République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire Présenté

En vue d'obtension du Diplôme de Master

Mathématiques Appliquées

Spécialité : Modélisation Mathématiques et Techniques de Décision

Par : CHEBIL Djedjiga
Et : BELGAID Asma

Algorithme d'Estimation de Distribution (EDA) pour l'Optimisation d'une fonction sur l'Ensemble des Solutions Efficaces d'un Problème Multiobjectif en Nombres Entiers (MOILP)

Soutenu à l'Université Abderrahmane Mira de Béjaia, Le 03/07/2023.
Devant le jury composé de :

D^{r} Belkacem. Brahmi	M.C. classe A	Président	à l'UAMB - Bejaia.
D^{r} Larbi. ASLI	M.C. classe A	Encadrant	à l'UAMB - Bejaia
D^{r} Ali. ZAIDI	M.A. classe A	Co-encadrant	à CRLCA- Bejaia.
M^{r} Abdelhak. LAOUAR	M.A. classe A	Examinateur	à l'UAMB - Bejaia.
M^{elle} Zahra. BOUZERIA	Doctorante	Examinateur	à l'UAMB - Bejaia.

Année Universitaire 2022 – 2023

Remerciement

Nous tenons tout d'abord à remercier le bon Dieu tout puissant d'avoir guidé nos pas vers les portes du savoir, en nous donnant le courage et la volonté nécessaires pour mener ce travail à terme.

Nous tenons à exprimer nos plus vifs remerciements à notre promoteur, le Dr L. Asli, ainsi qu'au Dr A. Zaidi, qui nous ont guidés et orientés tout au long de ce projet. Nous leur exprimons notre profonde gratitude pour l'attention et le temps qu'ils nous ont consacrés, ainsi que pour les précieux conseils qu'ils nous ont prodigués.

Nous avons eu un énorme plaisir de travailler avec eux .

Nous souhaitons également remercier les membres du jury qui nous ont fait l'honneur d'examiner ce travail.

Nous témoignons une reconnaissance particulière aux enseignants qui nous ont tenu et transmis le savoir, l'esprit de recherche et de perseverance dans nos etudes durant notre cursus.

Enfin, nous tenons à exprimer notre reconnaissance envers toutes les personnes anonymes qui ont participé à notre étude en fournissant des informations. Leur contribution a été essentielle à la réussite de ce mémoire.

Nous tenons à remercier tous ceux qui ont joué un rôle dans la réalisation de ce mémoire de fin d'étude. Votre soutien, votre expertise et votre présence ont été inestimables et ont contribué à faire de cette expérience un succès.

Dédicaces

Je dédie ce modeste travail à :

Mes chers parents, que nulle dédicace ne puisse exprimer ce que je leur dois, pour leur bienveillance, leur affection et leur soutien, ...

Je dédie ce travail à la mémoire de mes très chers grands-parents. Puisse Dieu, le tout puissant, les accueillir dans sa sainte miséricorde !

Mes chers frères Seif El Dine, Youcef que je remercie pour leurs encouragement et leurs soutien.

À ma sœur unique Nour El Houda, je dédie ce mémoire en reconnaissance de ton impact positif dans ma vie. Ta présence a été un phare de soutien et d'encouragement, et je suis honorée de t'avoir comme sœur.

À mes très chéres amies Sara,Chahinez, Lehna, Tina, Hadda, Nihad, Khalida merci pour votre encouragement, amitié, votre affection, Votre présence a illuminé chaque étape de cette aventure et a rendu cette expérience encore plus significative et mémorable. Merci du fond du cœur pour votre soutien inestimable.

Ma chère binôme Djedjiga que je remercie pour son soutien et sa patience pour réaliser ce modeste travail, ton amitié indéfectible et ton affection précieuse tout au long de notre parcours universitaire.

À Tous ceux qui m'aiment.

Asma

Dédicaces

Au nom de Dieu,

Je dédie humblement ce travail aux personnes qui me sont les plus chères :

En mémoire de mon père, ainsi qu'à mes tantes Djohar et Zahia,

En mémoire de mon amie Imad Bourai, qui nous a quittés en début d'année,

À ma très chère maman,

À mes sœurs Dihia et Wezna,

À mon frère Koceila,

À Redouane et Khalef,

À mon cher Amir,

À mes amis Sara, Tina, Hadda, Romaissa, Khalida, Nihad, Lahna, Laeticia, en particulier Chahinaz, et à tous ceux qui m'ont soutenu pendant ces 5 années.

Je tiens à vous exprimer ma profonde gratitude. Votre présence, votre soutien et votre amour ont été d'une importance capitale tout au long de mon parcours.

À ma mère, merci d'être mon roc, mon modèle et mon soutien inconditionnel. À mes sœurs, merci d'avoir soutenu mes décisions malgré tout.

À mes encadrants, Dr L. Asli, Dr A. Zaidi, merci pour votre expertise et vos précieux conseils.

À mes petites nieces Iliana et Milina, merci d'avoire amené la joix a mon coeur a chaque fois.

À Redouane, qui m'a soutenu depuis le début de mes études jusqu'à la fin.

À ma chère binôme Asma, merci de m'avoir accompagné dans cette aventure.

Djidji

Table des matières

Table des figures

Liste des algorithmes

Liste des tableaux

Notations

Symbole	signification
A	Matrice des coefficients des variables dans les contraintes linéaires .
AG	Algorithme génétique.
b	Vecteur des bornes.
c	Matrice des coefficients de la fonction objectif.
D_1^s	La population de solutions qui sélectionne les meilleurs individus.
D_0	La population d'individu.
$\mathcal{D}$	Ensemble des solutions réalisables entiers.
EDA	Algorithmes d'Estimation de Distributions.
IDE	Environnement de Développement Intégré
$MOILFP$	Problèmes Linéaire Fractionnaires Entiers Multi-objectifs.
$MOILP$	Problèmes Linéaires Multi-objectifs en Nombres Entiers.
N_E	Arête efficace
S	Espace de recherche.
X_E	Ensemble des solutions efficaces.
X	Ensemble des décisions.
Y_N	Ensemble des points non dominés.
$\mathbb{Z}$	Ensemble des nombre entiers relatifs.

Introduction générale

La recherche opérationnelle vise principalement à aider à prendre des décisions pour une gestion efficace, rationnelle et logique. La modélisation traditionnelle des problèmes de recherche opérationnelle se limitait aux modèles d'optimisation à objectif unique, mais une nouvelle vision plus réaliste consiste à modéliser les problèmes en prenant en compte tous les objectifs pertinents. Cette approche de la multiplicité des objectifs est raisonnable et conduit à des méthodes de gestion plus réalistes que les méthodes classiques [9].

L'optimisation multi-objectif est un domaine de recherche important pour les scientifiques et les ingénieurs car la plupart des problèmes réels impliquent plusieurs objectifs, et il reste encore beaucoup de questions en suspens dans ce domaine. Différentes techniques ont été développées pour traiter les problèmes à objectifs multiples en recherche opérationnelle, allant de la combinaison naïve des objectifs en un seul objectif à l'utilisation de la théorie des jeux pour coordonner l'importance relative de chaque objectif et le comportement de différents intervenants [6].

Il existe des situations où les décideurs ne nécessitent pas toutes les solutions efficaces d'un problème de programmation multi-objectifs, mais uniquement celles qui atteignent l'optimum pour un objectif différent de ceux déjà définis. Cela implique de rechercher la solution optimale d'un critère parmi l'ensemble des solutions efficaces du problème multi-objectifs. L'objectif de notre travail est d'étudier les problèmes de programmation mathématique linéaires multi-objectifs à variables discrètes, également connus sous le nom de MOILP (Multiple Objective Integer Linear Programming) [6].

L'optimisation est un domaine important de l'informatique et des mathématiques appliquées, avec de nombreuses applications dans les domaines de l'ingénierie, de la finance, de la logistique et de la planification, entre autres. L'optimisation multi-objectif en nombre entier (MOILP) traite les problèmes d'optimisation où plusieurs objectifs doivent être optimisés simultanément tout en respectant des contraintes qui exigent des solutions en nombres entiers.

Les MOILP sont des problèmes d'optimisation où plusieurs objectifs sont à considérer simultanément, et où les variables de décision sont des nombres entiers. Ces types de problèmes sont couramment rencontrés dans diverses applications industrielles telles que la planification de la production, la gestion de la chaîne d'approvisionnement, la conception de réseaux de transport, entre

autres.

Les méthodes traditionnelles d'optimisation pour les MOILP utilisent souvent des approches basées sur l'énumération exhaustive de l'ensemble des solutions possibles, ou sur des techniques heuristiques telles que la méthode du poids ou les méthodes de pondération des objectifs introduites par Tchebychev. Cependant, ces méthodes peuvent être coûteuses en temps de calcul ou peuvent ne pas garantir la découverte de toutes les solutions optimales [25].

Dans ce contexte, les algorithmes d'Estimation de Distribution (EDA) sont devenus une alternative intéressante aux méthodes d'optimisation classiques. Les EDA sont des algorithmes évolutionnaires qui utilisent des modèles probabilistes pour représenter et évoluer la distribution de probabilité des solutions possibles. Les EDA ont montré une grande efficacité dans la résolution des problèmes d'optimisation complexes et en particulier dans les problèmes MOILP [20].

Les Algorithmes d'Estimation de Distribution (EDA) ont été introduits dans les années 1990 par le chercheur allemand Dirk Thierens [20], en tant que méthodes évolutionnaires pour l'optimisation de fonctions continues et multi-modales. Leurs objectifs est de surmonter les limites des algorithmes génétiques (AG) et des stratégies d'évolution (ES), qui souffraient souvent de convergence prématurée et d'une lente convergence vers l'optimum global [32].

Le premier EDA a été proposé en 1996 par Lozano et al [26]. sous le nom de "EDA with marginal histograms" (EDA avec des histogrammes marginaux). Ils ont utilisé des histogrammes marginaux pour estimer la distribution marginale de chaque variable de décision et ont généré de nouvelles solutions en échantillonnant aléatoirement à partir de ces distributions marginales.

Depuis lors, de nombreux autres EDA ont été proposés, notamment l'EDA basé sur la maximisation de l'entropie (EMEDA) de Hwang et Choi en 1996, et l'EDA basé sur l'estimation de densité par noyau (KDEEDA) de Pelikan et Goldberg en 2000 [20].

Aujourd'hui, les EDA sont de plus en plus utilisés dans divers domaines, tels que l'ingénierie, la finance, la planification, la biologie, la physique, etc. Pour résoudre des problèmes d'optimisation complexes, dans le cas des MOILP, les EDA peuvent être utilisés pour explorer l'espace de recherche complexe des solutions potentielles qui satisfont à la fois les contraintes du problème et les objectifs multiples. L'utilisation des EDA dans ce contexte permet de trouver un ensemble de solutions diversifiées, appelé ensemble Pareto, qui représente les solutions efficaces dans le compromis entre les différents objectifs.

L'un des problèmes d'optimisation multiobjectif les plus connus actuellement est le problème d'allocation des ressources, où il s'agit de trouver une répartition optimale de certaines ressources limitées, pour satisfaire plusieurs critères simultanément conflictuels. Ce dernier est utilisé pour modéliser plusieurs situations pratiques, prenant par exemple, le cas où il s'agir de l'allocation de ressources financières, humaines, ou matérielles, ceci dans le but de maximiser les bénéfices, minimiser les coûts d'utilisation et optimiser la satisfaction des clients.

L'allocation des ressources est souvent un problème complexe, car il implique de prendre en compte des contraintes telles que les capacités des ressources, les priorités et les dépendances entre

les tâches, ainsi que les objectifs multiples. Les approches traditionnelles peuvent être limités par la complexité combinatoire de l'espace de recherche [11] .

Dans cette étude, nous adaptons l'utilisation des EDA pour résoudre un problème MOILP, ceci en améliorant l'ensemble des solutions efficaces, en explorant intelligemment l'espace de recherche des solutions entières, par la génération des solutions potentielles selon une distribution de probabilité spécifiée. A chaque génération, on modifié les paramètres de cette dernière selon l'importance des solutions par rapport à l'objectif primordial ϕ.

Selon la taille de l'instance du problème, ce processus est répété à plusieurs reprises. Pour illustrer notre approche de manière pratique, nous avons choisi de nous concentrer sur le problème de l'allocation des ressources multiobjectif. Notre démonstration met en évidence la manière dont les EDA (Algorithmes d'Estimation de Distribution) peuvent jouer un rôle crucial dans la recherche de solutions optimales. Ils parviennent à cela en modélisant la relation complexe entre les variables de décision, les contraintes du problème et les multiples objectifs à atteindre.

Ces concepts seront implémentés sur une machine via Python, et une série d'expériences numériques sera réalisée afin d'évaluer les performances de l'approche développée.

Des discussions des résultats feront l'objet de l'analyse de l'efficacité de cette approche.

Ce mémoire sera structuré de la façon suivante :

Après cette introduction générale mettant l'accent sur la thématique traitée, ainsi que sur nos motivations et la démarche suivie, le premier chapitre sera dédié à la présentation des préliminaires et des concepts de base de l'optimisation, tels que les principales définitions et propriétés liées à l'optimisation mono-objectif et multi-objectifs. Ensuite, quelques méthodes existantes dans la littérature, dédiées à la programmation multi-objectif, notamment à la programmation linéaire multi-objectif, seront explorées.

Le deuxième chapitre sera entièrement consacré aux outils utilisés dans notre travail. Nous aborderons les métaheuristiques ainsi que les algorithmes évolutionnaires basés sur les EDA.

Enfin, le troisième chapitre présentera la modélisation du problème d'allocation de ressources dans sa version multi-objectif, afin d'illustrer l'approche proposée.

Le quatrième chapitre contient la partie pratique où nous allons implémenté l'algorithme de l'approche developpée, illustée par des expérimentations numériques, disscussions et analyses.

Le travail s'achève par une conclusion générale, mettant l'accent sur les perspectives de recherche induites par les résultats obtenus.

Chapitre 1

Concepts et préliminaires sur l'optimisation des problèmes en nombres entiers

L'optimisation multi-objectif est une approche de l'optimisation qui vise à trouver le meilleur compromis entre plusieurs objectifs simultanément conflictuels. Contrairement à l'optimisation mono-objectif, qui cherche à maximiser ou minimiser une seule fonction objectif, l'optimisation multi-objectif cherche à trouver un ensemble de solutions qui offrent un équilibre entre plusieurs critères. Ces solutions sont appelées "solutions efficaces" ou "solutions Pareto-optimales", et sont souvent représentées dans un graphique appelé "front de Pareto".

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 [9].

Dans ce chapitre, nous allons explorer la notion d'optimisation mono-objectif et multiobjectif, aussi nous allons présenter quelques méthodes de résolution du problème MOILP sur l'ensemble des solutions efficaces dans deux cas : continue et discret.

1.1 Généralités sur l'optimisation

Définition 1.1.1. ***Optimisation :***
L'optimisation est une discipline mathématique qui vise à trouver les valeurs extrêmes d'une fonction tout en respectant éventuellement des contraintes sur les variables d'entrée. Elle aborde les problèmes liés à la prise de décision et joue un rôle essentiel en recherche opérationnelle. Son objectif est d'identifier les solutions optimales pour améliorer l'efficacité, la rentabilité ou d'autres critères pertinents dans divers domaines [13].

Définition 1.1.2. ***Problème d'optimisation*** *[13] :*
Un problème d'optimisation est défini par :

1. *Un espace de décision : ensemble de solutions ou de configurations possibles qui sont déterminées par les différentes valeurs attribuées aux variables de décision ;*
2. *Une ou plusieurs fonction(s) dites objectifs(s) à optimiser (minimiser ou maximiser) ;*
3. *Un ensemble de contraintes à respecter.*

1.1.1 Principaux concepts en optimisation

Des notions importantes sont nécessaires pour la définition d'un problème d'optimisation, ainsi que sa résolution, ci-après nous citons l'essentiel de ces derniers [37].

Fonction Objectif : Une fonction objectif est une fonction mathématique qui modélise le but à atteindre dans le problème d'optimisation. Il s'agit de la fonction qui doit être optimisée. Elle est notée $F(x)$; Dans le cas d'optimisation multiobjectif $F(x)$ est un vecteur $F(x) = [f_1(x), f_2(x), \ldots, f_k(x)]$. Elle est aussi appelée : critères d'optimisation, et peut s'agir de fonction coût, fonction d'adaptation, ou encore performance.

Paramètres : Un paramètre du problème d'optimisation, est une variable qui exprime une donnée quantitative ou qualitative sur une dimension du problème : coût, temps, taux d'erreurs, ... etc. Ces paramètres correspondent aux variables de la fonction objectif. Ils sont ajustés pendant le processus de modélisation.

Vecteur de décision : Un vecteur de décision est un vecteur correspondant à l'ensemble des variables du problème, il est noté : $x = [x_1, x_2, \ldots, x_n]^T$ avec n : le nombre de variables ou dimension du problème.

Critère de décision : Un critère est utilisé pour évaluer les vecteurs de décision et déterminer le meilleur parmi eux. Ce critère peut être une variable individuelle du problème ou une combinaison de plusieurs variables.

Contraintes : Une contrainte du problème est une condition que doivent respecter les vecteurs de décision du problème. Une contrainte est notée :
$g_i(x)$ avec : $i = 1, \ldots, m$. où m : le nombre des contraintes.

Espace de recherche : représente l'ensemble des valeurs pouvant être prises par les variables.

Espace réalisable : représentant l'ensemble des valeurs des variables satisfaisant les contraintes.

Espace des objectifs : l'images de l'espace réalisable, déterminé par toutes les valeurs possibles des fonctions objectifs.

1.2 Optimisation mono-objectif

Un problème d'optimisation mono-objectif consiste à optimiser (maximiser ou minimiser) une fonction objectif soumise à un certain nombre de contraintes.

1.2.1 Formulation mathématique d'un problème d'optimisation mono-objectif

Dans sa forme générale, un problème d'optimisation mono-objectif consiste en une fonction objectif à valeurs réelles à optimiser, et un certain nombre de contraintes d'égalité ou d'inégalité à respecter[9] :

$$\begin{cases} \min & f(x) \\ \text{s.c.} & g_i(x) \leq 0, i = \overline{1,q}. \\ & h_j(x) = 0, j = \overline{1,p}. \\ & x^l \leq x \leq x^u \end{cases} \tag{1.1}$$

où $x \in \mathbb{R}^n$ est un vecteur de variables de décision. $g_1, \ldots, g_q$ et $h_1, \ldots, h_p$ sont les fonctions des contraintes d'inégalité et d'égalité respectivement. Et x^l, x^u sont respectivement la borne inférieure et supérieure du domaine de recherche. Cet ensemble de contraintes définit l'espace faisable (réalisable) S des variables de décision avec :

$$S = \{x \in \mathbb{R}^n / h(x) = 0,\, g(x) \leq 0 \;\text{ et}\, x^l \leq x \leq x^u\}$$

1.2.2 Définitions et notions de base

Définition 1.2.1. ***Ensemble convexe [9] :***
Soit $\lambda x + (1-\lambda)y \in X$, $\forall x, y \in X; \forall \lambda \in [0,1]$ donc l'ensemble $X \subseteq \mathbb{R}^n$ X est convexe.

Propriété 1.2.1. *Si $X_1, \ldots, X_n$ sont des ensembles convexes, alors l'intersection $\cap_{i=1}^n X_i$ est convexe.*

Définition 1.2.2. ***Fonction convexe [8] :***
Soit l'ensemble convexe $X \subseteq \mathbb{R}^n$.
Une fonction $f : X \longmapsto \mathbb{R}^n$ est convexe si : $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y); \forall x, y \in X \, \forall \lambda \in [0,1]$.

Une fonction f est concave si $-f$ est convexe. Une fonction f est strictement convexe si l'inégalité ci-dessus est stricte pour tout $x, y \in \mathcal{X}$ tels que $x \neq y$ et tout $\lambda \in]0,1[$.

Définition 1.2.3. ***Fonction quasi-convexe [8] :***
On dit qu'une fonction $f(x)$ est quasi-convexe sur un intervalle I ($I \subseteq \mathbb{R}^n$) si et seulement si :

$$\forall (x_1, x_2) \in I^2, \forall \lambda \in [0,1], f(\lambda x_1 + (1-\lambda)x_2) \leq \max\{f(x_1), f(x_2)\}$$

On dit qu'une fonction f est quasi-concave sur l'intervalle I si et seulement si $-f$ est quasi-convexe.

1.2.2.1 Minima globaux et locaux

Soit l'ensemble $X \subseteq \mathbb{R}^n$ et une fonction $f : X \longmapsto \mathbb{R}$ non linéaire de classe C^1.
Considérons le problème : $\min_{x \in X} f(x)$.
Les minimaux locaux et globaux de f sur X sont définis de la manière suivante :

Définition 1.2.4. ***Minimum Local [9] :***
Le vecteur $x^* \in X$ est un minimum local de la fonction f sur X s'il présente un coût inférieur à celui de tous les vecteurs voisins.
De manière plus formelle, x^* sera considéré comme un minimum local de f sur X si :

$$\exists \epsilon > 0 \text{ tel que } \quad f(x^*) \leq f(x), \forall x \in X \text{ avec } \|x - x^*\| < \epsilon$$

où $\|v\|$: désigne la norme du vecteur v.
Le minimum local est strict si
$f(x^*) < f(x), \forall x \in X \quad et \quad x \neq x^* \quad avec \quad \|x - x^*\| < \epsilon.$

Définition 1.2.5. ***Minimum global [9] :***
Si le coût de x^* est inférieur à celui de tous les autres vecteurs de X, alors x^* est considéré comme un minimum global de f sur X. En revanche, si x^* est seulement inférieur aux coûts des autres vecteurs dans une petite région de X, alors il est considéré comme un minimum local de f sur X.
De manière formelle, x^* est un minimum global de f sur X si :

$$f(x^*) \leq f(x), \forall x \in X$$

Le minimum global est strict si :

$$f(x^*) < f(x), \forall x \in X \backslash \{x^*\}$$

Remarque 1.2.1. *Si la fonction objectif est convexe, alors il n'y a pas de différence entre le minimum local et global. Cela signifie que tout minimum local est également un minimum global.*

théorème 1.2.1. *Soit $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ une fonction convexe définie sur un ensemble convexe X. Alors, tout minimum local de f sur X est également un minimum global. Si f est strictement convexe, alors il existe au plus un minimum global de f.*

1.2.3 Espaces des solutions

Solution réalisable [18] : On appelle solution réalisable tout vecteur $x \in \mathbb{R}^n$ qui satisfait les contraintes.

