MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOULOUD MAMMERI DE TIZI OUZOU
FACULTÉ DES SCIENCES
DÉPARTEMENT : MATHÉMATIQUES

MÉMOIRE DE FIN D'ÉTUDE

En vue de l'obtention du diplôme

Master en Mathématiques

SPÉCIALITÉ : **Mathématiques appliquées à la gestion**

Thème

Scatter Search Pour le Problème du Sac à Dos

Présenté par :

BELABED Safia et SAHNOUN Cilia

Devant le jury d'examen composé de :

M^{me} F. Aklouche ;	MAA ;	U.M.M.T.O ;	Présidente.
M^{me} O. Bouarab ;	MCA ;	U.M.M.T.O ;	Examinatrice.
M^{me} R. Kheffache ;	MCB ;	U.M.M.T.O ;	Promotrice.

Soutenu le 01 /10 /2019

Remerciements

Dieu merci Dieu merci Dieu merci . . . !

Nous tenons à exprimer nos sincères remerciements à toutes les personnes qui nous ont aidées de prés ou de loin jusqu'à l'achèvement et la réussite de ce modeste travail.

Nous tenons aussi à exprimer nos plus vifs remerciements à notre promotrice M^{me} Kheffache Rezika pour avoir suivie constamment avec attention la réalisation de notre travail, ainsi que pour sa disponibilité, ce qui nous a permis d'une part d'approfondir nos connaissances, d'autre part de progresser dans notre travail et nous remercions également M^{me} Bouarab Ouiza et M^{me} Aklouche Fariza d'avoir accepter d'examiner notre travail sans oublier tous ceux qui ont contribuer à notre formation.

Enfin, nous vous remercions vous lecteurs et lectrices à qui ce modeste travail sera bénéfique.

Dédicaces

Je dédie ce modeste travail

A ma très chère mère pour son aide, son soutien et son encouragement tout au long de mes études et qui a fait de moi ce que je suis aujourd'hui et à mon cher père pour qui je souhaite la miséricorde, la paix et que Dieu l'accueille dans son vaste paradis ;

A mon cher mari qui ma toujours soutenu je profite de le remercier pour son encouragement, son aide et son soutien sans oublier sa famille ;

A mes chers frères Amar et Khelifa, leurs femmes et leurs enfants ;

A mes chères sœurs Fatiha, Fariza et Saida, leurs maries et leurs enfants ;

A toute la famille ;

A ma promotrice M^{me} Kheffache Rezika ;

A ma très chère amie et mon binôme Cilia et sa famille ;

A tous mes amis (es) ;

A toute la promotion 2018/2019 ;

A ceux que j'aime et qui m'aiment.

Safia

Dédicaces

Je dédie ce modeste travail à mes chers parents avec tous mes sentiments de respect , d'amour, de gratitude et de reconnaissance pour tous les sacrifices déployés pour m'élever dignement et assurer mon éducation dans les meilleures conditions,

A mes chers frères et sœurs

A mon cher mari et mes beaux parents

A Lounes et sa femme et à Sarah

A mes chères grand mères

A mes tantes, oncles , cousins et cousines

A ma promotrice M^{me} Kheffache Rezika

A ma chère binôme Safia

Et à tous ceux qui ont contribué de prés ou de loin à la réussite de ce modeste travail.

Cilia

Table des matières

Introduction générale

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire[14, 16].

La plupart des problèmes d'optimisation sont des problèmes NP-difficiles et donc ne possèdent pas d'algorithme, permettant de les résoudre en un temps polynomial. Étant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle et en intelligence artificielle[23].

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche.

Comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de tailles importantes.
Depuis plusieurs années, des progrès important ont été réalisés avec l'apparition des méthodes approchées puissantes et générales, souvent appelées métaheuristiques[24].

Une métaheuristique est constituée d'un ensemble de concepts fondamentaux qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation[6]. Ainsi les métaheuristiques sont adaptables et applicables à une large classe de problèmes.

Dans notre travail, nous nous sommes intéressées à la méthode dite scatter search qui est une méthode d'évolution proposée par Glover en 1977 basée sur une population de solutions. En accord avec ce que nous venons de présenter, ce document est organisé comme suit :

Le premier chapitre introduit l'optimisation combinatoire et les méthodes de résolution utilisées, tout en présentant quelques définitions et concepts de base nécessaire pour la comprehension de notre travail ainsi que la notion de complexité et les classes de problèmes.

Dans le deuxième chapitre, nous présentons le problème du sac à dos simple KP à variables binaires, ses applications, ses variantes, sa modélisation et quelques méthodes de résolution de ce problème.

Le troisième chapitre, concerne l'étude de la méthode de scatter search, là on parle du concept fondamental, le principe général de cette méthode, sa procédure, son domaine d'application et l'algorithme général de cette méthode.

Le quatrième chapitre, introduit un exemple d'application de la méthode de scatter search pour le problème du sac à dos suivant l'algorithme donné dans le troisième chapitre.

Dans le dernier chapitre, nous avons implémenté l'instance du problème du sac à dos que nous avons étudié en utilisant la programmation dynamique avec le langage Python sous windows.

Nous terminons le mémoire par une conclusion générale et une bibliographie.

Chapitre 1

Optimisation combinatoire et métaheuristiques

1.1 Introduction

Le but de ce chapitre est de situer le cadre théorique dans lequel s'inscrit notre travail de recherche. A ce titre, nous présentons la complexité algorithmique, les notions de base de l'optimisation combinatoire, ainsi que les différentes méthodes de résolution des problèmes d'optimisation, en particulier les méthodes approchées.

1.2 Optimisation combinatoire

L'optimisation combinatoire est une branche récente des mathématiques qui s'est développée en réponse au grand nombre de problèmes pratiques qu'elle peut modéliser et qu'elle se propose de résoudre.

Définition 1.2.1. Une instance d'un problème d'optimisation combinatoire discret est définie par la donnée d'un ensemble fini S et d'une application $f : S \rightarrow R$. Il s'agit de déterminer un élément s^* dans S tel que $f(s^*) \leq f(s)$, pour tout élément s de S (problème de minimisation).

Définition 1.2.2. L'ensemble de toutes les instances définit le problème d'optimisation combinatoire.

Á tout problème d'optimisation combinatoire

$$\begin{cases} minf(x) \\ s \in S. \end{cases}$$

on peut associer un problème de reconnaissance (ou de décision) en introduisant un seuil α correspondant à la fonction objectif. Le problème de décision devient :
Existe t-il une solution réalisable $\hat{s} \in S$ tel que $f(\hat{s}) \leq \alpha$?

1.3 Complexité

Afin de mesurer la difficulté d'un problème donné et le comparer avec celles d'autres problèmes pour pouvoir dire qu'un tel problème est plus facile à résoudre que l'autre, une théorie de complexité a été développée et permet de classer les problèmes faciles et difficiles.

