j^u\right]^t \cdot \delta_f \leq \beta \\ & v_i.\left[\nabla_x g_i^u\right]^t \cdot \delta_g \leq \beta \\ & t_l.\left[\nabla_x h_l^u\right]^t \cdot \delta_h = \beta \end{array} \right.$$

On a $\delta_f \in \mathbb{R}^k$, $\delta_g \in \mathbb{R}^m$, $\delta_h \in \mathbb{R}^p$. Où :

β : fonction scalaire de préférence ;

∇_x : opérateur nabla de la variable x, $\nabla_x A = \sum_i \frac{\partial A}{\partial_{x_i} x_i}$;

δ : direction du pas de minimisation.

w_j : coefficient de pondération de la j^e fonction objectif.

v_i : coefficient de pondération de la $i^{\text{ème}}$ contrainte d'inégalité ;

t_l : coefficient de pondération de la $l^{\text{ème}}$ contrainte d'égalité.

La solution de notre nouveau problème commence par le choix d'un vecteur de depart réalisable x^0 et par le calcul du vecteur des fonctions objectif correspondant $f(x^0)$.
Le choix du point de départ influence les alternatives futures possibles, en supposant que l'espace de recherche $\widehat{Y}^0$ soit restreint à :

$$\widehat{Y}^0 = \left\{ f(x) | f(x) \leq f(x^0) \right\}$$

En fonction du vecteur objectif $f(x)$, l'utilisateur choisit une fonction objectif f_i, qui doit être minimisée prioritairement. A cet effet, un pas de direction δ et de longueur appropriée l doit être déterminé, et il doit satisfaire la relation suivante :

$$x^{k+1} = x^k + \lambda^k \cdot \delta^k$$

Il faut que x^{k+1} et x^k satisfassent les contraintes du problème (h et g). De plus, il faut que ces points vérifient :

$$f_i^{k+1} < f_i^k \quad \forall i \in I = \{1, \ldots, k\}$$

$$f_j^{k+1} \leq f_j^k \quad \forall j \in J = \{I | j \neq i\}$$

On obtient la direction du pas en résolvant notre problème modifié (le problème de départ transformé). En agissant ainsi, les coefficients de pondération w_j, v_i et t_l choisis par l'utilisateur influencent la direction du pas. L'utilisateur peut maintenant choisir la longueur du pas λ^k dans un intervalle $[0, \lambda_{\max}]$. On trouve $\lambda_{\max}$ en résolvant le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & \lambda_{\max} \\ \text{sachant que} & f_i(x^k + \lambda_{\max} \cdot \delta^k) < f_i(x^k), \forall i \in I = \{1, \ldots, k\} \\ & f_j(x^k + \lambda_{\max} \cdot \delta^k) \leq f_j(x^k) \quad \forall j \in J = \{I | j \neq i\} \\ & g(x^k + \lambda_{\max} \cdot \delta^k) \leq 0 \\ & h(x^k + \lambda_{\max} \cdot \delta^k) = 0 \end{array} \right.$$

Le nouveau vecteur fonction objectif f^{k+1}, pour le pas de direction et de longueur données, est calculé $f^{k+1} = f(x^k + \lambda^k \cdot \delta^k)$. Si ce vecteur objectif n'est pas accepté, on détermine à la prochaine itération une nouvelle direction, en résolvant notre problème modifié.

2.3.5 Méthode de Geoffrion

Cette approche est similaire à la méthode de Jahn. Elle repose sur un algorithme : l'algorithme de Franck-Wolfe [25]

Présentation du la méthode On transforme le problème P de la manière suivante :

$$\left\| \begin{array}{ll} \text{Minimiser} & \left(\nabla_x p \cdot [fx]\right)^t \cdot \delta \\ \text{sachant que} & h(x) = 0 \\ & g(x) \leq 0 \end{array} \right.$$

avec $\delta \in \mathbb{R}^k$.

La fonction de preference $p \cdot [f(x)]$ n'est pas forcément connue de l'utilisateur. Pendant le déroulement de l'algorithme, l'utilisateur devra fournir des informations sur les compromis qu'il désire réaliser. Il doit aussi fournir la longueur du pas.

Deux méthodes sont possibles pour déterminer le point de depart x^0. Soit l'utilisateur sélectionne un vecteur en utilisant la méthode de Jahn, soit le vecteur est calcule en résolvant le problème de substitution suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^k w_i \cdot f_i(x) \\ \text{sachant que} & h(x) = 0 \\ \text{et} & g(x) \leq 0 \end{array} \right.$$

On reconnaît là la méthode de pondération des fonctions objectif décrite dans les sections précédentes. En résolvant ce problème, on obtient le point $x^0 = x^*$.

A ce moment, on peut résoudre le problème p'. Cette résolution nous donne la direction de recherche δ. Des informations à propos de la fonction de préférence sont maintenant requises. Il va falloir transformer le problème P' en utilisant la relation suivante :

$$\nabla_x p \cdot \big[f(x)\big] = \sum_{i=1}^{k} \big[\frac{\partial p}{\partial f_i}\big]_x \cdot \nabla_x \cdot f_i(x)$$

On obtient le problème suivant :

$$\left\| \begin{array}{ll} \text{Minimiser} & \sum_{i=1}^{k} w_i \cdot (\nabla_x f_i(x^k))^t \cdot \delta \\ \text{sachant que} & h(x) = 0 \\ \text{et} & g(x) \leq 0 \end{array} \right.$$

On obtient

$$w_i = \frac{\left[\frac{\partial p}{\partial f_i}\right]_x}{\left[\frac{\partial p}{\partial f_1}\right]_x}$$

avec $i \in \{1, \ldots, k\}$, et la direction de l'étape est donnée par $\delta = x^{k+1} - x^k$.

Les coefficients de pondération reflètent le compromis qui est fait par l'utilisateur entre la fonction objectif f_i et la fonction objectif de référence f_1.

Après résolution du problème de substitution ci-dessus, la longueur du pas λ doit être déterminée par l'utilisateur lui-même, qui doit considérer le but :

$$\left\| \begin{array}{ll} \text{Minimiser} & p\big[f(x_k + \lambda^k \cdot \delta^k)\big] \\ \text{sachant que} & h(x_k + \lambda^k \cdot \delta^k) = 0 \\ \text{et} & g(x_k + \lambda^k \cdot \delta^k) \leq 0 \end{array} \right.$$

en utilisant l'hypothèse suivante :

$$0 \leq \lambda^k \leq 1$$

Une fois que la longueur du pas est déterminée, le nouveau vecteur objectif peut être calculé :

$$f^{k+1} = f(x_k + \lambda^k \cdot \delta^k)$$

L'utilisateur évalue alors le résultat, et décide si le processus peut continuer, ou si la solution peut être acceptée comme solution de compromis. S'il décide de continuer, il peut alors parcourir la surface de compromis

2.3.6 Méthode de Simplexe

Cette méthode n'a rien à voir avec la méthode du simplexe utilisée en programmation linéaire. Il s'agit là d'une méthode de recherche séquentielle [Nelder et al. [52]] de l'optimum d'un problème d'optimisation. Le logiciel [Multisimplex] réalise cette optimisation pour un problème d'optimisation multi-objectifs de manière interactive.

Cette méthode utilise $k + 1$ essais (où k représente la dimension de la variable de décision x) pour définir une direction d'amélioration des fonctions objectif. L'amélioration des fonctions objectif est obtenue en utilisant une méthode d'agrégation floue.

On commence par choisir au hasard $k+1$ valeurs pour la variable de décision x. L'algorithme évalue alors les $k + 1$ points et supprime le point le moins "efficace". Il crée alors un nouveau point à partir du point supprimé (voir figure 2.5) et recommence l'évaluation.
L'algorithme comporte quelques règles, qui permettent de choisir des points qui évitent de tourner autour d'une mauvaise solution. Les deux principales règles sont les suivantes :

Règle 1 : rejeter les pires solutions. Une nouvelle position de la variable de decision x est calculée par réflexion de la position rejetée (voir la figure 2.5). Après cette transformation, on recherche le nouveau pire point. La méthode est alors répétée en eliminant ce point, etc. A chaque étape, on se rapproche de la zone où se trouve l'optimum recherché.

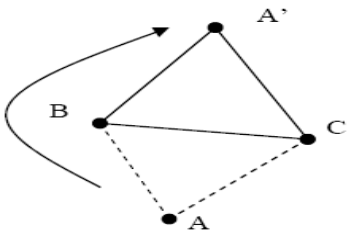

FIG. 2.4 – Choix d'un nouveau point par symétrie.

Règle 2 : ne jamais revenir sur un point qui vient juste d'être rejeté. Sans cette règle, l'algorithme pourrait osciller entre deux "mauvais" points. [13]
W : point le moins favorable ou point venant d'être rejeté.
B : point le plus favorable.
R : second point le plus favorable.

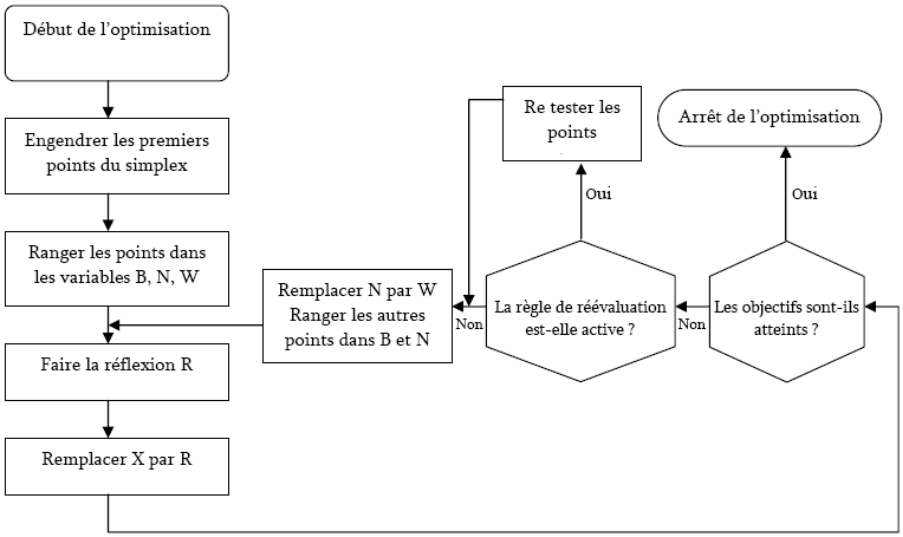

FIG. 2.5 – L'algorithme de la méthode de Simplex.

2.4 Méthodes floues ou brouillées

Dans la vie courante, tout ne peut pas être décrit de manière binaire. Par exemple, la transition entre le jour et la nuit se fait progressivement, l'action sur l'embrayage d'un véhicule est, elle aussi, progressive.
Longtemps, le seul outil de description en logique était binaire. Tout en logique a été décrit en termes de VRAI ou FAUX. Le problème de cette description simpliste en logique est qu'elle ne permet pas de traiter l'incertitude et l'imprécision des connaissances humaines.

L'automaticien L. A. Zadeh a élaboré une nouvelle logique basée sur les ensembles flous. Elle permet de traiter l'imprécision et l'incertitude dans la connaissance humaine ainsi que les transitions progressives entre états. La différence principale entre une logique classique et cette logique floue est l'existence d'une transition progressive entre le VRAI et le FAUX [87]

2.4.1 Méthode de Sakawa

Cette méthode fait intervenir la logique floue à tous les niveaux (sur les paramètres du problème ainsi que sur les paramètres des contraintes). L'ensemble de solutions que l'on va trouver, lui aussi, fera intervenir la logique floue. Ces solutions auront un niveau d'appartenance. Ce seront des solutions qui auront une corrélation variable avec l'objectif initial [Sakawa et al.[64]] (le niveau de corrélation avec l'objectif initial est fixé par l'utilisateur) [13].

On part du problème suivant :

$$\left\| \begin{array}{l} \min f_1(x), \ldots, f_k(x), \\ g(x) \leq 0, \end{array} \right.$$

2.4.2 Méthode de reardon

On présente une méthode simplifiée de traitement multi-objectifs utilisant la logique floue. Des exemples utilisant cette méthode peuvent être trouvés dans Reardon [61]

Présentation de la méthode On part du problème P. Pour chaque fonction objectif, on définit une fonction d'appartenance, qui aura la forme représentée à la figure 2.6.

Cette fonction d'appartenance permet d' "informer" l'algorithme que la fonction objectif a sa valeur située dans l'intervalle $[[O_i - E_i, O_i + E_i]$ (zone où la fonction d'appartenance vaut 0). Si la valeur de la fonction objectif est située en dehors de cette zone, on pénalise de plus en plus la fonction objectif. Dans ce cas, la pénalité varie entre 0 et S_{min} et S_{max}.
La signification des différents paramètres de la fonction d'appartenance est la suivante :

S_{min} **et** S_{max} **:** facteurs d'échelle flous.

f_{min} **:** valeur minimale de la fonction objectif numéro i.

f_{max} **:** valeur maximale de la fonction objectif numéro i.

O_i **:** valeur expérimentale de la fonction objectif numéro i : on voudrait que $f_i = O_i$.

E_i **:** marge d'erreur acceptable (l'objectif O_i est défini à $2 \cdot E_i$)près)

Algorithme 2 Algorithme de la méthode de Sakawa

Étape 1 : sous la contrainte $g(x) \leq 0$, on minimise, puis on maximise, chaque fonction objectif séparément, de manière a obtenir la plage de variation de la fonction d'appartenance des fonctions objectif.

Étape 2 : On définit les fonctions d'appartenance de la manière suivante :

$$\mu_i(x) = \frac{f_{i1} - f_i(x)}{f_{i1} - f_{i0}} \tag{2.1}$$

où

- f_{i0} est la valeur de la fonction d'appartenance la moins intéressante,
- f_{i1} est la valeur de la fonction d'appartenance la plus intéressante.

Étape 3 : On définit les fonctions de prise de décision DM_i (Decision Making) comme suit :

$$DM_i(x) = \begin{cases} 0, & \text{si } \mu_i(x) \leq 0\,; \\ \mu_i(x), & \text{si } 0 \leq \mu_i(x) \leq 1\,; \\ 1, & \text{si } \mu_i(x) \geq 1. \end{cases} \tag{2.2}$$

où

- $DM_i(x) = 0$, quand l'objectif n'est pas atteint,
- $DM_i(x) = 1$, quand l'objectif est atteint.

Étape 4 : On maximise les fonctions DM_i.

Étape 5 : S'il n'y a pas de solutions, ou si la solution ne satisfait pas l'utilisateur, alors il faut modifier les fonctions d'appartenance, et retourner à l'étape 3.

Étape 6 : Arrêt et affichage des résultats.

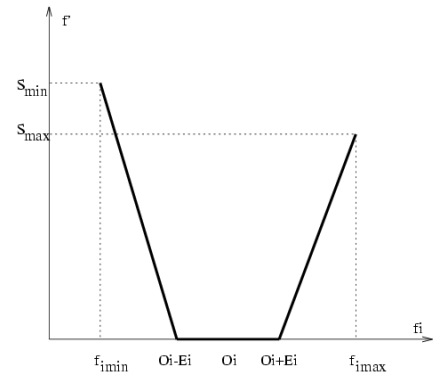

FIG. 2.6 – Fonction d'adhésion de Reardon

- si $f_i \leq (O_i - E_i)$ alors $f'(f_i) = [\frac{S_{max}}{f_{min} - (O_i - E_i)}] \cdot (f_i - (O_i - E_i))$;
- si $(O_i - E_i) \leq f_i \leq (O_i + E_i)$ alors $f'(f_i) = 0$;
- si $f_i \geq (O_i + E_i)$ alors $f'(f_i) = [\frac{S_{min}}{(O_i + E_i) - f_{max}}] \cdot (f_i - (O_i + E_i))$.

2.5 Métaheuristiques multi-objectifs

2.5.1 Qu'est-ce qu'une métaheuristique ?

Les métaheuristiques sont des méthodes générales de recherche dédiées aux problèmes de "l'optimisation difficile" [Sait et al. [63]]. Ces méthodes sont, en général, présentées sous la forme de concept. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Les principales métaheuristiques sont le recuit simulé, la recherche tabou et les algorithmes génétiques. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie comme les algorithmes évolutionnaires. Dans la figure suivante 2.7, on voit un classement de ces méthodes selon le principe d'inspiration utilisé, est ce qu'il est basé. Plusieurs définition ont été données pour clarifié le concepts de l'heuristique, parmi les quels on trouve les définitions suivantes :

Définition 2.1 *Une heuristique est une technique trouvant de bonnes solutions pour un coût de calcul raisonnable, sans pouvoir garantir l'admissibilité ou l'optimalité, ou même dans de nombreux cas, préciser la distance à l'optimum d'une solution particulière.[59]*

Typiquement, ce genre de méthodes est particulièrement utile pour les problèmes nécessitant une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch and Bound par exemple).

A côté des méthodes heuristiques, sont apparues des méthodes qualifiées de métaheuristiques. Plusieurs définitions ont été proposées pour décrire leurs particularités par rapport aux heuristiques.

Définition 2.2 (Métaheuristique selon Osman et Laporte) *Une métaheuristique est un processus itératif de génération guidant une heuristique subordonnée en combinant intelligemment*

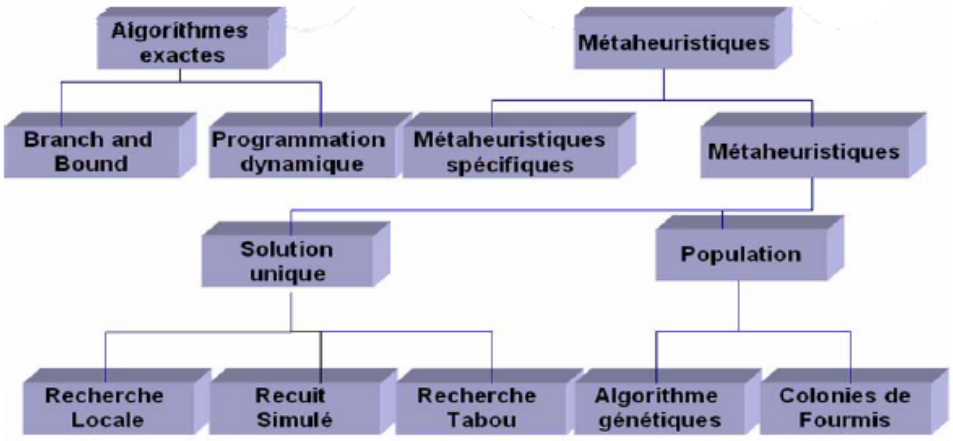

FIG. 2.7 – Schéma des méthodes basées sur les métaheuristiques

différents concepts pour explorer et exploiter l'espace de recherche en utilisant des stratégies pour structurer l'information de manière à trouver efficacement des solutions proches de l'optimum [30].