Solution optimal[18] : On appelle solution optimale, toute solution réalisable qui optimise l'objectif.

Solution de base [9] : une solution de base est une solution satisfaisante d'un problème composée de $n - m$ variables, où n est le nombre total de variables et m est le nombre de contraintes indépendantes. Cette solution est caractérisée par $(n - m)$ variables nulles et m variables non nulles, formant ainsi une base.

Solution de base réalisable [9] : On appelle solution de base réalisable toute solution de base satisfaisant les contraintes de non-négativité.

Polyèdre [9] : L'intersection de demi-espaces fermés est dit polyèdre fermé.

Ensemble borné [18] : Un ensemble $X \subset \mathbb{R}^n$ est dit borné si $\forall x \in X \ \|x\| \leq a$ (a est fini).

1.3 Optimisation multi-objectif

L'optimisation multi-objectifs est nécessaire pour résoudre des problèmes réels complexes avec des espaces de recherches vastes et des fonctions objectifs contradictoires. Contrairement à l'optimisation mono-objectif, elle cherche à optimiser plusieurs composants d'un vecteur de fonction coût pour obtenir un ensemble de solutions optimales appelées ensemble des solutions Pareto optimales [9] .

1.3.1 Classification de la résolution des problèmes d'optimisation multi-objectif

Il existe plusieurs classifications des méthodes de résolutions dédiés à l'optimisation multi-objectif. Deux classes différentes de ces approches se distinguent. Le premier classement adopte **un point de vue décideur**, et le second adopte **un point de vue concepteur** [9].

1.3.1.1 Classification "Point de vue décideur" [25]

A cet égard on distingue trois schémas possibles. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème mono-objectif. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par le solveur multiobjectif :

1-Les approches a priori : Le preneur de décision intervient en aval du processus d'optimisation, pour définir la fonction d'agrégation modélisant le compromis qu'on souhaite faire entre les différents objectifs. Dans ce cas, il est présumé que le décideur connaît a priori le poids de chaque objectif afin de les mélanger en une seule fonction. Cela revient à résoudre un problème mono-objectif. Toutefois, dans la plupart des cas, le décideur ne peut exprimer clairement sa fonction d'utilité, parce que les différents objectifs ne sont pas comparables.

2-Les approches interactives : En associant les processus de prise de décision et d'optimisation de façon cyclique et progressive, le décideur intervient pour modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. De cette façon, le décideur modifie de manière interactive le compromis entre ses préférences et les résultats. Cette démarche permet de tenir compte des préférences du décideur, mais exige sa présence tout au long de la recherche .

3-Les approches a posteriori : Cette démarche cherche à fournir au décideur un ensemble de bonnes solutions bien réparties, ensuite, au regard de cette ensemble, le décideur sélectionne

celle qui lui semble la plus appropriée. Cependant, fournir un ensemble de solution bien réparties est difficile et nécessite un temps de calcul vraiment important.

1.3.1.2 Classification "Point de vue concepteur"[25]

Cette classification adopte un point de vue plus théorique basé sur les notions de regroupement et de Pareto optimalité. Les approches utilisées pour résoudre des problèmes multi-objectifs peuvent être classées en deux catégories : les approches non Pareto et les approches Pareto.

1-Les approches non Pareto : Les approches non Pareto ne considèrent pas le problème en tant que véritable problème multiobjectif. Leurs but est de chercher à ramener le problème initial à un ou plusieurs problèmes mono-objectifs.

2-Les approches Pareto : Les approches Pareto ne modifient pas les objectifs du problème, ils ont traités sans aucune distinction lors de la résolution.

1.3.2 Méthodologie de résolution Multi-objectif

La résolution de problèmes à objectifs multiples relève de deux disciplines bien différentes, étant donné que la résolution d'un problème multiobjectif peut être divisé en deux phases :

Phase 1 : La recherche des solutions de meilleur compromis : c'est la phase d'optimisation multiobjectif ;

Phase 2 : Le choix de la solution à retenir : c'est la tâche du décideur qui, parmi l'ensemble des solutions de compromis, doit extraire celle(s) qu'il utilisera .

1.3.3 Structure d'un problème d'optimisation Multi-objectif

Un problème multiobjectif consiste à optimiser (maximiser ou minimiser) k fonctions objectif simultanément ($k \geq 2$). Il est généralement défini par :

Problème 1

$$(MOP) \begin{cases} \text{"}optimiser\text{"} & (f_1(x), f_2(x), \ldots, f_k(x)) \\ \text{s.c} & g_j(x) \leq 0 \end{cases} \tag{1.2}$$

où :

- k : Le nombre de fonction objectif, $k \geq 2$.
- $f_i(x)(i = 1, \ldots, k)$ et $g_j(x)(j = 1, \ldots, m)$ sont des fonctions à valeurs réelles du vecteur de décision $x \in \mathbb{R}^n$.
  Le problème (1.2) est dit problème de programmation mathématique multiobjectif.

$\surd$ Si les objectifs $f_i(x)$ et les fonctions $g_j(x)$ sont linéaires, on obtient un problème de programmation linéaire multiobjectif (**MOLP : " Multiple Objective Linear Programming"**), écrit comme :

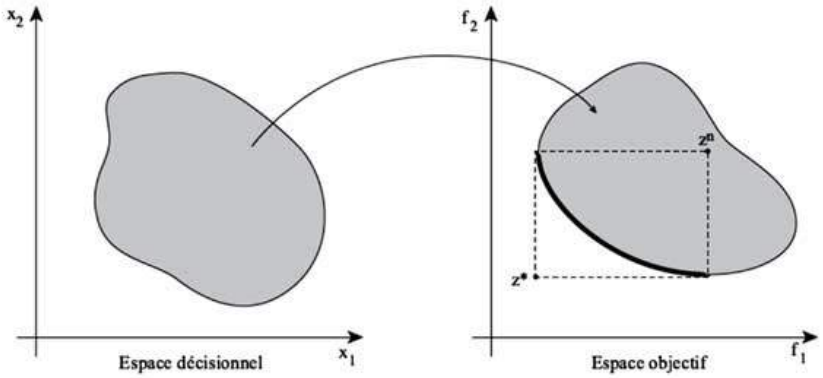


FIGURE 1.1 – Espace décisionnel et espace objectif d'un problème d'optimisation Multi-objectif (exemple avec deux variables de décision et deux fonctions objectif)

Problème 2

$$(MOLP)\begin{cases} \text{"optimiser"} & Z_i(x) = c^i(x), i = 1, \ldots, k \\ \text{s.c} & Ax \leq b \\ & x \geq 0 \end{cases} \tag{1.3}$$

où :

- $c^i \in \mathbb{R}^{1\times n}$
- $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}$, $b \in \mathbb{R}^m$, $m, n \in \mathbb{N}$.

$\surd$ Si de plus, l'ensemble des décisions est restreint aux entiers, le problème devient un problème de programmation linéaire multiobjectif en nombres entiers (**(MOILP) "Multiple Objective Integer Linear Program"**) qui peut être formulé mathématiquement comme suit :

Problème 3

$$(MOILP)\begin{cases} \text{"optimiser"} & Z_i(x) = c^i(x),\ i = \overline{1,k}. \\ \text{s.c} & Ax \leq b \\ & x \in \mathcal{D} \end{cases} \tag{1.4}$$

Où :

- $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$.
- C : Une matrice de dimension $(k \times n)$, et ses vecteurs lignes $c^i \in \mathbb{R}^n, i = \overline{1,k}$.
- $\mathcal{D}$: L'ensemble des solutions réalisables entières.

La résolution du ***problème 1*** consiste à trouver l'ensemble des solutions optimale au sens de Pareto, c'est à dire les solutions qui sont meilleures sur un critère et au moins bonne sur tous les autres critères.

Remarque 1.3.1. *Avant de présenter quelques méthodes de résolution d'un problème (MOILP), nous allons d'abord expliquer quelques concepts concernant les solutions efficaces.*

Remarque 1.3.2. *La difficulté des problèmes multiobjectifs réside dans l'absence d'une relation d'ordre total entre les solutions. Une solution peut être meilleure sur un critère et moins bonne sur un autre.*

Donc il est absolument difficile d'obtenir une solution unique qui procure simultanément la solution optimale pour l'ensemble des objectifs. Dans ce cas la solution optimale n'est plus une solution unique mais, un ensemble de solutions de compromis entre les différents objectifs à optimiser.

Pour identifier les meilleurs compromis il faut d'abord définir la relation d'ordre ente ces éléments. La relation la plus célèbre est la relation de ***dominance*** *au sens de* ***Pareto****. Dans ce qui suit nous allons présenter quelques définitions expliquant cette relation.*

1.3.4 Vocabulaire et définitions

Définition 1.3.1. *[3, 1]Soient u et v, deux vecteurs de même dimension :*

$$\left\| \begin{array}{lll} u = v, & ssi & \forall i \in \{1, 2, \ldots, n\}, u_i = v_i \, ; \\ u \leq v, & ssi & \forall i \in \{1, 2, \ldots, n\}, u_i \leq v_i \, ; \\ u < v, & ssi & u \leq v \bigwedge u \neq v. \end{array} \right.$$

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 $<, \leq, \ldots$ suffisent pour comparer les points, elles sont insuffisantes pour comparer des points issus des problèmes multiobjectif. 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 1.3.2. ***La dominance au sens de Pareto [3, 1] :***
Considérons un problème de minimisation. Soient u et v deux vecteurs de décision :

$$\left\| \begin{array}{lll} u \prec v (u \textit{ domine } v), & ssi & f(u) < f(v) \, ; \\ u \preceq v (u \textit{ domine faiblement } v), & ssi & f(u) \leq f(v)), \, f(u_i) < f(v_i) \, ; \\ u \sim v (u \textit{ est incomparable (ou non-dominé) avec } v), & ssi & f(u) \nleq f(v) \textit{ et } f(v) \nleq f(u). \end{array} \right.$$

Pour un problème de maximisation, ces relations sont définies de manière symétrique.

Définition 1.3.3. *Une solution $x \in \chi_E$ est dite non dominée par rapport à un ensemble $\chi_a \subseteq \chi_E$ 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 (figure 1.2). 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.

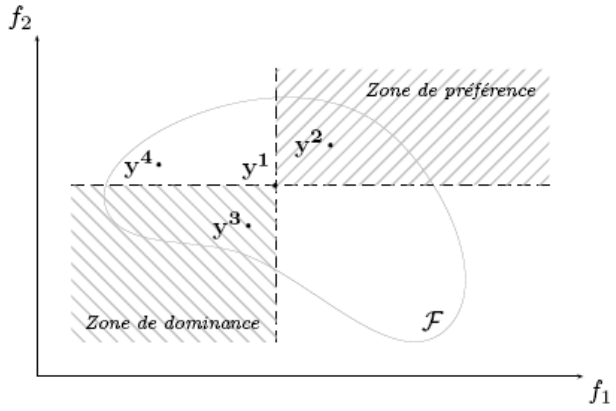


FIGURE 1.2 – Exemple de dominance.

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 multiobjectif. 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 1.3.4. *[3, 1] Un vecteur de décision $x \in X$ est dit Pareto globalement optimal si et seulement si : $\nexists y \in X$, $y \prec x$.*

Définition 1.3.5. *[3, 1] Un vecteur de décision $x \in X$ est dit Pareto optimal localement si et seulement si, pour un $\delta > 0$ fixé : $\nexists y \in X; f(y) \in B(f(x); \delta)$ et $y \prec x$, où $B(f(x); \delta)$ représente une boule de centre $f(x)$ et de rayon δ.*

La figure (1.3) 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.

Définition 1.3.6. ***La dominance au sens de Geoffrion [3] :***
Une dernière forme de dominance importante dans le monde de l'optimisation multiobjectif est la dominance au sens de Geoffrion. Les solutions optimales obtenues par ce type de dominance sont appelées les solutions Pareto optimales propres.

Définition 1.3.7. *Une solution $x^* \in X$ est appelée solution Pareto optimale propre si :*

- *elle est Pareto optimale,*
- *il existe un nombre $M > 0$ tel que $\forall i = 1, \dots k$ et $\forall x \in X$ vérifiant $f_i(x) < f_i(x^*)$, il existe un indice j tel que $f_j(x^*) < f_j(x)$ et :*

$$\frac{f_i(x^*) - f_i(x)}{f_j(x) < f_j(x^*)} \leq M;$$

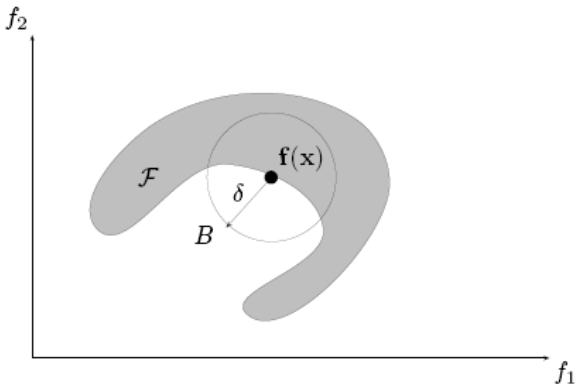


FIGURE 1.3 – Exemple d'optimalité locale .

Définition 1.3.8. ***Solution efficace [9] :***
Soit $x^* \in X$

- x^* *est dite efficace s'il n'existe aucune autre solution admissible* $x \in X$ *telle que* $f(x) \geq f(x^*)$. *On dit alors que* $y^* = f(x^*)$ *est un point non dominé.*
- x^* *est dite faiblement efficace s'il n'existe aucune autre solution admissible* $x \in X$ *telle que* $f(x) > f(x^*)$. *On dit alors que* $y^* = f(x^*)$ *est un point faiblement non dominé.*

Définition 1.3.9. ***Ensembles efficace [35] :***
L'ensemble des solutions efficaces est noté X_E *et l'ensemble des points non dominés est* Y_N .
Y_N *peut alternativement êtes défini par* $Y_N = \{y \in Y : (y - \mathbb{R}^k \cap Y = \{y\}\}$.

Définition 1.3.10. ***Efficacité et non dominance [16] :***

Les fonctions objectif sont supposées conflictuelles. De ce fait, il n'existe pas de solution admissible optimisant tous les objectifs simultanément. Contrairement au cas des problèmes mono-objectif là ou la notion d'optimalité est bien définie dans l'optimisation multi-objectif cette notion ne s'applique donc plus. Il est alors nécessaire de définir un contexte de résolution afin de donner un sens à "min" ou "max".

Définition 1.3.11. ***Equivalence [16] :***
Deux solutions admissibles x *et* $x' \in X$ *sont dites équivalentes si* $f(x) = f(x^*)$.

Définition 1.3.12. ***Ensemble de Solution complète [35] :***
L'ensemble des solutions complètes est également connu sous le nom de frontière de Pareto ou ensemble Pareto. Il représente l'ensemble des solutions qui ne peuvent pas être améliorées dans un critère d'optimisation sans détériorer au moins un autre critère.

Mathématiquement, soit X *l'ensemble des solutions et* $F(X)$ *le vecteur de fonctions objectifs correspondant à chaque solution* $x \in X$. *L'ensemble des solutions complètes est défini comme suit :* $Pareto(x) = \{x \in X \mid \forall x' \in X, F(x') \leq F(x) \Rightarrow F(x') = F(x)\}$.

Cela signifie que chaque solution de Pareto est dominante par rapport à toutes les autres solutions non dominantes, et aucune solution de Pareto ne peut être améliorée dans tous les critères objectifs en même temps .

Définition 1.3.13. ***Ensemble de solution non complète [37] :***
l'ensemble des solutions non complètes comprend toutes les autres solutions qui ne font pas partie de l'ensemble des solutions complètes. Ces solutions ne sont pas dominantes par rapport à toutes les autres et peuvent être améliorées dans au moins un critère objectif sans détériorer les autres critères.

Définition 1.3.14. ***Problème MOLP réduit [9] :***
le problème réduit est une approche qui transforme un problème MOLP avec plusieurs objectifs en un problème de programmation linéaire mono-objectif en utilisant la méthode de la pondération. Cela permet de simplifier la résolution en agrégeant les objectifs en une seule fonction objectif, avec des poids attribués pour refléter l'importance relative des objectifs.

Définition 1.3.15. ***Arête efficace [25] :*** *une arête efficace est une solution qui se trouve sur le front Pareto d'un problème multi-objectif, représentant un compromis optimal entre les différents objectifs. Ces arêtes efficaces sont importantes pour la prise de décision et sont recherchées lors de la résolution de problèmes multi-objectifs.*

Définition 1.3.16. ***Le tri non dominé [16] :*** *le tri non dominé est une méthode permettant de classer les solutions d'un problème d'optimisation multi-objectif en fonction de leur domination les unes par rapport aux autres. Il permet d'identifier les solutions non dominées qui constituent le front Pareto, représentant les solutions optimales où aucun objectif ne peut être amélioré sans détérioration sur un autre objectif.*

1.3.5 Caractérisation du front Pareto [1]

1.3.5.1 Solutions supportées / non supportées

Le front Pareto comporte plusieurs solutions de meilleur compromis appelées solutions pareto optimales. Nous allons caractériser les différents types de solutions composant ce front. Deux types de solutions le composent.

a- Les solutions supportées : leurs images dans Y se trouvent sur l'enveloppe convexe de Y. Chacune de ces solutions peut être trouvée en optimisant une agrégation linéaire des objectifs (théorème de Geoffrion).

b- Les solutions non-supportées : elles sont pareto optimales mais leurs images dans Y ne sont pas situées sur l'enveloppe convexe de Y. Aucune de ces solutions ne peut être trouvée en optimisant une agrégation linéaire des objectifs (corollaire du théorème de Geoffrion).

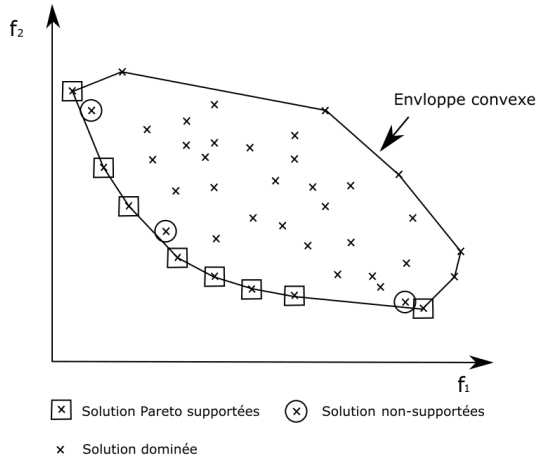


FIGURE 1.4 – Solutions supportés et non supportés

Remarque 1.3.3. *L'ensemble des solutions efficaces du problème (MOILP) est égal à l'union des solutions supportées SES et des solutions non supportées SENS [35].*

1.3.5.2 Ensemble Pareto minimal complet / Ensemble Pareto maximal complet [22]

Un ensemble Pareto est dit complet, si toutes les valeurs Pareto qui peuvent être obtenues sont représentées parmi les solutions de l'ensemble. Certaines méthodes ne permettent pas de trouver l'ensemble Pareto complet (ex : la méthode par agrégation).

Il est important de remarquer que le nombre de solutions Pareto dépend de l'espace considéré (espace décisionnel/ espace objectif). En effet, deux solutions différentes de l'espace décisionnel peuvent avoir le même vecteur coût et donc représentent le même point de l'espace des objectifs.

L'ensemble Pareto de l'espace décisionnel est appelé *ensemble Pareto maximal* et celui des objectifs est appelé *ensemble Pareto minimal*.

Il est donc nécessaire de définir l'ensemble recherché car cela fera varier le nombre de solutions Pareto. Ces notions sont illustrées dans la figure (1.4).

l'ensemble complet minimal ne contient qu'une seule solution pour chaque point non dominé dans l'espace des objectifs.

l'ensemble complet maximal contient toutes les solutions équivalentes pour chaque point non dominé dans l'espace des objectifs.

1.3.6 Complexité des algorithmes [1]

La complexité (temporelle) d'un algorithme est le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par cet algorithme. Ce nombre s'exprime en fonction de la taille n des données.

On s'intéresse au coût exact quand c'est possible, mais également au coût moyen (que se passe-t-il si en moyenne sur toutes les exécutions du programme sur des données de taille n ?), au cas le plus favorable, ou bien au pire cas.

On dit que la complexité de l'algorithme est $O(f(n))$ où f est d'habitude une combinaison de polynômes, logarithmes ou exponentielles. Ceci reprend la notation mathématique classique, et signifie que le nombre d'opérations effectuées est borné par $cf(n)$, où c est une constante, lorsque n tend vers l'infini.

Les algorithmes usuels peuvent être classés en un certain nombre de grandes classes de complexité :

- ✓ Les algorithmes sous-linéaires, dont la complexité est en général en $o(\log n)$. C'est le cas de la recherche d'un élément dans un ensemble ordonné fini de cardinal n ;
- ✓ Les algorithmes linéaires en complexité $o(n)$ ou en $o(n \log n)$ sont considérés comme rapides, comme l'évaluation de la valeur d'une expression composée de n symboles ou les algorithmes optimaux de tri ;
- ✓ Plus lents sont les algorithmes de complexité située entre $o(n^2)$ et $o(n^3)$, c'est le cas de la multiplication des matrices et du parcours dans les graphes ;
- ✓ Au delà, les algorithmes polynômiaux en $o(n^k)$ pour $k > 3$ sont considérés comme lents, sans parler des algorithmes exponentiels (dont la complexité, est supérieure à tout polynôme en n), que l'on s'accorde à dire inutilisables dès que la taille des données est supérieure à quelques dizaines d'unités.