1.3.1 Concepts de base

Définition 1.3.1. A toute instance I d'un problème (P), nous pouvons associer un nombre $\mu(I)$ qui mesure la longueur des données de cette instance et que nous appelons la taille de l'instance I.

Algorithme

Un algorithme est une procédure de calcul bien définie qui prend en entrée une valeur, ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforment l'entrée en sortie.

Complexité d'un algorithme

Si A est un algorithme de résolution d'un problème (P) et I est une instance de ce dernier, alors au couple (A,I) nous associons un entier $\tau(A, I)$ représentant le nombre d'opération élémentaires (addition, multiplication, comparaison, affectation) effectuées par l'algorithme A dans la résolution de l'instance I du problème (P). Le plus grand

nombre $\tau(A, I)$ sur l'ensemble de toutes les instances ayant la même taille est appelé complexité de l'algorithme A.

Algorithme efficace

Un algorithme est efficace si sa complexité est majorée par un polynôme en la taille des données. La notion d'efficacité d'un algorithme est liée à la manière dont le nombre d'opérations augmente avec la taille de l'instance à laquelle il est appliqué.

Complexité d'un problème

La complexité d'un problème (P) est la complexité du meilleur algorithme qui résout A.

1.3.2 Les classes de problèmes

La complexité des algorithmes a abouti à une classification des problèmes en fonction des performances des meilleurs algorithmes connus qui les résolvent.

1. La classe des problèmes $\mathcal{P}$

Un problème est dit facile s'il existe un algorithme polynomial pour le résoudre. L'ensemble de ces problèmes constitue la classe des problèmes polynômiaux notée $\mathcal{P}$.

2. La classe des problèmes NP et les algorithmes non déterministes

Définition 1.3.2. Un problème de reconnaissance (ou de décision) est un problème dont la réponse est soit "oui" ou "non".

Á tout problème d'optimisation combinatoire

$$\begin{cases} minf(x) \\ s \in S. \end{cases}$$

on peut associer un problème de reconnaissance (ou de décision) en introduisant un seuil α correspondant à la fonction objectif. Le problème de décision devient :
Existe t-il une solution réalisable $\hat{s} \in S$ tel que $f(\hat{s}) \leq \alpha$?

Remarque 1. Le problème de reconnaissance associé au problème d'optimisation combinatoire est au moins aussi facile que lui. En d'autres termes, si le problème de reconnaissance est difficile alors le problème d'optimisation combinatoire l'est aussi. Les problèmes de reconnaissance sont d'une très grande importance théorique. C'est grace à ces problèmes que la théorie de la complexité a trouvé son fondement.

Définition 1.3.3. Les algorithmes non déterministes sont des algorithmes contenant une instruction "choix", opérant sur un ensemble fini, choisit un élément, sans que la manière dont le choix est effectué ne soit précisée. Ils sont caractérisés par le fait que s'il existe une manière de procéder qui conduit au résultat, alors c'est selon cette manière qu'ils procèdent.

Définition 1.3.4. Un problème de reconnaissance appartient à la classe NP (Non deterministic polynomial) s'il existe un algorithme non déterministe polynomial pour le résoudre.

Remarque 2. Un problème de reconnaissance est dans la classe NP si pour toute instance de ce problème, nous pouvons vérifier en un temps polynomial par rapport à la taille de l'instance qu'une solution proposée permet d'affirmer que la réponse est "oui" pour cette instance.

3. La classe des problèmes NP-complet

Définition 1.3.5. Un problème (P_1) se réduit pôlynomialement à un problème (P_2) si et seulement s'il existe un algorithme polynomial de résolution de (P_1) faisant appel à la résolution de (P_2) comme procédure.

Définition 1.3.6. Un problème (P) est dit NP-complet s'il est dans NP et si tout problème de la classe NP se réduit pôlynomialement à (P).

1.4 Méthodes de résolution

Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe NP-difficiles[17]. Étant donné l'importance de ces problèmes, de nombreuses méthodes ont été développées en recherche opérationnelle. Elles sont classées en deux classes :

1.4.1 Méthodes exactes

Ces méthodes donnent une garantie de trouver la solution optimale pour des problèmes de tailles raisonnables. Parmi ces méthodes, on trouve : les méthodes de séparation et évaluation (Branch and Bound), la programmation linéaire pour l'optimisation continue, la programmation dynamique.

Comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes ne sont pas applicables pour les problèmes de grande taille.

1.4.2 Méthodes approchées

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grandes tailles. Parmi ces méthodes, on trouve les heuristiques et les métaheuristiques.

1. Heuristiques

Généralement une heuristique est conçue pour un problème particulier en s'appuyant sur sa structure propre. Lorsque les approches contiennent des principes plus généraux, on parle de métaheuristiques.

Il existe deux classes d'heuristiques : les algorithmes gloutons et les algorithmes par exploration locale.

a) Algorithmes gloutons

Les algorithmes gloutons sont caractérisés par l'examen des données dans un ordre prédéfini et par l'impossibilité de leur réexamen. Intuitivement, cela signifie que l'erreur n'est pas admise dans ce genre d'algorithmes car nous ne pouvons pas nous corriger en revenant en arrière "pas de retour en arrière car ce qui est fait est fait".

De plus, ils ne peuvent être appliqués qu'aux problèmes dont la fonction objectif est obtenue par la somme des "parts apportées" par les éléments de la solution. Ces problèmes sont dits à fonction objectif séparée.

b) Algorithmes par exploration locale

A partir d'une solution réalisable d'un problème, les algorithmes par exploration locale définissent un voisinage dans lequel ils cherchent une solution meilleure que celle dont ils disposent.

2. Les métaheuristiques

Le terme métaheuristique vient des mots grecs meta (au delà) et heuristique (trouver). Une métaheuristique est une heuristique générique qu'on peut adapter à tout problème. Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des points (presque) optimaux. Les métaheuristiques sont en général non déterministes et ne donnent aucune garantie d'optimalité[15, 18].

Classification des métaheuristiques

Il existe plusieurs façons de classer les métaheuristiques, on en donne quelques une, et nous adopterons celle faisant la différence entre les méthodes de trajectoire et les méthodes basées sur une population[5, 13].

a) Méthodes de trajectoire

Elles manipulent un seul point à la fois et tentent itérativement d'améliorer ce point. Elles construisent une trajectoire dans l'espace des points en tentant de se diriger vers des solutions. Parmi ces méthodes :

– **Le recuit simulé**

Le recuit simulé est une méthode de programmation empirique inspirée d'un processus utilisé en métallurgie. On atterne dans cette dernière des cycles de refroidissement lent et de réchauffage (recuit) qui ont pour effet de minimiser l'énergie du matériau. Cette méthode est transposée en optimisation pour trouver les extrema d'une fonction.