Cependant, cette définition ne fait pas clairement apparaître l'indépendance des métaheuristiques actuelles vis-à-vis des problèmes à traiter.

Définition 2.3 (Métaheuristique selon Pirlot) *Les métaheuristiques ne sont pas à proprement parler des heuristiques, mais des schémas généraux, des "moules" à heuristiques ; le schéma général doit être adapté à chaque type particulier de problème. [59]*

Malgré la grande diversité de métaheuristiques (algorithmes génétiques, méthode de bruitage, méthode GRASP, recherche à voisinage des variable, recherche dispersée, recherche tabou, recuit simulé, systèmes de fourmis, ...), leurs principes sont souvent assez proches. Taillard [74] distingue trois éléments principaux constitutifs de toutes les métaheuristiques :

- la structure de voisinage permettant de modifier une solution,
- la mémoire des solutions déjà obtenues,
- la construction de solutions entièrement nouvelles.

Ainsi, chaque métaheuristique utilise au moins l'un de ces trois éléments. De plus, l'hybridation de différentes métaheuristiques est une pratique de plus en plus courante permettant d'obtenir des heuristiques performantes. Taillard [73] propose d'ailleurs une unification de la plupart des métaheuristiques telles qu'elles sont implantées actuellement sous la dénomination de programmation à mémoire adaptatif.[18]

Une métaheuristique est donc une méthode très générale, qui nécessite quelques transformations (mineures en générale) avant de pouvoir être appliquée à la résolution d'un problème particulier. Si l'on considère les différentes métaheuristiques, on constate que celles-ci se répartissent principalement en trois familles :

2.5.2 Méthodes déterministes de recherche d'optimum local

Ces méthodes convergent rapidement mais, la plupart du temps, elles ne trouvent pas l'optimum global. Elles se contentent de trouver un optimum local et ne reposent pas sur un processus stochastique pour la recherche de l'optimum (voir 2.8(a)).

2.5.3 Méthodes déterministes de recherche d'optimum global

Ces méthodes permettent de trouver un optimum global rapidement et ne reposent pas sur un processus stochastique pour la recherche de l'optimum (voir 2.8(b)).

2.5.4 Méthodes stochastiques de recherche d'optimum global

Ces méthodes reposent sur un processus stochastique chargé d'effectuer la recherche de l'optimum. Elles sont moins performantes (du point de vue rapidité) que les méthodes déterministes, mais elles peuvent trouver, un optimum global difficile à atteindre(voir 2.8(c)).

Chromosome A	10110010110011100101
Chromosome B	11111110000000011111

(a) *Méthode déterministe de recherche de l'optimum local.*

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

(b) *Méthode déterministe de recherche d'optimum global.*

Chromosome A	1.235 5.323 0.454 2.321 2.454
Chromosome B	(left),(back),(left),(right),(forward)

(c) *Méthode stochastique de recherche d'optimum global.*

FIG. 2.8 – Différentes familles de métaheuristiques.

2.5.5 Méthode de recherche locale

La première classe des métaheuristiques présentées regroupe les méthodes utilisant les principes de la recherche locale. Ces méthodes résolvent le problème d'optimisation de manière itérative. Elles font évoluer la configuration courante en la remplaçant par une autre issue de son voisinage, ce changement de configuration est couramment appelé mouvement. Parmi les méthodes de recherche locale la plus simple et la plus utilisée dans la literature est La descente[6].

2.5.6 Descente

La descente est une méthode d'amélioration itérative simple permettant d'atteindre le premier optimum local. La descente pour un problème de minimisation peut être définie très simplement :

où N est la fonction de voisinage, f la fonction d'évaluation, et x_0 la configuration initiale servant de point de départ à l'algorithme.

Algorithme 3 Pseudo-code de la méthode de descente.

Entrées: Paramètres d'entrée : N ; f ; x_0

$x_{suiv} \leftarrow x_0$;

répéter

$x \leftarrow x_{suiv}$;

$x_{suiv} \in \{x' | x' \in N(x) \bigwedge f(x') = \min(\{f(y) | y \in N(x)\})\}$;

jusqu'à $f(x_{suiv}) > f(x)$;

Renvoyer x.

2.5.7 Recuit simulé

Cette méthode de recherche a été proposée par des chercheurs d'IBM qui étudiaient les verres de spin. Ici, on utilise un processus métallurgique (le recuit) pour trouver un minimum. En effet, pour qu'un métal retrouve une structure proche du cristal parfait (l'état cristallin correspond au minimum d'énergie de la structure atomique du métal), on porte celui-ci à une température élevée, puis on le laisse refroidir lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement (voir [Bonomi], [Siarry] et [Siarry][9, 69, 70]).

Ce processus métallurgique a été transposé à l'optimisation et a donné une méthode simple et efficace. Le pseudo-code du recuit simulé est représenté dans l'algorithme qui suit. Le fonctionnement de cet algorithme est le suivant :

- On commence par choisir un point de départ au hasard (x) ;
- On calcule un voisin de ce point ($\gamma = \mathcal{V}(x)$) ;
- On évalue ce point voisin et on calcule l'écart par rapport au point d'origine ($\Delta C = C(\gamma) - C(x)$) ;
- Si cet écart est négatif, on prend le point γ comme nouveau point de départ, s'il est positif, on peut quand même accepter le point y comme nouveau point de départ, mais avec une probabilité $e^{-\frac{\Delta C}{T}}$ (qui varie en sens inverse de la température T) ;
- Au fur et à mesure du déroulement de l'algorithme, on diminue la température $T(T = \alpha(T))$, souvent par paliers ;
- On répète toutes ces étapes tant que le système n'est pas figé (par exemple, tant que la température n'a pas atteint un seuil minimal).

Au début de la simulation, les points ont une grande capacité d'exploration de l'espace d'état car l'algorithme accepte des déplacements très importants. Au fur à mesure que T diminue P augmente, la capacité de déplacement d'un point diminue et les points améliorant leur valeur sont de plus en plus nombreux, la chance d'une solution négative d'être acceptée diminue. Quand $T \rightarrow 0$ seuls les points ameliorant leurs valeurs sont acceptés. A la fin il y a une chance de zéro que n'importe quel changement négatif de température peut être alloué. A ce point, le « refroidissement » est dit d'avoir lieu, et le système doit avoir convergé à la solution optimale.

Dans la littérature des méthodes d'optimisation multi-objectifs, on trouve deux importantes extensions de la méthode du Recuit Simulé basées sur les frontières de Pareto. La méthode P.A.S.A (Pareto Archived Simulated Annealing et La méthode M.O.S.A (Multiple objectif Simulated Annealing) Annealing).

Algorithme 4 Recuit simulé.

```
1: Init T (température initial), Init x (point de départ), Init ΔT (temperature) ;
2: tantque not end faire
3:    y = Voisin(x), ΔC = C(y) − C(x) ;
4:    si ΔC < 0 alors
5:       y = x ;
6:    sinon
7:       si alea(0,1) < e^{−ΔC/T} alors
8:          y = x T = α(T) ;
9:       finsi
10:   finsi
11:   si T < ΔT alors
12:
13:   finsi
14: fin tantque
15: REPEAT(while) ;
```

2.5.8 Recherche tabou

Cette méthode, mise au point par F. Glover, est conçue en vue de surmonter les minima locaux de la fonction objectif. C'est une technique d'optimisation combinatoire que certains présentent comme une alternative au recuit simulé.

A partir d'une configuration initiale quelconque, Tabou engendre une succession de configurations qui doit aboutir à la configuration optimale. A chaque itération, le mécanisme de passage d'une configuration, soit x_n, à la suivante, soit x_{n+1}, est le suivant :

- on construit l'ensemble des "voisins" de x_n, c'est-à-dire l'ensemble des configurations accessibles en un seul "mouvement" élémentaire à partir de x_n (si cet ensemble est trop vaste, on en extrait aléatoirement un sous-ensemble de taille fixée) : soit Voisinage(x_n) l'ensemble (ou le sous-ensemble) envisagé ;
- on évalue la fonction objectif f du problème pour chacune des configurations appartenant à Voisinage(x_n). La configuration x_{n+1}, qui succède à la configuration x_n dans la chaîne de Markov construite par Tabou, est la configuration de Voisinage(x_n) en laquelle f prend sa valeur minimale.

Notons que la configuration x_{n+1} est adoptée même si $f(x_{n+1}) > f(x_n)$: c'est grâce à cette particularité que Tabou permet d'éviter les minima locaux de f .

Cependant, telle quelle, la procedure ne fonctionne généralement pas, car il y a un risque important de retourner à une configuration déjà retenue lors d'une itération précédente, ce qui provoque l'apparition d'un cycle. Pour éviter ce phénomène, on tient à jour, à chaque iteration, une "liste Tabou" de mouvements interdits ; cette liste - qui a donne son nom à la méthode - contient les mouvements inverses ($x_{n+1} \rightarrow x_n$) des m derniers mouvements ($x_n \rightarrow x_{n+1}$) effectués (typiquement m=7). La recherche du successeur de la configuration courante x_n est alors restreinte aux voisins de x_n qui peuvent être atteints sans utiliser un mouvement de la liste tabou. La procédure peut être stoppée des que l'on a effectué un nombre donné d'itérations, sans améliorer la meilleure solution atteinte jusqu'ici.
Le pseudo-code de cette méthode est représenté dans l'algorithme 6.

Algorithme 5 Algorithme Tabou standard TS[6].

Soit x une configuration réalisable, et L le nombre d'itérations
$x^* = x$ (x^* est la meilleure solution obtenu auparavant) ;
pour $i = 0$ jusqu'à L **faire**
    Choisir le meilleur mouvement m autorisé selon $opt(x)$ qui est une fonction d'evaluation définie par le décideur ;
    Mettre à jour la liste Tabou en fonction de m
    si $f(x) < f(x^*)$ **alors**
        $x^* \leftarrow x$;
    finsi
fin pour
Renvoyer (x).

2.5.9 Méthode GRASP

La métaheuristique $GRASP$ a été proposée initialement par Féo et Resende [26]. C'est une méthode multi-départ en deux phases pour la résolution approchée de problèmes difficiles de l'optimisation combinatoire. La première phase correspond à la construction d'une solution initiale à l'aide d'une procédure gloutonne aléatoire. L'incorporation d'une composante aléatoire permet d'obtenir des solutions dans des zones diverses de l'espace des solutions. La seconde phase correspond à une recherche locale améliorant ces solutions. Ce processus est répété un grand nombre de fois. De plus, plusieurs nouveaux composants sont venus compléter le schéma de base de $GRASP$ comme par exemple :
Une implementation de $GRASP$ pour un problème particulier va être caractérisée par six éléments principaux :

- l'algorithme glouton utilisé,
- l'importance de la composante aléatoire (fixée par un paramètre $\alpha \in [0; 1]$, une valeur $\alpha = 1$ correspondant à un glouton pur et une valeur $\alpha = 0$ correspondant à un algorithme aléatoire),
- le type de recherche locale et le voisinage considéré,
- la phase d'intensification éventuelle,
- le critère d'arrêt,
- la phase post-optimisation éventuelle.

L'adaptation de $GRASP$ pour résoudre un problème particulier est assez facile lorsqu'il existe des algorithmes de construction et de recherche locale pour ce problème. $GRASP$ a ainsi été appliqué avec succès à un grand nombre de problèmes d'optimisation comme des problèmes d'ordonnancement, de routage ou encore des problèmes théoriques dans les graphes.

2.5.10 Méthode des Colonies de Fourmis

La recherche guidée par Marco Dorigo produit un nouveau membre de cette classe d'algorithmes : l'algorithme de «système de fourmis».[31]

Algorithme de base Le point de départ de cet algorithme se base sur l'observation des fourmis qui construisent des chemins entre une source de nourriture et leur nid. Les fourmis sont capables de déposer sur le sol une certaine quantité d'une substance chimique volatile (le phéromone) qu'elles peuvent détecter ensuite. Les fourmis se déplacent au hasard, dans ce cas, on peut dire qu'elles sont aveugles, mais sont attirées par les chemins de phéromone déposées par d'autres fourmis. Ainsi, plus les fourmis empruntent un chemin, plus il y aura de fourmis attirées par cet itinéraire. Mais dans le cas de cet algorithme de base, quelques modifications sont apportées aux capacités des fourmis telles que :

1. elles possèdent une mémoire ;
2. elles ne sont pas totalement aveugles ;
3. le temps est discret.

L'algorithme 7 donne la structure générale de système de fourmis pour le TSP.

Algorithme 6 Algorithme de colonies de fourmis de base "Ant System"

Initialisation : $\tau ij \longrightarrow \tau_0 \quad \forall (i,j)$
aléatoirement chaque fourmi sur une ville.
pour t = 1, ...,t_max **faire**
  pour chaque fourmi k = 1,...,m **faire**
    Choisir une ville au hasard ;
    pour chaque ville non visitée i **faire**
      Choisir une ville j, dans la liste J_i^k des villes restantes ;
    fin pour
    Déposer une piste $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$;
  fin pour
  Évaporer les pistes
fin pour

Autres variantes

Il existent d'autres variantes de la méthode telles que : AS rank qui est une version élitiste de AS, le Max-Min Ant System qui introduit l'utilisation des valeurs τ_{max} et τ_{min} pour l'intensité de la trace de phéromone. Ceci permet d'éviter que certains chemins soient trop favorisés, Ant Colony System (ACS) a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles. Elle est fondée sur des modifications du AS.

2.5.11 Méthode Monté Carlo

Les méthodes Monté Carlo consistent en des simulations expérimentales ou informatiques des problèmes mathématiques ou physiques, basées sur le tirage des nombres aléatoires. Généralement, on utilise des séries de nombres pseudo-aléatoires générées par des algorithmes spécialisés. Les propriétés de ces séries sont très proches de celles d'une véritable suite aléatoire.

Le grand avantage de cette méthode est sa simplicité. Elle permet entre autres de visualiser l'effet de différents paramètres et de donner ainsi des orientations, d'étudier des structures intéressantes qui auraient été a priori écartées et de trouver facilement des structures que l'on n'aurait pas aussi bien optimisées «à la main».

Algorithme 7 Algorithme de Monté Carlo :

étape 1 On génère un point initial x dans l'espace d'état, considéré comme solution courante.

étape 2 On génère aléatoirement un point x'.

étape 3 Si x' est meilleur que x alors x' devient la solution courante.

étape 4 Si le critère d'arrêt est satisfait alors fin sinon retour en à la deuxième étape

2.5.12 Optimisation par essaims de particules

Ł'optimisation par essaims de particules (OEP) est une méthode née en 1995 aux États-Unis sous le nom de Particle Swarm Optimization (PSO). Initialement, ses deux concepteurs, Russel Eberhart et James Kennedy [10], cherchaient à modéliser des interactions sociales entre des «agents» devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Un modèle simple fut alors élaboré.

Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants convergeant parfois en plusieurs sous-essaims vers des sites intéressants. Ce comportement se retrouve dans bien d'autres modèles, explicitement inspirés des systèmes naturels Ici, la métaphore la plus pertinente est probablement celle de l'essaim d'abeilles, particulièrement du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ses consoeurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement.

2.5.13 Algorithmes évolutionnaires

On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Holland, 1975 ; Goldberg, 1989], les stratégies d'évolution [Schwefel, 1981] et la programmation évolutive [Fogel, 2000]. Ces méthodes se distinguent par la manière de représenter l'information et par la façon de faire évoluer la population d'une génération à l'autre. Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- une ***population*** constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné,
- un ***mécanisme d'évaluation des individus*** permettant de mesurer l'adaptation de l'individu à son environnement,
- un ***mécanisme d'évolution de la population*** permettant, grâce à des opérateurs predéfinés, d'éliminer certains individus et d'en créer de nouveaux.

Parmi les composants d'un algorithme évolutionnaire, l'***individu*** et la ***fonction d'évaluation*** correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. Le mécanisme d'évolution est composé de plusieurs opérateurs tels que la sélection, la mutation et le croisement.

La ***sélection*** a pour objectif de sélectionner des individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante. Le ***croisement*** ou recombinaison est un opérateur permettant de construire de nouveaux individus enfants à partir des caractéristiques d'individus parents sélectionnés. La ***mutation*** effectue des légères modifications de certains individus. Comme exemple des algorithmes évolutionnaires, nous présentons maintenant les algorithmes génétiques [6].

2.5.14 Algorithmes génétiques

Les algorithmes évolutionnaires sont parmi les métaheuristiques à base de population. Ils sont inspirés de la biologie et introduisent le théorème de Darwin dans l'évolution des espèces. Les algorithmes génétiques (AGs) ont été introduits par Holland [Holland, 1975 [35]] comme un modèle de méthode adaptative. Ils s'appuient sur un codage de l'information sous forme de chaînes binaires de longueur fixe et d'un ensemble d'opérateurs génétiques : la sélection, la mutation, le croisement,... Un individu sous ce codage, appelé un chromosome, représente une configuration du problème [6].

Récemment, beaucoup de recherches ont été menées sur l'application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectifs. Celles-ci ont permis de mettre en avant l'intérêt d'utiliser des méthodes d'optimisation basées sur le concept de population. Nous présentons maintenant deux algorithmes évolutionnaires représentatifs, résolvant des problèmes d'optimisation multi-objectifs [6].

Non dominated sorting genetic algorithm II"

$NSGA-II$ [6] est un algorithme élitiste n'utilisant pas d'archive externe pour stocker l'élite. Pour gérer l'élitisme, NSGA-II assure qu'à chaque nouvelle génération, les meilleurs individus rencontrés soient conservés.

Strength Pareto evolutionary algorithm

La méthode SPEA [6] implementée par Zitzler et Thiele [Zitzler and Thiele, 1998b] est l'illustration même d'un algorithme évolutionnaire élitiste. Pour réaliser cet élitisme, $SPEA$ maintient une archive externe contenant le meilleur front de compromis rencontré durant la recherche.

Algorithme 8 Pseudo-code de l'algorithme $NSGA-II$.

tantque critère_d'arrêt_non_rencontré **faire**
  Créer $R_t = P_t \bigcup Q_t$
  Calculer différents fronts F_i de la population R_t par un algorithme de "ranking" ;
  Mettre $P_{t+1} = \phi$; et $i = 0$,
  tantque $|P_{t+1}| + |F_i| < N$ **faire**
    $P_{t+1} = P_t \bigcup F_i$
    $i = i + 1$
  fin tantque
  Inclure dans P_{t+1} les $(N - |Pt + 1|)$ individus de F_i les mieux répartis
  au sens de la distance de "crowding"
  Sélectionner dans P_{t+1} et créer de Q_{t+1} par application des opérateurs de croisement et mutation ;
fin tantque

La première étape consiste à créer une population initiale (constituée par exemple d'individus générés aléatoirement). L'archive externe, au départ initialisée à l'ensemble vide, est mise à jour régulièrement en fonction des individus non dominés de la population.