1.4 Compromis entre exploration et exploitation [1]

La capacité d'exploitation est l'aptitude d'une méthode à utiliser des résultats obtenus pour faire converger l'algorithme.

La capacité d'exploration est l'aptitude d'une méthode à explorer l'espace d'état. Cette aptitude a tendance à ralentir la convergence. Lors de la conception d'une méthode d'optimisation, le chercheur doit choisir entre maintien de la diversité et convergence alors que le décideur doit choisir entre *efficacité et efficience* d'une méthode.

Une méthode efficace fournira un résultat optimal ou proche de l'optimum. Dans ce cas, le temps de calcul n'a aucune importance. Pour assurer un résultat optimale, ces méthodes doivent d'effectuer une recherche exploratoire très importante de façon à ne laisser aucune partie de l'espace des états non visitée.

Une méthode efficiente est une méthode capable de donner un résultat satisfaisant en un temps de calcul relativement court. Dans ce genre de méthode, le décideur privilégie le temps de calcul à la précision du résultat.

La conception de ces méthodes est basée sur une capacité d'exploration importante des résultats précédents. Ainsi, le processus de convergence est accéléré au risque de voir la méthode trompée, et dirigée vers une zone de l'espace non optimale. Cette réutilisation systématique des caractères des génération passées entraîne une perte rapide de la diversité. L'adaptation des individus n'a pas le temps de devenir optimale.

1.5 Problème d'optimisation linéaire multi-objectif en nombre entier (MOILP)

Un problème linéaire multi-objectif en nombre entier est constitué :

— d'un système de contraintes linéaires définissant un domaine discret de solutions réalisables,
— un ensemble de fonctions linéaires à optimiser (maximiser ou minimiser) définissant des objectifs conflictuels entre eux.

Le problème consiste à déterminer toute solution réalisable entière, telle qu'il n'y a aucune autre solution réalisable qui offre des valeurs au moins aussi bonne que celles sur chaque objectif et encore mieux sur au moins un objectif.

1.5.1 Structure générale d'un problème MOILP

$$(P)\begin{cases} \text{"}opt\text{"} & Z_i(x) = c^i(x),\ i = 1, \ldots, k \\ \text{s.c} & x \in \mathcal{D} \end{cases}$$

où :

$\mathcal{D} = S \cap \mathbb{Z}^n$, $\mathbb{Z}$ étant l'ensemble des nombres entiers relatifs.
$S = \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}$; $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $c^i \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^m$; $m, n \in \mathbb{N}$.

1.6 Quelques méthodes de résolution des problèmes MOILP

L'optimisation d'un problème MOILP sur l'ensemble des solutions efficaces entières est un problème courant dans l'optimisation combinatoire.

Dans ce contexte, une solution efficace est définie comme une solution qui satisfait toutes les contraintes du problème.

Dans des application pratiques de prise de décision, à critère multiples, les décideurs doivent souvent choisir un certain point préféré de l'ensemble des solutions efficaces X_E, ceci nous conduit

à trouver les solutions efficaces en décrivant la structure de X_E, comme dans beaucoup de cas, les critères sont en conflit, les décideurs essayent alors d'optimiser un certain critère de compromis sur l'ensemble X_E de solutions efficaces qui n'est pas convexe.

On va se concentrer sur l'optimisation d'une fonction linéaire, dénotée ϕ, sur l'ensemble des solutions efficaces d'un problème multi-objectifs en nombres entiers noté MOILP. Nous examinerons le cas général où ϕ est une fonction linéaire, Une approche directe consisterait à rechercher l'ensemble de toutes les solutions efficaces X_E du problème MOILP et par la suite, à optimiser ϕ sur l'ensemble X_E.

Considérons le problème MOILP défini par l'équation (1.5).

$$(P(D)) \begin{cases} "max" & Z_i(x) = c^i(x), i = 1, \ldots, k \\ \text{s.c} & x \in \mathcal{D} = S \cap \mathbb{Z}^n. \end{cases} \tag{1.5}$$

Le problème principale étudier est décrit par l'équation suivante :

$$(PLI_E) \begin{cases} "max" & \phi(x) = d^{'}x \\ \text{s.c} & x \in X_E(P) \end{cases} \tag{1.6}$$

Naïvement, on peut résoudre le problème (PLI_E) en formant la liste de toutes les solutions efficaces $X_E(P(D))$ du problème $(P(D))$ et en optimisant $\phi(x) = dx$ sur cette liste. On vise à éviter cependant l'énumération explicite de toutes les solutions efficaces .

Remarque 1.6.1. *Dans cette partie quelque méthodes de résolution du problème MOILP sont exposées, quelques approches nécessitent la présence du décideur, d'autre donnent l'ensemble des solutions efficaces sons l'intervention de ce dernier.*

1.6.1 Cas continue

H.P. Benson, S. Sayin [5] , se sont intéressés à l'optimisation d'une fonction sur l'ensemble X_E des solutions efficaces d'un problème de programmation linéaire multiobjectifs (MOLP). Étant donné un problème MOLP, l'optimisation sur l'ensemble des solutions efficaces X_E est la maximisation (respectivement la minimisation) d'une fonction donnée, $\phi(x)$, sur X_E .

En particulier, H.P.Benson [13] 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 appropriés que les programmes linéaires multi-objectifs plus habituels .

Pour notre part nous travaillons sur l'ensemble des solutions efficace du problème d'optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces du problème MOILP introduit précédemment.

Ce problème surgit à chaque fois qu'une fonction linéaire est disponible en tant que critère pour mesurer l'importance ou pour distinguer parmi les solutions efficaces disponibles. En optimisant cette fonction linéaire sur l'ensemble des solutions efficace, les calcules exiges pour produire l'ensemble de toutes les solutions efficaces sont évités. C'est potentiellement tout a fait salutaire puisque

la charge de calcul pour produire cet ensemble se développe rapidement avec la taille du problème.

Dans la suite de ce chapitre en décrit deux algorithmes d'optimisation sur l'ensemble des solutions efficaces cas continue, celui de YOSHITSUGU Yamamoto[9, 37] et celui de ECKER J.G et SONG H.G.[16]

1.6.1.1 Methode de Yamamoto

L'algorithme de Yamamoto [9, 37] est une méthode pour trouver une solution optimale à un problème d'optimisation linéaire en nombres entiers en optimisant une fonction quasi-convexe sur l'ensemble des solutions efficaces.

1.6.1.2 Algorithme de Yamamoto

Algorithm 1 Algorithme de Yammamoto

Entrée : Un problème d'optimisation linéaire en nombres entiers (MOILP),
Sortie : Une solution optimale de (MOILP),
Etape 1 : Calculer l'ensemble des points extrêmes (sommets) du polyèdre X qui représente l'ensemble des solutions efficaces de (MOILP),
Etape 2 : Sélectionner un sommet x de X qui appartient à la fois à X et à l'ensemble des solutions efficaces, et qui maximise la fonction ϕ sur l'ensemble des sommets liés à x par une arête efficace ($N_E(x)$),
Etape 3 : Si la valeur de ϕ en x est supérieure à la valeur de ϕ en tous les autres sommets de X qui appartiennent à la fois à X et à l'ensemble des solutions efficaces, **alors** x est une solution optimale de (MOILP),
Sinon, se déplacer vers un sommet adjacent efficace qui maximise la fonction ϕ, et répéter le processus jusqu'à ce qu'une solution optimale soit trouvée.

1.6.1.3 Methode de Ecker et Song

L'algorithme de Ecker et Song [16] est une méthode pour trouver une solution optimale à un problème d'optimisation linéaire en nombres entiers en optimisant une fonction linéaire sur l'ensemble des solutions efficaces en utilisant une technique de pivotage pour se déplacer d'un sommet efficace à un sommet adjacent efficace.

La méthode de Ecker et Song [16] est une méthode exacte pour résoudre des problèmes MOILP avec une fonction ϕ linéaire, mais elle peut être coûteuse en temps de calcul pour des problèmes de grande taille.

1.6.1.4 Algorithme de Ecker et Song

Algorithm 2 Algorithme de Ecker et Song

Entrée : Un problème d'optimisation linéaire en nombres entiers (MOILP),
Sortie : Une solution optimale de (MOILP),
Etape 1 :Déterminer un point efficace x' du polyèdre X associé au problème MOILP,
Etape 2 : Calculer les coefficients de la fonction ϕ pour chaque variable de décision j,
Etape 3 : Trouver la variable de décision j* avec le plus grand coefficient ϕ_j,
Etape 4 : Si $\phi_{j^*} \leq 0$, **alors** x' est la solution optimale.
Sinon, sélectionner une arête efficace adjacente à x' qui augmente la valeur de x_{j^*},
Etape 5 : Effectuer un pivotage pour passer de x' à un nouveau point efficace x'' en utilisant l'arête efficace sélectionnée,
Etape 6 : Répéter les étapes 2 à 5 jusqu'à ce que la solution optimale soit trouvée.

1.6.2 Cas discret

1.6.2.1 Méthode de Klein-Hannan

La méthode de Klein et Hannan (1982) [9, 38] est une méthode pour identifier l'ensemble de toutes les solutions efficaces d'un problème MOILP ou pour en caractériser une partie seulement.

La méthode de Klein et Hannan [9, 38] utilise une technique de coupe pour éliminer les solutions dominées et éviter les solutions non dominées déjà trouvées. Les contraintes ajoutées à chaque étape éliminent les solutions efficaces déjà trouvées et garantissent que la nouvelle solution optimale est différente de toutes les solutions efficaces précédentes. La méthode de Klein et Hannan est une méthode exacte pour résoudre des problèmes MOILP, mais elle peut être coûteuse en temps de calcul pour des problèmes de grande taille.

1.6.2.2 Algorithme de Klein-Hannan

Algorithm 3 Algorithme de Klein-Hannan

Entrée : Un problème d'optimisation linéaire en nombres entiers (MOILP),
Sortie : Une solution optimale de (MOILP),
Etape 1 : Résoudre le problème de programmation linéaire en nombres entiers associé au problème MOILP,
Etape 2 : Si la solution optimale est inefficace, alors le problème MOILP n'a pas de solution efficace,
Etape 3 : Sinon, ajouter la solution optimale x_1 à l'ensemble des solutions efficaces Y_1,
Etape 4 : Ajouter des contraintes pour éliminer la solution optimale x_1 et résoudre le problème MOILP sous ces nouvelles contraintes,
Etape 5 : Si le problème MOILP est irréalisable, alors toutes les solutions efficaces ont été trouvées. Sinon, ajouter la nouvelle solution optimale x_2 à l'ensemble des solutions efficaces Y_2,
Etape 6 : Répéter les étapes 4 et 5 en ajoutant des contraintes pour éliminer les solutions efficaces déjà trouvées jusqu'à ce que toutes les solutions efficaces soient identifiées.

1.6.2.3 Méthode de Jorge

La méthode de Jorge (2009) [38] est une méthode pour produire une solution optimale globale d'un problème MOILP sans énumérer l'ensemble de toutes les solutions efficaces.

1.6.2.4 Algorithme de Jorge

Algorithm 4 Algorithme de Jorge

Entrée : Un problème d'optimisation linéaire en nombres entiers (MOILP),
Sortie : Une solution optimale de (MOILP),
Etape 1 : Résoudre le problème de programmation linéaire en nombres réel associé au problème MOILP,
Etape 2 : Si la solution optimale est inefficace, **alors** le problème MOILP n'a pas de solution efficace,
Etape 3 :Sinon, résoudre le problème de programmation linéaire en nombres entiers associé au problème MOILP en utilisant la solution optimale du problème de programmation linéaire en nombres réels comme borne supérieure,
Etape 4 :Si la solution optimale est inefficace, **alors** la solution optimale du problème de programmation linéaire en nombres réels est la solution optimale globale du problème MOILP.
Sinon, tester la solution optimale du problème de programmation linéaire en nombres entiers pour l'efficacité,
Etape 5 : Si la solution optimale est inefficace, **alors** elle est éliminée en ajoutant une nouvelle contrainte au problème MOILP. **Sinon**, la solution optimale est ajoutée à l'ensemble des solutions efficaces et une nouvelle borne supérieure est calculée en utilisant la solution optimale du problème de programmation linéaire en nombres entiers,
Etape 6 : Répéter les étapes 3 à 5 jusqu'à ce que toutes les solutions efficaces soient identifiées.

Conclusion

L'optimisation multi-objectif est souvent confrontée à des problèmes de conflits entre les différents objectifs, qui nécessitent une analyse minutieuse des compromis possibles. Dans ce contexte, l'ensemble des solutions efficaces est un concept clé qui permet de déterminer l'ensemble des solutions compromis pour chaque objectif tout en prenant en compte les interactions entre eux.

Plusieurs approches ont été proposées pour résoudre ce type de problèmes, notamment les métaheuristiques telles que les algorithmes génétiques et les EDA. Les algorithmes génétiques ont été largement utilisés dans ce contexte, car ils sont capables de traiter plusieurs objectifs simultanément et de rechercher efficacement l'ensemble des solutions efficaces. Cependant, leur convergence peut être lente pour les problèmes MOILFP et MOLP [5].

Dans le prochain chapitre, nous allons nous intéresser plus particulièrement aux EDA et à leur utilisation pour l'optimisation des MOILP, en explorant en particulier les avantages et les limites de ces approches.

Chapitre 2

Algorithme d'Estimation de Distribution (EDA)

Introduction

L'algorithme d'estimation de distribution (EDA) est une méthode puissante utilisée dans le domaine de l'optimisation et de l'intelligence artificielle. Il fait partie des métaheuristiques, des techniques d'optimisation inspirées de processus naturels tels que l'évolution biologique ou le comportement des insectes sociaux [30].

Le principe des EDA consiste à estimer la distribution probabiliste des solutions prometteuses dans l'espace de recherche afin de guider la génération de nouvelles solutions. Cela diffère des autres approches d'optimisation qui se basent sur la manipulation directe des solutions individuelles[20]. Ils ont été proposés pour la première fois dans les années 1990 par des chercheurs tels que Larranaga, Lozano, and Martínez [32].

Les premiers travaux dans le domaine des EDA se sont concentrés sur la construction des modèles probabilistes simples, tels que les modèles de mélange gaussien, pour estimer la distribution des solutions. Ces modèles ont été utilisés pour générer de nouvelles solutions par échantillonnage stochastique, basé sur la distribution estimée [26].

Dans ce chapitre, nous allons présenter les métaheuristiques, en mettant l'accent sur l'algorithme génétique, l'une des approches les plus couramment utilisées. Ensuite, nous introduirons les EDA en décrivant leurs différentes étapes et en expliquant leur fonctionnement.

De plus, nous examinerons le comportement général des algorithmes d'estimation de distribution et présenterons différentes variantes, notamment les algorithmes paramétriques et non paramétriques.

Nous aborderons ensuite, les applications des algorithmes d'estimation de distribution, nous

soulignerons leurs utilisations dans différents domaines, en particulier pour la modélisation de données. Nous discuterons également des avantages et des limites de l'utilisation de ces algorithmes dans des contextes spécifiques.

Enfin, nous présenterons un exemple concret du problème "ONE MAX" pour illustrer l'application de l'EDA. Nous conclurons en analysant les avantages et les inconvénients de l'utilisation de l'EDA par rapport aux autres méthodes d'optimisation, mettant en évidence ses caractéristiques distinctives.

Ce chapitre offre une introduction complète à l'algorithme d'estimation de distribution, en présentant ses principes fondamentaux, ses étapes, ses modèles de distribution, ses applications et ses avantages. Il fournit une base solide pour approfondir la compréhension et l'application de cette méthode d'optimisation efficace.

2.1 Les métaheuristiques

Les métaheuristiques sont une classe d'algorithmes de recherche heuristique qui sont utilisés pour résoudre des problèmes d'optimisation complexes. Ces algorithmes ne garantissent pas de trouver la solution optimale, mais ils sont souvent très efficaces pour trouver des solutions de haute qualité dans des temps raisonnables [4].

Définition 2.1.1. *[2]**(Métaheuristique selon Osman et Laporte)** Une métaheuristique est un processus itératif de génération guidant une heuristique subordonnée en combinant intelligemment 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.*

Il existe de nombreuses métaheuristiques différentes, telles que la recherche locale, les algorithmes génétiques, les essaims de particules et les colonies de fourmis .... Chacune de ces techniques a ses propres avantages et inconvénients, et leur efficacité dépend du type de problème d'optimisation et des contraintes qui y sont associées [15].

2.1.1 Concepts généraux de métaheuristiques [4]

1 **Exploration de l'espace de recherche :** Les métaheuristiques explorent l'espace de recherche des solutions possibles pour trouver une solution de haute qualité. Cette exploration est réalisée en utilisant des techniques de recherche heuristique qui sont conçues pour trouver des solutions dans des espaces de recherche très grands ou complexes.

2 **Évaluation des solutions :** Les métaheuristiques évaluent chaque solution dans l'espace de recherche en utilisant une fonction d'évaluation qui mesure la qualité de la solution. Cette fonction d'évaluation peut être conçue pour maximiser ou minimiser une fonction objectif, ou pour respecter certaines contraintes.

3 **Recherche locale et globale :** Les métaheuristiques peuvent utiliser des techniques de recherche locale et globale pour trouver des solutions. La recherche locale se concentre sur l'exploration des régions locales de l'espace de recherche pour trouver des solutions voisines de la solution actuelle. La recherche globale explore tout l'espace de recherche pour trouver la meilleure solution possible.

4 **Population de solutions :** Certains types de métaheuristiques, tels que les algorithmes génétiques et les essaims de particules, utilisent une population de solutions pour explorer l'espace de recherche. Cette population est mise à jour itérativement en utilisant des opérateurs de mutation et de croisement pour créer de nouvelles solutions à évaluer.

5 **Adaptation :** Les métaheuristiques peuvent être adaptatives, c'est-à-dire qu'elles peuvent ajuster leurs paramètres de contrôle au fil du temps pour mieux s'adapter au problème d'optimisation en cours. Par exemple, l'algorithme d'optimisation par essaim de particules peut ajuster la vitesse et la direction des particules pour améliorer la recherche.

6 **Paramètres stochastiques :** Les métaheuristiques peuvent utiliser des paramètres stochastiques pour ajouter une certaine variabilité à la recherche. Ces paramètres peuvent inclure des probabilités de mutation ou de croisement aléatoires pour créer de nouvelles solutions dans la population.

7 **Critère d'arrêt :** Les métaheuristiques doivent avoir un critère d'arrêt pour arrêter la recherche lorsque la solution de haute qualité est atteinte ou lorsque la recherche prend trop de temps. Les critères d'arrêt peuvent être basés sur un nombre maximal d'itérations, un temps maximal de calcul ou une qualité minimale de la solution.

2.2 Algorithme Génétique

Un algorithme génétique est une méthode de résolution de problème d'optimisation, avec ou sans contraintes basée sur un processus de sélection naturelle [7].

Dans un tel algorithme, une population de solution est modifier à plusieurs reprise. A chaque fois, l'algorithme sélectionne au hasard des individus dans la population et les utilise comme parents pour produire les enfantes de la génération suivantes. Au fil des génération successives, la population "évolue" vers une solution optimal.

On peut utiliser un algorithme génétique afin de résoudre des problèmes pour lesquels les algorithmes d'optimisation standard ne sont pas vraiment adaptés. Il peut s'agir par exemple de problèmes avec une fonction objectif discontinue, stochastique ou particulier non linéaire [4].

2.2.1 Principes des algorithmes génétiques

L'algorithme génétique suit les principes suivants :

Génération aléatoire : Une population initiale d'individus est créée de manière aléatoire.

Sélection des parents : Pour passer de la génération k à la génération $k+1$, les parents sont sélectionnés en fonction de leur adaptation. Des couples de parents P_1 et P_2 sont choisis.

Opérateur de croisement : Un opérateur de croisement est appliqué aux couples de parents P_1 et P_2 avec une probabilité P_c (généralement autour de 0.6). Cela engendre des enfants C_1 et C_2.

Sélection d'autres individus : En plus des parents, d'autres individus de la population k sont sélectionnés en fonction de leur adaptation.

Opérateur de mutation : Un opérateur de mutation est appliqué aux individus sélectionnés avec une probabilité P_m (P_m est généralement bien inférieur à P_c). Cela engendre des individus mutés P'.

Évaluation des enfants et des individus mutés : Les enfants (C_1, C_2) ainsi que les individus mutés P' sont évalués pour déterminer leur adaptation.

Création de la nouvelle population : Les enfants et les individus mutés remplacent les parents dans la nouvelle population. Les meilleurs individus sont conservés pour la génération suivante.

Ces étapes sont répétées pour plusieurs générations jusqu'à atteindre une solution satisfaisante ou un critère d'arrêt spécifié.

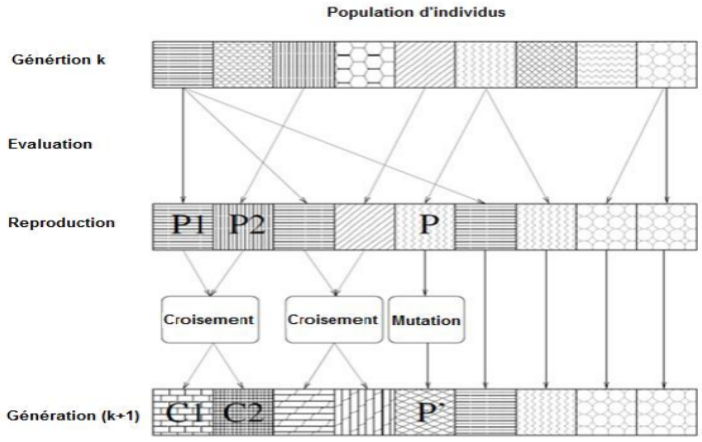


FIGURE 2.1 – Principe générale des algorithmes génétique

2.2.2 Fonctionnement des algorithmes génétiques

Les algorithmes génétiques fournissent des solutions aux problèmes n'ayant pas des solutions calculables en temps raisonnable de façon analytique ou algorithmique [15].