– **La recherche tabou**

La recherche tabou est une métaheuristique d'optimisation présentée par Fred Glover en 1986[20]. Le principe de cette méthode est : à chaque itération le voisinage de la solution courante est examiné et la solution minimisant le coût est sélectionnée. Pour éviter le phénomène de cyclage, la méthode interdit de revisiter une solution déjà visitée. Pour cela, une liste taboue contenant les solutions visitées est utilisée.

b) Approches "population"

Elles consistent à travailler avec un ensemble de solution simultanément, que l'on fait évoluer graduellement. L'utilisation de plusieurs solutions simultanément permet naturellement d'améliorer l'exploration de l'espace de recherche. Parmi ces méthodes, on trouve :

– **Algorithmes génétiques**

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnaires. Inspirés de la théorie de Darwin du XIX^e siècle[1]. Dans cette théorie, une population d'individus évolue grâce au mécanisme de la reproduction sexuée[19, 21].

– **Algorithmes de colonies de fourmis**

Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis à la recherche de nourriture. Initialement proposé par Marco Dorigo[22] dans les années 1990, pour la recherche de chemins optimaux dans un graphe. Il est connu que les fourmis sont capables de déterminer le chemin le plus court entre leur nid et une source de nourriture. Ceci est possible grâce à la phéromone qui est une substance que les fourmis déposent sur le sol lorsqu'elles se déplacent. Lorsqu'une fourmi doit choisir entre deux directions, elle choisit avec une plus grande probabilité celle comportant une plus forte concentration de phéromone. L'idée originale s'est diversifiée pour résoudre une

classe plus large de problèmes.

– **Méthode à mémoire adaptative**

La recherche à mémoire adaptative est une méthode proposée par Rochat et Taillard en 1995. C'est une extension de la Recherche Tabou qui permet de réaliser automatiquement une diversification et une intensification de la recherche.

1.5 Conclusion

Nous avons présenté dans ce chapitre les principaux concepts de l'optimisation combinatoire indispensables à la compréhension de notre travail. Ces concepts sont centrées sur la complexité algorithmique, les différentes méthodes de résolution des problèmes d'optimisation, en particulier, les méthodes approchées.

Chapitre 2

Le problème du sac à dos

2.1 Introduction

Le problème du sac à dos noté KP(en anglais Knapsack problem) est l'un des 21 problèmes NP-complets de Richard Karp, exposé dans son article de 1972. Il est intensivement étudié depuis le milieu du xx^{e} siècle. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou une partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans les sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

2.2 Les applications du problème du sac à dos

Nous pouvons retrouver le problème du sac à dos dans de nombreux domaines :

1. Dans les systèmes financiers, où l'idée est la suivante : étant donné un certain montant d'investissement dans des projets, quel projet choisir pour que tout rapporte le plus d'argent possible ;
2. Pour découper des matériaux, afin de minimiser les pertes dues aux chutes ;
3. Dans le chargement de cargaisons (avions, camions, bateaux...) ;
4. Ou encore, dés qu'il s'agit de préparer une valise ou un sac à dos pour une randonnée.

Une autre raison de s'intéresser à ce problème est son apparition dans certaines utilisations de méthodes de génération de colonnes (ainsi pour le problème de "bin packing" c'est à dire le problème algorithmique où il s'agit de ranger des objets avec un nombre minimum de boîtes).

2.3 Les variantes du problème du sac à dos

Il existe plusieurs variantes du sac à dos. Les particularités de ces variantes se font sur le domaine des variables, le nombre de valeurs des objets, les dimensions du sac, ... etc. Ces particularités peuvent aussi êtres combinées.

2.3.1 Variables continues

Le problème du sac à dos en variables continues (LKP) est obtenu en enlevant la contrainte d'intégrité sur les variables. C'est à dire que l'on s'autorise à ne prendre qu'une fonction des objets dans le sac à dos : $x_i \in [0, 1]$. LKP appartient à la classe de complexité $\mathcal{P}$.

2.3.2 Variables entières

Dans le problème du sac à dos en variables entières, on considère que l'on a plusieurs exemplaires de chaque objet. Le problème consiste donc à trouver le nombre d'exemplaires à prendre de chaque objet. Si le nombre d'exemplaires est limité, on parlera du sac à dos borné (BKP), sinon on parlera du sac à dos non borné (UKP).

2.3.3 Sac à dos multidimensionnel

On considère ici que le sac à dos a d dimensions, avec $d > 0$ (d-KP). Par exemple, on peut imaginer une boîte. Chaque objet a trois dimensions, et il ne faut déborder sur chacune des dimensions. En pratique la version multidimensionnelle peut servir à modéliser et résoudre le problème du remplissage d'un conteneur dont le volume et la charge maximale sont limitées.

2.3.4 Sac à dos multi-objectif

Cette variante consiste, à partir d'objets ayant plusieurs valeurs, à maximiser plusieurs fonctions objectifs, c'est le problème du sac à dos multi-objectif (MOKP). On rentre donc dans le domaine de l'optimisation multi-objectif.

2.3.5 Sac à dos quadratique

Le problème du sac à dos quadratique est noté QKP. On a ici un gain $g_{i,j}$ supplémentaire lorsque deux objets (i et j) sont pris simultanément.

2.3.6 Sac à dos à choix multiple

Dans le problème du sac à dos à choix multiple (MCKP), les objets sont regroupés en classes, et il ne faut prendre qu'un seul représentant pour chaque classe. Par exemple, on est en train de confectionner une boîte à outils. Si on a cinq clés à molette, on peut soit choisir la plus légère, afin de prendre un marteau performant, ou alors choisir la clé la plus performante et un marteau bas de gamme, ou alors faire un compromis. L'idée générale est qu'on ne peut pas prendre plus d'une clé, ni plus d'un marteau.

2.3.7 Sac à dos multiple

Le problème du sac à dos multiple (MKP) consiste à répartir un ensemble d'objets dans plusieurs sacs à dos de capacités différentes. La valeur d'un objet dépend maintenant du sac dans lequel il est placé. Il existe une variante de ce problème où tous les sacs ont la même capacité.

2.4 Modélisation du problème KP

Dans cette section, nous nous contentons de donner uniquement la modélisation du problème du sac à dos à variables binaires, le problème sur lequel nous nous sommes intéressées.

Étant donnés n objets numérotées par l'indice i variant de 1 à n. Les nombres w_i et p_i représentent respectivement le la valeur et le poids de l'objet numéro i. La capacité du sac sera notée W. Il faut indiquer pour chaque élément s'il est pris ou non. On peut utiliser un codage binaire : l'état du i-ème élément vaudra $x_i = 1$ si l'élément est mis dans le sac, ou $x_i = 0$ s'il est laissé de côté. Une façon de remplir le sac est donc complètement décrite par un vecteur $X = (x_1, x_2, ..., x_n)$; tels que :

1. la somme des valeurs associées aux objets soit maximale.
2. la somme des poids associés aux objets choisis ne dépasse pas la capacité du sac.