À chaque itération de l'algorithme, on retrouve les étapes classiques *sélection* + *croisement* + *mutation*. Puis les nouveaux individus non dominés découverts viennent s'ajouter à l'archive, et les individus de l'archive dominés par le nouvel arrivant sont supprimés. Si l'archive vient à excéder une certaine taille (fixée au départ), alors une phase de clustering est appliquée dans le but de garder les meilleurs représentants. La figure 2.9 (extraite de la thèse de Zitzler [Zitzler, 1999]) illustre le schéma général de fonctionnement de l'algorithme.

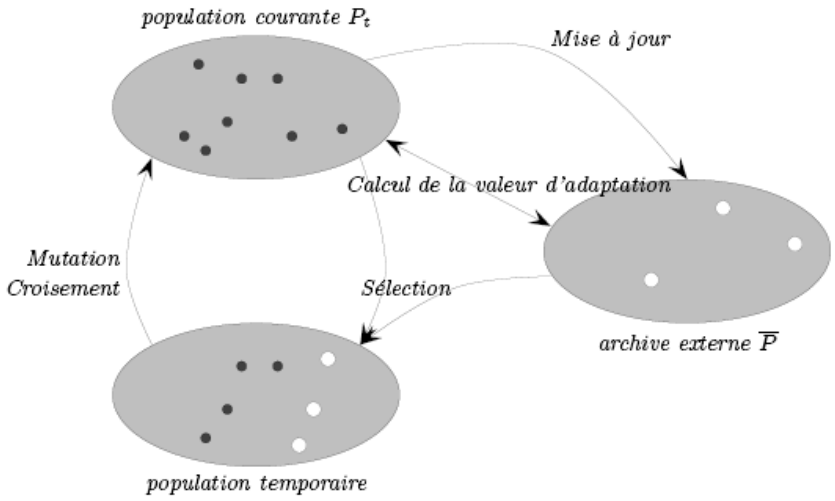

FIG. 2.9 – Fonctionnement général de l'algorithme SPEA.

2.6 Méthodes d'aide à la décision

Les méthodes que nous avons présentées jusqu'ici sont basées sur la relation de dominance. Cette relation (que l'on peut définir de plusieurs manières : la dominance de Pareto, la dominance lexicographique par exemple) permet de filtrer les éléments d'un ensemble, et de ne retenir que les éléments incomparables entre eux. Cependant, il existe une autre approche pour obtenir un ensemble de solutions, qui repose sur l'établissement d'une relation d'ordre entre les différents éléments. Ainsi, on peut, en fonction de la relation d'ordre définie, obtenir un ensemble de solutions (relation d'ordre partiel) ou une et une seule solution (ordre complet). L'autre différence majeure, par rapport aux méthodes d'optimisation multi-objectifs classiques, vient du fait que les méthodes d'aide à la décision ne travaillent que sur des ensembles discrets de points (les méthodes d'optimisation multi-objectifs "classiques" peuvent travailler, elles, sur des ensembles continus).

De plus, les méthodes d'aide à la décision permettent de répondre à plusieurs problématiques, réunies dans le tableau 2.1.

Problématique	Résultat	Procédure
Choix d'un sous-ensemble des actions les "meilleures" ou, a défaut, les plus "satisfaisantes".	Choix	Sélection
Tri par affectation des actions à des catégories prédéfinies.	Tri	Affectation
Rangement de classes d'équivalence composées d'actions, ces classes étant ordonnées de façon complète ou partielle.	Rangement	Classement

TAB. 2.1 – Les divers problèmes répondus par des méthodes d'aide de décision

L'aide à la décision a été développée à partir de la constatation explicitée maintenant.
Dans certains cas, lorsque l'on est amené à comparer trois actions, on peut rencontrer un bouclage entre ces actions. Ce phénomène est appelé l'intransitivité de la préférence, et de l'indifférence, ou paradoxe de Condorcet. Il se résume de la manière suivante : soient trois actions A, B et C, on peut avoir $A \geq B$, $B \geq C$ et $C \geq A$ (ici, le symbole $\geq$ désigne la preference).

L'aide à la décision nous propose donc une prise en compte de ces propriétés d'intransitivité des relations d'ordre, et elle nous propose aussi des familles de méthodes destinées à résoudre des problématiques différentes (Sélection, Tri, Rangement) de celles traitées par l'optimisation multi-objectifs classique.

Lorsque l'on est confronté au domaine de l'aide à la décision, on remarque un certain nombre de termes redondants : Action désigne un objet, une décision, un candidat ou autre chose encore. C'est sur cette entité que va s'opérer la sélection (ou le tri, ou le classement).
Une méthode d'aide à la décision va donc permettre de choisir la meilleure action, ou de classer des actions en fonction d'un ou plusieurs critères.

En fonction du type de classement que l'on désirera effectuer, on pourra utiliser différents types de règles de classement ou de choix. Ces règles vont définir un ordre complet (si toutes les actions sont classées) ou partiel (après le classement, il subsiste des actions incomparables, donc non classées).
Il existe plusieurs méthodes d'aide à la décision, on cite par example la famille ELECTRE (I, IS, II, III, IV et TRI), et la famille PROMETHEE (I et II).

2.7 Difficultés de l'optimisation multi-objectifs

Dans cette section, nous présentons toutes les difficultés rencontrées par le processus d'optimisation multi-objectifs. Un processus d'optimisation multi-objectifs doit résoudre les deux tâches suivantes :

- guider le processus de recherche vers la frontière de Pareto,
- maintenir une diversité des solutions pour assurer une bonne répartition sur la frontière Pareto.

L'accomplissement de ces tâches est très délicat car les difficultés rencontrées dans un problème multi-objectifs sont identiques à celles d'un problème monobjectif. Elles sont amplifiées par la présence d'objectif dépendant les uns des autres.

2.7.1 Guider le processus de recherche vers la frontière Pareto

Le processus de recherche est souvent ralenti ou totalement dérouté par des fonctions possédant une des caractéristiques suivantes : **multimodalité** (lorsque une fonction possède plusieurs optimums globaux, dès lors, chaque optimum exerce une attraction sur le processus de résolution ce qui peut piéger la convergence de la méthode), **isolation d'un optimum** (il existe des problème dans lesquels un optimum peut être entouré de grandes zones pratiquement plates. Cet optimum se trouve alors isolé car l'espace de recherche qui l'entoure ne peu pas guider vers lui les individus de la population) et **tromperie** (un problème est trompé lorsqu'il guide la convergence vers une zone non optimale de la fonction).

2.7.2 Maintenir la diversité sur le front Pareto

La difficulté à maintenir une bonne répartition des solutions sur la frontière de Pareto résulte principalement des caractéristiques suivantes : **convexité ou non convexité** de la frontière de Pareto (certains problèmes ont une frontière de Pareto non convexe), **discontinuité** de cette frontière (si une frontière de Pareto est discontinue, nous retrouvons le même principe que pour une fonction multimodale, les différentes parties de cette frontière vont exercer proportionnellement à leurs tailles, une attraction plus ou moins importante sur les individus d'une population, certaines d'entre elles pourront donc ne pas être découvertes) **non uniformité** de la distribution (les solutions sur la frontière de Pareto peuvent ne pas être réparties uniformément, la raison principale vient du choix des fonctions objectifs, si une des fonction objectif est multimodale, elle va influencer de manière très différente la répartition des solutions sur la frontière de Pareto).

3

Les Algorithmes Génétiques.

Ce chapitre présente les notions essentielles permettant de comprendre la structure de base d'un algorithme génétique et son fonctionnement. Nous présentons quelques principes des algorithmes génétiques, les étapes de fonctionnement (initialisation, évaluation des individus, codage, selection, croisement, mutations, décodage). Nous présentons également les différentes méthodes relatives à chaque étape et les applications de ces algorithmes.

Contents

L'idée d'utiliser les principes des processus d'évolution organique en tant que technique d'optimisation globale a émergé indépendamment des deux côtés de l'océan Atlantique il y a une vingtaine d'années. Ces deux approches reposent sur l'imitation du phénomène d'apprentissage collectif d'une population naturelle, basée sur les observations de Charles Darwin et sur la théorie moderne de l'évolution. Ces deux courants ont évolué parallèlement jusqu'à ces dernières années, chacun ayant son champ d'application particulier, tous deux devenant actuellement de plus en plus attirants aussi bien pour les chercheurs que pour les industriels, grâce notamment à la vulgarisation des calculateurs parallèles [7, 57].

Les premiers travaux sur les algorithmes génétiques ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland, sur la base des travaux précédents, développa les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique et ont trouvé un premier aboutissement en 1975 avec la publication de *Adaptation in Natural and Artificial Systems*. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. C'est cependant l'ouvrage de David Goldberg *Genetic algorithms in search, optimization and machine learning* qui a largement contribué à développer les algorithmes génétiques. En 1992 John Koza a utilisé les algorithmes génétiques pour évoluer des programmes pour exécuter certaines tâches. Il a appelé sa méthode " Programmation génétique " [4, 20].

3.1 Principes des algorithmes génétiques

Les algorithmes génétiques sont des procédures qui s'inspirent des mécanismes de *sélection naturelle* et des phénomènes génétiques. Le principe de base consiste à simuler le processus d'évolution naturelle dans un environnement hostile. Ces algorithmes utilisent un vocabulaire similaire à celui de la génétique, cependant, les processus auxquels ils font référence sont beaucoup plus complexes.

On parlera ainsi d'*individu* dans une population. L'individu est composé d'un ou plusieurs *chromosomes*. Les chromosomes sont eux-mêmes constitués de *gènes* qui contiennent les caractères héréditaires de l'individu. Les principes de *sélection*, de *croisement*, de *mutation* introduits dans ce cadre artificiel, s'appuient sur les processus naturels du même nom.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'état. On lui associe la valeur du critère à optimiser. L'algorithme génère ensuite de façon itérative des populations d'individus sur lesquelles on applique des processus de sélection, de croisement et de mutation. La sélection a pour but de favoriser les meilleurs éléments de la population, tandis que le croisement et la mutation assurent une exploration efficace de l'espace d'état.

Le mécanisme consiste à faire évoluer, à partir d'un tirage initial, un ensemble de points de l'espace vers le ou les optimas d'un problème d'optimisation.Par analogie avec la génétique, on parle alors de générations successives. L'ensemble du processus s'effectue à taille de population constante, que nous notons N, de sorte que les générations successives comportent toutes N

individus. Afin de faire évoluer ces populations de la génération k à la génération $k+1$, trois opérations sont effectuées pour tous les individus de la génération k :

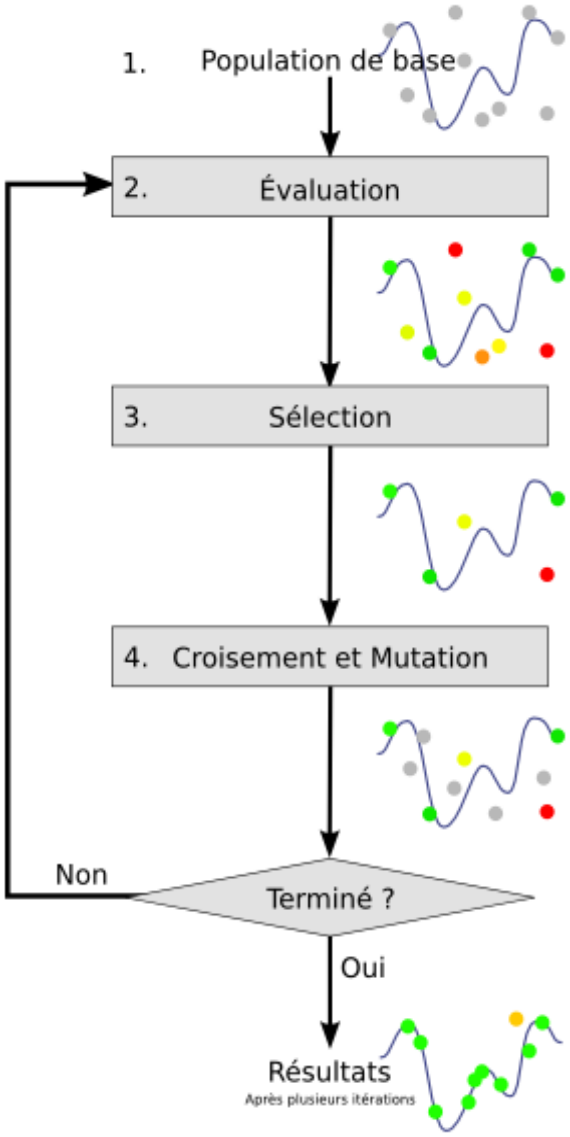

FIG. 3.1 – Principe général des algorithmes génétiques

1. Une **sélection** d'individus de la génération k est effectuée en fonction du critère à optimiser ou plus généralement du critère d'adaptation au problème (fitness), on cherche ainsi à privilégier la reproduction des " bons " éléments au détriment des " mauvais ". Des opérateurs d'exploration de l'espace sont ensuite utilisés pour " élargir " la population et introduire de la nouveauté d'une génération sur l'autre.
2. L'opérateur de **croisement** est appliqué avec une probabilité Pc à deux éléments de la génération k (parents) qui sont alors transformés en deux nouveaux éléments (les enfants) destinés à les remplacer dans la génération $k+1$.
3. Certaines composantes (les gènes) de ces individus peuvent ensuite être modifiés avec une probabilité Pm par l'opérateur de **mutation**. Cette procédure vise à introduire de la nouveauté dans la population.

Enfin, les nouveaux individus sont évalués et intégrés à la population de la génération suivante,cette procédure en trois points est ensuite renouvelée à taille de population constante. Plusieurs critères d'arrêt de l'algorithme sont possibles. Les critères d'arrêts sont alors :

- Arrêt après un nombre de générations fixé *a priori*. C'est la solution retenue lorsqu'un impératif de temps de calcul est imposé.
- Arrêt lorsque la population cesse d'évoluer ou n'évolue plus suffisamment rapidement, on est alors en présence d'une population homogène dont on peut penser qu'elle se situe à proximité du ou des optimums.

Il s'agit donc d'un *" algorithme stochastique itératif "* qui opère sur des ensembles de points *codés*, à partir d'une *population initiale*, et qui est bâti à l'aide de trois opérateurs : *croisement*, *mutation*, *sélection*. Les deux premiers sont des opérateurs d'exploration de l'espace, tandis que le dernier fait évoluer la population vers les optima du problème. Nous détaillons dans la section suivante les différentes phases de l'algorithme ainsi les principes de fonctionnement de ces opérateurs.

Pour utiliser un algorithme génétique sur un problème d'optimisation, on doit donc disposer d'un principe de codage des individus, d'un mécanisme de génération de la population initiale et d'opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche.
Les AGs sont alors basés sur les phases suivantes :

1. **Initialisation**. Une population initiale de N chromosomes est tirée aléatoirement.
2. **Évaluation**. Chaque chromosome est codé, puis évalué.
3. **Sélection**. Création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.
4. **Reproduction**. Possibilité de croisement et mutation au sein de la nouvelle population.
5. **Retour** à la phase d'évaluation jusqu'à l'arrêt de l'algorithme [78].

Le fonctionnement de l'algorithme est représenté par l'organigramme de la figure suivante :

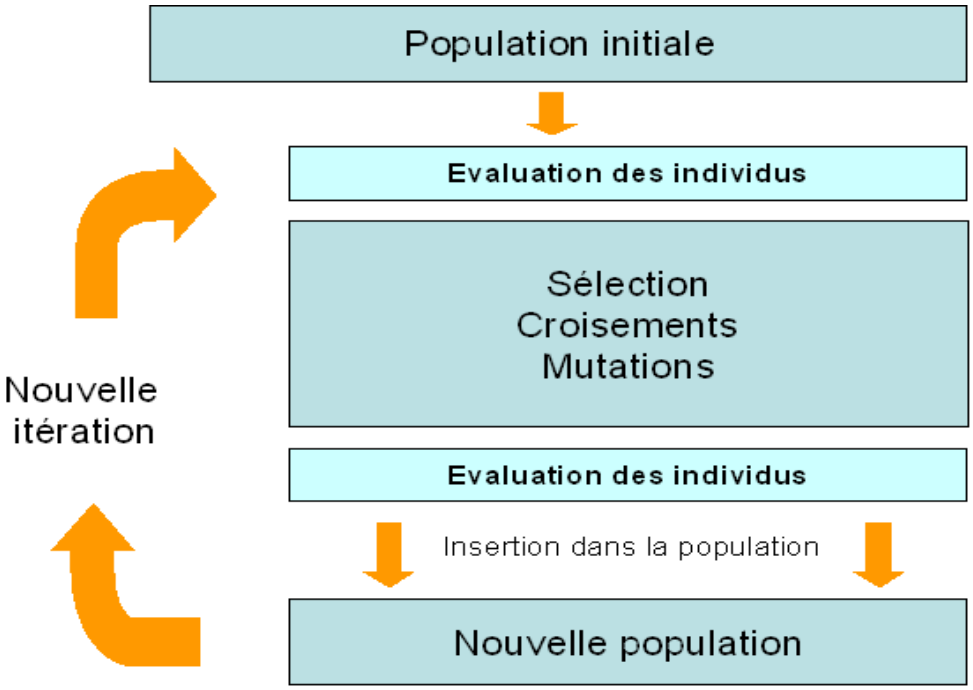

FIG. 3.2 – Algorithme génétique de base

Voyons maintenant plus en détail les autres phases de l'algorithme génétique. Nous présentons ces opérateurs sous l'hypothèse implicite que le codage est binaire.

3.2 Éléments majeurs d'un algorithme génétique

3.2.1 Codage des chromosomes :

La première étape est de définir et de coder convenablement le problème.Historiquement le codage utilisé par les algorithmes génétiques était représenté sous forme de chaîne binaire contenant toute l'information nécessaire a la description d'un individu (Chromosome). D'un point de vue informatique, nous utilisons dans notre algorithme un *codage binaire*[1]. Un chromosome est un tableau de gènes (figure 3.4). Un individu est un tableau de chromosomes. La population est un tableau d'individus[27]. D'autres formes de codage sont possibles a savoir le *codage réel*,*codage de gray*,... .

On aboutit à une structure présentant cinq niveaux d'organisation (figure 3.3), d'où résulte le comportement complexe des AG :

[1]Un des avantages du codage binaire est que l'on peut ainsi facilement coder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes, des chaînes de caractères... Cela nécessite simplement l'usage de fonctions de codage et décodage pour passer d'une représentation à l'autre.

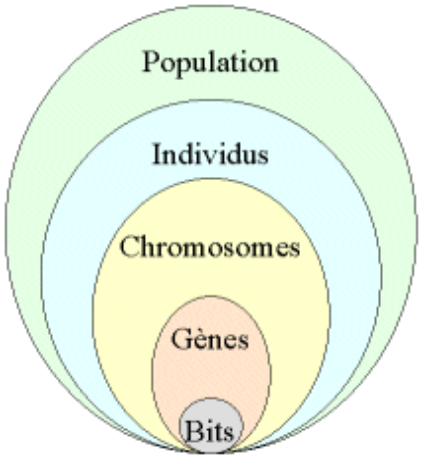

FIG. 3.3 – Les cinq niveaux d'organisation de notre Algorithme Génétique.