Selon cette méthode, des milliers de solutions (génotypes) plus ou moins bonnes sont créées au hasard. Ensuite, ces solutions sont soumises à un processus d'évaluation de leur pertinence, qui imite l'évolution des espèces "les plus adaptées". Les solutions optimales, c'est-à-dire celles qui sont les plus optimales pour résoudre le problème, survivent davantage que celles qui le sont moins. La

population évolue donc au fil des générations en croisant les meilleures solutions entre elles et en les faisant muter. Ce processus est répété un certain nombre de fois pour tendre vers la solution optimale [15].

Le mécanisme d'évolution et de sélection est indépendant du problème à résoudre. Cependant, il existe trois fonctions qui sont spécifiques à chaque problème :

- **Fonction de représentation** : Cette fonction est responsable de la représentation du problème en codant chaque information caractérisant une solution possible. Elle utilise un codage spécifique où chaque information est un gène et les différentes valeurs possibles pour cette caractéristique sont les allèles du gène. En concaténant tous les gènes, on obtient un chromosome qui représente une solution complète.

- **Fonction d'inversion** : Cette fonction permet de décoder le génome, c'est-à-dire de partir d'un chromosome pour obtenir une solution concrète qui correspond aux informations codées.

- **Fonction d'évaluation** : Cette fonction évalue l'adaptation d'une solution à un problème spécifique en lui attribuant une mesure de pertinence ou de qualité.

Cette technique inspirée de l'évolution biologique est d'application générale.

Un algorithme génétique prend généralement la forme suivante :

Algorithm 5 Algorithme génétique

[1] Initialiser la population initiale P.
[2] Évaluer P.
[3] **Tant Que** (Pas de convergence) **faire** :
  [a] $P' =$ Sélectionner les parents dans P.
  [b] $P' =$ Appliquer l'opérateur de croisement sur P'.
  [c] $P' =$ Appliquer l'opérateur de mutation sur P'.
  [d] Remplacer les individus de P par leurs descendants de P'.
  [e] Évaluer P.
Fin Tant que

2.3 Algorithme d'estimation de distribution (EDA)

Les Algorithmes à Estimation de distribution (Estimation of Distribution Algoritm, EDA), également appelés algorithmes génétiques à construction de modèle probabilistes (Probabilistic Model-Building Genetic Algorithm, PMBGA) et algorithmes évolutionnaires à estimation de densité itéré (Iterated Density Estimation Evolutionary Algorithm, IDEA), sont des méthodes d'optimisation qui utilisent des modèles probabilistes pour générer des solutions de haute qualité pour un problème donné [7].

L'idée principale est de modéliser la distribution de probabilité des variables d'un problème d'optimisation et d'utiliser cette distribution pour générer des solutions potentielles [7].

La plupart des EDA se basent sur des modèles de probabilité construits à partir d'un ensemble de solution générée au hasard. Ces solutions sont évaluées à l'aide d'une fonction objectif, et les caractéristiques les plus importantes de ces solutions sont utilisées pour mettre à jour le modèle de probabilité. Ce processus est répété jusqu'à est-ce que le modèle de probabilité soit suffisamment précis pour générer des solutions de haute qualité [30].

Parmi les algorithmes d'EDA les plus courants, on peut citer :

- **L'algorithme de distribution marginale (MDE)** : il est basé sur la modélisation des distributions marginales des variables de décision. Il est largement utilisé pour résoudre des problèmes de planification de tâches.
- **L'algorithme de modèle probabiliste de Markov (PMMA)** : il utilise un modèle de probabilité de Markov pour modéliser la distribution de probabilité des variables de décision. Il est couramment utilisé pour résoudre des problèmes de conception de circuit.
- **L'algorithme de distribution de fréquence (FEDA)** : il est basé sur la modélisation de la distribution de fréquence des variables de décision. Il est souvent utilisé pour résoudre des problèmes de placement d'objets.

2.3.1 Description [21]

Dans un problème d'optimisation donné, les algorithmes à estimation de distribution (EDA) font évoluer une population de solutions potentielles.

Les algorithmes à estimation de distribution (EDA) fonctionnent de manière similaire aux algorithmes évolutionnaires standard en commençant par générer une population de solutions potentielles de manière aléatoire.

À chaque itération, la population est mise à jour en utilisant des opérateurs de sélection et de variation. L'opérateur de sélection choisit une population de solutions prometteuses à partir de la population actuelle.

Toutefois, au lieu d'utiliser des opérateurs de variation inspirés de l'évolution et de la génétique, les EDA créent de nouvelles solutions en construisant un modèle probabiliste des solutions sélectionnées, puis en échantillonnant le modèle construit pour générer de nouvelles solutions.

Les nouvelles solutions sont ajoutées à la population initiale et le processus de sélection et de variation est répété jusqu'à ce qu'un critère d'arrêt soit satisfait.

Idéalement, la qualité des solutions générées par le modèle probabiliste s'améliore au fil du temps, et, après un nombre raisonnable d'itérations, l'échantillonnage du modèle devrait générer seulement la meilleure solution. Dans cette mise en œuvre, le modèle probabiliste construit pour la

population des solutions sélectionnées forme un arbre de dépendance où chaque variable est conditionnée à son prédécesseur.

La distribution globale est ensuite donnée comme le produit de la probabilité marginale de la variable à la racine de l'arbre et les probabilités conditionnelles des variables restantes compte tenu de leurs prédécesseurs.

2.3.2 Les pré-requis pour l'estimation de distribution [7]

L'utilisation des algorithmes d'estimation de distribution peut être complexe et nécessite une certaine connaissance préalable en statistique et en mathématiques.

Les pré-requis nécessaires pour utiliser ces algorithmes incluent :

1. **Connaissances de base en probabilités et en statistiques :** Il est important de comprendre les concepts de base en probabilités et en statistiques tels que les distributions de probabilité, les moments, les fonctions de densité de probabilité, les lois de probabilité, etc.

2. **Compréhension des différents types d'algorithmes d'estimation de distribution :** Il est important de comprendre les différences entre les algorithmes paramétriques et non paramétriques, ainsi que les avantages et les limites de chaque type d'algorithme.

3. **Maîtrise d'un langage de programmation :** La plupart des algorithmes d'estimation de distribution sont implémentés en utilisant un langage de programmation tel que Python, R, Matlab, etc. Il est donc important de maîtriser au moins l'un de ces langages.

4. **Capacité à manipuler des données :** Les algorithmes d'estimation de distribution nécessitent souvent la manipulation de grandes quantités de données. Il est donc important de savoir comment manipuler et préparer les données avant de les utiliser avec ces algorithmes.

5. **Capacité à interpréter les résultats :** Les résultats produits par les algorithmes d'estimation de distribution peuvent être difficiles à interpréter. Il est donc important de savoir comment interpréter les résultats pour les appliquer efficacement à des problèmes réels.

2.3.3 Les différentes étapes de l'algorithmes d'estimation de distribution [23]

Pour commencer, un ensemble de solutions (individus) est généré de manière aléatoire, en utilisant par exemple une distribution uniforme. Ensuite, ces solutions sont évaluées en utilisant une fonction objectif, qui permet d'évaluer l'impact de chaque solution sur le problème à optimiser.

À partir de cette évaluation, un nombre de solutions est sélectionné, comprenant les solutions qui ont de bonnes valeurs de la fonction objectif. Ensuite, un modèle probabiliste des solutions sélectionnées est créé, et une population de solutions est générée à partir de ce modèle. Ce processus

est répété de manière itérative jusqu'à ce que l'optimum soit atteint.

Les étapes de l'algorithme à estimation de distribution sont les suivantes :

1 On génère une population $P(t)$ de T solutions aléatoirement suivant une distribution uniforme.
2 Évaluation des T solutions générées à l'aide d'une fonction objectif.
3 Sélectionner S solutions (individus) à partir des T solutions, ces solutions sélectionnées forment la population $S(t)$.
4 Estimer les paramètres du modèle à partir des M individus (construction du modèle probabiliste à n dimensions $M(t)$.
5 Générer N nouvelles solutions $P(t+1)$ à partir du modèle probabiliste $M(t)$ (Échantillonnage).
6 Reprendre les étapes 2,3,4,5 jusqu'à ce que le critère d'arrêt soit satisfait.

Chaque solution est représentée par un vecteur de valeurs qui est considéré comme une variable statistique. Le modèle graphique probabiliste est construit en utilisant les individus sélectionnés, et les relations entre les différentes variables qui composent chaque individu sont exprimées de manière explicite en utilisant les distributions de probabilité conjointe associées aux individus sélectionnés à chaque itération.

2.3.4 Procédure principale d'un algorithme d'estimation de distribution [29]

Les EDA, ciblent principalement à maintenir un modèle probabiliste explicite qui représente une distribution de probabilité sur les solutions.

Dans chaque itération, le modèle est ajusté en fonction des résultats de l'évaluation des solutions trouvées. L'ajustement a pour but d'obtenir des solutions meilleures dans les itérations suivantes.

Les chercheurs font souvent la distinction entre deux principaux catégories d'EDA :

2.3.4.1 Les algorithmes d'estimation de distribution basés population

Ils utilisent un modèle probabiliste pour représenter la relation entre les variables du problème. Dans un EDA basé population, une population initiale est générée et évaluée pour estimer la distribution des variables. Cette distribution est ensuite utilisée pour générer de nouvelles solutions potentielles, qui sont évaluées et utilisées pour mettre à jour la distribution.

Ce processus itératif se poursuit jusqu'à ce qu'une solution satisfaisante soit trouvée. Les EDA basés population sont appréciés pour leur efficacité à explorer l'espace de recherche et à trouver des solutions de qualité.

Un pseudo-code d'un EDA basé sur la population est représenté dans l'algorithme suivant :

Algorithm 6 Algorithme d'un EDA basé sur la population

[1] $T \leftarrow 0$
[2] Générer une population initiale $P(t)$
[3] **Tant que** le critère d'arrêt n'est pas satisfait **Faire**
[4] Évaluer toutes les solutions dans $P(t)$;
[5] Choisir parmi $P(t)$ un groupe de parents $S(t)$ en favorisant les meilleures solutions ;
[6] Construire un modèle probabiliste $M(t)$ à partir de $S(t)$;
[7] Générer de nouvelles solutions $O(t)$ en échantillonnant $M(t)$;
[8] Construire $P(t+1)$ en combinant $O(t)$ et $P(t)$.
[9] **Fin Tant que**

Principales composantes d'un EDA basé-population :

1 **Collecte de données** : Il est important de collecter des données pertinentes et représentatives de la population cible.

2 **Nettoyage des données** : Les données collectées peuvent contenir des erreurs, des valeurs manquantes ou des données aberrantes. Il est donc important de nettoyer les données avant de procéder à une analyse.

3 **Résumé statistique** : Il s'agit d'une analyse descriptive qui permet de résumer les caractéristiques des données collectées, telles que la moyenne, la variance, la médiane, etc.

4 **Visualisation de données** : La visualisation des données permet de représenter graphiquement les données collectées pour mieux comprendre les tendances et les relations entre les variables.

5 **Analyse de corrélation** : Il s'agit de l'étude des relations entre les différentes variables et leur influence sur les résultats.

6 **Analyse de régression** : L'analyse de régression permet d'étudier la relation entre deux ou plusieurs variables et d'établir un modèle mathématique qui peut être utilisé pour prédire les résultats.

7 **Tests statistiques** : Les tests statistiques sont utilisés pour valider ou rejeter les hypothèses formulées sur la base des données collectées.

2.3.4.2 Les algorithmes d'estimation de distribution Incrémenteaux (EDI) :

Ils commencent avec une seule solution initiale et utilisent un modèle probabiliste pour estimer la distribution des variables. Ensuite, une nouvelle solution est générée en utilisant cette distribution, et le modèle est mis à jour avec cette nouvelle solution.

Ce processus est répété de manière incrémentale, en utilisant les nouvelles solutions générées pour améliorer le modèle et obtenir des solutions de plus en plus performantes.

Principales composantes d'un EDA incrémental :

1 **Modèle de distribution :** Il s'agit d'un modèle probabiliste qu'est utilisé pour estimer la distribution des solutions prometteuses. Ce modèle représente les relations entre les variables et est mis à jour de manière itérative à mesure que de nouvelles solutions sont générées.
2 **Échantillonnage :** L'échantillonnage est utilisé pour générer de nouvelles solutions candidates en fonction du modèle de distribution. Cela implique de tirer aléatoirement des échantillons du modèle pour créer de nouvelles solutions potentielles
3 **Évaluation :** L'évaluation est effectuée pour estimer la qualité des solutions candidates générées. Cela implique d'appliquer une fonction d'évaluation aux solutions pour déterminer leur aptitude ou leur valeur.
4 **Mise à jour du modèle :** Après l'évaluation des solutions, le modèle de distribution est mis à jour en utilisant des informations sur les meilleures solutions obtenues. Cela permet d'affiner progressivement le modèle pour qu'il représente de mieux en mieux les caractéristiques souhaitées des solutions.
5 **Critères d'arrêt :** Les critères d'arrêt définissent les conditions qui indiquent la fin de l'algorithme. Cela peut être basé sur le nombre d'itérations, le nombre de solutions évaluées ou l'atteinte d'un critère de convergence.

2.3.5 Comportement [34]

Le graphique représente les distributions des valeurs des optimums trouvés (sur un grand nombre d'exécutions), l'algorithme passe d'une population de solution très dispersée (A) à une population plus centré sur l'optimum trouvé (B) (2.2).

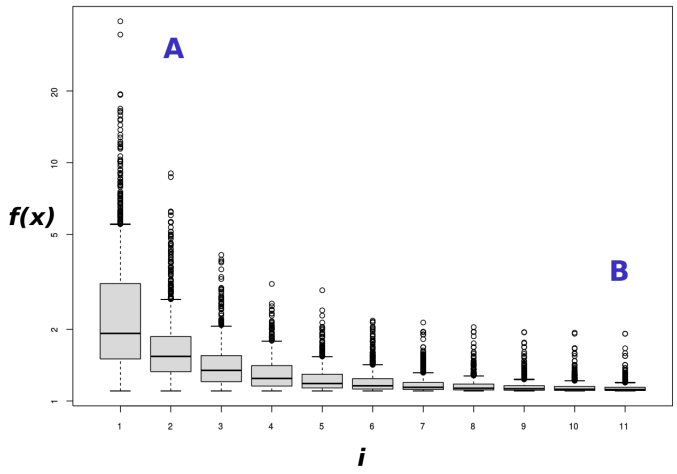


FIGURE 2.2 – Comportement des algorithmes d'estimation de distribution

La distribution A représente la population initiale de solutions générées par l'algorithme. Comme indiqué, cette distribution est caractérisée par une dispersion importante, ce qui signifie que les solutions trouvées initialement sont très variées et éloignées de l'optimum recherché.

Au fur et à mesure que l'algorithme progresse, il effectue des itérations et utilise des mécanismes de recherche et d'optimisation pour générer de nouvelles solutions. La distribution B représente l'évolution de la population de solutions au fil des itérations. On peut observer que cette distribution se resserre progressivement autour de la région où se trouve l'optimum recherché.

Cela indique que l'algorithme parvient à améliorer la qualité des solutions au fur et à mesure de son exécution. Les solutions générées deviennent de plus en plus proches de l'optimum, ce qui suggère une convergence vers une solution optimale ou quasi-optimale. L'évolution de la distribution des optima trouvés témoigne donc de l'amélioration progressive des performances de l'algorithme au cours de ses itérations.

2.3.6 Différents types d'algorithmes d'estimation de distribution

Les algorithmes paramétriques et non paramétriques sont deux types d'algorithmes utilisés en statistiques et en apprentissage automatique pour modéliser la relation entre les variables d'un ensemble de données.

2.3.6.1 Les algorithmes paramétriques [30]

Les algorithmes paramétriques sont des modèles statistiques qui supposent que la distribution des données suit une forme paramétrique spécifique, telle que la distribution normale ou la distribution de Poisson.

Ces modèles ont un nombre fixe de paramètres qui doivent être estimés à partir des données. Une fois que les paramètres sont estimés, le modèle peut être utilisé pour prédire de nouvelles données.

Les algorithmes paramétriques sont souvent plus efficaces et plus précis que les algorithmes non paramétriques lorsqu'ils sont correctement ajustés aux données, mais leur performance peut se dégrader rapidement si l'hypothèse de la forme paramétrique est incorrecte.

Voici quelques exemples d'algorithmes paramétriques couramment utilisés :

- **Régression linéaire :** La régression linéaire est un algorithme paramétrique qui vise à modéliser la relation linéaire entre une variable dépendante et une ou plusieurs variables indépendantes.
  Le modèle de régression linéaire suppose que la relation entre les variables est linéaire et suit une distribution normale. et voici la forme générale d'un modèle de régression linéaire :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$$

  où :
  - Y représente la variable dépendante que l'on cherche à prédire .
  - $\beta_0, \beta_1, \beta_2, ..., \beta_p$ sont les coefficients de régression correspondant aux variables indépendantes $X_1, X_2, ..., X_p$ respectivement .
  - $X_1, X_2, ..., X_p$ sont les variables indépendantes utilisées pour prédire Y .
  - ϵ est le terme d'erreur aléatoire

Dans cette forme, chaque variable indépendante X est multipliée par son coefficient de régression correspondant β, et la somme de ces termes est ajoutée à β_0 (l'ordonnée à l'origine) pour obtenir la valeur prédite de Y.

- **Distribution normale :** La distribution normale, également appelée distribution de Gauss, est une distribution de probabilité continue qui est largement utilisée en statistiques.

  Elle est caractérisée par deux paramètres : la moyenne et l'écart-type, qui déterminent la forme de la distribution.

  La notation mathématique de la distribution normale est la suivante :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  où $f(x)$ est la densité de probabilité à la valeur x, μ est la moyenne et σ est l'écart-type.

- **Modèles ARIMA :** Les modèles ARIMA (Autoregressive Integrated Moving Average) sont des algorithmes paramétriques utilisés pour modéliser les séries temporelles.

  Ces modèles supposent que les données suivent une distribution normale et qu'il existe une relation autoregressive (AR) et une moyenne mobile (MA) entre les observations .

  La forme générale du modèle ARIMA est 'ARIMA(p,d,q)' où :

  * p : représente l'ordre de l'autorégression, c'est-à-dire le nombre de termes AR inclus dans le modèle.
  * d : représente l'ordre de la différenciation intégrée, qui correspond au nombre de fois que les données doivent être différenciées pour les rendre stationnaires.
  * q : représente l'ordre de la moyenne mobile, c'est-à-dire le nombre de termes MA inclus dans le modèle.

  En termes mathématiques, un modèle ARIMA(p, d, q) peut être représenté par l'équation suivante :

$$((1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p)(1-L)^d X_t) = (1 + \theta_1 L + \theta_2 L^2 + ... + \theta_q L^q)\epsilon_t$$

  où :
  — X_t est la série temporelle.
  — L est l'opérateur de retard, qui décale les observations d'un pas en arrière ($L^k X_t = X_{t-k}$).
  — $\phi_1, \phi_2, .., \phi_p$ sont les coefficients d'autorégression.
  — $\theta_1, \theta_2, ..., \theta_q$ sont les coefficients de la moyenne mobile.
  — ϵ_t est le terme d'erreur, qui est généralement supposé être un bruit blanc.

2.3.6.2 Les algorithmes non paramétriques [30]

Les algorithmes non paramétriques ne contraignent pas la forme de la distribution des données. Ils utilisent généralement des techniques de classification ou de régression qui peuvent s'appliquer à des données de toute forme.

Les algorithmes non paramétriques offrent souvent plus de souplesse que les algorithmes paramétriques, mais ils peuvent être plus sensibles au bruit et à la variabilité des données.

Voici quelques exemples des Algorithmes non paramétriques :

1. **Arbres de décision :** Les arbres de décision sont des algorithmes non paramétriques utilisés pour la classification et la régression.
   L'algorithme construit un arbre en divisant les données en différents sous-ensembles en fonction de leur distribution et de leur relation avec la variable cible.

2. **Forêts aléatoires :** Les forêts aléatoires sont des ensembles d'arbres de décision construits à partir de différents sous-échantillons des données et de différentes variables.
   L'algorithme agrège ensuite les prédictions de chaque arbre pour obtenir une prédiction finale.

3. **Méthodes de noyau :** Les méthodes de noyau sont des algorithmes non paramétriques utilisés pour la régression, la classification et la réduction de dimensionnalité.
   L'idée de base de la méthode de noyau est de projeter les données dans un espace de plus grande dimension où les données sont plus facilement séparables, en utilisant une fonction de noyau pour mesurer la similarité entre les points de données.

2.3.6.3 Les principales différences entre les algorithmes paramétrique et non paramétrique

1. **Hypothèses :** Les algorithmes paramétriques supposent que la distribution des données suit une forme paramétrique spécifique, tandis que les algorithmes non paramétriques n'imposent pas d'hypothèses sur la forme de la distribution.

2. **Complexité :** Les algorithmes paramétriques ont souvent une complexité algorithmique plus faible que les algorithmes non paramétriques, car ils nécessitent l'estimation d'un nombre fini de paramètres. Les algorithmes non paramétriques ont généralement une complexité algorithmique plus élevée car ils peuvent nécessiter l'utilisation de méthodes de recherche dans l'espace des caractéristiques.