Ainsi le problème du sac à dos KP sera modéliser comme suit :

$$\begin{cases} maxZ(x) = \sum_{i=1}^{n} w_i x_i \\ \sum_{i=1}^{n} p_i x_i \leq W \\ x_i \in \{0,1\} \text{ ,i=1,2,...,n} \end{cases}$$

Sans perte de généralité, les coefficients w_i, p_i et W sont considérés comme des entiers positifs et $\sum_{i=1}^{n} p_i > W$.

2.5 Méthodes de résolution

2.5.1 Résolution exacte

Le problème du sac à dos, dans sa version classique, a été étudié en profondeur[25]. Il existe donc de nombreuses méthodes pour le résoudre. La plupart de ces méthodes correspondent à une version améliorée d'une des méthodes suivantes :

a) Programmation dynamique

Le problème du sac à dos possède la propriété de sous-structure optimale, c'est-à-dire que l'on peut construire la solution optimale du problème à i variables à partir du problème à $i - 1$ variables.

Cette propriété permet d'utiliser la programmation dynamique comme méthode de résolution [9, 10].

b) Procédure de séparation et évaluation

Le problème du sac à dos peut être aussi résolu à l'aide d'une procédure de séparation et évaluation (PSE). La fonction d'évaluation du nœud consiste souvent à résoudre le problème en variables continues. L'implementation proposée par Martello et Toth (1990)[11] est devenue une référence.

c) Approches hybrides

L'approche hybride n'est pas réellement une nouvelle méthode de résolution. Elle consiste simplement à combiner les deux méthodes précédentes, ainsi on applique une procédure de séparation et évaluation (PSE) jusqu'à une profondeur de recherche où le sous-problème devient assez petit pour pouvoir être résolu par la programmation dynamique. Les précurseurs de cette approche sont Plateau et Elkihel (1985)[12]. Il ya eu d'autres améliorations depuis. Les approches hybrides restent meilleures que les méthodes précédentes.

2.5.2 Résolution approchée

Comme pour la plupart des problèmes NP-complets[8], il peut être intéressant de trouver des solutions réalisables mais non optimales. De préférence avec une garantie sur l'écart entre la valeur de la solution trouvée et la valeur de la solution optimale.

On appelle efficacité d'un objet le rapport de sa valeur sur son poids. Plus la valeur de l'objet est importante par rapport à ce qu'il consomme, plus l'objet est efficace.

a) Algorithme glouton

L'algorithme le plus simple est l'algorithme glouton. L'idée est d'ajouter en priorité les objets les plus efficaces, jusqu'à saturation du sac.

b) Métaheuristiques

Les méthodes métaheuristiques comme les algorithmes génétiques ou les optimisations basées sur des algorithmes de colonies de fourmis, permettent d'obtenir une approximation raisonnable tout en évitant de monopoliser trop de ressources.

Les algorithmes génétiques sont souvent utilisés pour la résolution du problème du sac à dos. Ils sont relativement faciles à mettre en œuvre et permettent d'obtenir rapidement une solution satisfaisante même si la taille du problème est importante.

2.6 Conclusion

Dans ce chapitre, nous avons évoqué le problème du sac à dos ainsi que ses variantes, nous avons aussi présenté les différentes méthodes de résolution de ce problème.

Chapitre 3

Scatter search

3.1 Introduction

La recherche dispersée ou "scatter search" en anglais, est une méthode d'évolution qui a été proposée par Glover en 1977[27]. Le concept fondamental et le principe général de cette méthode ont été proposés en premier lieu dans les années 70 comme une heuristique basée sur une formulation donnée dans les années 60 pour les règles de décision de combinaison et les contraintes du problème[2].

3.2 Principes de scatter search

Scatter search ou la recherche dispersée du point de vue classification des métaheuristiques est une méthode d'évolution qui est basée sur une population de solutions. Elle est fondée sur trois principes :

1. Les informations pertinentes de l'emplacement de la solution optimale sont généralement contenues dans un ensemble diversifié de l'ensemble de solutions « élite »(générées).
2. Grâce à une bonne combinaison de solutions, il est possible de réaliser d'autres régions intéressantes qui pourraient contenir la meilleure solution ou les solutions qui peuvent conduire à la solution optimale.
3. Tenir compte de multiples solutions comme fondation d'en créer de nouvelles, Scatter

search favorise l'exploration de l'information ne figurant pas dans chaque solution individuellement.

3.3 Procédure de scatter search

Scatter search comprend cinq étapes :

1. Une méthode de génération de diversification pour produire une collection de solutions diverses, en utilisant une solution d'essai arbitraire comme entrée.
2. Une méthode d'amélioration pour transformer une solution initiale en une ou plusieurs solutions d'essai améliorées.
3. Une méthode de mise à jour de l'ensemble de référence pour construire et maintenir un ensemble de référence comprenant les meilleures solutions trouvées. L'incorporation de solutions à l'ensemble de référence est effectuée selon leur qualité ou leur diversité.
4. Une méthode de génération d'un sous-ensemble pour opérer sur l'ensemble de référence pour produire un sous ensemble de ces solutions comme base pour créer des solutions combinées.
5. Une méthode de combinaison de solutions pour transformer un sous ensemble donné de solutions produites par la méthode de génération d'un sous ensemble en un ou plusieurs vecteurs combinés de solutions[2].

a) Population initiale

La population initiale est générée en utilisant des stratégies de diversification et d'intensification.

b) Diversification (ou exploration)

C'est une stratégie qui sert à conduire la recherche vers des nouvelles régions afin d'avoir un échantillon représentatif de l'espace de recherche dans le but d'atteindre l'optimum global c'est-à-dire stratégie pour l'algorithme en entier, dont l'objectif est d'atteindre la meilleure solution.

c) Intensification (ou exploitation)

Elle vise à utiliser l'information déjà récoltée pour définir et parcourir les zones intéressantes de l'espace de recherche pour atteindre la meilleure solution.

Le générateur de diversification est utilisé pour commencer à générer l'ensemble P de solutions initiales. Le cardinal de l'ensemble P est généralement égal au max(100, 5) où b est le cardinal de l'ensemble de référence.

Des stratégies de diversification sont utilisées pour avoir une population variée qui permettra d'explorer des régions diverses de l'espace de recherche.

Pour avoir une population de solutions qui représente des zones diverses de l'espace de recherche, on utilise une méthode de diversification.