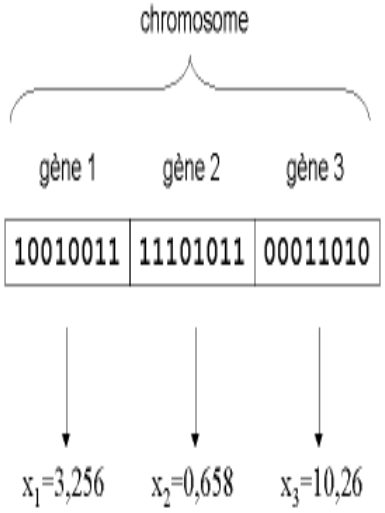

FIG. 3.4 – Illustration schématique du codage des variables d'optimisation .

Il existe deux types de difficultés dans le choix d'un codage.D'une part celui-ci doit pouvoir être adapté au problème de façon à limiter au mieux la taille de l'espace de recherche,et aussi de façon que les nouveaux chromosomes engendrés par les opérateurs de recherche soient significatifs le plus souvent possible,c'est à dire qu'il puissent coder des solutions valides respectant les contraintes du problème.

Avant d'aller plus loin il nous faut définir quelques termes importants généralement définis sous l'hypothèse de codage binaire.

Définition 1 (Séquence/Chromosome/Individu (Codage binaire)) [78]
Nous appelons une séquence (chromosome, individu) A de longueur $l(A)$ une suite $A = \{a_1, a_2, ..., a_l\}$

avec $\forall i \in [1, l], a_i \in V = \{0, 1\}$.

Un chromosome est donc une suite de bits en codage binaire, appelé aussi chaîne binaire. Dans le cas d'un codage non binaire, tel que le codage réel, la suite A ne contient qu'un point, nous avons $A = \{a\}$ avec $a \in \mathbb{R}$.

Quelques autres méthodes de codage

1. ***Le codage réel :*** Une autre solution pour avoir une représentation génétique proche du problème d'optimisation réel est d'employer le *codage réel* qui consiste à prendre comme chromosome le vecteur des variables de contrôle. L'alphabet a alors une cardinalité infni et le chromosome est le plus court possible (sa longueur est la dimension du problème d'optimisation N). Chaque gène a pour valeur la composante du vecteur des variables de contrôle qui lui est associé [21].
2. ***Le codage de permutation :*** Utilisable pour divers problèmes combinatoires,comme le problème du voyageur de commerce ,et des problèmes d'ordonnancement [20].

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

FIG. 3.5 – Le codage avec permutation

3. ***Le codage avec valeur :*** Utilisé dans les problèmes où les valeurs sont compliqués, tel que les nombres réels,où le codage binaire ne suffirait pas.
   Bon pour quelques problèmes, mais souvent nécessaire un développement de quelque croisement spécifique et techniques de la mutation pour ces chromosomes [82].

Chromosome A	1.235 5.323 0.454 2.321 2.454
Chromosome B	(left),(back),(left),(right),(forward)

FIG. 3.6 – Le codage avec valeur

4. ***Codage d'arbre syntaxique :*** Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes où les solutions n'ont pas une taille finie.En principe, des arbres de taille quelconque peuvent être formés par le biais de crossing-over et de mutations.
   Ce codage, est généralement utilisé pour la résolution de problèmes par programmation automatique [20].

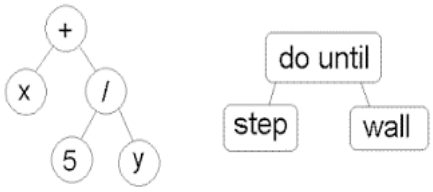

FIG. 3.7 – Le codage avec arbre

3.2.2 Génération aléatoire de la population initiale

Comme dans tout problème d'optimisation, une connaissance de "bons" points de départ conditionne la rapidité de la convergence vers l'optimum.

Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes.

Si par contre, des informations *a priori* sur le problème sont disponibles, il parait bien évidemment naturel de générer les individus dans un sous domaine particulier afin d'accélérer la convergence.

Une nouvelle fois, les contraintes du problème pourront être incorporées (ou non) dans le tirage de la génération initiale.

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état, c'est le rôle des opérateurs de croisement et de mutation. Toutefois cette méthode diffère de la méthode de recherche aléatoire, puisque les générations successives doivent être évaluées et modifiées afin de converger, c'est ici qu'intervient l'opérateur de sélection.

3.2.3 La Fonction Fitness

Pour mieux concrétiser le processus d'évolution, il est nécéssaire de pouvoir faire les distinctions entre chromosomes les *plus adaptés* et les *moins adaptés*. Ceci est possible par l'assignation d'une valeur d'adaptation à chaque chromosome.

La mise au point d'une bonne fonction d'adaptation (Fitness Function)doit réspecter plusieurs critères qui se rapportent à sa complexité ainsi qu'à la satisfaction des contraintes du problème. Si le problème doit satisfaire des contraintes et que les chromosomes produit par les opérateurs de recherche codent des individus invalides, une solution parmi d'autres est d'attribuer une mauvaise *fitness* à l'élément qui a violé les contraintes afin de favoriser la reproduction des individus valides [4].

Définition 2 (Fitness d'une séquence) : [78]
Nous appelons fitness d'une séquence toute valeur positive que nous noterons $f(A)$, où f est typiquement appelée fonction de fitness.

3.2.4 Opérateurs de sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Il existe plusieurs méthodes de sélections, parmi lesquelles on trouve :

1. ***La méthode de la roulette biaisée (Roulette Wheel Selection) :*** La méthode RWS(Roulette Wheel Selection)consiste à associer à chaque individu de la population un segment dans la roulette.La largeur de ce segment est proportionnelle a sa fitness ou, plus precisement, à une probabilité d'être sélectioné proportionnelle à sa fitness [20]. Le principe de la méthode roulette-wheel se fait comme suit :
   (a) Évaluez la fonction Fitness,f_i,de chaque individu dans la population.
   (b) Calculez la probabilité (slot size),p_i,de chaque membre sélectionné de la population : $p_i = f_i / \sum_{j=1}^{N} f_j$,où N est la dimension de la population.
   (c) Calculez la probabilité cumulative,q_i,pour chaque individu : $q_i = \sum_{j=1}^{i} p_j$.
   (d) Générez un nombre aléatoire uniforme,$r \in]0, 1]$.
   (e) Si $r < q_1$ alors sélectionnez le premier chromosome,x_1,sinon sélectionnez l'individu x_i tel qui $q_{i-1} < r \leq q_i$.
   (f) Répétez les étapes 4-5 ,avec N fixe, pour créer les N candidats dans la mare de l'accouplement [65].

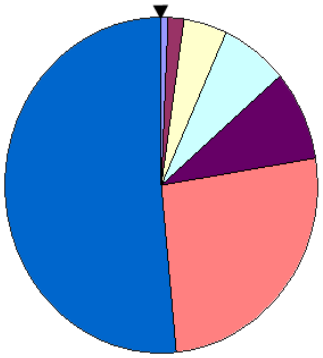

FIG. 3.8 – La selection Roulette-Wheels

2. ***La selection par tournoi :*** La sélection par tournois est une alternative à la sélection proportionnelle.
   Tournoi déterministe : Le tournoi le plus simple consiste à choisir aléatoirement un nombre k d'individus dans la population, et à sélectionner pour la reproduction celui qui a la meilleure performance.Au cours d'une étape de sélection , il y a autant de tournois que d'individus sélectionnés.Les individus qui participent à un tournoi sont remis dans la population, ou en sont retirés, selon le choix de l'utilisateur.Un tirage sans remise permet de faire $\lfloor N/k \rfloor$ tournois avec une population de N individus.Cette méthode de sélection est très utilisée, car elle est beaucoup plus simple à mettre en oeuvre qu'une reproduction proportionnelle.

Tournoi stochastique : Avec le tournoi binaire stochastique, sur deux individus en compétition, le meilleur gagne avec une probabilité p comprise entre 0.5 et 1.

3. ***La selection par Elitiste :*** Du fait que le caractère aléatoire de la sélection ne garantit pas que le meilleur individu soit conservé, la méthode de sélection par élitisme surpasse cette inconvénient en recopiant automatiquement le meilleur individu de chaque génération directement dans la génération suivante. Le modèle élitiste, en fait, n'est pas une méthode de sélection en soi, mais plutôt une variante qu'on ajoute à d'autres méthodes.
4. ***La selection par troncature :*** Dans cette méthode, la sélection n'est pas laissée au hasard. Les individus sont triés selon leur fonction d'adaptation ,ensuite sont choisit les T premiers individus de la génération pour générer la suivante. Cela revient en quelque sorte à généraliser le modèle élitiste à tout le processus de sélection.Le problème avec cette méthode est qu'on ne peut pas maintenir une diversité génétique suffisante dans la population.
5. ***La selection déterministe :*** Cette sélection est très simple à mettre en œuvre, puisqu'elle ne fait que choisir les n meilleurs individus d'une population, n étant un paramètre que l'utilisateur doit fixer.
6. ***La sélection par rang :*** La sélection par rang est une variante du système de roulette. Il s'agit également d'implémenter une roulette, mais cette fois ci les secteurs de la roue ne sont plus proportionnels à la qualité des individus, mais à leur rang dans la population triée en fonction de la qualité des individus. D'une manière plus parlante, il faut trier la population en fonction de la qualité des individus puis leur attribuer à chacun un rang. Les individus de moins bonne qualité obtiennent un rang faible (à partir de 1). Et ainsi en itérant sur chaque individu on finit par attribuer le rang N au meilleur individu (où N est la taille de la population). La suite de la méthode consiste uniquement en l'implémentation d'une roulette basée sur les rangs des individus. L'angle de chaque secteur de la roue sera proportionnel au rang de l'individu qu'il représente.

3.2.5 Le Croisement (Crossover)

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus).Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis un (potentiellement plusieurs) site de croisement est tiré aléatoirement . Enfin, selon une probabilité P_c que le croisement s'effectue, les segments finaux (dans le cas d'un seul site de croisement) des deux parents sont alors échangés autour de ce site[23].Soit i un individu choisi par l'opérateur de sélection. Soit r_i un nombre aléatoire, $0 \leq r_i \leq 1$.On a :

Si $r_i \geq P_c$ alors appliquer l'opérateur de croisement sur i.

Si $r_i < P_c$ alors on ne peut pas appliquer l'opérateur de croisement sur i.

Classiquement, les croisements sont envisagés avec deux parents(P1 et P2)et génèrent deux enfants (C1 et C2).Cette combinaison des parents utilise la notion de *points de coupures* qui correspond aux points au niveau desquels s'effectuera l'échange de matériel génétique.

Les méthodes de croisement

Pour une représentation binaire, il existe trois variantes de croisement classique :

Le Croisement à un point : Le croisement à un point est le croisement le plus simple. Pour éffectuer ce type de croisement, on sélectionne aléatoirement un point de coupure k identique sur les deux chaines binaires qui soit compris entre 1 et l (l est la longueur du chromosome) puis on subdevise le chromosome de chacun des parents en deux parties de part et d'autre part de ce point.On échange ensuite les deux sous-chaînes situés à droite de chacun des deux chromosomes, ce qui produit deux enfants.

$$C1(i) = \begin{cases} P1(i) \text{ si i appartient a [1,k]} \\ P2(i) \text{ si i appartient a [k+1,L]} \end{cases}$$

$$C2(i) = \begin{cases} P2(i) \text{ si i appartient a [1,k]} \\ P1(i) \text{ si i appartient a [k+1,L]} \end{cases}$$

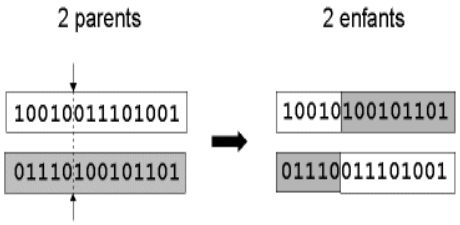

FIG. 3.9 – Représentation schématique du croisement en 1 point.

Le Croisement à deux points : On choisit au hasard deux points de croisement (Figure 1.10). Par la suite, nous avons utilisé cet opérateur car il est généralement considéré comme plus efficace que le précédent. Néanmoins nous n'avons pas constaté de différence notable dans la convergence de l'algorithme.

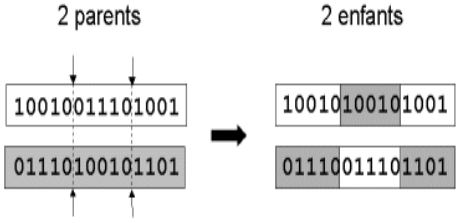

FIG. 3.10 – Représentation schématique du croisement en 2 points.

Notons que d'autres formes de croisement existent, du croisement en k points jusqu'au cas limite du croisement uniforme...

Le Croisement uniforme : Le croisement uniforme peut être vu comme un croisement multi-points dont le nombre de coupures est indéterminé à priori.Pratiquement on utilise un "masque de croisement",engendré aléatoirement pour chaque couple d'individus, qui est un mot binaire de même longueur que les chromosomes.Un "0" à la n-ième position du masque laisse inchangé les symboles à la n-ième position des deux génotypes, un "1" déclenche un échange des symboles correspondants.

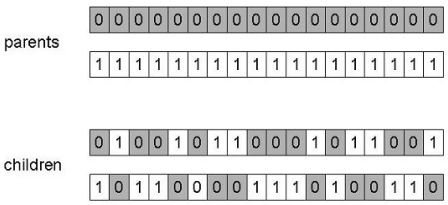

FIG. 3.11 – Représentation schématique du croisement uniforme

3.2.6 Opérateur de mutation

Le rôle de la mutation est d'introduire de nouvelles caractéristiques génétiques, ou de les réintroduire, en modifiant quelques gènes des individus enfants. Elles permettent d'assurer une recherche aussi bien globale que locale, selon le poids et le nombre des bits mutés. De plus, elles garantissent mathématiquement que l'optimum global peut être atteint.

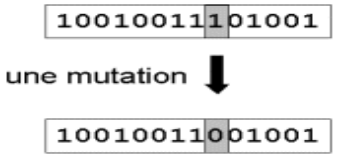

FIG. 3.12 – Représentation schématique d'une mutation dans un chromosome.

Pour chaque gène de chaque enfant, une permutation est réalisée avec une probabilité P_m par bit et par génération est fixée entre 0,001 et 0,01. On peut prendre également $P_m = 1/l$ où l est le nombre de bits composant un chromosome. Il est possible d'associer une probabilité différente à chaque gène. Et ces probabilités peuvent être fixes ou évoluer dans le temps. Cette valeur est généralement faible, de manière à ne pas transformer l'algorithme en une simple recherche aléatoire. La mutation permet d'explorer l'espace de recherche aléatoirement et de réintroduire certains gènes qui ont disparu au cours des générations.

3.2.7 Autres paramètres [78]

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance. La valeur de ces paramètres influence la réussite ou non d'un algorithme génétique. Ces paramètres sont les suivants :

– La taille de la population, N, et la longueur du codage de chaque individu l (dans le cas du codage binaire). Si N est trop grand le temps de calcul de l'algorithme peut s'avérer très important, et si N est trop petit, il peut converger trop rapidement vers un mauvais chromosome. Cette importance de la taille est essentiellement due à la notion de *parallélisme implicite* qui implique que le nombre d'individus traité par l'algorithme est au moins proportionnelle au cube du nombre d'individus.

– La probabilité de croisement P_c. Elle dépend de la forme de la fonction de fitness. Son choix est en général heuristique (tout comme pour P_m). Plus elle est élevée, plus la population subit de changements importants. Les valeurs généralement admises sont comprises entre 0.5 et 0.9.
– La probabilité de mutation P_m. Ce taux est généralement faible puisqu'un taux élevé risque de conduire à une solution sous-optimale.

3.3 Applications des Algorithmes Génétiques [46]

Les algorithmes génétiques classiques, tels que nous venons de les décrire, ont été étendus d'une part par imitation des phénomènes d'évolution naturelle comme la création de niches écologiques (pour l'optimisation de fonctions multimodales) ou la coévolution de différentes populations (où l'on recherche un état d'équilibre). D'autre part, l'emploi d'autres types de codes que les codes binaires, ont permis des applications orientées Intelligence Artificielle comme les systèmes de classeurs, qui manipulent des chromosomes représentant des règles, ou la programmation génétique, où les chromosomes représentent directement des programmes arborescents .

Les algorithmes génétiques et les algorithmes évolutifs en général intéressent des chercheurs et des ingénieurs de disciplines très diverses, par exemple :

– **En optimisation :** lorsque les fonctions à optimiser sont complexes, de forte dimensionnalité, irrégulières, mal connues ;
– **En intelligence artificielle et sciences cognitives :** où l'on exploite plutôt les capacités adaptatives des algorithmes génétiques, et les techniques fondées sur les systèmes de classeurs, (réseaux de neurones, évolution de langages, grammaires) ;
– **En robotique :** où l'on s'intéresse aux MOBOTS (MObile roBOTS) qui doivent pouvoir se mouvoir et agir dans des environnement inconnus, variables (programmation génétique, systèmes de classeurs) ;
– **En physique et en ingénierie :** en tant que méthode d'optimisation pour les problèmes réels complexes (pour l'optimisation de structures par exemple) ;
– **En économie :** pour la modélisation de comportements d'agents par exemple ;
– **En traitement d'images, du signal :** pour détecter des formes caractéristiques, problème que l'on peut soit comprendre comme une optimisation, soit comme une application de règles de décision ;
– **En théorie des graphes et théorie des jeux :** le problème du voyageur de commerce, notamment a beaucoup intéressé les chercheurs.

4

Optimisation d'un critère sur l'ensemble des solutions efficaces

Contents

Présenté par : Ali ZAIDI *Sous la direction de : CHAABANE Djamel, Professeur à l'USTHB*

4.1 Introduction

Depuis les années 1970, des auteurs se sont intéressés à l'optimisation d'une fonction sur l'ensemble E des solutions efficaces d'un problème de programmation linéaire multiobjectifs (MOLP). Le but de ce chapitre est de fournir un aperçu de ces développements. Étant donnée un problème MOLP, l'optimisation sur l'ensemble des solutions efficaces E est la maximisation d'une fonction donnée, ϕ, sur E.
L'importance et la motivation de ce problème, noté (PE), ont été discutées en détail dans la littérature (voir par exemple [32, 8, 47, 53]). En particulier, Benson [8] prouve que dans quelques problèmes de modélisation impliquant des objectifs multiples, les modèles d'optimisation sur l'ensemble des solutions efficaces (PLE) sont plus réalistes et appropries que les programmes linéaires multi-objectifs plus habituels (MOLP). En outre, la solution de ces problèmes d'optimisation, avec l'ensemble efficace défini implicitement, évite les difficultés informatiques d'énumérer tous les points extrêmes efficaces. L'exemple de planification de la production donné par Benson [8] illustre bien ces points. La difficulté du problème est principalement due à la non-convexité de l'ensemble des solutions efficaces. Les algorithmes existants pour résoudre ce problème peuvent être classés en plusieurs groupes selon leur principe de fonctionnement ; citons

- les algorithmes de recherche de sommet adjacent,
- les algorithmes de recherche de sommets non adjacents,
- les algorithmes basés sur la méthode de séparation et évaluation,
- les algorithmes basés sur la méthode de relaxation Lagrangienne,
- les algorithmes basés sur la méthode dual et de bisection [86].