3. **Interprétabilité :** Les algorithmes paramétriques ont souvent des résultats plus facilement interprétables, car les paramètres du modèle ont une signification statistique claire. Les algorithmes non paramétriques peuvent être plus difficiles à interpréter, car ils peuvent être basés sur des méthodes de classification ou de régression plus complexes.

2.3.7 Méthodes d'estimation de distribution [33]

Il existe plusieurs méthodes d'estimation de distribution, chacune avec ses propres avantages et limitations.

Voici une brève présentation des principales méthodes :

- **L'estimation par maximum de vraisemblance (EMV) :** Cette méthode est utilisée pour estimer les paramètres d'une distribution de probabilité paramétrique en maximisant la vraisemblance des données observées.

  L'EMV est largement utilisée en statistiques et en apprentissage automatique.

  Soit $X = (X_1, X_2, ..., X_n)$ un échantillon de n observations indépendantes et identiquement distribuées selon une certaine loi de probabilité, caractérisée par une fonction de densité de probabilité $f(x; \theta)$, où θ est un vecteur de paramètres à estimer.

  La fonction de vraisemblance $L(\theta; X)$ est définie comme le produit des densités de probabilité pour chaque observation :

  $$f(X_n, \theta) = f(X_1, \theta) \times f(X_2, \theta) \times ... \times f(X_n, \theta) = \prod_i f(X_i, \theta)$$

  L'estimation par maximum de vraisemblance consiste donc à trouver $\hat{\theta}$ qui maximise $ln\ L(\theta; X)$ tel que $\hat{\theta} = argmax_t L(\theta; X)$.

  En utilisant le logarithme naturel (ln), on peut également maximiser la log-vraisemblance plutôt que la vraisemblance elle-même :

  $$lnL(\theta, X) = lnf(X_1; \theta) + lnf(X_2; \theta) + ... + lnf(X_n; \theta) = \sum_i lnf(X_i, \theta)$$

- **L'estimation par moments :** Cette méthode est également utilisée pour estimer les paramètres d'une distribution de probabilité paramétrique, mais elle repose sur l'égalisation.

  La méthode des moments peut donc être formulée comme suit :

  1. **Calcul des moments empiriques (Moment empirique d'ordre K)** :

     $$\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} X_i^k (k = 1, ..., K)$$

  2. **Calcul des moments théoriques (Moment théorique d'ordre k) :**

     $$\mu_k(\theta)(k = 1, ..., K)$$

3. **Égalisation des moments empiriques et théoriques [33] :**

$$\hat{\mu}_k = \mu_k(\theta)(k = 1, 2, ..., K)$$

4. **Résolution du système d'équations** : Résoudre le système d'équations pour estimer les valeurs de paramètre θ.

2.3.8 Exemples d'utilisation des algorithmes d'estimation de distribution

2.3.8.1 Modélisation de données [21]

Un exemple concret d'utilisation des algorithmes d'estimation de distribution pour la modélisation de données pourrait être dans la finance, où les investisseurs peuvent utiliser des modèles d'estimation de distribution pour modéliser les rendements des actifs financiers, tels que les actions ou les obligations.

Ces modèles peuvent être utilisés pour comprendre la distribution de probabilité des rendements, ce qui peut aider les investisseurs à prendre des décisions éclairées sur l'achat, la vente ou la détention d'un actif particulier.

Par exemple, un investisseur peut utiliser un modèle d'estimation de distribution pour estimer la probabilité qu'une action donnée augmente ou diminue de valeur dans un certain délai, en fonction des données historiques des rendements de cette action. Ces estimations de probabilité peuvent ensuite être utilisées pour évaluer le risque associé à l'investissement dans cette action et pour prendre des décisions de placement en conséquence.

Supposons qu'un investisseur s'intéresse à l'action ABC. Il dispose des données historiques des rendements de cette action sur une période de 1 an, avec des mesures quotidiennes. L'investisseur souhaite estimer la probabilité que l'action ABC augmente ou diminue de valeur dans les 30 prochains jours.

1- **Collecte des données :** L'investisseur collecte les données quotidiennes des rendements de l'action ABC au cours de l'année passée. Les rendements sont exprimés en pourcentage.
Exemple de données historiques des rendements de l'action ABC :

Jours	Rendement quotidien (%)
jour 1	0,5
jour 2	-0,8
jour 3	1,2
...	...
jour 365	-0,3

2- **Estimation de la distribution :** L'investisseur utilise un modèle d'estimation de distribution, tel que la distribution normale, pour modéliser le comportement des rendements de l'action ABC. En ajustant le modèle aux données historiques, il obtient les paramètres de la

distribution.

Supposons que le modèle ajusté donne les paramètres suivants pour la distribution normale :

- Moyenne(μ) =0,2 %
- Écart type =1,5 %

3- **Estimation de la probabilité :** À l'aide de la distribution normale estimée, l'investisseur peut maintenant estimer la probabilité que l'action ABC augmente ou diminue de valeur dans les 30 prochains jours.

- **Probabilité d'augmentation :** L'investisseur calcule la probabilité que les rendements de l'action ABC soient supérieurs à zéro dans les 30 prochains jours en utilisant la distribution normale. Supposons qu'il obtienne une probabilité de 70 % d'augmentation.
- **Probabilité de diminution :** L'investisseur calcule la probabilité que les rendements de l'action ABC soient inférieurs à zéro dans les 30 prochains jours en utilisant la distribution normale. Supposons qu'il obtienne une probabilité de 30 % de diminution.

4- **Évaluation du risque et prise de décision :** À partir de ces estimations de probabilité, l'investisseur peut évaluer le risque associé à l'investissement dans l'action ABC.

Dans cet exemple, l'investisseur pourrait conclure que l'investissement présente un risque modéré, avec une probabilité plus élevée d'augmentation de la valeur par rapport à la diminution. Il pourrait utiliser ces informations pour prendre des décisions de placement éclairées, telles que l'allocation de son portefeuille ou la gestion de ses positions sur l'action ABC, en fonction de son appétit pour le risque.

Cet exemple illustre comment un modèle d'estimation de distribution peut être utilisé dans la modélisation de données pour évaluer le risque et prendre des décisions de placement en conséquence.

2.3.8.2 Problème 'One max' [34]

Dans le problème du « one max », on cherche à maximiser le nombre de 1 sur un nombre de dimensions donné. Pour un problème à 3 dimensions, une solution $x_1 = \{1, 1, 0\}$ aura donc une meilleure qualité qu'une solution $x_2 = \{0, 1, 0\}$, l'optimum étant $i^* = \{1, 1, 1\}$. On cherche donc à maximiser une fonction $f(x) = \sum_{i=1}^{3} x_i$, où x_i peut prendre la valeur 0 ou 1.

La première étape consiste à tirer au hasard les individus, avec pour chaque variable, une chance sur deux de tirer un 1 ou un 0, autrement dit, on utilise une distribution de probabilité $p_0(x) = \prod_{i=3}^{3} p_0(x_i)$, où $p_0(x_i)$ est la probabilité que chaque élément soit égal à 1. La distribution est donc factorisée comme un produit de 3 distributions marginales univariantes de Bernoulli, de paramètre $0, 5$.

Exemple de tirage de la population D_0, avec une population de 6 individus, la dernière ligne indique la probabilité $p(x)$ pour chaque variable :

i	x_1	x_2	x_3	$f(x)$
1	0	1	0	1
2	0	1	0	1
3	1	0	1	2
4	1	0	1	2
5	0	1	1	2
6	1	0	0	1
$p(x)$	0.5	0.5	0.5	

L'étape suivante consiste en la sélection des meilleurs individus, pour former D_1^S. Dans notre exemple, il s'agit simplement de ne garder que les 3 meilleurs individus :

3	1	0	1	2
4	1	0	1	2
5	0	1	1	2
$p(x)$	0.7	0.3	1	

On constate que les trois paramètres $(p_i(x))$ caractérisant la distribution de probabilité (D_1^S) ont changé après la sélection.
En utilisant cette nouvelle distribution, on peut tirer une nouvelle population :

i	x_1	x_2	x_3	$f(x)$
1	1	1	1	3
2	0	1	1	2
3	1	0	1	2
4	1	0	1	2
5	1	0	1	2
6	0	0	1	1
$p(x)$	0.7	0.3	1	

Et ainsi de suite jusqu'à vérifier un critère d'arrêt (par exemple quand tous les individus sont à l'optimum, comme l'individu 1 du tableau ci-dessus).

Algorithme du problème one max :

Algorithm 7 Algorithme du problème 'One max'

Etape 1 :
Générer une population initiale de vecteurs binaires aléatoires de longueur n.
Etape 2 :
Tant qu'un critère d'arrêt n'est pas atteint :
- Calculer les fréquences de chaque bit dans la population actuelle.
- Utiliser les fréquences pour estimer la distribution de probabilité des bits.
- Générer une nouvelle population de vecteurs binaires en suivant la distribution de probabilité estimée.
- Évaluer la nouvelle population et sélectionner les meilleurs individus pour former la population suivante.

Etape 3 :
Retourner la meilleure solution trouvée.

2.3.8.3 Avantages et limites de l'utilisation des algorithmes d'estimation de distribution dans les différents domaines

Les algorithmes d'estimation de distribution offrent des avantages dans de nombreux domaines, notamment :

1. **Modélisation de données :** Les EDA peuvent être utilisés pour modéliser les données, comprendre leurs caractéristiques et estimer les paramètres de la distribution de probabilité sous-jacente.

2. **Prédiction et classification :** Les EDA peuvent être utilisés pour la prédiction et la classification en utilisant des modèles probabilistes qui reposent sur la distribution de probabilité estimée.

3. **Analyse des risques financiers :** Les EDA peuvent être utilisés pour modéliser les risques financiers, tels que les fluctuations des marchés financiers, les variations des taux de change, etc.

4. **Analyse des données géospatiales :** Les EDA peuvent être utilisés pour l'analyse des données géospatiales, telles que la prédiction des mouvements des masses d'air, des courants marins, etc.

Cependant, il y a aussi des limites à l'utilisation des algorithmes d'estimation de distribution :

1. **Dépendance aux données :** Les algorithmes d'estimation de distribution dépendent fortement des données d'entrée. Des données insuffisantes ou inexactes peuvent entraîner des estimations incorrectes de la distribution de probabilité.

2. **Préjugés :** Les algorithmes d'estimation de distribution peuvent être biaisés si la distribution de probabilité réelle des données est différente de celle supposée par l'algorithme.

3. **Interprétation :** L'interprétation des résultats d'un algorithme d'estimation de distribution peut être difficile, en particulier pour les non-experts.

4. **Complexité :** Certains algorithmes d'estimation de distribution, tels que les algorithmes non paramétriques, peuvent être plus complexes à mettre en œuvre et à comprendre que les algorithmes paramétriques.

2.3.9 Avantages et inconvénients de l'utilisation des algorithmes d'estimation de distribution par rapport aux autres méthodes d'optimisation

L'utilisation d'un Algorithme d'Estimation de Distribution (EDA) présente plusieurs avantages et inconvénients par rapport aux autres méthodes d'optimisation telles que les métaheuristiques ou les algorithmes génétiques. Voici quelques avantages et inconvénients de l'utilisation de l'EDA :

Avantages :

- L'EDA permet une exploration plus efficace de l'espace de recherche, ce qui peut entraîner une convergence plus rapide vers la solution optimale.
- Les EDAs peuvent être utilisés pour résoudre une variété de problèmes d'optimisation, qu'il s'agisse de problèmes continus ou discrets, mono-objectifs ou multi-objectifs.
- Les EDAs peuvent également être utilisés pour résoudre des problèmes à variables mixtes, c'est-à-dire des problèmes qui comportent à la fois des variables continues et discrètes.
- L'EDA peut être utilisé pour trouver une approximation de l'ensemble de Pareto, ce qui est important pour les problèmes multi-objectifs.
- Les EDAs sont capables de trouver des solutions de haute qualité sans nécessiter une connaissance approfondie du problème à résoudre.

Inconvénients :

- L'EDA peut être plus lent que d'autres méthodes d'optimisation, en particulier pour les problèmes de grande taille.
- Les EDAs peuvent nécessiter des ajustements de paramètres, tels que la taille de la population et la stratégie de sélection, pour obtenir de bons résultats.
- L'EDA peut être sensible à la qualité des données d'entrée, notamment lorsque les données sont bruyantes ou comportent des valeurs manquantes.
- Les EDAs peuvent également être sensibles aux hypothèses sur la distribution des données et peuvent être moins efficaces lorsque ces hypothèses ne sont pas satisfaites.
- L'EDA peut être plus difficile à implémenter que certaines autres méthodes d'optimisation, en particulier pour les problèmes complexes.

Conclusion

L'algorithme d'estimation de distribution (EDA) est une méthode puissante utilisée dans l'optimisation et beaucoup d'autres domaines. Basé sur la modélisation de la distribution des solutions prometteuses, l'EDA guide la recherche de la solution optimale.

Dans ce chapitre nous avons introduit les concepts essentiels des EDA basés sur une population et des EDA incrémentaux, ainsi que les modèles de distribution utilisés. Nous avons également abordé les métaheuristiques, en mettant l'accent sur les algorithmes génétiques, par suite nous avons ainsi exploré les méthodes d'estimation de distribution, leurs applications et leurs avantages et inconvénients.

Enfin, un exemple du problème "One max" a été présenté pour illustrer l'application de l'EDA. Ces concepts fournit une base solide pour approfondir la compréhension et l'application de l'EDA.

Dans le prochain chapitre, nous étudierons les problèmes d'allocation de ressources multiobjectifs. Nous examinerons les objectifs multiples souvent impliqués dans ces problèmes et discuterons des approches couramment utilisées pour les résoudre.

Chapitre 3

Problème d'allocation des ressources

Introduction

L'allocation de ressources est un domaine où l'optimisation multiobjectif est particulièrement pertinente. En effet, l'allocation de ressources consiste à répartir de manière optimale des ressources limitées entre différentes activités ou tâches [24].

Ce problème peut être particulièrement complexe car il peut y avoir des contraintes et des objectifs concurrents à prendre en compte. Par exemple, dans le domaine de la production industrielle, il peut être nécessaire de minimiser les coûts tout en maximisant l'efficacité et en réduisant les déchets. Dans le domaine de la logistique, il peut être nécessaire de minimiser les coûts tout en optimisant la qualité de service et en réduisant les temps d'attente [24].

Pour résoudre ces problèmes complexes, l'optimisation multiobjectif utilise des techniques d'optimisation mathématique et des méthodes de recherche opérationnelle. Ces techniques permettent de modéliser le problème, d'évaluer les solutions possibles et d'en sélectionner les meilleures parmi celles-ci [26].

Dans ce chapitre, nous allons explorer l'optimisation multiobjectif appliquée au problème d'allocation de ressources. Tout d'abord, nous allons présenter les modèles mono-objectif utilisés dans l'allocation de ressources.

Ensuite, nous allons aborder les modèles multi-objectifs en définissant les différents objectifs, et nous illustrerons trois variantes possibles.

3.1 Modélisation mono-objectif

La modélisation mono-objectif d'un problème d'allocation des ressources consiste à formuler le problème de manière à optimiser un seul objectif. Dans le contexte de l'allocation des ressources,

l'objectif peut être de maximiser les bénéfices, minimiser les coûts, optimiser l'utilisation des ressources, ou atteindre tout autre objectif spécifique.

3.1.1 Modèle mathématique

Considérons un ensemble fini d'indices $i = 1, 2, \ldots, n$, où n correspond au nombre de tâches à réaliser, et un ensemble d'indices $j = 1, 2, \ldots, m$, où m correspond au nombre de ressources disponibles.

Les $r_{i,j}$ sont des valeurs réelles représentant les coûts ou les bénéfices associés à l'utilisation de la ressource j pour la tâche i.

Soit $x_{i,j}$ la variable de décision qui représente la quantité de la ressource j allouée à la tâche i. La quantité de la ressource j allouée doit respecter les contraintes de disponibilité, c'est-à-dire que la somme des quantités allouées à la tâche i ne peut pas dépasser la quantité totale disponible.

Soit b_j la quantité totale disponible de la ressource j. La contrainte de disponibilité est définie comme suit :

$$\sum_{i=1}^{n} x_{i,j} \leq b_j, \quad j = 1, 2, \ldots, m \tag{3.1}$$

L'objectif du problème d'allocation de ressources est de maximiser le bénéfice total de l'allocation des ressources. Le bénéfice total est calculé en sommant les bénéfices associés à chaque tâche pour toutes les ressources.

Ainsi, le problème d'allocation de ressources peut être formulé mathématiquement comme suit :

$$\begin{cases} \max & f = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} r_{i,j}x_{i,j} \\ \text{s.c.} & \displaystyle\sum_{i=1}^{n} x_{i,j} \leq b_j, \quad j = 1, 2, \ldots, m \\ & x_{i,j} \geq 0, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m \end{cases}$$

où f représente la fonction objectif que l'on cherche à maximiser en ajustant les quantités allouées $x_{i,j}$ des ressources, tout en respectant les contraintes de disponibilité. Les coefficients de pondération des ressources sont donnés par les $r_{i,j}$ et b_j représente la quantité totale disponible pour la ressource j.

3.1.2 Variantes du problème

Il existe plusieurs variantes d'un modèle mathématique mono-objectif de problème d'allocation de ressources, en fonction des spécificités du problème et des objectifs visés. Voici quelques exemples :

1. **Problème d'allocation de ressources avec coûts variables [10] :** Dans le cas du problème d'allocation de ressources avec coûts variables, les coûts associés à l'allocation d'une

ressource peuvent varier en fonction du moment où elle est allouée. Ainsi, le modèle mathématique doit tenir compte de cette variable supplémentaire dans la formulation de la fonction de coût.

Soit n le nombre de tâches à allouer et m le nombre de ressources disponibles. On suppose que les coûts d'allocation sont représentés par une matrice $C_{n\times m}$, où chaque élément C_{ij} représente le coût d'allocation de la ressource j à la tâche i. On suppose également que la quantité de ressources allouées est représentée par une matrice $X_{n\times m}$, où chaque élément X_{ij} représente la quantité de ressource j allouée à la tâche i.

Ainsi, la fonction de coût F pour le problème d'allocation de ressources avec coûts variables est donnée par :

$$F = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}.X_{ij}(t)$$

où t est l'instant d'affectation de la ressource j à la tâche i. Cette fonction de coût prend en compte la variable supplémentaire t, qui représente le moment d'affectation de la ressource.

Les contraintes du modèle peuvent être similaires à celles du modèle mono-objectif classique, avec l'ajout d'une contrainte supplémentaire pour le moment d'affectation de chaque ressource :

$$\sum_{i=1}^{n} X_{ij}(t) \leq B_j(t) \qquad \forall j \in \{1, ..., m\}, \forall t$$

où $B_j(t)$ représente la capacité maximale de la ressource j au moment t. Cette contrainte garantit que la quantité de ressources allouées à chaque moment ne dépasse pas la capacité maximale de la ressource à ce moment-là.

Enfin, pour résoudre ce problème, des techniques d'optimisation peuvent être utilisées pour trouver la meilleure allocation de ressources qui minimise la fonction de coût F tout en respectant les contraintes du problème.

2. **Problème d'allocation de ressources avec contraintes de temps[31] :** Le problème d'allocation de ressources avec contraintes de temps est une variante du problème d'allocation de ressources qui prend en compte des contraintes temporelles supplémentaires. Dans ce cas, chaque tâche doit être accomplie avant une certaine échéance, et l'allocation de ressources doit être effectuée de manière à respecter ces échéances.

   Pour modéliser ce problème mathématiquement, nous pouvons utiliser un modèle de programmation linéaire mixte. Soit N l'ensemble des tâches à effectuer et M l'ensemble des ressources disponibles. Pour chaque tâche $i \in N$, nous avons les données suivantes :

   d_i : l'échéance de la tâche i.

Pour chaque ressource $j \in M$, nous avons les données suivantes :

c_j : le coût d'utilisation de la ressource j.

Nous cherchons à minimiser le coût total de l'allocation de ressources tout en respectant les échéances de chaque tâche. Nous pouvons formuler ce problème de la manière suivante :

$$\begin{cases} \text{Min} & f = \sum_{j \in M} c_j x_j \\ \text{s.c.} & \sum_{j \in M} a_{ij} x_j \geq 1 \qquad \forall i \in N \\ & \sum_{j \in M} a_{ij} x_j \leq M_i y_i \qquad \forall i \in N \\ & y_i = 1 \text{ si la tâche } i \text{ est accomplie avant son échéance, } y_i = 0 \text{ sinon.} \\ & x_j \in 0,1 \qquad \forall j \in M \\ & y_i \in 0,1 \qquad \forall i \in N \end{cases}$$

La première contrainte garantit que chaque tâche est effectuée en utilisant au moins une ressource. La deuxième contrainte garantit que chaque tâche est effectuée avant son échéance, en imposant une pénalité M_i si la tâche i n'est pas accomplie à temps. Les variables binaires x_j indiquent si la ressource j est utilisée ou non, et les variables binaires y_i indiquent si la tâche i est accomplie à temps ou non.

3. **Problème d'allocation de ressources multi-niveaux [24] :** Le problème d'allocation de ressources multi-niveaux consiste à allouer des ressources à des tâches qui nécessitent des ressources à différents niveaux hiérarchiques. Le modèle mathématique doit prendre en compte cette hiérarchie en définissant une structure d'allocation de ressources en plusieurs niveaux.

On peut utiliser la programmation linéaire pour résoudre ce problème. La fonction objectif est définie comme suit :

$$\min \quad f = \sum_{i \in I} \sum_{k \in K_i} \sum_{j \in J} c_{ijk} x_{ijk}$$

où c_{ijk} est le coût de l'allocation de la ressource j pour la tâche i au niveau k, et x_{ijk} est la variable de décision binaire qui prend la valeur 1 si la ressource j est allouée à la tâche i au niveau k, et 0 sinon.