3.3.1 Méthode de génération de diversification

Il existe plusieurs types de diversifications, nous indiquons uniquement celle de la méthode appliquée aux problèmes d'optimisation en 0-1.

Génération de vecteurs en 0-1

Soit x un n-vecteurs de composantes x_i, $i = 1, 2, ..., n$ de valeur 0 ou 1 où n est le nombre de variables du problème considéré. Soit h un paramètre tel que $h \leq n - 1$.

Soit $x = (0, 0, 0, ..., 0)$ la solution initiale et pour chaque valeur de $h = 1, 2, ..., n - 1$; on génère deux types de solutions $x^{'}$ et $x^{''}$ de la manière suivante :

Pour type 1 :

$x^{'}_1$ = 1 - x_1

$x^{'}_{1+hk}$ = 1 - x_{1+hk} pour k = 1,2,...,$\lfloor \frac{n}{h} \rfloor$, où $\lfloor \frac{n}{h} \rfloor$ est la partie entière.

Pour type 2 :

$x^{''}$ est le complémentaire de $x^{'}$. ($x^{''}$=1-$x^{'}$).

3.3.2 Ensemble de solution de référence

L'ensemble de référence Refset, est une collection de solutions de qualité et de solutions diverses qui sont employées pour produire de nouvelles solutions par l'application de la méthode de combinaison de solutions. L'ensemble de référence comprend l'union de deux sous-ensembles, $Refset_1$ et $Refset_2$, de taille b_1 et b_2 respectivement. C'est-à-dire $|Refset| = b = b_1 + b_2$.

Définition 3.3.1. $Refset_1$ c'est les solutions élites qui comporte b_1 solutions. Elles constituent les meilleures solutions de la population selon la valeur de la fonction objectif.

Définition 3.3.2. $Refset_2$ contient b_2 solutions prises de la population restante $P - b_1$. Elles sont les plus éloignées des meilleures solutions de b qui représentent des solutions diversifiées.

La construction d'un ensemble de référence initial commence par le choix des meilleures solutions b_1 à partir de P. Ces solutions sont ajoutées à Refset et supprimées de P.

Pour chaque solution x améliorée dans P-Refset, le minimum des distances aux solutions y dans Refset est calculé comme suit :

$$d_{min}(\mathrm{x}) = min_{y \in Refset} d(x, y)$$

On définit une distance entre deux solutions x et y de la manière suivante :

$$d(x, y) = \sum_i |x_i - y_i|$$

Puis, la solution avec le maximum de ces distances minimales est choisie. Cette solution est ajoutée à Refset et supprimée de P. On répète le processus jusqu'à avoir b_2 solutions qui formeront l'ensemble $Refset_2$.

L'ensemble de référence résultant a les solutions b_1 de haute qualité et les solutions b_2 diverses.

Le nombre de solutions contenues dans l'ensemble de référence doit être ≤ 20.

3.3.3 Méthode de génération de sous-ensembles

Cette méthode consiste à générer les sous-ensembles qui seront utilisés pour créer la structure des combinaisons de l'étape suivante en utilisant les éléments de Refset construits dans l'étape précédente.

On distingue quatre types de sous-ensembles :

Type 1 : Combiner les solutions de l'ensemble de référence Refset deux à deux.

Type 2 : Combiner les solutions de Refset trois à trois en prenant chaque sous-ensemble de type1 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type 3 : Combiner les solutions de Refset quatre à quatre en prenant chaque combinaison de type2 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type 4 : Combiner les i meilleures solutions de Refset au sens de la fonction objectif avec i variant de 5 à b.

Noter que le nombre de sous-ensembles de type1 produits égal à $\frac{b^2-b}{2}$.

Si l'ensemble de référence contient b solutions ,la procédure examine approximativement $\frac{(3b-7)*b}{2}$ combinaisons de quatre type cités précédemment.

3.3.4 Méthode de combinaison des solutions

Cette méthode emploie les sous-ensembles produits avec la méthode de génération pour combiner les éléments dans chaque sous-ensemble en vue de créer de nouvelles solutions. Généralement, la méthode de combinaison de solutions est un mécanisme spécifique du problème, puisqu'on le lie directement à la représentation de solutions. Selon la forme spécifique de la méthode de combinaison de solutions, chaque sous-ensemble peut créer une ou plusieurs nouvelles solutions.

Considérons la méthode suivante de combinaison pour les solutions en 0-1 :

On définit un score pour chaque variable en se basant sur les solutions du sous-ensemble et la valeur de la fonction objectif correspondante comme suit : Score de la variable i qui correspond à la solution d'un sous-ensemble E est :

$$Score(i) = \frac{\sum_{x \in E} Z(x) * x_i}{\sum_{x \in E} Z(x)}$$

$Z(x)$: valeur de la fonction objectif donnée par la solution x.

x_i : la valeur de la $i^{ème}$ variable dans la solution x.

La solution sera construite de la manière suivante :

$$x_i^{'} = \begin{cases} 1 & si\ le\ score(i) > 0,5 \\ 0 & si\ le\ score(i) \leq 0,5. \end{cases}$$

Une fois les solutions sont calculées, on peut obtenir celles qui ne sont pas réalisables, à l'aide d'une méthode d'amélioration appliquée à chacune des solutions, on obtient une ou plusieurs solutions de chaque type.

Si une solution x n'est pas dans Refset et que la valeur de la fonction objectif de x est meilleure que la valeur de la fonction objectif du plus mauvais élément de $Refset_1$ alors ajouter x à $Refset_1$ et supprimer le mauvais élément en cours dans $Refset_1$ (le mauvais élément est celui qui a la mauvaise valeur de la fonction objectif).

Si x n'est pas dans $Refset_2$ et $d_{min}(x)$ est supprimer à $d_{min}(y)$ par une solution dans $Refset_2$ alors :

Ajouter à $Refset_2$ et supprimer le mauvais élément en cours dans $Refset_2$ (le mauvais élément est la solution y avec la valeur la plus petite de $d_{min}(y)$).

3.4 Algorithme de scatter search

Pour illustrer les différentes phases de la méthode de scatter search[3, 4], on définit les paramètres suivants :

Psize : la taille de l'ensemble de solutions générées par la méthode de génération de

diversification.

b **:** cardinal de l'ensemble de référence.

b_1 **:** cardinal du sous ensemble de référence contenant les meilleures solutions au sens de la fonction objectif.

b_2 **:** cardinal du sous ensemble de référence contenant les solutions diverses.

Maxlter : nombre maximum d'iterations.

Refset : ensemble de référence.

$Refset_1$ **:** sous ensemble contenant les meilleures solutions.

$Refset_2$ **:** sous ensemble contenant les solutions diverses.