Le problème à été étudié la première fois en 1972 par Philip [131], qui décrit un algorithme basé sur le déplacement sur les sommets efficaces adjacents dans le cas où ϕ est une fonction linéaire ; et un bon nombre de chercheurs ont suivi cette voie. Plus tard, Isermann et Steuer [32] ont décrit une procédure semblable pour la solution de (PE) où la fonction objectif $\phi(x) = dx$ est l'un des objectifs $c^i x$ dans le problème MOLP. Ces méthodes emploient un hyperplan de coupe et essayent de trouver un point efficace sur la face de la coupe qui a une arrête efficace adjacente rapportant une augmentation de la fonction objectif $\phi(x)$. Comme discuté par Benson, qui a étudié des cas plus généraux du problème (P_E), y compris le cas où S est un ensemble général convexe, aucune de ces procédures n'indique explicitement comment trouver un point sur la face de la coupe qui a une arrête efficace adjacente rapportant une augmentation de $\phi(x)$. Pour notre part, nous travaillons en variables discrètes et étudions le problème d'optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces du problème MOILP introduit précédemment. Ce problème surgit toutes les fois qu'une fonction linéaire est disponible en tant que critère pour mesurer l'importance ou pour distinguer parmi les solutions efficaces qui sont disponibles. En optimisant cette fonction linéaire sur l'ensemble des solutions efficaces, les calculs exigés pour produire l'ensemble de toutes les solutions efficaces sont évités. C'est potentiellement tout à fait salutaire, puisque la charge de calcul pour produire cet ensemble se développe rapidement avec la taille de problème [40, 16, 37, 50].

4.2 Quelques méthode de résolution du problème PL_E avec des variables de décision continue

4.2.1 Méthode de Yamamoto[86]

Rappelons que l'on cherche à résoudre le problème (PL_E) où ϕ est quasi convexe.
Soit S_V l'ensemble des sommets du polyèdre S.
Pour $x, \acute{x} \in S_V$, $[x, \acute{x}]$ représente l'arête reliant x et $\acute{x}$. Pour$x \in S_V \cap E$ soit

$$N_E(x) = \{\acute{x} | \acute{x} \in S_V \cap E; [x; \acute{x}] \subset E\}$$

i.e. l'ensemble des sommets efficaces liés à x par une arrête efficace.
En utilisant la quasi-convexité de ϕ, on a :

Lemme 1.*Soit $x \in S_V \cap E$ et supposons que $\{\acute{x} | \acute{x} \in N_E(x); \phi(\acute{x}) > \phi(x)\} = \emptyset$. Alors, x est un maximum local pour le problème (PL_E).*

Ce lemme caractérise la solution optimale locale du problème PL_E quand elle existe, et il est utilisé dans l'algorithme ci-dessous pour brancher vers la deuxième boucle (Boucle(k)).

4.2.2 Méthode d'Ecker et Song [37]

Les auteurs ont développé une méthode pour la résolution de (PL_E) dans le cas où ϕ est linéaire ; on résout donc

$$(PL_E) \begin{cases} \max & \phi(x) = dx \\ t.q & x \in E \end{cases} .$$

où $d \in \mathbb{R}_p = \mathbb{R}^{1 \times p}$

Dans cette méthode on pivote seulement sur la région admissible d'un problème MOLP ou d'un problème réduit (MOLP) afin de surmonter le problème d'optimisation locale qui surgit quand, au cours de la recherche, un point extrême efficace produit n'est lié à aucune arrête efficace améliorant la valeur de la fonction $\phi(x)$.

En effet, étant donné un point efficace courant $\widetilde{x}$, une technique de pivotement pour trouver les arrêtes efficaces adjacentes à $\widetilde{x}$ est utilisée, s'il n'y a aucune arrête adjacente conduisant à une augmentation de f(x), ce qui indique que $\widetilde{x}$ est une solution optimale locale de (PL_E), alors nous ajoutons une coupe à la région réalisable S et nous considérons le programme linéaire multi-objectifs suivant :

$$(MOLP) \begin{cases} \max & z_k = c_k x & \text{k=1,2,....p} \\ t.q & x \in \overline{S} & . \end{cases}$$

où $\overline{S} = \{x \in S | dx \geq \widetilde{dx}\}$. Soit $\overline{E}$ l'ensemble des solutions efficaces pour le problème (MOLP).

Soit le problème (I_j) défini par :

$$(MOLP) \begin{cases} \max & c^i x & \text{k=1,2,....p} \\ t.q & x \in \overline{S} & . \end{cases}$$

Associons au problème (PL_E) le problème relaxé (PL_R)

$$(PL_R)\begin{cases} \max & \phi(x) = dx \\ t.q & x \in S \end{cases} .$$

L'efficacité d'une solution admissible x^* est testée en résolvant le problème linéaire suivant :

$$(P_{x^*})\begin{cases} \max & é\Psi \\ t.q & Cx = \Psi + Cx^* \quad . \\ & x \in S \quad . \\ & \Psi \quad . \end{cases}$$

où $é$ est un vecteur ligne de composantes égales à 1 et $\Psi \in \mathbb{R}^p$

Si la solution optimale de (P_{x^*}) est$\Psi = 0$, alors il n'existe pas dans S une solution x telle que $Cx > Cx^*$
Dans l'algorithme, on teste aussi s'il existe une solution optimale x_j du problème I_j telle que $dx > d\widetilde{x}$ et $x^j \in E$ pour un certain j. Si une telle solution existe, on fait $\widetilde{x} = x^j$, solution efficace courante, et le processus de recherche de l'arrête efficace améliorant $\phi(x)$ continue. Sinon, on pivote si possible, vers une meilleure solution efficace ou bien la solution efficace courante $\widetilde{x}$ est une solution optimale du problème (PL_E). L'algorithme utilise également une technique de pivotement (Ecker et Kouada [38]) pour identifier une solution efficace.
Soit $\widetilde{x}$ une solution efficace initiale du problème (MOLP) et le tableau simplexe correspondant, tableau suivant,

	$\widetilde{x}_N$	$\widetilde{x}_B$
Z	-C	0
b	$\widetilde{A}$	I

où $\widetilde{x}_N$ représente les variables hors-base et $\widetilde{x}_B$ représente les variables de base. L'arrête efficace incidente à $\widetilde{x}$ est identifiée en utilisant le résultat du théorème suivant :

Théorème. [37] *Étant donné le tableau au dessus qui correspond à la solution efficace $\widetilde{x}$. Considérons le programme linéaire (Q^j) défini par :*

$$(Q^j)\begin{cases} \max & Z_Q = és \\ t.q & Cx = s + C(j) \\ & x \geq 0 \\ & s \geq 0 \end{cases}$$

où c(j) désigne la colonne j de C dans le tableau au dessus. Soit F^j l'arrête incidente à $\widetilde{x}$ correspondant à l'entrée j. Alors $F^j \in E$ si et seulement si $Z_Q = 0$dans le problème (Q^j).

Considérons le dual de (Q^j) qui est :

$$(\overline{Q}^j)\begin{cases} \min & W_Q = -ć(j)y \\ t.q & Ćy \geq 0 \\ & -y \geq e \end{cases}$$

Posons y = -υ - e ; on obtient :

$$(\overline{\overline{Q}}^j)\begin{cases} \min & W_Q = -\acute{e}c(j) - \acute{\upsilon}c(j) \\ t.q & \acute{C}\upsilon \geq -\acute{C}e \\ & \upsilon \geq 0 \end{cases}$$

Les auteurs ont montré que le problème (Q^j) admet une solution optimale nulle si et seulement si l'ensemble

$$\Omega = \{(\upsilon, \omega_N) \geq 0 | \acute{C}\upsilon + \omega_N) = -\acute{C}e, \omega_{N_j} = 0)\} = \emptyset$$

Les équations définissant l'ensemble peuvent .etre représentées par le tableau suivant : on suppose que$-\acute{C}e \geq 0$, sinon on pivote pour obtenir une colonne constante non-négative. On observe que, si ω_N peut sortir de la base en maintenant la colonne constante positive ou nulle, l'arrête F^j est efficace. Cette technique est utilisée dans l'algorithme et elle est dite technique de sous-problème.

	υ	$\widetilde{\omega}_N$
$-\acute{C}e$	$\acute{C}$	I

5

Optimisation d'un critère sur l'ensemble des solutions efficaces entières pour le problème du voyageur de commerce.

Contents

Présenté par : Ali ZAIDI *Sous la direction de : CHAABANE Djamel, Professeur à l'USTHB*

5.1 Introduction

Dans des applications pratiques de prise de décision à critères multiples, les décideurs doivent souvent choisir un certain point préféré de l'ensemble des solutions efficaces E ; ceci nous conduit à trouver les solutions efficaces en décrivant la structure de E. Comme, dans beaucoup de cas, les critères sont en conflit, les décideurs essayent alors d'optimiser un certain critère de compromis - généralement linéaire - sur l'ensemble E, ce qui mène à rechercher une méthode qui optimise une fonction linéaire sur un ensemble de solutions efficaces qui n'est pas convexe.

Dans ce chapitre, nous concentrons notre intérêt sur l'optimisation d'une fonction linéaire, dénotée Φ sur un sous ensemble des solutions efficaces d'un problème de voyageur de commerce multi-objectifs noté MOTSP .Nous examinerons le cas général où Φ est une fonction linéaire (pas nécessairement obtenue comme combinaison linéaire des objectifs du problème MOTSP).Une approche métaheuristique (Algorithmes génétiques) consisterait à rechercher le sous ensemble $\acute{E}$ inclue dans l'ensemble des solutions efficaces E du problème MOTSP,cette sous ensemble$\acute{E}$ contient les points efficaces qui minimise la distance d_i entre le point efficace et le point idéal. L'avantage de cette étape est de réduire la taille de l'ensemble des solutions efficaces à parcourir, d'une taille de l'ensemble E grande et exponentielle à une taille réduite et maîtrisable de l'ensemble $\acute{E}$;
le problème est décrit par

$$(MOTSP)\begin{cases} \min & Z_k = \sum_{i=1}^{n}\sum_{j=1}^{j=n} c_{kij}x_{ij} & \text{k=1.....p ;} \\ & & \\ t.q & \sum_{i=1}^{n} x_{ij} = 1 & \text{i=1.....n ;} \\ t.q & \sum_{j=1}^{n} x_{ij} = 1 & \text{j=1.....n ;} \\ & \sum_{i,j\in S} x_{ij} <= |S| - 1 & \text{S\{1.....n ;.\}} \\ & x_{ij} \in \{0,1\} & \text{i=1.....n, j=1.....n.} \end{cases}$$

le problème principale est décrit par

$$(PL_E)\begin{cases} \min & \Phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{j=n} \phi_{ij}x_{ij} & \\ t.q & x_{ij} \in E(MOTSP) & \text{i=1.....n, j=1.....n ;} \\ & x_{ij} \in \{0,1\} & \text{i=1.....n, j=1.....n.} \end{cases}$$

5.2 Le problème du voyageur de commerce

Le problème du voyageur de commerce, étudié depuis le 19^{eme} siècle, est l'un des plus connus dans le domaine de la recherche opérationnelle. Jouez à trouver le meilleur parcours possible... et découvrez différentes méthodes informatiques proposées pour résoudre ce problème.

C'est déjà sous forme de jeu que William Rowan Hamilton a posé pour la première fois ce problème, dès 1859. Sous sa forme la plus classique, son énoncé est le suivant : « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur ». Ce problème d'optimisation combinatoire appartient à la classe des problèmes NP-Complets .

Les domaines d'application sont nombreux : problèmes de logistique, de transport aussi bien de marchandises que de personnes, et plus largement toutes sortes de problèmes d'ordonnancement.

Certains problèmes rencontrés dans l'industrie se modélisent sous la forme d'un problème de voyageur de commerce, comme l'optimisation de trajectoires de machines outils : comment percer plusieurs points sur une carte électronique le plus vite possible ?

5.3 Présentation technique de l'algorithme

Nous donnons une description technique de l'algorithme présenté dans l'introduction.

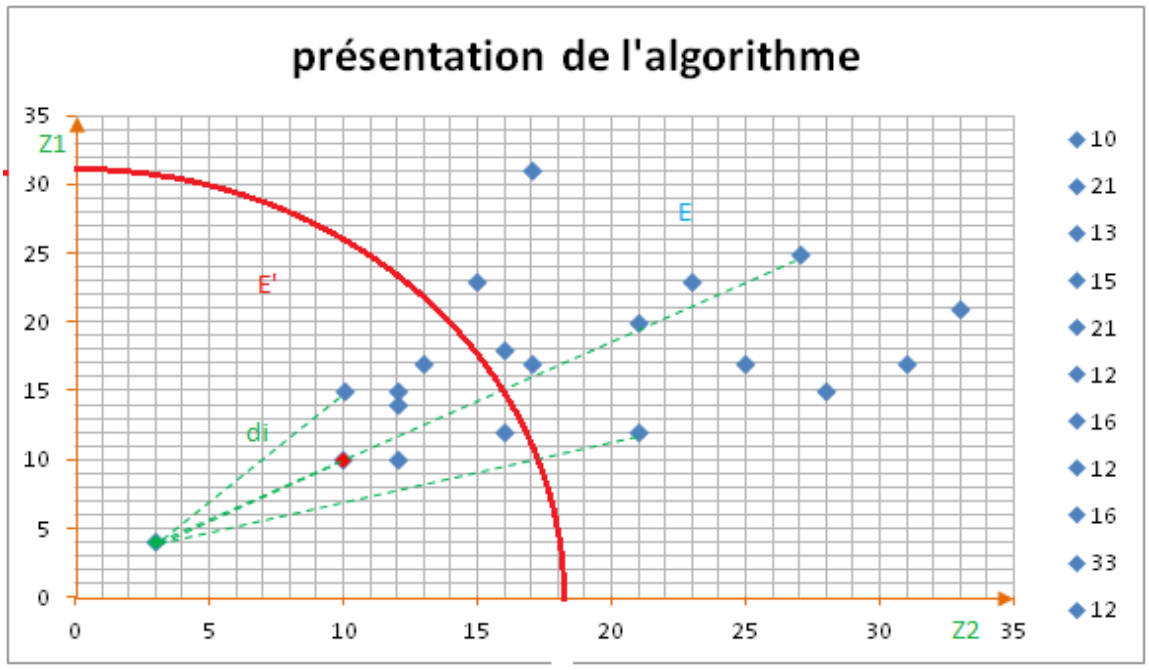

FIG. 5.1 – Representation des solutions sur les axes Z_1, Z_2

Algorithme 9 Algorithme principal

1. **Étape 1 :** Nous entrons les données suivantes :
   - les fonctions objectifs Z_j ;
   - la fonction principale ϕ ;
   - le nombre de villes v du problème MOTSP ;
   - le nombre de critères p.
   - la taille de la population de l'algorithme génétique n.
   - le nombre d'itérations de l'algorithme génétique.
2. **Étape 2 :** à cette étape , nous calculons le point idéalZ_{ideal}. Nous minimisons chaque critère en utilisons l'algorithme génétique.
3. **Étape 3 :** Nous réduisons la taille de l'ensemble des solutions d'une taille exponentielle à une taille maîtrisable inférieur ou égale à la taille de la population choisie dans l'algorithme génétique.nous réduisons cette taille en gardons les points qui minimisent la distance d_i avec le point idéalZ_{ideal}. cette étape est basé sur un algorithme génétique qui minimise une fonction fitness

$$f_i = \sqrt{\sum_{j=1}^{p}(Z_{ij} - Z_{ideal})^2} \quad i \in \{1, 2, ..., n\} \tag{5.1}$$

   à la fin de cette étape nous réussirons a avoir un ensemble de solutions $\acute{E}$ d'une taille inférieur ou égale à n.
4. **Étape 4 :** nous vérifions l'efficacité des points de l'ensemble $\acute{E}$ en appliquant le théorème suivant : **théorème :***Soit X^*. un élément arbitraire de la région D. $X \in E(P)$ si et seulement si la valeur optimale de la fonction objectif Θ est nulle dans le problème de programmation linéaire mixte suivant :*

$$(E(P)) \begin{cases} \max & \Theta = \Sigma_{i=1}^{p}\psi_i \\ t.q & Cx + I\Psi = CX^* \\ & x \in D \quad . \\ & \psi_i \in \mathbb{R}^+ \forall i \quad . \end{cases}$$

   où C est une matrice $(p \times n)$,dont la i^{eme} ligne correspond à $c^i, i = 1, 2, ..., p$, I est la matrice identité $(p \times p)$ et $\Psi = (\psi_i)_{i=1,...,p}$.
   à cette étape si le point vérifier est n'est pas efficaces , l'application affecte la valeur de X à X^* jusqu'a ce que X^* soit efficace. Donc à la fin de cette étape on aura un ensemble réduit des solutions efficaces $\acute{E}$.
5. **Étape 5 :** tant que $\acute{E}$ est une ensemble des solutions efficaces réduit nous évaluons la fonction de critère ϕ et nous choisirons la solution X^* qui donne le minimum des évaluations.

5.4 Présentation technique de l'algorithme génétique

Représentation d'une solution (codage) :

Comme nous l'avons déjà dit le voyageur de commerce doit revenir à son point de départ et passer par toutes les villes une fois et une seule. Nous avons donc codé une solution par un tableau comptant autant d'éléments qu'il y a de villes. Chaque ville y apparaît une fois. Il est alors évident que selon la ville de départ que l'on choisit on peut avoir plusieurs représentations différentes du même parcours.

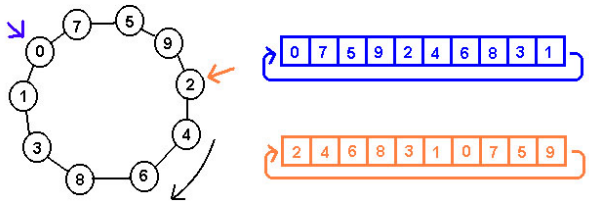

FIG. 5.2 – codage.