Les contraintes du problème peuvent être définies comme suit :

La somme des ressources allouées à la tâche i doit être égale à la demande totale de la tâche i :

$$\sum_{k \in K_i} \sum_{j \in J} x_{ijk} = d_i \qquad \forall i \in I.$$

La capacité de chaque ressource j doit être respectée :

$$\sum_{i \in I} \sum_{k \in K_i} a_{jk} x_{ijk} \leq b_j \qquad \forall j \in J.$$

Les ressources ne peuvent être allouées qu'à des tâches qui ont une priorité inférieure ou égale à leur niveau hiérarchique :

$$x_{ijk} = 0, \qquad \forall i \in I,\ j \in J,\ k \in K_i \text{ tels que } k > h_i.$$

où h_i est le niveau hiérarchique maximal requis par la tâche i.

La méthode de résolution consiste à résoudre le problème pour chaque niveau hiérarchique, en utilisant les résultats de l'optimisation pour le niveau précédent comme contraintes pour le niveau suivant.

4. **Problème d'allocation de ressources avec incertitudes [27] :** Dans le problème d'allocation de ressources avec incertitudes, les demandes en ressources et les capacités des ressources peuvent varier de manière imprévisible. Pour prendre en compte cette incertitude, il est possible d'utiliser des techniques de programmation stochastique ou de théorie des jeux.

   La programmation stochastique consiste à modéliser les incertitudes sous forme de variables aléatoires, et à résoudre le problème en utilisant des outils de la théorie des probabilités. La formulation mathématique du problème d'allocation de ressources avec incertitudes en programmation stochastique peut se faire comme suit :

   Soit n le nombre de tâches et m le nombre de ressources disponibles. On suppose que les demandes de chaque tâche pour chaque ressource sont des variables aléatoires indépendantes, notées d_{ij}, où $i = 1, \ldots, n$ et $j = 1, \ldots, m$. De même, on suppose que les capacités de chaque ressource sont des variables aléatoires indépendantes, notées b_j, où $j = 1, \ldots, m$.

   Le but est d'allouer les ressources aux tâches de manière à minimiser le coût moyen total d'allocation de ressources, sachant que les demandes et les capacités sont aléatoires. On peut formuler le problème d'optimisation comme suit :

$$\begin{cases} min \quad f = \mathbb{E}\left[\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} b_j x_{ij}\right] \\ \text{s.c.} \quad \mathbb{E}\left[\displaystyle\sum_{j=1}^{m} d_{ij} x_{ij}\right] \geq p_i, \quad i = 1, \ldots, n \\ \qquad \displaystyle\sum_{i=1}^{n} x_{ij} \leq s_j, \quad j = 1, \ldots, m \\ \qquad x_{ij} \geq 0, \quad i = 1, \ldots, n,\ j = 1, \ldots, m \end{cases} \tag{3.2}$$

   où x_{ij} est la quantité de la ressource j allouée à la tâche i, p_i est la demande moyenne de la tâche i, s_j est la capacité moyenne de la ressource j, et $\mathbb{E}$ représente l'espérance mathématique.

La première contrainte assure que la demande moyenne de chaque tâche est satisfaite, la deuxième contrainte assure que la capacité moyenne de chaque ressource n'est pas dépassée, et la troisième contrainte impose que les quantités allouées soient positives.

3.1.3 Etat de l'art

L'allocation de ressources est un domaine de recherche très actif dans les sciences de gestion, l'informatique et l'ingénierie. Voici quelques exemples de travaux récents dans ce domaine :

- **Optimisation de l'allocation de ressources dans les réseaux de télécommunications** [28] **:** plusieurs études récentes ont examiné les méthodes d'allocation de ressources pour maximiser l'efficacité des réseaux de télécommunications, notamment dans les réseaux 5G et les réseaux de communication par satellite.

- **Allocation de ressources pour l'optimisation de la production d'énergie** [10] **:** les énergies renouvelables, comme l'énergie solaire et l'énergie éolienne, nécessitent une allocation efficace des ressources pour maximiser leur production. Plusieurs études ont examiné les méthodes d'allocation de ressources pour optimiser la production d'énergie renouvelable.

- **Allocation de ressources dans les systèmes de transport intelligents** [17] **:** l'efficacité des systèmes de transport intelligents dépend de l'allocation efficace des ressources, notamment en ce qui concerne la gestion du trafic et l'optimisation de l'utilisation des véhicules.

- **Optimisation de l'allocation de ressources pour les systèmes informatiques** [27] **:** les centres de données, les réseaux de capteurs et les systèmes de traitement de données nécessitent une allocation efficace des ressources pour optimiser leurs performances. Plusieurs études ont examiné les méthodes d'allocation de ressources pour optimiser les performances de ces systèmes.

- **Allocation de ressources dans les systèmes de soins de santé** [36] **:** l'allocation efficace des ressources est un enjeu majeur dans les systèmes de soins de santé, où les ressources sont souvent limitées. Plusieurs études ont examiné les méthodes d'allocation de ressources pour optimiser les soins de santé, notamment dans les domaines de la planification des traitements et de la gestion des lits d'hôpitaux.

3.2 Modélisation multi-objectif

Le problème d'allocation de ressources multi-objectif consiste à allouer des ressources limitées à des tâches concurrentes, tout en cherchant à optimiser plusieurs objectifs simultanément. Ce problème peut être formulé mathématiquement de la manière suivante :

3.2.1 Modèle Mathématique

Soit un ensemble fini de tâches $A = a_1, a_2, ..., a_n$ et un ensemble de ressources $R = r_1, r_2, ..., r_m$. Chaque tâche a_i requiert un certain montant de ressources pour être réalisée, représenté par un

vecteur $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ où x_{ij} est la quantité de la ressource r_j nécessaire pour la tâche a_i.

Le problème consiste à allouer les ressources de manière optimale entre les tâches, en maximisant (ou minimisant) plusieurs objectifs simultanément.

Nous considérons k le nombre d'objectifs à optimiser, représentés par des fonctions objectifs $f_1(x), f_2(x), ..., f_k(x)$, où x est le vecteur de toutes les allocations de ressources pour toutes les tâches.

Chaque fonction objectif peut être définie de manière à maximiser ou minimiser un certain critère, comme par exemple la maximisation du profit, la minimisation du coût ou la minimisation du temps de réalisation.

Ainsi, le problème d'allocation de ressources multi-objectif peut être formulé mathématiquement comme suit :

$$(MORAP) \begin{cases} \text{Opt} & F(x) = f_l(x) & l = 1, ..., k, \\ & \sum_{i=1}^{n} x_{ij} \leq b_j & j = 1, 2, ..., m \\ & x_{ij} \geq 0 & i = 1, 2, ..., n, \quad j = 1, 2, ..., m \end{cases}$$

où b_j est la quantité maximale de la ressource r_j disponible.

Les contraintes indiquent que la somme des quantités allouées de chaque ressource doit être inférieure ou égale à la quantité totale disponible pour cette ressource, et que les quantités allouées doivent être positives ou nulles.

Remarque 3.2.1. *Les contraintes d'un problème d'allocation de ressources multi-objectif varient d'un modèle à un autre en fonction du nombre de fonctions objectif et du type de problème.*

Le problème d'allocation de ressources multi-objectif est souvent un problème d'optimisation difficile car il peut y avoir des trade-offs entre les différents objectifs, et il peut ne pas y avoir de solution unique optimale qui maximise ou minimise tous les objectifs simultanément. C'est pourquoi des méthodes d'optimisation multi-objectif sont utilisées pour trouver des solutions qui sont "Pareto-optimales", c'est-à-dire des solutions qui ne peuvent pas être améliorées sur un objectif sans détériorer au moins un autre objectif.

3.2.2 Définition des objectifs

Dans le problème d'allocation de ressources multi-objectif, plusieurs objectifs peuvent être considérés en même temps. Ces objectifs peuvent être de nature différente, tels que la maximisation du profit, la minimisation du coût, la maximisation de la satisfaction du client, la minimisation du temps de réalisation, etc.

Dans cette section, nous définissons quelques-uns des objectifs les plus couramment utilisés dans la littérature.

— **Maximisation du profit :** L'objectif de maximisation du profit est l'un des plus courants dans les problèmes d'allocation de ressources multi-objectif. Il vise à maximiser les revenus générés par les activités tout en minimisant les coûts associés. Dans ce cas, la fonction objectif peut être définie comme la maximisation de la somme des profits de chaque activité, où le profit est calculé comme la différence entre les revenus et les coûts.

— **Minimisation du coût :** L'objectif de minimisation du coût vise à minimiser les coûts associés aux activités tout en respectant les contraintes de ressources. Dans ce cas, la fonction objectif peut être définies comme la minimisation de la somme des coûts de chaque activité, où les coûts peuvent inclure les coûts de production, les coûts de main-d'œuvre, les coûts de transport, etc.

— **Maximisation de la satisfaction du client :** L'objectif de maximisation de la satisfaction du client vise à maximiser la satisfaction des clients en respectant les contraintes de ressources. Dans ce cas, la fonction objectif peut être définies comme la maximisation de la somme des scores de satisfaction des clients pour chaque activité, où le score de satisfaction peut être mesuré à l'aide d'une échelle de satisfaction ou d'un indicateur de performance.

— **Minimisation du temps de réalisation :** L'objectif de minimisation du temps de réalisation vise à minimiser le temps nécessaire pour réaliser les activités, tout en respectant les contraintes de ressources. Dans ce cas, la fonction objectif peut être définies comme la minimisation de la somme des temps de réalisation de chaque activité, où le temps de réalisation peut inclure le temps de production, le temps de transport, le temps d'attente, etc.

— **Equilibrage d'utilisation des ressources :** L'objectif de l'équilibrage dans le problème d'allocation des ressources est de répartir équitablement les ressources disponibles entre les différentes entités ou activités qui en ont besoin. L'équilibrage vise à éviter les déséquilibres ou les disparités excessives dans l'allocation des ressources, ce qui pourrait entraîner une utilisation inefficace ou injuste des ressources.

Remarque 3.2.2. *Il est important de noter que ces objectifs peuvent être en conflit les uns avec les autres et qu'il peut être difficile d'optimiser tous les objectifs simultanément. C'est pourquoi il est souvent nécessaire de trouver un compromis entre les différents objectifs en utilisant des méthodes d'optimisation multi-objectif.*

3.2.3 Méthodologie de résolution

La méthodologie de résolution d'un problème d'allocation de ressources dépend du modèle mathématique utilisé et des objectifs à optimiser. Toutefois, il existe une méthodologie générale qui peut être appliquée pour résoudre un problème d'allocation de ressources mono-objectif ou multi-objectif. Voici les étapes clés de cette méthodologie :

1. **Formulation du problème :** La première étape consiste à formuler le problème d'allocation de ressources de manière précise et complète. Cela implique de définir les variables de décision, les contraintes, les objectifs à optimiser et les critères d'évaluation.

2. **Choix du modèle mathématique :** La deuxième étape consiste à choisir le modèle mathématique le plus adapté au problème d'allocation de ressources. En fonction des spécificités du problème et des objectifs à optimiser, il peut s'agir d'un modèle linéaire, non-linéaire, multi-objectif, etc.

3. **Résolution du problème :** La troisième étape consiste à résoudre le problème d'allocation de ressources en utilisant une méthode d'optimisation appropriée. Selon le modèle mathématique choisi, il peut s'agir de méthodes analytiques ou de méthodes heuristiques.

4. **Analyse des résultats :** La quatrième étape consiste à analyser les résultats obtenus pour déterminer la qualité de la solution optimale et son adéquation par rapport aux objectifs à optimiser. Cela implique de vérifier si les contraintes sont respectées et si les objectifs sont atteints.

5. **Validation de la solution :** La cinquième étape consiste à valider la solution optimale en effectuant des simulations et en réalisant des tests sur le terrain si possible. Il s'agit de vérifier si la solution optimale peut être mise en œuvre dans des conditions réelles et si elle est efficace.

6. **Mise en œuvre et suivi :** La dernière étape consiste à mettre en œuvre la solution optimale et à assurer un suivi régulier pour vérifier si elle reste efficace et si elle peut être améliorée.

3.3 Illustration par application

Considérons un problème d'allocation de ressources multiobjectif, où nous avons l'intérêt d'allouer j ressources disponibles à i tâches différentes.

Nous allons présenter trois variantes du problème d'allocation des ressources, et pour chaque variante, nous utiliserons un modèle mathématique. Les données du modèle seront les suivantes :

Données du problème :

— n : nombre de tâches ,
— m : nombre de ressources disponibles,
— s_i : la valeur de la satisfaction d'un client lors d'une réalisation d'une tâche i,
— c_{ij} : coût d'affectation de la ressource j à la tâche i,
— t_{ij} : temps d'affectation de la tâche i à une ressource j,
— b_j : la quantité de la ressource disponible j, $\quad j = 1 \dots m$,
— Q_{ij} : la portion de la ressource j affecté à la tâche i.
— x_{ij} : la quantité de la ressource j nécessaire pour la tâche i.
— p_{ij} : le profit associé à l'affectation de la ressource j à la tâche i,
— h_i : une mesure de l'utilisation de la ressource j pour la tâche i, (par exemple, une mesure de charge, de capacité ou d'utilisation).

3.3.1 Variantes 1

Objectifs du problème :

1. La fonction ϕ est représenté par la maximisation de la satisfaction du clients.
2. La fonction objectif f_1 sera la minimisation du coût.
3. La fonction objectif f_2 sera minimisation du temps de réalisation.

3.3.1.1 Modèle mathématique

$$(Var1)\begin{cases} \max & \phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} s_i x_{ij} \\ \min & f_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} \\ \min & f_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} x_{ij} \\ \text{s.c.} & \sum_{i=1}^{n} Q_{ij} x_{ij} \leq b_j \quad j = 1 \dots m \\ \text{s.c.} & x_{ij} \in \mathbb{N} \quad \forall i, j \end{cases} \tag{3.3}$$

3.3.2 Variante 2

Objectifs du problème :

1. La fonction ϕ représentera la minimisation du coût.
2. La maximisation du profit est représenté par la fonction f_1.
3. La maximisation de la satisfaction des clients est représenté par la fonction f_2.

3.3.2.1 Modèle mathématique

$$(Var2)\begin{cases} \min & \phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} \\ \max & f_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} x_{ij} \\ \max & f_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} s_i x_{ij} \\ \text{s.c.} & \sum_{i=1}^{n} Q_{ij} x_{ij} \leq b_j \quad j = 1 \dots m \\ \text{s.c.} & x_{ij} \in \mathbb{N} \quad \forall i, j \end{cases} \tag{3.4}$$

3.3.3 Variante 3

Les objectifs de cette variante sont :

1. Maximisation du profit est représenté par la fonction ϕ .
2. f_1 : Minimisation du temps de réalisation.
3. f_2 : Minimisation de coût.
4. f_3 : Maximisation de la satisfaction des clients.
5. f_4 : Équilibrage d'utilisation des ressources.

3.3.3.1 Modèle mathématique

$$\text{(Var 3)} \begin{cases} \max & \phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij} \\ \min & f_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij}x_{ij} \\ \min & f_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \max & f_3(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} s_i x_{ij} \\ \max & f_4(x) = \sum_{j=1}^{m}(\frac{1}{m}\sum_{i=1}^{n} h_i - \frac{1}{n}\sum_{i=1}^{n} h_i)x_{ij} \\ \text{s.c.} & \sum_{i=1}^{n} Q_{ij}x_{ij} \leq b_j \quad j = 1 \dots m \\ \text{s.c.} & x_{ij} \in \mathbb{N} \quad \forall i, j \end{cases} \tag{3.5}$$

Remarque 3.3.1. *La fonction $f_4(x)$ mesure l'écart moyen entre l'utilisation des ressources par chaque tâche et la moyenne globale des utilisations des ressources. L'objectif est de maximiser cette fonction afin d'atteindre un équilibrage optimal de l'utilisation des ressources entre les tâches.*

Conclusion

L'optimisation multiobjectif est une approche puissante pour résoudre des problèmes complexes d'allocation de ressources. Les modèles mathématiques permettent de modéliser les problèmes, d'identifier les solutions optimales et de prendre des décisions éclairées en prenant en compte les différents objectifs et les contraintes du problème.

Dans ce chapitre, nous avons examiné la modélisation mono-objectif et multi-objectif du problème d'allocation de ressources. Dans la modélisation mono-objectif, nous avons présenté un modèle mathématique et exploré différentes variantes. Pour la modélisation multi-objectif, nous avons

discuté de la définition des objectifs et de la méthodologie de résolution. Nous avons également présenté des applications concrètes du problème d'allocation des ressources à travers trois variantes spécifiques.

Dans le dernier chapitre à venir, nous présenterons notre algorithme et appliquerons notre approche à l'une des variantes du problème (variante 2). Notre objectif sera de trouver une solution optimale et de présenter des résultats numériques qui illustreront l'efficacité de notre approche.

Chapitre 4

Modélisation et implémentation de l'approche développée

Introduction

Le problème d'optimisation d'une fonction sur l'ensemble des solutions efficaces d'un problème multi-objectif en nombre entier est un défi majeur en recherche opérationnelle. Cependant, la recherche de solutions efficaces à ce problème est souvent très difficile, voire impossible en raison de la complexité combinatoire de l'espace de recherche.

En général, un ensemble de solutions est souvent la cible des méthodes de résolutions, ce qui complique le choix du décideur. Dans ce contexte, les algorithmes d'estimation de distribution (EDA) sont de plus en plus utilisés pour résoudre les problèmes d'optimisation multi-objectif. Cependant, l'utilisation de ces algorithmes sur l'ensemble des solutions efficaces d'un MOILP pose des défis particuliers, tels que la nécessité de prendre en compte les contraintes en nombres entiers et la préservation de la diversité des solutions dans la population.

Dans ce chapitre, nous proposons un modèle mathématique pour résoudre ce type de problèmes et nous présentons l'adaptation de l'algorithme EDA pour prendre en compte les spécificités des MOILP en nombres entiers. Enfin, notre algorithme utilise les EDAs pour obtenir une solution unique optimale au problème considéré.

4.1 Modèle mathématique

Dans cette section, nous décrivons le modèle mathématique utilisé pour le problème d'optimisation d'une fonction sur l'ensemble des solutions efficaces d'un problème multi-objectif en nombres entiers. Le modèle peut être formulé comme suit :

$$(MOILP)\begin{cases}\text{minimiser} & Z_i(x) = c^i(x),\ i = 1, \dots k \\ \text{s.c} & Ax \leq b \\ & x \in \mathbb{Z}\end{cases} \tag{4.1}$$

où :

- A est une matrice de dimension $(m \times n)$ et b est un vecteur de dimension $(m \times 1)$.
- c^i estun vecteur de dimension $(n \times 1)$ qui représente les coefficients de la fonction objectif $Z_i(x)$ pour chaque objectif $i = 1, \dots, k$.
- $\mathbb{Z}$ est l'ensemble de solutions entières.

Nous notons par $\mathbb{F}$ l'ensemble des solutions efficaces du problème (4.1). Le problème que nous allons considérer est le $(MOILP_{\mathbb{F}})$ défini comme suit :

$$(MOILP_{\mathbb{F}})\begin{cases}\min & \Phi = \phi^{\top} x, \\ \text{s.c.} & x \in \mathbb{F}.\end{cases} \tag{4.2}$$

où ϕ représente un vecteur de dimension n.

Le modèle mathématique (4.2) décrit un problème d'optimisation mono-objectif où l'objectif est de trouver une solution optimale, tout en respectant les contraintes du problème MOILP (4.1), qui sont représentées par un ensemble de solutions efficaces.

Dans les sections suivantes, nous décrivons notre démarche de résolution de ce problème qui utilise les algorithmes EDA.

4.2 Choix et adaptation de l'algorithme d'estimation de distribution

L'algorithme EDA est une méthode d'optimisation stochastique qui repose sur l'estimation de la distribution de probabilité des solutions réalisables. Cette approche est utilisée pour estimer une distribution de probabilité à partir d'une population initiale de solutions réalisables, puis générer une nouvelle population de solutions en se basant sur cette distribution.

Nous commençons initialement par introduire le problème en question, avec $A_{m\times n}$ la matrice des contraintes où n : est le nombre de variables et m : est le nombre de contraintes. De même, $C_{k\times n}$: est la matrice des objectifs avec k comme nombre d'objectifs. b_m : est le vecteur de droite et ϕ_n : est l'objectif principal.

Les paramètres de notre algorithme sont initialisés de la manière suivante :

npo **:** taille de la population,

T **:** représente le nombre d'itérations,

θ **:** portion de solutions dominées pour la génération K, est initialisé à 1,

ϵ **:** indice de mise à jour où $\epsilon = \frac{2\times\theta}{T}$,

sol_d **:** ensemble de solutions dominantes initialisées à l'ensemble vide,

S **:** la taille de la population générée par la loi uniforme et plasée dans l'ensemble des solutions aléatoire où $S = \frac{1}{5}$.

Et nous générons notre population initiale en utilisant une loi uniforme $U(0.5, npo)$.

Le déroulement de notre algorithme commence par la vérification des contraintes, suivie de la vérification de la dominance. Nous récupérons les solutions dominantes et les ajouter à l'ensemble Sol_d. Ensuite, nous sélectionnons θ solutions qui donnent les meilleures valeurs de ϕ parmi les solutions dominées, et puis nous les plaçons dans l'ensemble des solutions_dominées.

Ensuite, nous générons un sous-ensemble de taille S en utilisant une loi uniforme $U(0.5, S)$ et le placer dans l'ensemble des solutions_aléatoires. Nous regroupons les trois ensembles dans un ensemble unique appelé **Solution**, soit Solution = Sol_d ∪ Solutions_dominées ∪ Solutions_aléatoires.