Algorithm 1 Scatter Search

1. Commencer par $P = \emptyset$. Utiliser une méthode de génération de diversification pour construire une solution x. Appliquer une méthode d'amélioration à x pour obtenir une solution améliorée x^*.
   Si $x^* \notin P$, $P = P \bigcup x^*$, sinon omettre x^*. Répéter jusqu'à $|P| = PSize$.

2. Ordonner les solutions de P par rapport à la fonction objectif de la meilleure à la mauvaise solution.

   Pour(iter=1 **à** Maxlter)

3. Etablir $Refset = Refset_1 \bigcup Refset_2$ de P avec $|Refset| = b$, $|Refset_1| = b_1$, $|Refset_2| = b_2$.

   Prendre les premières solutions b_1 dans P et les ajouter à $Refset_1$. Pour chaque solution dans $P - Refset$ et $y \in Refset$, calculer une mesure de distance de dissimilitude $d(x, y)$
   solution dans P- Refset et y Refset, calculer une mesure de distance de dissimilitude .
   Choisir la solution $x^{'}$ qui maximise $d_{min}(x) = \min\limits_{y \in refset} \{d(x, y)\}$, ajouter $x^{'}$ à $Refset_2$, jusqu'à $Refset_2 = b_2$.
   Faire Nouveaux Eléments = VRAI.

   Tant que (Nouveaux Eléments) **Faire**

4. Calculer le nombre de sous ensembles (MaxSubset) qui incluent au moins un nouvel élément.
   Faire Nouveaux Eléments = FAUX.
   Pour(Subset =1,...., MaxSubset) **Faire**

5. Générer le sous ensemble s suivant avec la méthode de génération de sous ensemble. Cette méthode génère l'un des quatre types des sous ensembles avec le nombre d'éléments rangés de 2 à $Refset$. Poser le sous ensemble $s = \{s_1, s_2, ...s_k\}$ pour $2 \leq k \leq |Refset|$.
   (Nous considérons que la méthode de génération de sous ensembles saute les sous ensembles pour lesquels les éléments considérés n'ont pas changé dans les itérations précédentes).

6. Appliquer la méthode de combinaison des solutions à s pour obtenir une ou plusieurs nouvelles solutions x_s.

7. Appliquer la méthode d'amélioration à x_s pour obtenir une solution améliorée x_s^* ;
   Si(x^* n'est pas dans $Refset$ et que la valeur de la fonction objectif de x_s^* est meilleure que la valeur de la fonction objectif du plus mauvais élément de $Refset_1$) **alors**

8. Ajouter x_s^* à $Refset_1$ et supprimer le mauvais élément en cours dans $Refset_1$ (le mauvais élément est celui qui a la mauvaise valeur de la fonction objectif).

9. **Faire** Nouveaux Eléments = VRAI.

Si (x_s^* n'est pas dans $Refset_2$ et $d_{min}(x_s^*)$ est supérieur à $d_{min}(x)$ pour une solution x dans $Refset_2$)**alors**

10. Ajouter x_s^* à $Refset_2$ et supprimer le mauvais élément en cours dans $Refset_2$.
    (le mauvais élément est la solution s avec la valeur la plus petite de $d_{min}(x)$.

11. Faire Nouveaux Eléments = VRAI.
    Si($iter < Maxlter$) **alors**

12. Construire un nouvel ensemble P en utilisant la méthode de génération de diversification. Initialiser le processus de génération avec les solutions en cours de $Refset_1$. Les premières b_1 solutions dans le nouvel ensemble P sont les meilleurs b_1 solutions en cours dans $Refset$.

13. Si x_s^* n'est pas dans Refset et que la valeur de la fonction objectif de x_s^* n'est pas meilleure que la valeur de la fonction objectif du plus mauvais élément de Refset1 et aussi d_{min} est inférieure à $d_{min}(x)$ pour une solution x dans Refset2 alors :
    On n'ajoute pas x_s^* à Refset1 et on ne l'ajoute pas aussi à Refset2. Stop La solution du problème sera la solution ayant la meilleure valeur de la fonction objectif de Refset1

3.5 Domaines d'application

Les applications de la recherche dispersée sont nombreuses, Dont on cite :

Description	Références
Méthodologie	Glover (1998), Binato et al.(2001), Glover(1994a-1999), Glover, Laguna and Marti(2003a-b, 2004a-b), Greistorfer(2004), Greistorfer and Voss(2004), Laguna(2002), Laguna and Marti(2003), Marti et al.(2004), Reeves and Yamada(1999), Resende and Ribeiro(2002, 2003b)
Affectation	Alfandari et al.(2001), Alfandari et al.(2004), Cung et al.(1997), Marti et al.(2OOO), Oliveira et al.(2003), Yagiura et al.(2002), Yagiura et al.(2004)
Coloration	Hamiez and Hao(2002)
Problèmes de graphes	Laguna and Marti(1999) Alvarez et al.(2001), Bastos et al.(2001), Cavique et al.(2001), Dell'Amico et al.(2004), Piñana et al.(2004), Souza et al.(2003), Xu et al.(2000), Festa et al.(2000), Ribeiro and Rosseti(2002), Ribeiro et al.(2002), Zhang and Lai(2004)
Knapsack	Da Silva et al.(2004), Diaz et al.(2004)
Programmation en nombres entiers	Glover et al.(2000)
Multi Objective	Beausoleil(2001), Beausoleil(2004)
Problèmes de Permutation	Campos et al.(2003), Marti, Laguna and Campos(2004),
Problèmes de cheminement	Chu et al.(2004), Corberan et al.(2002), Greistorfer(1999), Greistorfer(2001), Rego(2000), Resende and Ribeiro(2003a), Chiang and Russell(2004)
Ordonnancement	Aiex et al.(2003), Debels et al.(2004), Nowicki and Smutnicki(2004a-b), Reeves and Yamada(1998), Yamashita et al.(2004)

TAB. 3.1 – Publications sur la méthode de scatter search (Source : Rafael Marti(2004))

3.6 Conclusion

La méthode de scatter search est une métaheuristique qui est basée sur l'évolution de la population. Durant la recherche, elle capture l'information dans les sous ensembles pour construire de nouvelles solutions, et elle conserve les deux grandeurs (diversité, qualité) de l'ensemble de référence. La méthode de scatter search est constituée essentiellement par des méthodes de base. Entre ces méthodes, les méthodes de combinaison, et les méthodes d'amélioration, qui sont liées aux problèmes à traiter.

Chapitre 4

Scatter search pour le problème du sac à dos

4.1 Introduction

Dans ce chapitre, nous allons illustré la méthode de scatter search pour le problème du sac à dos en résolvant une instance à huit variables.