Evaluation :

pour évaluer chaque individu de la population ,on utilisant la fonction fitness

$$f_i = \sqrt{\sum_{j=1}^{p}(Z_{ij} - Z_{ideal})^2} \quad i \in \{1, 2, ..., n\} \tag{5.2}$$

Selection :

indent Pour sélectionner les individus père ,on utilise la roulette biaisée bien détaillé dans le chapitre 2.

Croisement :

Étant donné deux parcours il faut combiner ces deux parcours pour en construire deux autres.

Nous avons suivi la méthode suivante :

- On choisi aléatoirement deux points de découpe.
- On interverti, entre les deux parcours, les parties qui se trouvent entre ces deux points.
- On supprime, à l'extérieur des points de coupe, les villes qui sont deja placées entre les points de coupe.
- On recense les villes qui n'apparaissent pas dans chacun des deux parcours.
- On remplit aléatoirement les trous dans chaque parcours.

Mutation :

Il s'agit de modifier un des éléments constitutif de la solution, ici c'est une ville. Quand une ville doit être mutée, on choisit aléatoirement une autre ville dans ce problème et on intervertit les deux villes.

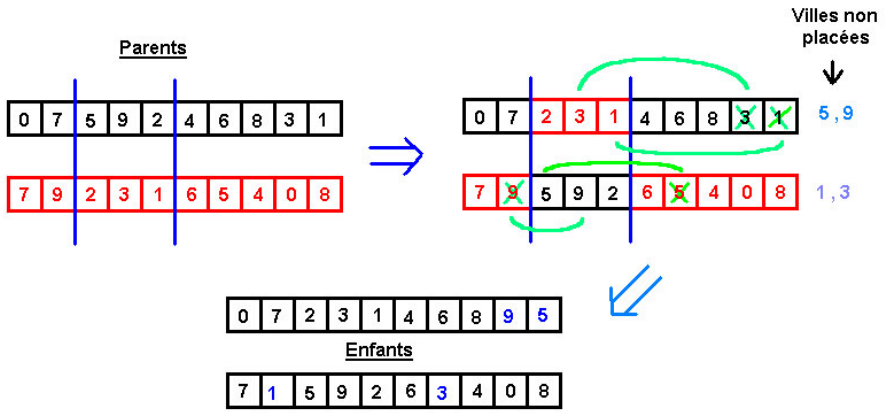

FIG. 5.3 – croisement.

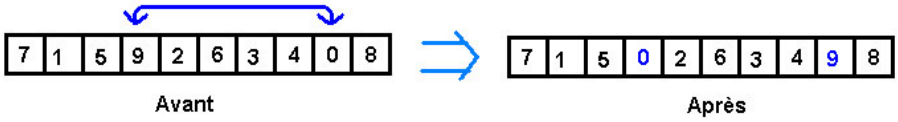

FIG. 5.4 – Mutation.

5.5 présentation de l'application

L'application programme l'algorithme précédent pour objectif de résoudre des problèmes de voyageur de commerce à grands dimensions 500,1000 et plus de villes , elle est programmé à base de langage MATLAB , elle est structuré d'une manier simple a manipulé.

Les entrées de l'application

- nombre de villes(v).
- le fichier exel qui correspond au nombre de villes choisie.
- la taille de la population(p).
- le nombre d'itérations de l'application(k).

Les sorties de l'application

- la solution de problème.
- l'indice de convergence.
- le temps d'execution.
- le graphe de convergence.
- le tableau des solutions efficaces.
- le graphe des solutions efficaces.

Remarque :
IC(indice de convergence) représente le percentage de distance parcourue vers le point idéal,commençant le départ dans le point qui représente la distance minimal dans la population à l'itération 0 et

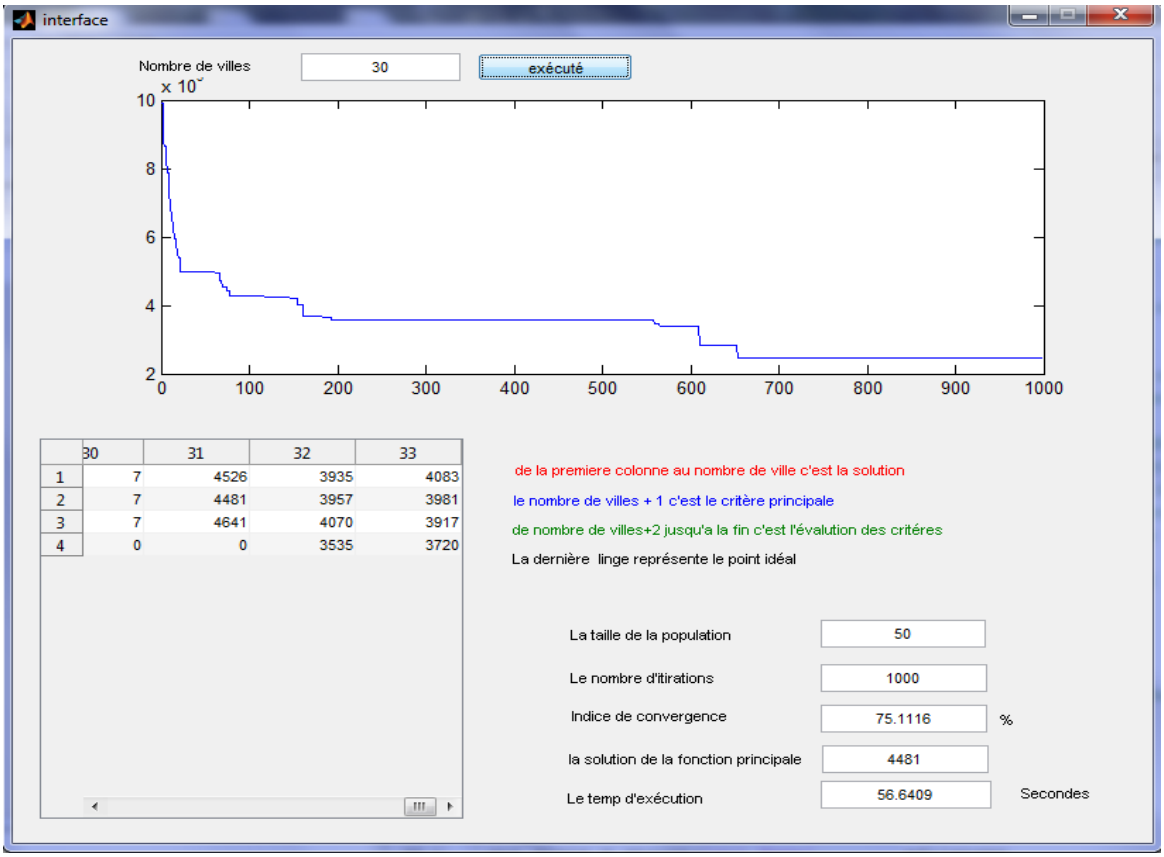

FIG. 5.5 – Application

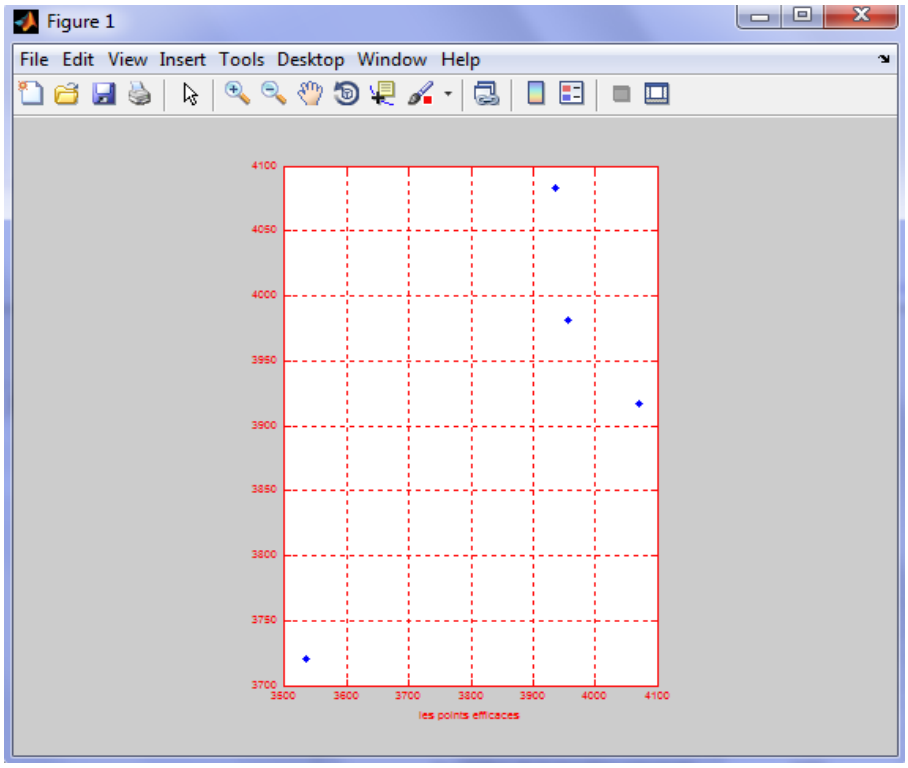

FIG. 5.6 – Graphe de l'application

l'arrivé dans le point qui représente la distance minimal de la population à l'itération final(k).

$$IC = \frac{d_{min}(t=0) - d_{min}(t=k)}{d_{min}(t=0)} * 100$$

5.6 Résultats et interprétations des résultats

5.6.1 résultats

Dans cette partie nous avons exécuté l'application sous un ordinateur portable **QOMPAQ Presario CQ50** intel core(TM)2 duo CPU P8600@ 2.4 GHz 2.4Ghz et une RAM 2.00 Go.
Le tableau suivant présente et classifier tous les temps d'exécutions(minimal,maxima et moyen) et tous les indices de convergences par les tailles du population et les nombres d'itérations.

tableau des évaluations															
p		50				100				150					
v		t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
4	500	7.29	7.77	7.58	0	500	18.09	21.20	19.91	0	500	31.78	38.38	34.44	0
	1000	10.11	10.76	10.43	0	1000	26.27	28.79	27.69	0	1000	33.66	41.35	37.72	0
5	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	7.369	10.04	9.15	0	500	30.05	36.08	34.37	0	500	51.21	73.79	62.21	0
	1000	10.95	19.70	14.17	0	1000	51.30	63.48	57.50	0	1000	53.30	98.89	77.41	0
20	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	19.77	33.42	25.55	73.95	500	60.35	142.13	94.27	63.56	500	134.43	251.71	196.06	79.65
	1000	37	61	47.69	71.96	1000	69.88	195.64	138.09	71.69	1000	140.68	508.31	262.19	66.97
30	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	21.36	34.75	30.26	75.69	500	119.56	175.39	136.03	84.27	500	270.34	485.64	358.56	83.32
	1000	32.02	75.54	46.82	78.96	1000	107.41	301.87	191.28	76.36	1000	223	363	322.69	70.97
50	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	50.19	76.81	70.50	76.74	500	171.83	282.87	205.66	89.65	500	395.21	654.19	538.83	88.95
	1000	52.36	125.43	92.03	84.89	1000	229.53	520.34	406.62	83.08	1000	533.73	1058.53	761.26	90.95
70	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	81.16	105.	87.38	80.49	500	243.21	464.56	338.89	85.71	500	656.77	904.06	7741	77.42
	1000	73.68	168.95	130.97	71.76	1000	453	1183	654.03	88,68	1000	1456.5	1513	1484.14	83.85
100	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	100.24	159.83	123.64	86.18	500	401.59	770.15	540.46	83.01	500	1227.19	2735.24	1782.09	82.23
	1000	141.12	332.54	213.84	86.35	1000	577.41	1469.56	840.64	86.97	1000	1287.4	1941.23	1650.83	87.1
250	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	270.79	883.55	468.12	84.60	500	1373.32	3943.26	2825.48	86.19	500	2898	8814.12	6646.76	89.59
	1000	336.18	1282.12	657.47	83.08	1000	2725.31	4332.49	3034.83	84.78	1000	9987.37	12922.1	11516.15	85.84
500	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC	itéra	t min	t max	tmoy	IC
	500	4523.12	5120.32	4990.02	81.23	500	23102.8	27213.2	25051.6	84.43	500	32456.2	37564.1	35397.7	89.85
	1000	4054.99	7439.64	5747.31	82.89	1000	25612.1	29523.4	28829.7	91.96	1000	41235.01	45640.2	43380.23	86.05

TAB. 5.1 – Tableau des résultats

5.6.2 interprétation des résultats

Nous avons introduit ce nouveau algorithme qui travaille dans l'espace des critères, produisant en même temps un sous-ensemble de solutions non dominées selon la direction de convergence des points vers le point idéal . Le nombre réduit des solutions efficaces de problème MOTSP nous facilite le choix de la solution efface qui optimise le critère principale . la minimisation de la distance entre les points et le point idéal nous donne plus de chance d'avoir plus de points efficaces dans le domain réduit et le test d'efficacité nous permit de réduire plus la taille de l'ensemble et de gardé que les points efficaces ou changé les points non efficace par des points efficaces. l'avantage de cette algorithme est le nombre grand de villes qu'il peut traité(100,250,500 et plus) , les résultats suivant preuve que les temps d'exécutions de cette algorithme suivent des régressions polynomials suivantes :

Régressions			
Itérations/population	50	100	150
500	$y = 0.029x^2 - 3.46x$	$y = 0.144x^2 - 16.10x$	$y = 0.166x^2 - 7.425x$
1000	$y = 0.032x^2 - 3.177x$	$y = 0.166x^2 - 17.92x$	$y = 0,165x^2 + 4,802x$

TAB. 5.2 – Tableau des régressions

La variation de l'indice de convergence (IC) entre 63% et 91% , montre la convergence des points vers le point idéal et nous garantie une amélioration pertinente de résultat final et nous assure la convergence de l'algorithme.