Le dernier ensemble obtenu (Solution) nous permet de paramétrer la loi qui génère les variables de notre population P^{t+1} à l'itération $t+1$. Nous répétons cette procédure jusqu'à atteindre $t = T$.

À la fin, nous obtenons la solution unique et optimale x^*, ainsi que la valeur optimale ϕ^* de l'objectif principal.

4.2.1 Algorithme

L'approche proposée est implémentée sur machine en utilisant le langage de programmation libre Python.

Algorithm 8 Optimisation d'une fonction sur l'ensemble des solutions efficaces d'un problème multi-objectif en nombres entiers

Entrée :
$\downarrow n$: nombre de variables ;
$\downarrow m$: nombre de contraintes ;
$\downarrow k$: nombre de d'objectifs ;
$\downarrow npo$: taille de la population ;
$\downarrow$ n_interval = nombre de répartition des intervalles ;
$\downarrow T$ = nombre d'itérations ;
Sortie :
$\uparrow x^*$: solution optimale du problème (4.1) ;
$\uparrow \phi^*$: valeur optimale d'objectif principale ;
(Initialisation) :
$t \leftarrow 0$, $\theta \leftarrow 1$, $\epsilon \leftarrow \frac{2\times\theta}{T}$, $S \leftarrow \frac{1}{5}$, Sol_d $= \varnothing$;
Générer la population initiale P^0 ;
TANT QUE $t < T$ **FAIRE**
  Vérifier la réalisabilité du P^t ;
  Sol_d $\leftarrow$ Sol_d $\cup$ Solutions_dominantes de P^t ;
  Solutions_dominées $\leftarrow$ récupérer θ solutions dominées de P^t ;
  Solutions_aléatoire $\leftarrow$ Générer avec la loi uniforme U(0.5, S) ;
  Solutions $\leftarrow$ Sol_d $\cup$ Solutions_dominées $\cup$ Solutions_aléatoires ;
  Construction de la nouvelle population P^{t+1} ;
  t < t + 1 ;
Fin TANT QUE.

4.3 Exemple didactique

Pour faciliter la compréhension de l'approche et de sa démarche, nous présentons l'exemple suivant comme une illustration (variante 2).

Considérons un problème d'allocation de ressources multiobjectif, où nous avons l'intérêt d'allouer 2 ressources disponibles à 2 tâches différentes, et un nombre d'itérations $T = 3$.

$$(MORAP)\begin{cases} \min \Phi(x) = ((3)x_{11} + (2)x_{12} + (1)x_{21} + (4)x_{22}) \\ S.C. \\ \quad \begin{cases} \max f_1(x) = \quad (1)x_{11} + (3)x_{12} + (2)x_{21} + (1)x_{22} \\ \max f_2(x) = \quad (2)x_{11} + (1)x_{12} + (1)x_{21} + (1)x_{22} \\ \\ S.c. \\ \qquad\qquad (1)x_{11} + (0)x_{12} + (1)x_{21} + (0)x_{22} \leq 3 \\ \qquad\qquad (0)x_{11} + (1)x_{12} + (0)x_{21} + (1)x_{22} \leq 2 \\ \qquad\qquad 0 \leq x_{ij} \leq 2\,, i \in \{1,2\},\ j \in \{1,2\} \end{cases} \end{cases} . \tag{4.3}$$

Les paramètres de problème (4.3) sont les suivant :

$n = 4$: nombre de variables,
$m = 2$: nombre de contraintes,
$k = 3$: nombre d'objectifs,
$npo = 10$: taille de la population,
n_interval = 3 : nombre de répartition des intervalles,
$T = 3$: nombre d'itérations,

Remarque 4.3.1. *chaque vecteur x_{ij} représente une instance individuelle et séparée dans l'espace des vecteurs.*

Après l'exécution de problème on aura ces résultat dans les tableaux suivants :

	x_{11}	x_{12}	x_{21}	x_{22}	$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_1	2	0	1	2	15	6	7
s_2	0	0	0	1	4	1	1
s_3	0	0	0	2	8	2	2
s_4	0	0	1	0	1	2	1
s_5	2	1	0	1	12	6	6
s_6	0	0	1	2	9	4	3
s_7	1	2	2	0	9	11	6
s_8	2	1	1	1	13	8	7
s_9	0	0	2	2	10	6	4
s_{10}	2	2	1	0	11	10	7
Probabilités							
variable \ intervalle	0	1	2				
x_{11}	0.33	0.33	0.33				
x_{12}	0.33	0.33	0.33				
x_{21}	0.33	0.33	0.33				
x_{22}	0.33	0.33	0.33				

TABLE 4.1 – La population générée dans la première itération

Le tableau (4.1) est obtenu en sélectionnant dix solutions de la population générées de manière aléatoire. Dans cette première étape, chaque variable a la même probabilité d'être choisie. Dans notre cas, puisque x_{ij} prend les valeurs 0, 1 et 2, la probabilité de chaque valeur est de 0,33. Cela est basé sur la répartition uniforme de l'intervalle [0, 1] en trois valeurs équivalentes.

	x_{11}	x_{12}	x_{21}	x_{22}	$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_1	2	0	1	2	15	6	7
s_7	1	2	2	0	9	11	6
s_8	2	1	1	1	13	8	7
s_{10}	2	2	1	0	11	10	7
Probabilités							
variable \ intervalle	0	1	2				
x_{11}	0.33	0.33	0.33				
x_{12}	0.33	0.33	0.33				
x_{21}	0.33	0.33	0.33				
x_{22}	0.33	0.33	0.33				

TABLE 4.2 – Les solutions non dominées de la premiere population

Le tableau (4.2) représente les solutions dominantes (s_1, s_7, s_8, s_{10}), c'est-à-dire les solutions non dominées, par rapport aux fonctions objectifs $f_1(x)$ et $f_2(x)$.

	x_{11}	x_{12}	x_{21}	x_{22}	$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_1	1	2	0	0	7	7	4
s_2	2	0	1	2	15	6	7
s_3	1	2	1	0	8	9	5
s_4	0	2	2	0	6	10	4
s_5	1	2	2	0	9	11	6
s_6	0	0	0	0	0	0	0
s_7	0	0	2	0	2	4	2
s_8	2	2	1	0	11	10	7
s_9	1	0	2	0	5	5	4
s_{10}	1	0	2	2	13	7	6
Probabilités							
variable \ intervalle	0	1	2				
x_{11}	0.3	0.5	0.2				
x_{12}	0.5	0	0.5				
x_{21}	0.2	0.3	0.5				
x_{22}	0.8	0	0.2				

TABLE 4.3 – La population générée dans la deuxième itération

Le tableau (4.3) est obtenu de la manière suivante : d'abord, nous avons gardé les solutions dominantes, puis nous avons généré une nouvelle population. Cette étape est réalisée car il est possible de générer de nouvelles solutions qui pourraient dominer les solutions déjà trouvées.

Cette fois-ci, les probabilités que chaque variable prenne les valeurs 0, 1 et 2 changent. Ces probabilités sont calculées en sommant le nombre de chaque valeur pour chaque variable dans la population actuelle. Ainsi, la nouvelle population de solutions est générée en se basant sur ces probabilités ajustées.

	x_{11}	x_{12}	x_{21}	x_{22}	$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_2	2	0	1	2	15	6	7
s_5	1	2	2	0	9	11	6
s_8	2	2	1	0	11	10	7
Probabilités							
variable \ intervalle	0	1	2				
x_{11}	0.3	0.5	0.2				
x_{12}	0.5	0	0.5				
x_{21}	0.2	0.3	0.5				
x_{22}	0.8	0	0.2				

TABLE 4.4 – Les solutions non dominées de la deuxième population

Le tableau (4.4) représente les solutions non dominées (s_2, s_5, s_8) par rapport aux fonctions objectifs $f_1(x)$ et $f_2(x)$.

	x_{11}	x_{12}	x_{21}	x_{22}	$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_1	0	0	2	0	2	4	2
s_2	0	2	0	0	4	6	2
s_3	0	2	2	0	6	10	4
s_4	1	0	1	0	4	3	3
s_5	1	0	2	0	5	5	4
s_6	1	0	2	2	13	7	6
s_7	1	2	0	0	7	7	4
s_8	1	2	1	0	8	9	5
s_9	1	2	2	0	9	11	6
s_{10}	2	2	1	0	11	10	7
Probabilités							
variable \ intervalle	0	1	2				
x_{11}	0.3	0.6	0.1				
x_{12}	0.4	0	0.6				
x_{21}	0.2	0.3	0.5				
x_{22}	0.9	0	0.1				

TABLE 4.5 – La population générée dans la troisième itération

En générant de la même manière lors de la deuxième itération et en se basant sur les probabilités du tableau (4.3) et (4.4), nous avons obtenu le tableau (4.5).

	x_{11}	x_{12}	x_{21}	x_{22}		$\Phi(x)$	$f_1(x)$	$f_2(x)$
s_9	1	2	2	0	9	11	6	
s_{10}	2	2	1	0	11	10	7	
Probabilités								
variable \ intervalle	0	1	2					
x_{11}	0.3	0.6	0.1					
x_{12}	0.4	0	0.6					
x_{21}	0.2	0.3	0.5					
x_{22}	0.9	0	0.1					

TABLE 4.6 – Les solutions non dominées de la troisième population

Le tableau (4.6) représente les solutions dominantes (s_9, s_{10}) par rapport aux fonctions objectifs $f_1(x)$ et $f_2(x)$. Cependant, puisque nous cherchons une solution unique, il est nécessaire de choisir entre ces deux solutions. Étant donné qu'elles sont incomparables en termes de dominance, nous devons utiliser un critère supplémentaire, ϕ. Dans notre cas, puisque ϕ doit être minimisé, la solution optimale du problème (4.3) est donc s_9.

4.4 Expérimentation Numérique

Afin de tester l'efficacité de notre approche, nous procédons dans ce qui suit à l'exécution de cette dernière sur des instances numériques de problème 4.1 de différente tailles.

4.4.1 Python

Python est un langage de programmation interprété, orienté objet et de haut niveau. Il a été créé par Guido van Rossum et sa première version a été publiée en 1991. Python se distingue par sa syntaxe claire et lisible, ce qui en fait un langage très populaire et apprécié des développeurs.

Python est un langage de programmation polyvalent que nous avons utilisé pour implémenter notre projet.

D'abord, nous avons opté pour l'installation d'Anaconda, une plate-forme qui comprend un environnement de développement intégré (IDE) appelé Spyder. L'installation d'Anaconda était simple et rapide, et elle nous a permis de bénéficier d'un ensemble complet d'outils pour le développement en Python.

Spyder est un environnement de développement puissant et convivial. Avec sa mise en page intuitive et ses fonctionnalités avancées, il nous a permis d'écrire et d'exécuter le code Python de manière efficace. Nous avons particulièrement apprécié son éditeur de code, qui propose des fonctionnalités telles que la coloration syntaxique, l'autocomplétion et la suggestion intelligente, ce qui a grandement facilité notre processus de programmation.

Python et l'utilisation de Spyder comme IDE ont été une combinaison extrêmement puissante pour notre projet de programmation. La simplicité et la lisibilité du langage Python, combinées

à la facilité d'utilisation de Spyder, ont contribué à rendre notre expérience de programmation productive.

4.5 Résultats expérimentaux

Pour montrer la performance et l'efficacité de notre approche, nous allons l'exécuter sur des instances de problème de plusieurs taille(Objectifs, contraintes, variables). puis nous prélevons les performances associé à chaque exécution.

Nous considérons comme indices de performance le temps d'exécution et la convergence vers la solution optimale. Ces résultats seront présentés dans les tableaux suivants, où nous avons varié à chaque fois le nombre d'objectifs k, le nombre de variables n et le nombre de contraintes m.

k=5		
$m \times n$	Temps	Convergence
20×30	101.2654	0.6257
50×70	276.7441	0.5363
80×100	423.4512	0.9524
100×200	1169.8068	1.0245

TABLE 4.7 – Résultats de l'exécution de l'algorithme avec différentes dimensions du problème cas : $k = 5$

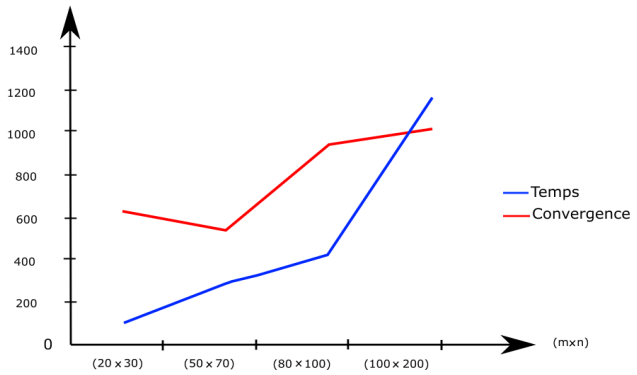
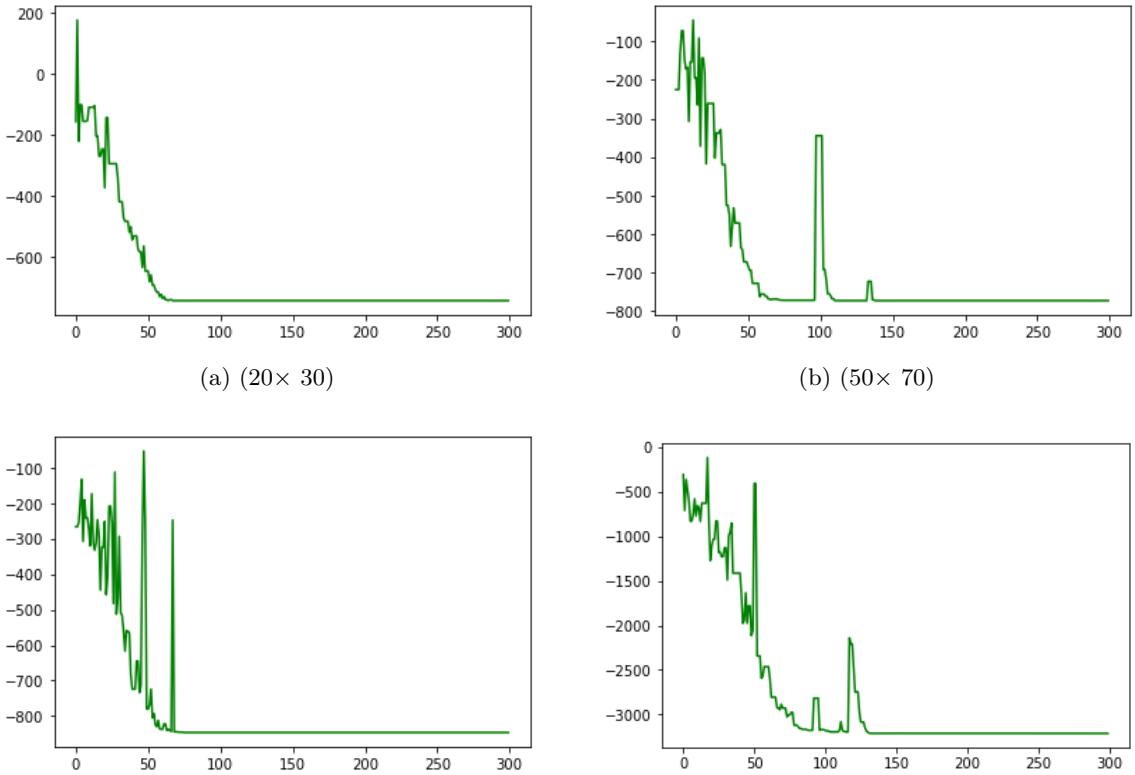


FIGURE 4.1 – Le comportement d'indices de performence cas : $k = 5$



(a) (20× 30)

(b) (50× 70)

(c) (80× 100)

(d) (100× 200)

FIGURE 4.2 – Les valeurs de la fonction ϕ en fonction de nombre d'itération cas : $k = 5$

k=10		
$m \times n$	Temps	Convergence
20×30	105.7855	0.9227
50×70	299.6925	1.3332
80×100	477.2317	1.3023
100×200	2378.2114	0.8473

TABLE 4.8 – Résultats de l'exécution de l'algorithme avec différentes dimensions du problème cas : $k = 10$

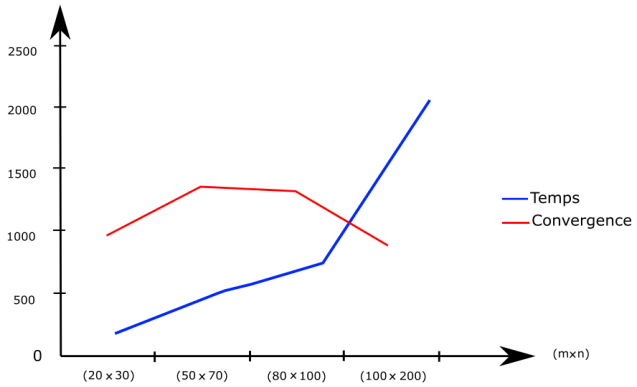
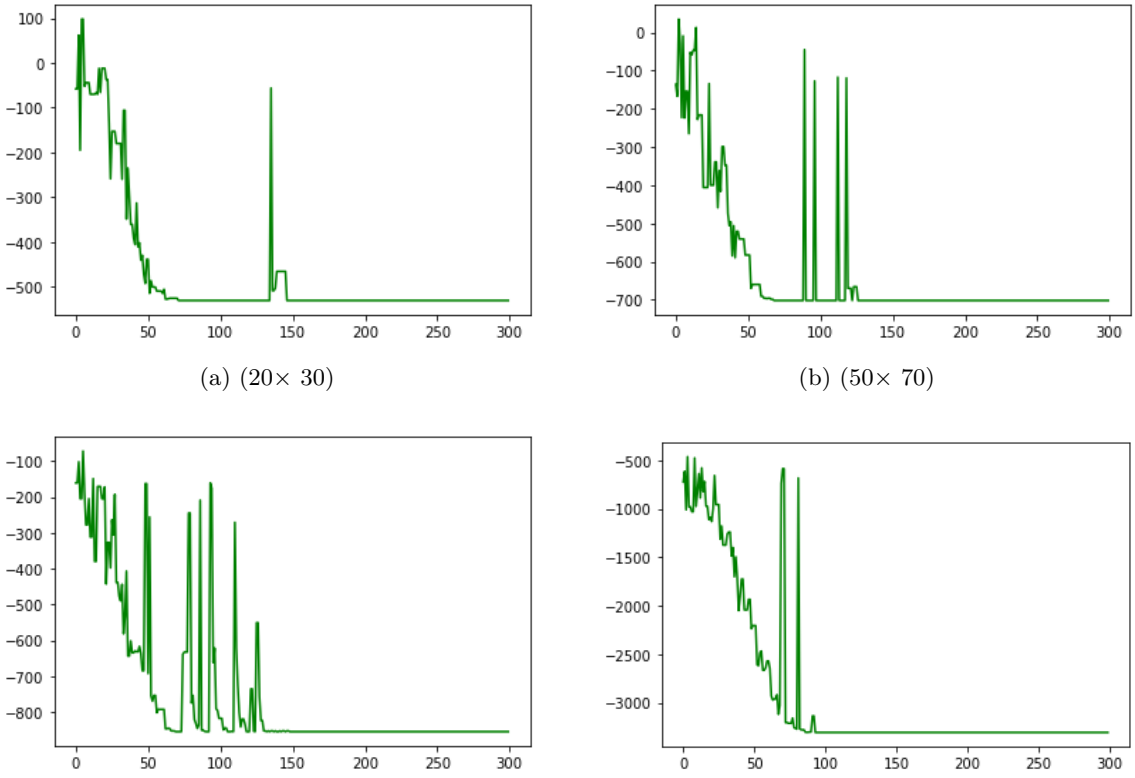


FIGURE 4.3 – Le comportement d'indices de performence cas : $k = 10$



(a) (20× 30)

(b) (50× 70)

(c) (80× 100)

(d) (100× 200)

FIGURE 4.4 – Les valeurs de la fonction ϕ en fonction de nombre d'itération cas : $k = 10$

k=20		
$m \times n$	Temps	Convergence
20×30	108.6834	0.8348
50×70	287.2989	0.8532
80×100	506.8144	0.9790
100×200	3017.446339	1.3315

TABLE 4.9 – Résultats de l'exécution de l'algorithme avec différentes dimensions du problème cas : $k = 20$