4.2 Exemple d'application

$Z = 15x_1 + 100x_2 + 90x_3 + 60x_4 + 40x_5 + 15x_6 + 10x_7 + 20x_8 \rightarrow max$

$35x_1 + 28x_2 + 20x_3 + 24x_4 + 38x_5 + 25x_6 + 60x_7 + 16x_8 \leq 115$

$x_i \in \{0, 1\}$, $i = 1, ..., 8$

4.2.1 Étape1 : Génération d'une population diversifiante

Soit $x = (0, 0, 0, ..., 0)$ une solution initiale du problème.

Et soit h un paramètre tel que $0 < h \leq n - 1$, n est le nombre de variables.

On choisit $h = 1, 2, ..., 7$. Pour chaque valeur de h deux solutions sont générées, les solutions de type1 notées $x^{'}$ et les solutions de type2 notées $x^{''}$ telle que :

Type1 : $x^{'}_1 = 1 - x_1$

$x^{'}_{1+hk} = 1 - x_{1+hk}$, $k = 1, 2, ..., \lfloor \frac{n}{h} \rfloor$

Type2 : $x_i^{''} = 1 - x_i^{'}$ ($x^{''}$ est le complémentaire de $x^{'}$).

h	$x^{'}$	$x^{''}$
1	(1,1,1,1,1,1,1,1)	(0,0,0,0,0,0,0,0)
2	(1,0,1,0,1,0,1,0)	(0,1,0,1,0,1,0,1)
3	(1,0,0,1,0,0,1,0)	(0,1,1,0,1,1,0,1)
4	(1,0,0,0,1,0,0,0)	(0,1,1,1,0,1,1,1)
5	(1,0,0,0,0,1,0,0)	(0,1,1,1,1,0,1,1)
6	(1,0,0,0,0,0,1,0)	(0,1,1,1,1,1,0,1)
7	(1,0,0,0,1,0,0,1)	(0,1,1,1,1,1,1,0)

Les solutions générées sont utilisées pour créer l'ensemble de référence en utilisant la méthode d'amélioration. Si une solution générée n'est pas réalisable, la méthode d'amélioration va la rendre réalisable.

Le tableau suivant nous donne les solutions qui ne sont pas réalisables.

Solution	Solution d'essai	Valeur de Z	Poids total
1	(1,1,1,1,1,1,1,1)	246	$246 > 115$ (non réalisable)
2	(1,0,1,0,1,0,1,0)	155	$153 > 115$ (non réalisable)
3	(1,0,0,1,0,0,1,0)	85	$119 > 115$ (non réalisable)
4	(1,0,0,0,1,0,0,0)	55	$73 < 115$ (réalisable)
5	(1,0,0,0,0,1,0,0)	30	$60 < 115$ (réalisable)
6	(1,0,0,0,0,0,1,0)	25	$95 < 115$ (réalisable)
7	(1,0,0,0,0,0,0,1)	35	$51 < 115$ (réalisable)
8	(0,0,0,0,0,0,0,0)	0	$0 < 115$ (réalisable)
9	(0,1,0,1,0,1,0,1)	195	$93 < 115$ (réalisable)
10	(0,1,1,0,1,1,0,1)	265	$127 > 115$ (non réalisable)
11	(0,1,1,1,0,1,1,1)	295	$173 > 115$ (non réalisable)
12	(0,1,1,1,1,0,1,1)	320	$186 > 115$ (non réalisable)
13	(0,1,1,1,1,1,0,1)	325	$151 > 115$ (non réalisable)
14	(0,1,1,1,1,1,1,0)	315	$195 > 115$ (non réalisable)

4.2.2 Étape2 : Méthode d'amélioration

Dans la méthode d'amélioration, les solutions non réalisables deviennent réalisables. La procédure ce fait de la manière suivante :

On calcule pour chaque x_i, $i = 1, 2, .., 8$, la valeur $\frac{w_i}{p_i}$.

Pour chaque solution S_i, si :

- le poids total est supérieur à la capacité du sac, On change la valeur de x_i ayant la plus petite valeur de $\frac{w_i}{p_i}$ de 1 vers 0.
- le poids total est inférieur à la capacité du sac, On calcule la difference entre ces deux valeurs et La valeur dont le poids est inférieur à cette difference, on la change de 0 vers 1.

Solutions de type 1

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_1 = (1,1,1,1,1,1,1,1)$	350	$246 > 115$	Pour x_1 : 0,42 x_2 : 3,57 x_3 : 4,5 x_4 : 2,5 x_5 : 1,05 x_6 : 0,6 x_7 : 0,16 x_8 : 1,25 On change la valeur de x_i ayant la plus petite valeur de $\frac{w_i}{p_i}$ de 1 vers 0. $x_7 = 0$
$S_1 = (1,1,1,1,1,1,0,1)$	340	$186 > 115$	$x_1 = 0$
$S_1 = (0,1,1,1,1,1,0,1)$	325	$151 > 115$	$x_6 = 0$
$S_1 = (0,1,1,1,1,0,0,1)$	310	$126 > 115$	$x_5 = 0$
$S_1 = (0,1,1,1,0,0,0,1)$	270	$88 < 115$	On calcule : 115-88=27 La valeur dont le poids est ≤ 27, on la change de 0 vers 1 $x_6 = 1$
$S_1 = (0,1,1,1,0,1,0,1)$	285	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_2 = (1,0,1,0,1,0,1,0)$	155	$153 > 115$	$x_7 = 0$
$S_2 = (1,0,1,0,1,0,0,0)$	145	$93 < 115$	115-93=22 $x_8 = 1$
$S_2 = (1,0,1,0,1,0,0,1)$	165	$109 < 115$	115-109=6

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_3 = (1,0,0,1,0,0,1,0)$	85	$119 > 115$	$x_7 = 0$
$S_3 = (1,0,0,1,0,0,0,0)$	75	$59 < 115$	115-59=56 $x_5 = 1$
$S_3 = (1,0,0,1,1,0,0,0)$	115	$97 < 115$	115-97=18 $x_8 = 1$
$S_3 = (1,0,0,1,1,0,0,1)$	135	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_4 = (1,0,0,0,1,0,0,0)$	55	$73 < 115$	115-73=42 $x_2 = 1$
$S_4 = (1,1,0,0,1,0,0,0)$	155	$101 < 115$	115-101=14

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_5 = (1,0,0,0,0,1,0,0)$	30	$60 < 115$	115-60=55 $x_5 = 1$
$S_5 = (1,0,0,0,1,1,0,0)$	70	$98 < 115$	115-98=17 $x_8 = 1$
$S_5 = (1,0,0,0,1,1,0,1)$	90	$114 < 115$	115-114=1

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_6 = (1,0,0,0,0,0,1,0)$	25	$95 < 115$	115-95=20 $x_3 = 1$
$S_6 = (1,0,1,0,0,0,1,0)$	115	$115 = 115$	115-115=0

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_7 = (1,0,0,0,0,0,0,1)$	35	$51 < 115$	115-51=64 $x_7 = 1$
$S_7 = (1,0,0,0,0,0,1,1)$	45	$111 < 115$	115-111=4