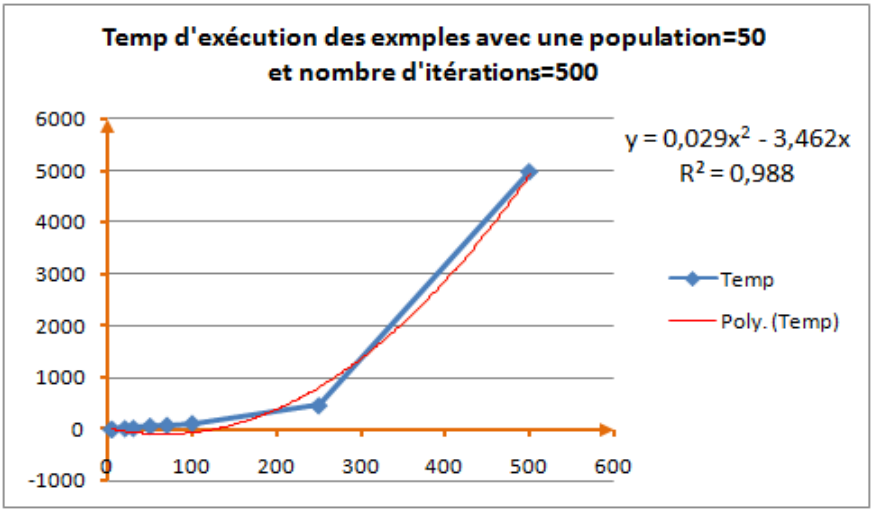

FIG. 5.7 – Graphe des exécutions 50/500

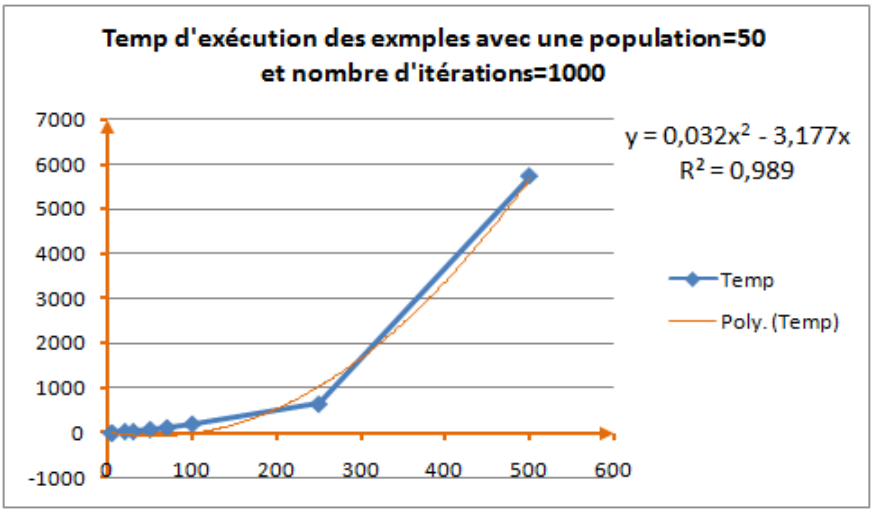

FIG. 5.8 – Graphe des exécutions 50/1000

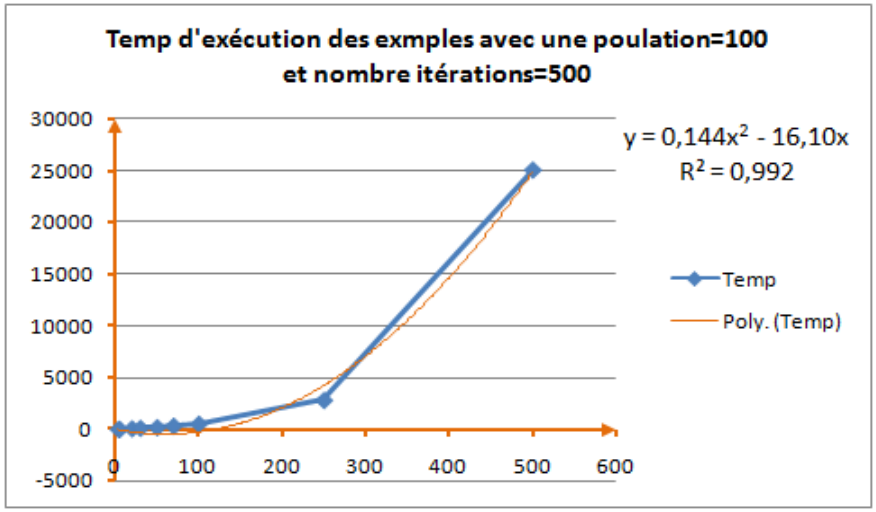

FIG. 5.9 – Graphe des exécutions 100/500

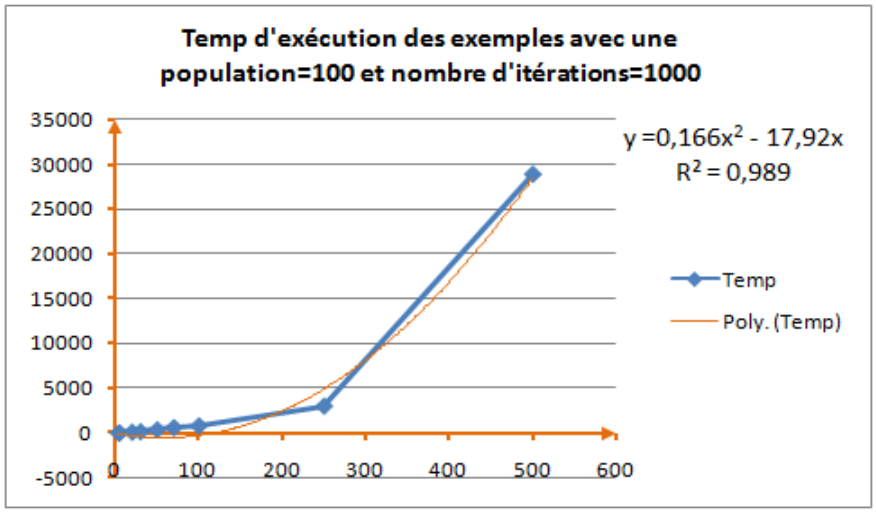

FIG. 5.10 – Graphe des exécutions 100/1000

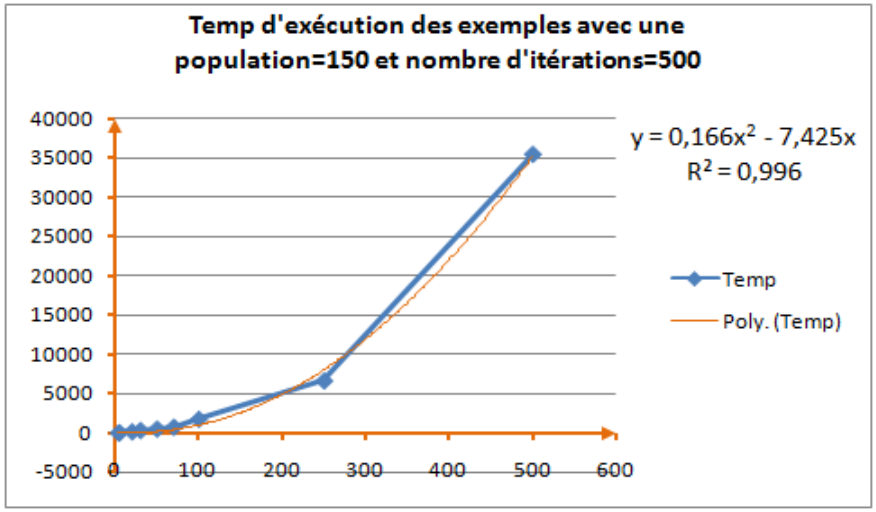

FIG. 5.11 – Graphe des exécutions 150/500

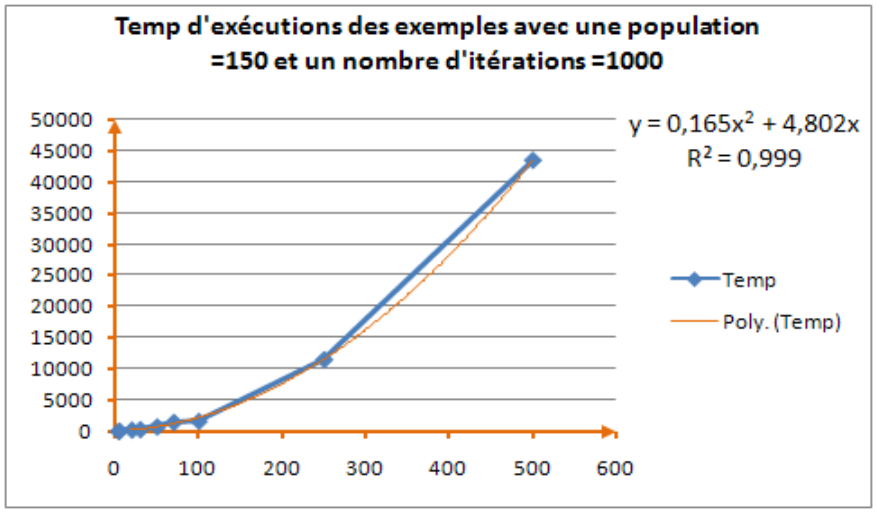

FIG. 5.12 – Graphe des exécutions 150/1000

5.7 Exemple illustratif

L'objectif de cette exemple illustratif est de comparer les résultats de l'application avec les résultats exactes du problème et pour cela nous choisirons un exemple de 5 villes , 2 critères et un critère principal. qui possède 5 !=124 solutions possible et $\frac{(5-1)!}{2} = 12$ de points sur l'ensemble des critères.

5.7.1 Exemple

le problème est décrit par

$$(MOTSP)\begin{cases} \min & Z_1 = \sum_{i=1}^{n}\sum_{j=1}^{j=n} c1_{ij}x_{ij} & ; \\ \min & Z_2 = \sum_{i=1}^{n}\sum_{j=1}^{j=n} c2_{ij}x_{ij} & ; \\ t.q & \sum_{i=1}^{n} x_{ij} = 1 & \text{i=1.....n}\,; \\ t.q & \sum_{j=1}^{n} x_{ij} = 1 & \text{j=1.....n}\,; \\ & \sum_{i,j\in S} x_{ij} <= |S|-1 & \text{S\{1.....n}\,;.\} \\ & x_{ij} \in \{0,1\} & \text{i=1.....n, j=1.....n.} \end{cases}$$

le problème principale est décrit par

$$(PL_E)\begin{cases} \min & \Phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{j=n} \phi_{ij}x_{ij} & \\ t.q & x_{ij} \in E(MOTSP) & \text{i=1.....n, j=1.....n}\,; \\ & x_{ij} \in \{0,1\} & \text{i=1.....n, j=1.....n.} \end{cases}$$

la matrice de premier objectif

$$(Z_1)\begin{pmatrix} \infty & 4 & 9 & 10 & 6 \\ 4 & \infty & 7 & 8 & 9 \\ 9 & 7 & \infty & 1 & 2 \\ 10 & 8 & 1 & \infty & 3 \\ 6 & 9 & 2 & 3 & \infty \end{pmatrix}$$

La solution optimal de premier objectif est la tourné 12345 et $Z_1^* = 21$.

la matrice de deuxième objectif

$$(Z_2)\begin{pmatrix} \infty & 8 & 7 & 2 & 9 \\ 8 & \infty & 11 & 1 & 6 \\ 7 & 11 & \infty & 5 & 3 \\ 2 & 1 & 5 & \infty & 4 \\ 9 & 6 & 3 & 4 & \infty \end{pmatrix}$$

La solution optimal de deuxième objectif est la tourné 24135 et $Z_2^* = 19$.

la matrice de critère principal

$$(\Phi)\begin{pmatrix} \infty & 2 & 9 & 1 & 9 \\ 2 & \infty & 4 & 10 & 4 \\ 9 & 4 & \infty & 8 & 2 \\ 1 & 10 & 8 & \infty & 8 \\ 9 & 4 & 2 & 8 & \infty \end{pmatrix}$$

La solution optimal de l'objectif principal est la tourné 21435 et $\Phi^* = 17$.

Le point idéal de cette exemple est de $Z_{ideal} = (21, 19)$.
Nous trouvons ci-joint dans l'annexe le tableaux de toutes les solutions possibles de l'exemple
.
les coordonnées des points des solutions sur les axes Z_1, Z_2 sont :

$\{(21, 37), (33, 32), (33, 28), (26, 30), (26, 24), (21, 26), (33, 33), (33, 26), (38, 30), (38, 19), (26, 23), (26, 28)\}$

les coordonnées des points efficaces sont :$Z_{eff} = \{(21, 26), (38, 19), (26, 23)\}$

les résultats de la fonction Φen replaçant les x des solutions efficaces respectivement sont :
$\Phi(x_{eff}) = \{31, 26, 31\}$.
Donc le point efficace qui minimise la fonction Φ est le point $Z(eff) = \{(38, 19)\}$.

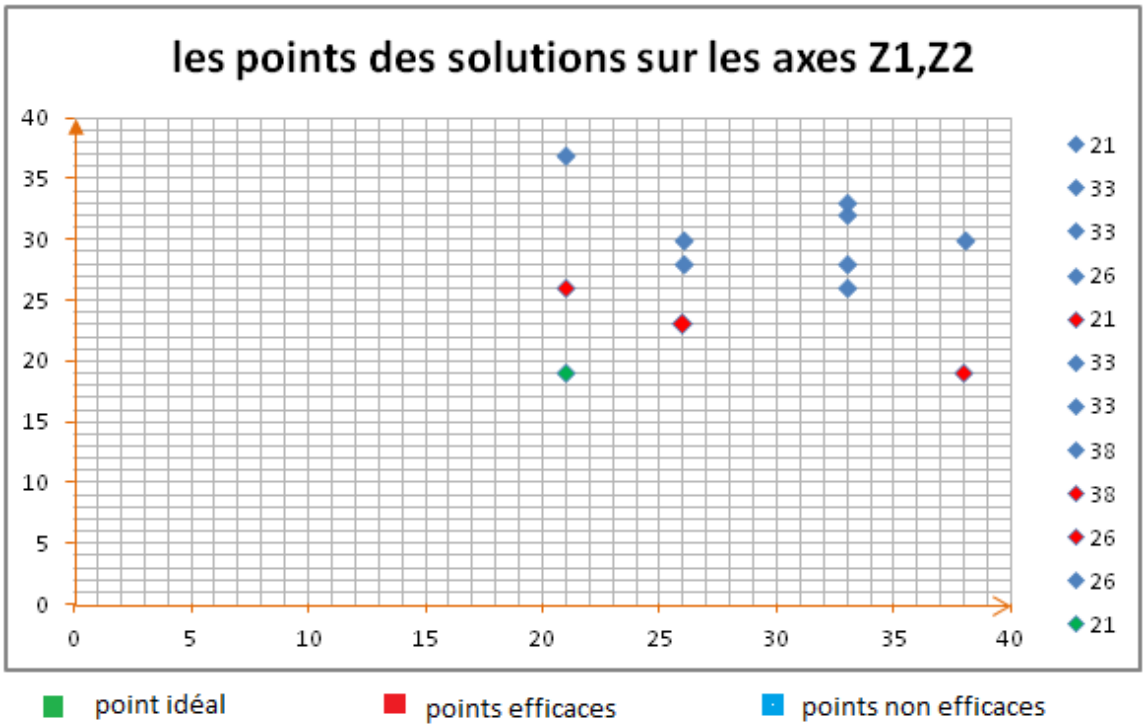

FIG. 5.13 – Representation des solutions sur les axes Z_1, Z_2

les résultats données par l'application pour cette exemple

- le temps d'exécution =12.221 secondes.
- $\Phi^* = 26$
- IC=0 ;
- la distance minimal des points est $\sqrt{41}$
- les points efficaces sont :$\{(21, 26), (26, 23), (38, 19)\}$
- les tournés sont respectivement :$\{(12435), (42135), (24135)\}$
- le point idéal est :$\{(21,19)\}$.
- le graphe des points efficaces.

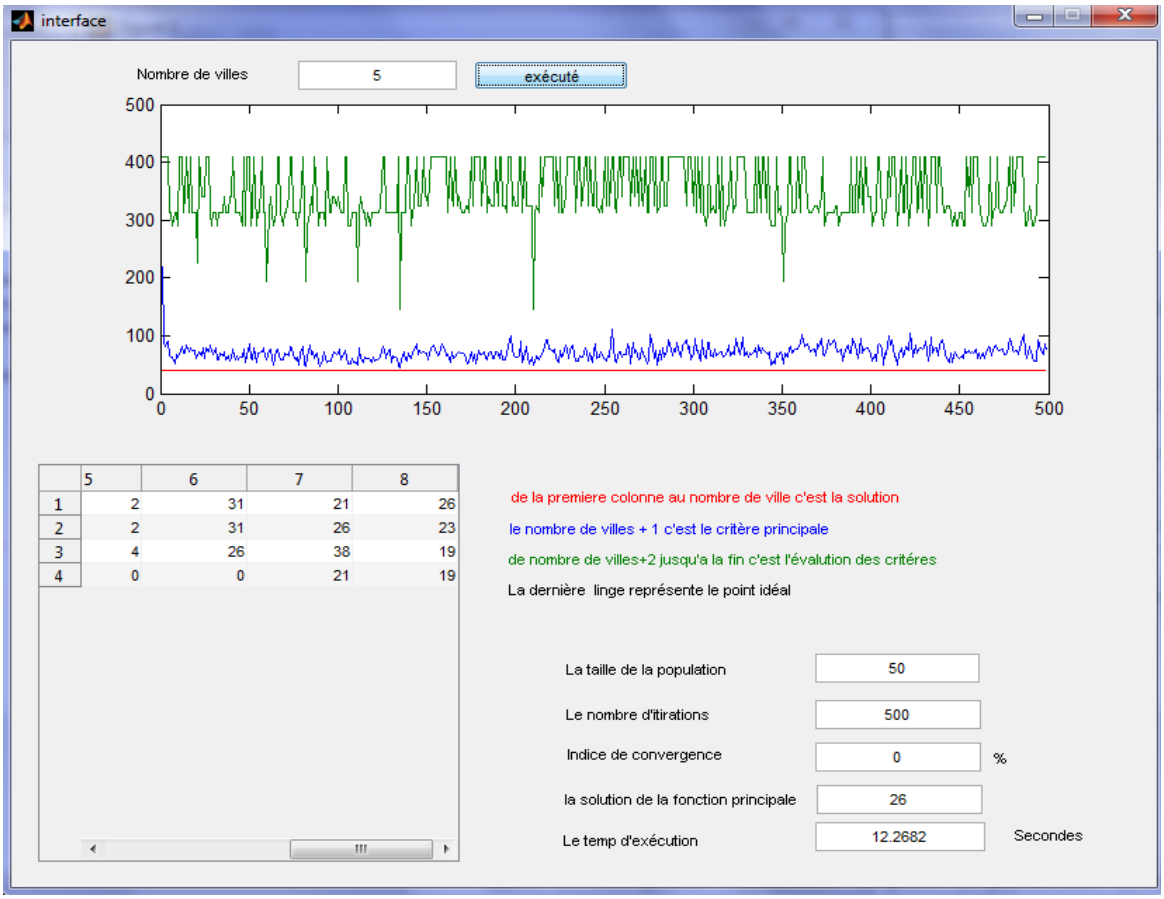

FIG. 5.14 – Résultats de l'exemple illustratif.

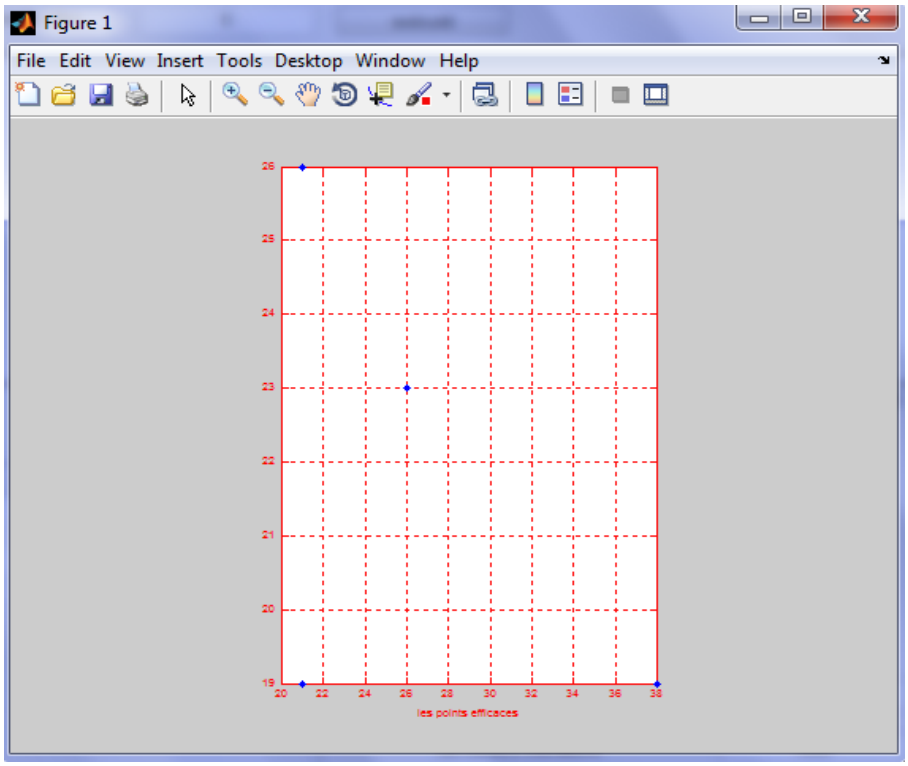

FIG. 5.15 – Graphe des points efficaces

5.8 Conclusion

dans cette partie nous avons réussie a présenté en détaille l'algorithme et l'application. Le temps d'exécution du programme dépend du type de machine utilisée et de la taille considérable des données du problème.

Ces résultats sont très encourageants. Bien qu'appliqué à des taille très grande , les régressions polynomial nous permet d'avoir une vision sur les grands tailles plus que 500 villes et l'indice de convergence nous assure une bonne amélioration de résulta final sur tous quant on possède une bonne solution de départ, ce qu'est le cas pour dans la réalité.

cette méthode et cette application nous donne plusieurs idées , pour l'optimisation d'un critère sur l'ensemble des solutions efficaces.

Conclusion générale

Les problèmes d'optimisation à objectifs multiples sont très variés et correspondent à des situations de décision très difficiles. L'objectif principal de notre mémoire a été de contribuer à l'étude de l'un des problèmes d'optimisation multi-objectifs à variables discrètes (problème de voyageur de commerce TSP). Le travail présenté s'articule autour de deux volets : la recherche des solutions efficaces d'un problème de programmation multi-objectifs en variables entières et l'optimisation d'un critère principal sur l'ensemble des solutions efficaces d'un tel problème.

Dans le premier chapitre nous avons présenté un panorama sur l'optimisation multi-objectifs et multi-objectifs combinatoires.

Dans le deuxième chapitre nous avons fait une présentation de différents méthodes de résolution des problèmes multi-objectifs .

Dans le troisième nous avons présenté en détaille les algorithmes génétiques , parce que ils sont utilisé dans le développement de notre algorithme et comme notre travaille a comme objectif l'optimisation d'un critère sur l'ensemble des solution efficaces, nous avons présenté dans le quatrième chapitre quelque méthodes d'optimisation d'un critère sur l'ensemble des solutions efficaces (Yamamoto , Eker et song) , et dans le dernier chapitre nous avons présenté notre algorithme qui s'articule sur la minimisation de la distance entre les points réalisables et le pints idéal.

L'utilisation des algorithmes génétiques pour un problème de voyageur de commerce nous a permit de résoudre pas mal de problèmes complexes ,par exemple la quadraticité de la fonction du distance a minimisé et la réalisabilité des solutions.