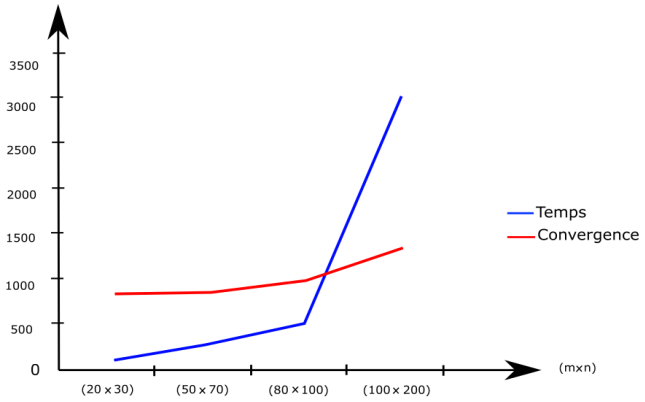
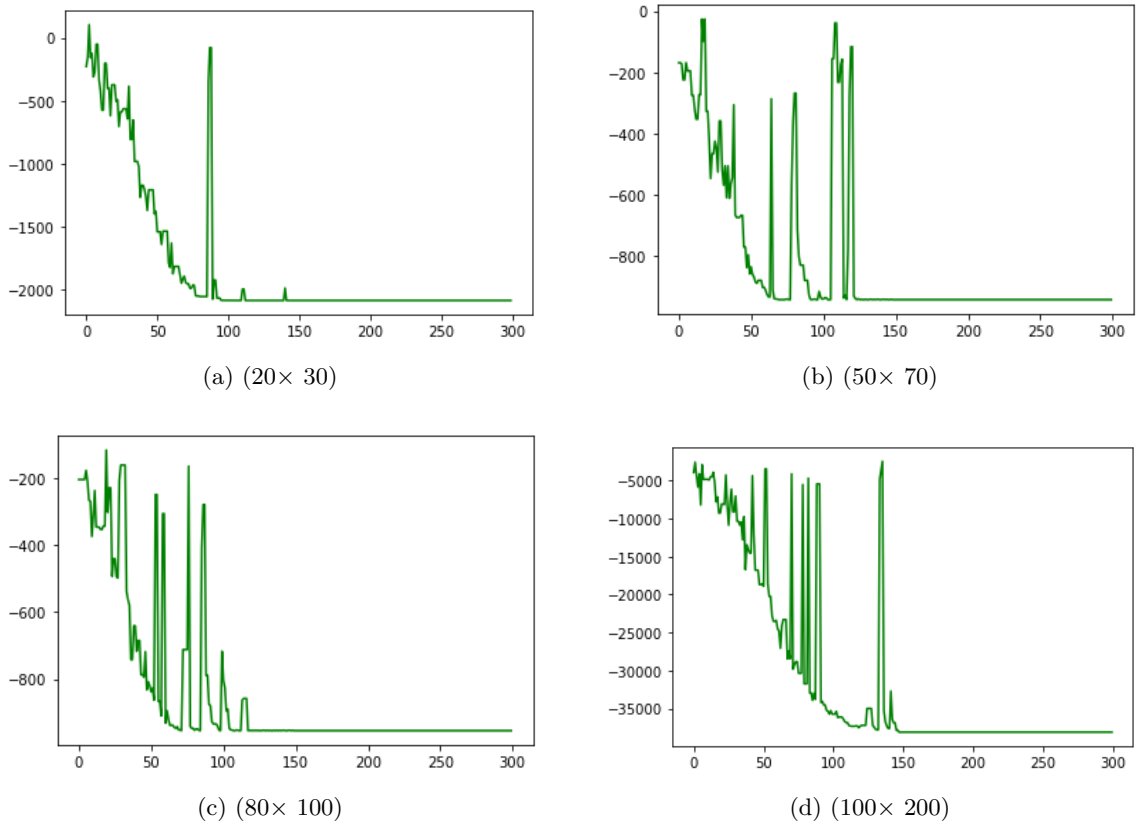


FIGURE 4.5 – Le comportement des indices de performence cas : $k = 20$



(a) (20× 30)

(b) (50× 70)

(c) (80× 100)

(d) (100× 200)

FIGURE 4.6 – Les valeurs de la fonction ϕ en fonction de nombre d'itération cas : $k = 20$

k=50		
$m \times n$	Temps	Convergence
20×30	114.0612	0.9524
50×70	294.8201	0.7462
80×100	526.7914	1.0751
100×200	3193.4694	1.1382

TABLE 4.10 – Résultats de l'exécution de l'algorithme avec différentes dimensions du problème cas : $k = 50$

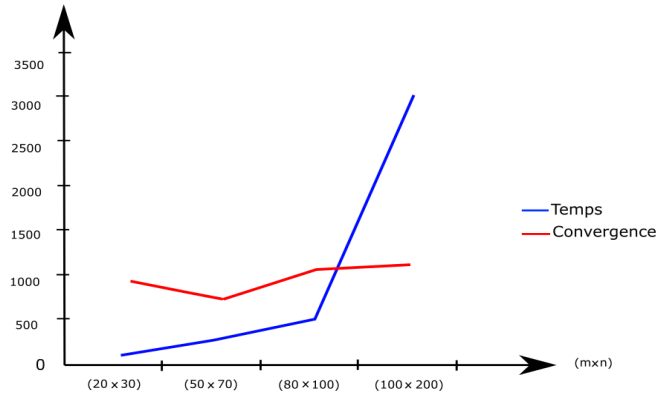
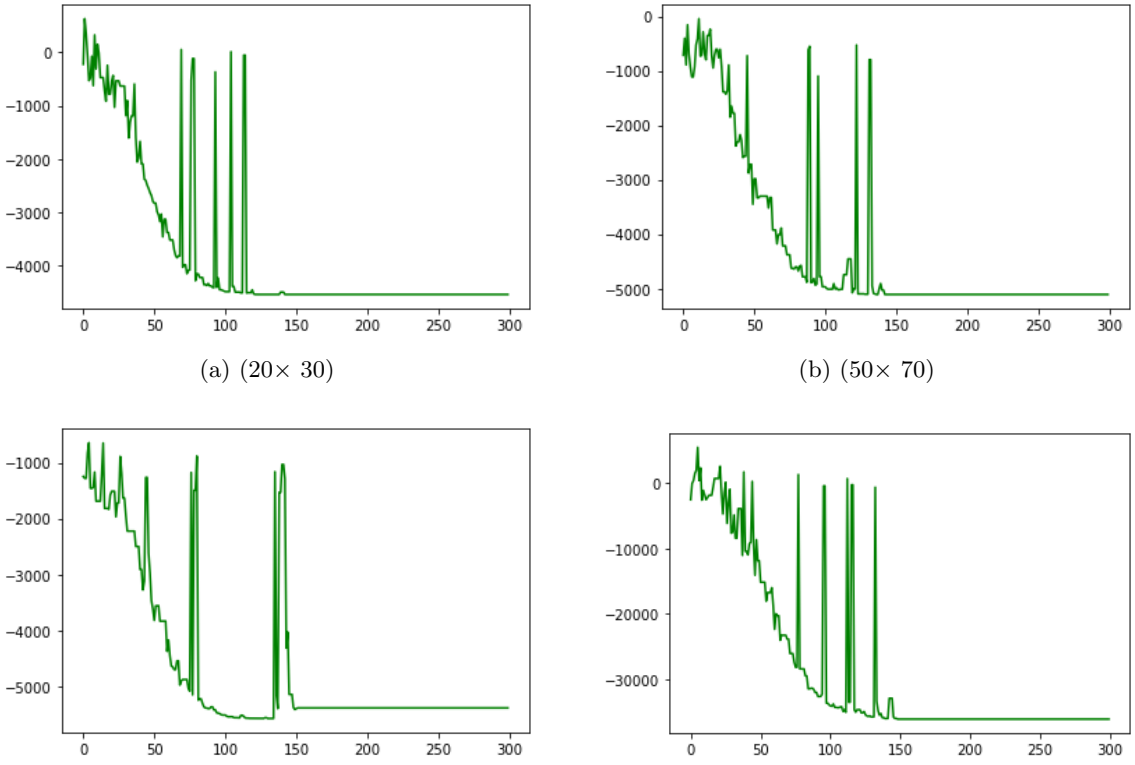


FIGURE 4.7 – Le comportement des indices de performence cas $k = 50$



(a) (20× 30)

(b) (50× 70)

(c) (80× 100)

(d) (100× 200)

FIGURE 4.8 – Les valeurs de la fonction ϕ en fonction de nombre d'itération cas : $k = 50$

4.6 Discussion des résultats

Les résultats des tableaux fournissent des informations importantes sur les performances de l'algorithme pour différentes tailles de problème. Une analyse plus approfondie de ces résultats révèle plusieurs aspects importants à considérer.

Tout d'abord, en ce qui concerne le temps d'exécution, on peut observer une tendance générale à l'augmentation du temps lorsque le nombre d'objectifs (k) et le nombre de variables (n) augmentent. Cependant, il est crucial de noter que cette relation n'est pas linéaire, car d'autres facteurs tels que la complexité de l'algorithme et la puissance de calcul de la machine utilisée peuvent également influencer les temps d'exécution. Par conséquent, il est essentiel d'examiner attentivement ces facteurs lors de l'interprétation des résultats.

En ce qui concerne la convergence, les valeurs d'indice de convergence varient d'une exécution à l'autre et ne montrent pas de tendance claire en fonction de la taille de la population ou du nombre d'objectifs. Il peut y avoir des variations significatives dans la capacité de l'algorithme à converger vers une solution optimale, ce qui peut être dû à des facteurs tels que la nature stochastique de l'algorithme ou la complexité du problème lui-même. Il serait intéressant d'explorer davantage ces variations et de les analyser en profondeur pour comprendre les conditions dans lesquelles l'algorithme atteint une convergence plus élevée.

De plus, la taille de la population ($m \times n$) a également une influence sur les résultats. On peut observer que pour une même valeur de k, lorsque la taille de la population augmente, le temps d'exécution a tendance à augmenter également. Cela peut être dû à une plus grande quantité de calcul nécessaire pour évaluer une population plus importante. Il convient de noter que l'impact de la taille de la population sur la convergence peut varier et mériterait une analyse plus approfondie.

l'analyse des résultats indique que l'algorithme présente une sensibilité à la complexité du problème, ce qui se traduit par une augmentation du temps d'exécution lorsque la taille de l'instance augmente. Cette sensibilité peut limiter la faisabilité pratique de l'approche pour les problèmes réels de grande taille.

En effet, le comportement des graphes des valeurs de la fonction ϕ en fonction du nombre d'itérations représentant dans les figures (4.2), (4.4), (4.6), (4.8) montrent une exploration initiale de l'espace de recherche, où la fonction ϕ diminue rapidement et atteint des valeurs différentes. Ensuite, la fonction ϕ atteint un plateau où elle oscille autour d'une valeur minimale, qui représente la solution optimale. Cette valeur minimale est atteinte à une certaine itération et reste stable pour les itérations suivantes.

Il est important de souligner que ces résultats sont basés sur les données fournies dans les tableaux, et une analyse plus complète nécessiterait une évaluation statistique plus rigoureuse, y compris des tests de significativité et une comparaison avec d'autres approches existantes. De plus, l'utilisation de métriques supplémentaires pour évaluer les performances de l'algorithme, telles que l'efficacité de la solution obtenue, pourrait fournir des informations plus complètes sur sa qualité.

Conclusion

Dans ce chapitre, nous avons développé une approche pour résoudre un problème MOILP sur l'ensemble des solutions efficaces, ou nous avons présenté un modèle mathématique définissant les objectifs du problème.

Nous avons ainsi choisi et adapté l'algorithme EDA (Estimation of Distribution Algorithm) pour résoudre ce problème. À l'aide de Python, nous avons réalisé des expérimentations numériques en paramétrant la méthode et en analysant les résultats obtenus.

Cette approche a été illustrée par un exemple didactique qui est le problème d'allocation de ressources où les objectifs sont contradictoires(la minimisation des coûts, la maximisation du profit et la satisfaction maximale des clients).

Les expérimentations ont montré l'efficacité de notre approche pour atteindre un compromis entre les objectifs contradictoires de l'allocation de ressources. Les solutions obtenues ont démontré de bonnes performances en termes de profit, de temps de réalisation, de coûts et de satisfaction client.

Par suite, des expérimentations numériques sur différentes tailles de problèmes MOILP ont été munis. Ceci offre des solutions prometteuses pour ce dernier. Elle permet ainsi, de trouver des solutions équilibrées et optimales, tout en prenant en compte les multiples objectifs du problème. Des perspectives d'amélioration et d'extension de cette approche sont envisageables pour des problèmes plus complexes et pour de nouveaux domaines d'application.

En synthèse, les résultats de l'analyse mettent en évidence une corrélation entre la taille de l'instance et le temps d'exécution de l'algorithme, indiquant une sensibilité à la complexité du problème. Cette sensibilité peut constituer un obstacle à la faisabilité pratique de l'approche pour les problèmes de grande envergure.

Conclusion générale

La recherche opérationnelle a évolué pour prendre en compte la multiplicité des objectifs dans la modélisation des problèmes. L'optimisation multi-objectif est devenue essentielle pour résoudre les problèmes réels qui impliquent plusieurs objectifs concurrents. Les problèmes de programmation multi-objectifs à variables discrètes, tels que les MOILP, sont couramment rencontrés dans divers domaines.

Les approches traditionnelles pour résoudre les MOILP peuvent être limitées en termes de temps de calcul. C'est pourquoi les Algorithmes d'Estimation de Distribution (EDA) ont été introduits comme une alternative prometteuse. Les EDA utilisent des modèles probabilistes pour représenter et évoluer la distribution de probabilité des solutions possibles, ce qui leur permet d'explorer efficacement l'espace de recherche complexe des solutions entières.

Dans ce contexte, notre travail consiste à adapter l'utilisation des EDA pour résoudre des problèmes MOILP en améliorant l'ensemble des solutions efficaces. Nous avons utilisé une distribution de probabilité uniforme pour générer de nouvelles solutions potentielles et modifié les paramètres de cette distribution en fonction de l'importance des solutions par rapport à l'objectif primordial.

Nous avons choisi de nous concentrer sur le problème de l'allocation des ressources multi-objectif pour illustrer notre approche. Les EDA nous permettent de modéliser la relation complexe entre les variables de décision, les contraintes du problème et les multiples objectifs à atteindre. En utilisant les EDA, nous pouvons trouver un ensemble de solutions diversifiées, appelé ensemble Pareto, qui représente les solutions efficaces dans le compromis entre les différents objectifs.

À travers des expérimentations numériques, nous avons évalué les performances de notre approche. Les EDA se révèlent être des outils efficaces pour résoudre les problèmes d'optimisation multi-objectif en nombre entier, offrant des solutions précises et diversifiées.

En revanche, notre travail contribue à améliorer la compréhension et l'utilisation des EDA dans la résolution des problèmes MOILP. Les perspectives de recherche futures incluent l'exploration de variantes d'EDA, l'application à d'autres problèmes MOILP et l'amélioration des performances de l'approche proposée.

L'optimisation multi-objectif reste un domaine passionnant et en évolution, offrant des opportunités pour développer de nouvelles méthodes et approches pour aider à la prise de décision efficace et rationnelle dans divers domaines d'application.

Bibliographie

[1] L. Asli. "Approche hybride pour les problémes d'optimisation combinatoire multiobjectif : cas des problémes de type sac-á-dos". Thèse de doct. Mémoire de magister, Université USTHB, Alger, 2010.

[2] L. Asli. "Les encheres combinatoires multiobjectif dynamiques". Thèse de doct. Thése de doctorat en sciences, Université USTHB, Alger, 2019.

[3] V. Barichard. "Approches hybrides pour les problémes multiobjectifs". Thèse de doct. Ecole Doctorale d'Angers, 2003.

[4] M. Basseur et al. "Metaheuristics for multiobjective combinatorial optimization : review and recent issues". In : (2006).

[5] H. Beddiaf et al. "Optimisation de tourne des vehicules de transport de gasoil a l'aide d'un algorithme evolutionnaire". In : (2018).

[6] K. Belkeziz et A. Metrane. "Optimisation d'une fonction linéaire sur l'ensemble des solutions efficaces d'un problème multicritère quadratique convexe". In : *Annales Mathematiques Blaise Pascal*. T. 11. 1. 2004, p. 19-33.

[7] P. Bosman et D. Thierens. "The balance between proximity and diversity in multiobjective evolutionary algorithms". In : *IEEE transactions on evolutionary computation* 7.2 (2003), p. 174-188.

[8] B. Brahmi. "Contribution dans l'optimisation non convexe". Thèse de doct. Faculté des Mathématiques, 2016.

[9] D. Chaabane. "Contribution a l'optimisation multicritere en variables discretes". In : *Faculté Polytechnque de Mons* (2007).

[10] Y. Collette et P. Siarry. *Multiobjective Optimization*. Cranfield Bedford MK43 0AL UK, August 2003.

[11] B. Dakkak, Y. Chater et A. Talbi. "Modélisation d'un problème d'allocation des agents de maintenance". In : *9ème Rencontres Internationales de la Recherche en Logistique* (2012), p. 1-14.

[12] K. Deb. "Algorithms, Multi-Objective Optimization Using Evolutionary". In : *Wiley-Interscience series in systems and optimization, New York* (2001).

[13] K. Deb. “Multi-objective optimization using evolutionary algorithms John Wiley & Sons”. In : *Inc., New York, NY* (2001).

[14] K. Deb et al. “A fast and elitist multiobjective genetic algorithm : NSGA-II”. In : *IEEE transactions on evolutionary computation* 6.2 (2002), p. 182-197.

[15] M. Dorigo, M. Birattari et T. Stutzle. “Ant colony optimization”. In : *IEEE computational intelligence magazine* 1.4 (2006), p. 28-39.

[16] M. Ehrgott et X. Gandibleux. “Multiobjective combinatorial optimization—theory, methodology, and applications”. In : *Multiple criteria optimization : State of the art annotated bibliographic surveys* (2002), p. 369-444.

[17] I. Fajjari. “Resource allocation algorithms for virtual networks within cloud backbone network”. Thèse de doct. Paris 6, 2012.

[18] K. Gana et S. Jemam. “Optimisation de la distribution des produits agricoles”. Thèse de doct. Université Abderhmane Mira-Béjaia, 2021.

[19] S. Khakurel. “Energy-aware resource allocation in multi-user OFDM systems”. In : (2013).

[20] P. Larrañaga et JA. Lozano. *Estimation of distribution algorithms : A new tool for evolutionary computation*. T. 2. Springer Science & Business Media, 2001.

[21] P. Larrañaga et JA. Lozano. *Estimation of distribution algorithms : A new tool for evolutionary computation*. T. 2. Springer Science & Business Media, 2001.

[22] J. Lemesre. “Méthodes exactes pour l’optimisation combinatoire multi-objectif : conception et aplication”. Thèse de doct. U.F.R. D’I.E.E.A, 2006.

[23] JA. Lozano, I. Pedro P. Larrañaga et IE. Bengoetxea. *Towards a new evolutionary computation : advances on estimation of distribution algorithms*. T. 192. Springer Science & Business Media, 2006.

[24] P. Luong. “Energy-efficient resource allocation in limited fronthaul capacity cloud-radio access networks”. Thèse de doct. École de technologie supérieure, 2018.

[25] T. Mekhtoub et L. Toubal. “Optimisation d’une Fonction sur l’ensemble pareto d’un problème multi-objectifs discret”. Thèse de doct. UMMTO, 2019.

[26] L. Mellal. “Seuillage d’histogrammes basé sur un algorithme à estimation de distribution”. Thèse de doct. Université Mouloud Mammeri, 2012.

[27] L. Mihoubi et S. Gherbi. “Optimal Allocation and sizing of Dispersed Generation for minimizing active losses using Harris Hawks Optimization”. Thèse de doct. Université Kasdi Merbah Ouargla, 2022.

[28] O. Moselhi et P. Lorterapong. “Least impact algorithm for resource allocation”. In : *Canadian journal of civil engineering* 20.2 (1993), p. 180-188.

[29] M.Pelikan, E. David et G. Fernando. “A survey of optimization by building and using probabilistic models”. In : *Computational optimization and applications* 21 (2002), p. 5-20.

[30] H. Mühlenbein et G. Paass. “From recombination of genes to the estimation of distributions I. Binary parameters”. In : *Parallel Problem Solving from Nature—PPSN IV : International Conference on Evolutionary Computation—The 4th International Conference on Parallel Problem Solving from Nature Berlin, Germany, September 22–26, 1996 Proceedings 4*. Springer. 1996, p. 178-187.

[31] B. Obando et D. German. “Distributed methods for resource allocation : a passivity based approach”. Thèse de doct. Nantes, Ecole des Mines, 2015.

[32] M. Pelikan, DE. Goldberg et FG. Lobo. “A survey of optimization by building and using probabilistic models”. In : *Computational optimization and applications* 21 (2002), p. 5-20.

[33] M. Shahbazi. “Copula-based quantile regression”. Thèse de doct. Université du Québec à Montréal, 2020.

[34] Johann Dréo—Patrick Siarry. “Métaheuristiques pour l’optimisation et auto-organisation dans les systèmes biologiques”. In : *LERISS, A* 412 (2004).

[35] E. Talbi. *Metaheuristics : from design to implementation*. John Wiley & Sons, 2009.

[36] P. Tangpattanakul. “Multi-objective optimization of Earth observing satellite missions”. Thèse de doct. Toulouse, INSA, 2013.

[37] Y. Yamamoto. “Optimization over the efficient set : overview”. In : *Journal of Global Optimization* 22 (2002), p. 285-317.

[38] X. Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

Résumés

Résumé

Ce travail se focalise sur l'application des algorithmes d'Estimation de Distribution (EDA) pour résoudre des problèmes de programmation mathématique linéaire multi-objectifs à variables discrètes (MOILP). L'approche proposée vise à améliorer l'ensemble des solutions efficaces d'un problème multi-objectifs en explorant de manière intelligente l'espace de recherche des solutions entières afin d'optimiser l'objectif principal. Pour démontrer les performances de cette approche, nous avons utilisé le problème d'allocation des ressources multi-objectif comme exemple concret. Des expériences numériques ont été menées pour évaluer les performances de l'approche développée. En outre, ce travail aborde également les concepts fondamentaux de l'optimisation multi-objectif, les outils tels que les métaheuristiques et les EDA, ainsi qu'une modélisation du problème d'allocation des ressources.

Mots clés : MOILP, EDA, problème d'allocation des ressources, programmation mathématique linéaires multi-objectifs, optimisation multi-objectif, métaheuristiques.

Abstract :

This work focuses on the application of Estimation of Distribution Algorithms (EDA) to solve multi-objective linear mathematical programming problems with discrete variables (MOILP). The proposed approach aims to improve the set of efficient solutions for a multi-objective problem by intelligently exploring the search space of integer solutions to optimize the main objective. To demonstrate the performance of this approach, we used the multi-objective resource allocation problem as a concrete example. Numerical experiments were conducted to evaluate the performance of the developed approach. Additionally, this work also covers the fundamental concepts of multi-objective optimization, tools such as metaheuristics and EDAs, as well as a modeling of the resource allocation problem.

Keywords : MOILP, EDA, resource allocation problem, multi-objective linear mathematical programming, multi-objective optimization, metaheuristics.