Cette méthode est implémenté sur machine sous le logiciel matlab 7, puis des exécutions sont faite sur notre application.Ces exécutions ont montré une bonne convergences des points de la population vers le point point idéal, cette convergence nous donne une grande assurance sur l'efficacité des points resté dans l'ensemble et le teste d'efficacité mit après, nous filtre les points efficaces et il change les non efficaces par d'autres efficaaces.A la fin, la taille réduite de cette ensemble nous permit de choisir facilement par evaluation le point efficace qui minimise la fonction principale.

Ce travail ouvre des perspectives de recherche nombreuses et intéressantes, parmi lesquelles nous privilégions les suivantes :

- Utlisation des algorithmes génétiques pour agrandir la taille des problèmes a résoudre pour quelques méthodes exactes qui optimise un critère sur l'ensemble des solutions efficaces (SEEVD,OLFIES).

- Adapter l'algorithme à d'autre problèmes d'optimisation multi-objectifs, tels que le problème de rotations de véhicules , sac à dos . . .
- Amélioration de l'algorithme en changent la partie des algorithmes génétiques par une méthode exacte.

Bibliographie.

Présenté par : Ali ZAIDI *Sous la direction de : CHAABANE Djamel, Professeur à l'USTHB*

Bibliographie

[1] A. Hertz, B. Jaumard, C. R., and Filho., W. F. A multi-criteria tabu search approach to cell formation problems in group technology with multiple objectives. *RAIRO Operations Research, 28(3)* (1994.), 303–328.

[2] Abdelaziz, F., K. S., and Chaouachi, J. Meta-heuristics : Advances and trends in local problems,. *Kluwer Academic Publishers.* ((1999)), 205–212.

[3] Allenson, R. Genetic algorithms with gender for multi-function optimisation. *Technical Report EPCC-SS92-01, Edinburg Parallel Computing Center, Edinburg, Scotland.* (1992).

[4] Amokrane, S. Algorithme génétique pour le problème d'ordonnancement dans un synthèse de haut niveau pour contrôleurs dédiés. Master's thesis, Université de Batna, Algeria, 2002.

[5] azarm. s. multiobjective optimum desig. *www.glue.umd.edu/azarm/optimum_notes/multi/multi.html* (1996).

[6] barichard. v. *Approches hybrides pour les problémes multiobjectifs.* PhD thesis, Ecole Doctorale d'Angers, 24 Novembre 2003.

[7] Barnier, N., B. P. Optimisation par algorithme génétique sous contraintes. *Technique et science informatiques 18* (1999).

[8] Benson, H. Optimization over the efficient set,. *Journal of Mathematical Analysis 98* (1984), 562–580.

[9] bonomi. e ; j. l. lutton. *Le recuit simulé, pour la science.* number 129, pages 68-77, July 1988.

[10] clerc. m ; p. siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires. Tech. rep., France Télécom R D ; Université Paris 12, 17/09/2004.

[11] Coello., C. C. Using a min-max method to solve multiobjective optimization problems with genetic algorithms. *In IBERAMIA'98, Springer Verlag. 1993,* (1998.), 303–314.

[12] coello coello. c. a. An updated survuey of g.a based multiobjective optimization techniques. *Technical report Lania-RI-98-08 Laboratorio Nacional Avanzada, Xalapa, Veracruz, Mexico* (Deceùmber 1998).

[13] COLLETTE. Y ; SIARRY P. *Multiobjective Optimization.* Cranfield Bedford MK43 0AL UK, August 2003.

[14] CURRENT, J., AND MARSH, M. Multiobjective transportation network design and routing problems : taxonomy and annotation. *European Journal of Operational Research, 65* ((1993)), 4–19.

[15] DAHL, G., J. K., AND LOKKETANGEN, A. A tabu search approach to the channel minimization problem. *In Liu, G., Phua, K.-H.,Ma, J., Xu, J., Gu, F., and He, C., editors, Optimization - Techniques and Applications, ICOTA'95, volume 1, Chengdu, China. World Scientific.* (1995), 369–377.

[16] DAUER, J. Optimization over the efficient set using an active constraint approach. *ZOR-Methods and Models of Operations Research 35* (1994), 541–563.

[17] DEGOUTIN, F., E. G. X. Un retour d'expérience sur la résolution de problèmes combinatoires bi-objectifs. *Proceedings de la Conférence sur la Programmation Mathématique MultiObjectif (PM2O).* (2002).

[18] DELORME. X. *Modélisation et résolution de problémes liés à léxploitation dínfrastructures ferroviaires.* PhD thesis, lúniversité de Valenciennes et du Hainaut-Cambrésis, 2003.

[19] DRÉO, J., P. A. S. P. E. T. E. Métaheuristiques pour l'optimisation difficile. *Paris : Eyrolles.* (2003).

[20] DRÉO, J., P. A. S. P. T. E. *Métaheuristiques pour l'optimisation difficile.* EYROLLES, 2005.

[21] DUVIGNEAU, R. *Introduction aux méthodes d'optimisation sans Gradient pour l'optimisation et le contrôle en mécanique des fluides.* INRIA, Sophia-Antipolis, France, 2006.

[22] EHRGOTT, M., E. G. X. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Specktrum , 22* (2000), 425–460.

[23] EHRGOTT, M., E. G. X. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Specktrum , 22,.* (2000), 425–460.

[24] EHRGOTT, M. *Multicriteria optimization*, vol. 2005. Springer.

[25] ESCHENAUER. H ; J. KOSKI, A. O. *Multicriteria design optimisation : Procedures and Applications.* Springer, 1990.

[26] FEO T. A ; RESENDE M. A probabilistic heuristic for a computationally difficult set covering problem. Tech. rep., Opérations Research Letters 8,67-71, 1988.

[27] FRIESZ, T., A. G. M. N. N. K. S. S., AND TOBIN, R. The multiobjective equilibrium network design problem revisited : A simulated annealing approach. *European Journal of Operational Research, 65* (1993), 44–57.

[28] FUJIMURA, K. Path planning with multiple objectives. *IEEE Robotics and Automation Society Magazine, 3(1)* (1996), 33–38.

[29] Fujita, K., H. N. A. S. K. S. Y. H. In design engineering technical conferences detc'98,. *Multi-objective optimal design of automotive engine using genetic algorithm., Atlanta, Georgia.* (1998), 1–11.

[30] G, O. I. H. L. Metaheuristics : a bibliography. *Annals of Operations Research* (63 :513-623, 1996).

[31] gagne c ; m. gravel. optimisation multi-objectifs a l'aide d'un algorithme de colonie de fourmis. In *Departement d'informatique et de mathimatique universite du Quebec. Canada G7H 2BI Courriel : caroline_gagne@uqac.ca. marc_gravel@uqac.ca* (WILSON L. PRICE Faculte des Sciences de I 'administration. Universite Laval Sfe-Foy, Quebec. Canada GIK 7P4 Courriel : wilson.price@fsa.ulavaica).

[32] H. Isermann, E. S. Computational experience concerning payoff tables and minimum values over the efficient set. *European Journal of Operational Research 33* (1987), 91–97.

[33] haimes. y. y ; w. a. hall ; h. t. freedman. A multiobjective optimization in water resources systems. *Elsevier Scientific* (1975).

[34] Halhal, D., W. G. O. D., and Savic, D. Water network rehabilitation with a structured messy genetic algorithm. journal of water resources planning and management,123(3). 137–146.

[35] holland. j.h. *Adaptation in natural and artifcial systems.* Phd thesis, University of Michigan Press, 1975.

[36] Hwang, C., and Masud, A. Multiple objective decision making - methods and applications. *In Lectures Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin. 164* (1979).

[37] J. G. Ecker, H. S. Optimizing a linear function over an efficient set. *Journal of Optimization Theory and Applications 83* (1994), 541–563.

[38] J.G. Ecker, I. A. K. Finding efficient points for multi-objective linear programs. *Mathematical Programming 8* (1975), 75–377.

[39] Jones, B., C. W., and Lyrintzis, A. Aerodynamic and aeroacoustic optimization of airfoils via a parallel genetic algorithm. *In Proc. of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, number AIAI-98-4811.* (1998), 1–11.

[40] J.P. Dauer, A. S. Constructing the set of efficient outcome values in multiple objective linear programs. *European Journal of Operational Research 46* (1990), 358–365.

[41] keeney. r. l ; h. raaiffa. Decision with multiple objective : Preference and value tradoff. *Cambridge University Press* (1993).

[42] Kursawe, F. A variant of evolution strategies for vector optimization. *In Schwefel, H. and Manner, R., editors, Parallel Problem Solving from Nature, of Lecture Notes in Computer Science, Berlin. Springer-Verlag. 496* (193-197).

[43] Lemesre, J., D. C. e. T. E.-G. A parallel exact scheme to solve bicriteria problems. *Pooceedings of the International Multiobjective Programming and Goal Programming Conference (MOPGP'04). Hammamet (Tunisia).* (2004).

[44] Lis, J., and Eiben, A. A multi-sexual genetic algorithm for multi-objective optimization. *In Fukuda, T. and Furuhashi, T., editors, Int. Conf. on Genetic Algorithms ICGA, Nagoya, Japan.* (1996), 59–64.

[45] Loughlin, D., and Ranjithan, S. The neighborhood contraint method : A genetic algorithmbased multiobjective optimization technique. *In Back, T., editor, Seventh Int. Conf. on Genetic Algorithms ICGA'97, San Mateo, California. Morgan Kaufmann.* (1997), 666–6773.

[46] Lutton, E. *Etat de l'art des algorithmes génétiques.* INRIA, Sofia-Antipolis, France, 1994.

[47] M. Shigeno, I. Takahashi, Y. Y. Minimum maximal flow problem - an optimization over the efficient set. *Journal of Global Optimization 25* (2003), 425–443.

[48] miettienen. k. m ; m. m. mäkelä. Proper pareto optimality in non convex problems, characterization with tangent and normal cones, technical report. *Jyväskylä University* (November 1998).

[49] Murata, T., and Ishibuchi., H. A multi-objectives genetic loacal search algorithm and its application flow-shop scheduling. *IEEE Transaction System, 28(3) .* (1998), 392–403.

[50] Muu, L. D. Computational aspects of optimization problems over the efficient set. *Vietnam Journal of Mathematics 23* (19995), 85–106.

[51] Nagar, A., H. J., and Heragu, S. Multiple and bicriteria scheduling : A literature survey. *European Journal of Operational Research,81* ((1995)), 88–104.

[52] nelder. j. a ; r. mead. A simplex method for function minimization. *Computer Journal 7* (1965), 308–313.

[53] Nguyen, N. An algorithm for optimizing a linear function over the integer efficient set. *Konrad-Zuse-Zentrum fur Informationstechnik Berlin* (1992).

[54] Obayashi, S., T. S., and Takeguchi, Y. Niching and elitist models for multiobjective genetic algorithms. *In Parallel Problem Solving from Nature PPSN'5,* ((1998)), 260–269.

[55] Papadimitriou, C. H., e. S. K. Combinatorial optimization : algorithms and complexity. *Prentice-Hall.* (1982).

[56] Park, Y., and Koelling, C. An interactive computerized algorithm for multicriteria vehicle routing problems. *Computers and Industrial Engineering, 16* ((1989)), 477–490.

[57] Pham, D., and Karaboga, D. *Intelligent optimisation techniques.* Springer-Verlag, London, 2000.

[58] PHILIP, J. Algorithms for the vector maximization problem. *Mathematical Programming* (207-229), 1972.

[59] PIRLOT. M. Métaheuristiques pour lóptimisation combinatoire : un aperçu général.dans j. teghem et m. pirlot, éditeurs, optimisation approchée en recherche opérationnelle - recherches locales, réseaux neuronaux et satisfaction de contraintes, pages 25-55. *Hermés Science Publications* (Paris, 2002).

[60] PRZYBYLSKI, A., G. X. E. E. M. Seek and cut algorithm computing minimal and maximal complete efficient solution sets for the biobjective assignment problem. *Pooceedings of the International Multiobjective Programming and Goal Programming Conference (MOPGP'04). Hammamet (Tunisia).* (2004).

[61] REARDON. B. J. Fuzzy logic us niched pareto muli-objective genetic algorithm optimization : Part i : Schaffer's problem. Technical report la- ur-97-3675, Los Alamos National Laboratory, 1997.

[62] RICHARDSON, J., P. M. L. G., AND HILLIARD, M. Some guidelines for genetic algorithms with penalty functions. *In Third Int. Conf. on Genetic Algorithms ICGA'3.* (1989), 191–197.

[63] SAIT. S. M ; H. YOUSSEF. *iterative computer algorithms with applications ininginee-ring : solving combinatorial optimization problems.* IEEE computerssociety, 1999.

[64] SAKAWA. M ; H. YANO. *An interactive fuzzy satisfcing method for multiobjective linear programming problems witth fuzzy parametres*, vol. volume 28. Fuzzy sets and Systems, 1988.

[65] SASTRY, K., G. D. *Genetic algorithms.* University of Illinois, USA.

[66] SEN, T., R. M., AND DILEEPAN, P. A branch and bound approach to the bicriterion scheduling problem involving total flowtime and range of lateness. *Management Science,34(2)* (254-260), 1998.

[67] SERAFINI, P. Simulated annealing for multiple objective optimization problems. *In Tenth Int. Conf. on Multiple Criteria Decision Making,Taipei.* ((1992)), 87–96.

[68] SHAW, K., AND FLEMING, P. Initial study of multi-objective genetic algorithms for scheduling the production of chilled ready meals. *In Mendel'96 2nd Int. Conf. on Genetic Algorithms,Brno, Czech Republic.* (1996).

[69] SIARRY. P. La méthode du recuit simulé en électronique. *Habilitation report, Paris-sud University(Orsay)* (1994).

[70] SIARRY. P. Optimisation et classification de données. *Lectures from the dea at Paris University* (1999).

[71] SRINIVAS, N., AND DEB., K. Multiobjective optimization using nondominated sorting in genetics algorithms. *Evolutionary Computation, 2(3) :* (1994.), 221–248.

[72] SURRY, P., R. N., AND BOYD, I. A multi-objective approach to constraint optimisation of gas supply networks : The comoga method. *In Fogarty, T., editor, Evolutionary Computing, AISB Workshop, LNCS, Sheffield, U.K. Springer-Verlag.* (1995), 166–180.

[73] TAILLARD. E. *La programmation á mémoire adaptative et les algorithmes pseudo-gloutons : nouvelles perspectives pour les métaheuristiques.* PhD thesis, Thése dḥabilitation, Université de Versailles Saint-Quentin-en-Yvelines, Versailles, France, 1998.

[74] TAILLARD. E. Principes dímplémentation des métaheuristiques. dans j. teghem et m. pirlot, éditeur, optimisation approchée en recherche opérationnelle - recherches locales, réseaux neuronaux et satisfaction de contraintes, pages 57-79. *Hermés Science Publications* (Paris, 2002).

[75] TODD, D., AND SEN, P. A multiple criteria genetic algorithm for container loading. *In Back,T., editor, Seventh Int. Conf. on Genetic Algorithms ICGA'97,San Mateo,California. Morgan Kaufmann.* (1997), 674–681.

[76] ULUNGU, E., AND TEGHEM, J. Multi-objective combinatorial optimization problems : A survey. *Foundation of Computing and Decision Science , 20* ((1994)).

[77] ULUNGU, E. L., E. T. J. The two phases method : an efficient procedure to solve bi-objective combinatorial optimization problems. *Foundation of Computing and Decision Science , 20,* (1995), 149–156.

[78] VALLÉE, T., AND YILDIZOGLU, M. *Présentation des algorithmes génétiques et de leurs applications en économie.* Université de Nantes, 2001.

[79] VELDHUIZEN, D. V., S. B. M. R.-L. G., AND TERZUOLI, A. Finding improved wire-antenna geometries with genetic algorithms. *In Chawdhry, P., Roy, R., and Pant, P., editors, Soft Computing in Engineering Design and Manufacturing,London. Springer Verlag.* (1997), 231–240.

[80] VELDHUIZEN. D. A. VAN. *Multiobjective Evolutionary Algorithms :Classifications, Analyses and New Innovation.* PhD thesis, Graduate School of Engineering ; Air Force Institute of Technology ; Wright Patterson AFB, Ohio,USA, Janvier ; 1999.

[81] WARBURTON, A. Approximation of pareto optima in multiple-objective shortest-path problems. *Operations Research, 35* ((1987)), 70–79.

[82] WASILEWSK, A. *Genetic Algorithms.* 2006.

[83] WEINBERG, B. A co-evolutionary meta-heuristic for the assignment frequencies in cellular networks. *In EvoCop.* (2001).

[84] WHITE, D. The set of efficient solutions for multiple-objectives shortest path problems. *Computers and Operations Research, 9* (1982), 101–107.

[85] Y. COLLETTE, P. S. *Optimisation multiobjectif,Etude (broché). Paru en 10* (2002).

[86] YAMAMOTO, Y. Optimization over the efficient set. *Overview, Kluwer Academic Publishers* (2001).

[87] ZADEH L. A. *Fuzzy sets*, vol. volume 8. Information and Control, 1965.

[88] ZELENY, M. Multiple criteria problem solving. *McGraw-Hill, New York.* ((1982)).

[89] ZHOU, G., AND GEN, M. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research,114* ((1999)), 141–152.

Résumé

Résumé *: Optimisation sur l'ensemble des solutions efficaces pour quelques problèmes d'optimisation combinatoire.*

L'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines applicatifs, tel que le traitement d'images, la conception de systèmes, la conception d'emplois du temps,... Ces problèmes sont, en majorité qualifies de difficiles, car pour leur résolution, il n'est pas en général possible de fournir dans tous les cas des solutions en un temps raisonnable. Il en est ainsi du problème de voyageur de commerce.Dans le cas où plusieurs critères sont à optimiser, la problématique se complique davantage car ces critères sont souvent contradictoires. l'optimisation d'un critère principal sur l'ensemble des solutions efficaces complique plus le problème .Des méthodes d'optimisation sont utilisées,pour résoudre ces problèmes mais la limitation de ces méthodes exactes à des tailles réduite,les rend non applicable dans la réalité. Dans ce mémoire, nous développons pour le problème d'optimisation d'un critère sur l'ensemble des solution efficaces d'un problème de voyageur de commerce multi-objectifs un algorithmes fondé sur les algorithmes génétiques . Cet algorithme met en évidence l'importance de réduire la distance entre les points et le point idéal.Cet algorithme montre aussi l'importance de réduire l'ensemble des solutions efficaces .Cet algorithme est implémenté et les résultats expérimentaux obtenus sont discutés.

Mots clés : *problèmes d'optimisation multi-objectifs, algorithmes génétiques , voyageur de commerce, optimisation combinatoire, optimisation d'un critère principal, solutions efficaces*