Solutions de type2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_8 = (0,0,0,0,0,0,0,0)$	0	$0 < 115$	115-0=115 $x_7 = 1$
$S_8 = (0,0,0,0,0,0,1,0)$	10	$60 < 115$	115-60=55 $x_5 = 1$
$S_8 = (0,0,0,0,1,0,1,0)$	50	$98 < 115$	115-98=17 $x_8 = 1$
$S_8 = (0,0,0,0,1,0,1,1)$	70	$114 < 115$	115-114=1

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_9 = (0,1,0,1,0,1,0,1)$	195	$93 < 115$	115-93=22 $x_3 = 1$
$S_9 = (0,1,1,1,0,1,0,1)$	285	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_{10} = (0,1,1,0,1,1,0,1)$	265	$127 > 115$	$x_6 = 0$
$S_{10} = (0,1,1,0,1,0,0,1)$	250	$102 < 115$	115-102=13

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_{11} = (0,1,1,1,0,1,1,1)$	295	$173 > 115$	$x_7 = 0$
$S_{11} = (0,1,1,1,0,1,0,1)$	285	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_{12} = (0,1,1,1,1,0,1,1)$	320	$186 > 115$	$x_7 = 0$
$S_{12} = (0,1,1,1,1,0,0,1)$	310	$126 > 115$	$x_5 = 0$
$S_{12} = (0,1,1,1,0,0,0,1)$	270	$88 < 115$	115-88=27 $x_6 = 1$
$S_{12} = (0,1,1,1,0,1,0,1)$	285	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_{13} = (0,1,1,1,1,1,0,1)$	325	$151 > 115$	$x_6 = 0$
$S_{13} = (0,1,1,1,1,0,0,1)$	310	$126 > 115$	$x_5 = 0$
$S_{13} = (0,1,1,1,0,0,0,1)$	270	$88 < 115$	115-88=27 $x_6 = 1$
$S_{13} = (0,1,1,1,0,1,0,1)$	285	$113 < 115$	115-113=2

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S_{14} = (0,1,1,1,1,1,1,0)$	315	$195 > 115$	$x_7 = 0$
$S_{14} = (0,1,1,1,1,1,0,0)$	305	$135 > 115$	$x_6 = 0$
$S_{14} = (0,1,1,1,1,0,0,0)$	290	$110 < 115$	115-110=5

Après la méthode d'amélioration, on obtient les solutions suivantes :

Solutions	Z	Poids total
$S_{14} = (0,1,1,1,1,0,0,0)$	290	110
$S_1 = (0,1,1,1,0,1,0,1)$	285	113
$S_{10} = (0,1,1,0,1,0,0,1)$	250	102
$S_2 = (1,0,1,0,1,0,0,1)$	165	109
$S_4 = (1,1,0,0,1,0,0,0)$	155	101
$S_3 = (1,0,0,1,1,0,0,1)$	135	113
$S_6 = (1,0,1,0,0,0,1,0)$	115	115
$S_5 = (1,0,0,0,1,1,0,1)$	90	114
$S_8 = (0,0,0,0,1,0,1,1)$	70	114
$S_7 = (1,0,0,0,0,0,1,1)$	45	111

4.2.3 Étape 3 : Méthode de mise à jour de l'ensemble de référence

Cette méthode est utilisée pour créer et maintenir un ensemble de référence de solutions $Refset = Refset_1 \cup Refset_2$ tels que $|Refset| = b$; $|Refset_1| = b_1$ et $|Refset_2| = b_2$

$(b = b_1 + b_2)$.

Les b_1 solutions sont celles ayant la plus grande valeur de la fonction objectif. Dans cet example, on choisit b = 5 avec $b_1 = 3$ et $b_2 = 2$.

$b_1 = \{S_{14}, S_1, S_{10}\}$.

Et, on choisit les b_2 solutions parmi les solutions de l'ensemble P - $Refset_1$ de la manière suivante :

On définit une distance entre deux solutions x et y comme suit :

$d(x, y) = \sum_i |x_i - y_i|$ où $x = (x_i)_i$ et $y = (y_i)_i$

Solution candidate	Distance			Distance minimale
	S_{14}	S_1	S_{10}	
S_2	4	6	2	2
S_4	3	6	3	3
S_3	4	5	4	4
S_6	5	6	5	5
S_5	6	5	4	4
S_8	5	6	3	3
S_7	7	6	5	5

La solution qui maximise les distances minimales sera ajouté à l'ensemble de référence $Refset$. Dans cet exemple, on a :

$max \;\; d_{min} = max(2, 3, 4, 5) = 5$ qui correspond aux solutions S_6 et S_7.

Ainsi on obtient $b = \{S_{14}, S_1, S_{10}, S_6, S_7\}$ avec $b_1 = \{S_{14}, S_1, S_{10}\}$ et $b_2 = \{S_6, S_7\}$.

Après avoir construit l'ensemble de référence initial, cet ensemble sera modifié par la méthode de génération de sous ensembles.

4.2.4 Étape 4 : Méthode de génération de sous-ensembles

Cette méthode consiste à générer le sous-ensemble qui sera utilisé pour créer la structure des combinaisons dans l'étape suivante en utilisant les éléments de b construit dans l'étape précédente.

$b = \{S_{14}, S_1, S_{10}, S_6, S_7\}$.

Type1 : Construire les combinaisons à deux solutions de l'ensemble b.

Type2 : Combiner les solutions de Refset trois à trois en prenant chaque combinaison de type1 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type3 : Combiner les solutions de Refset quatre à quatre en prenant chaque combinaison de type2 et on lui ajoute la meilleure solution au sens de la fonction objectif qui n'est pas dans cet ensemble.

Type4 : Contient les sous-ensembles qui sont constitués des i meilleures solutions pour $i = 5$ jusqu'à b.

Type1	(S_{14}, S_1) ; (S_{14}, S_{10}) ; (S_{14}, S_6) ; (S_{14}, S_7) ; (S_1, S_{10}) ; (S_1, S_6) ; (S_1, S_7) ; (S_{10}, S_6) ; (S_{10}, S_7) ; (S_6, S_7)
Type2	(S_{14}, S_1, S_{10}) ; (S_{14}, S_6, S_1) ; (S_{14}, S_7, S_1) ; (S_{10}, S_6, S_{14}) ; (S_{10}, S_7, S_{14}) ; (S_6, S_7, S_{14})
Type3	$(S_{14}, S_1, S_{10}, S_6)$; $(S_{14}, S_7, S_1, S_{10})$; (S_6, S_7, S_{14}, S_1) ;
Type4	$(S_{14}, S_1, S_{10}, S_6, S_7)$

4.2.5 Étape 05 : Méthode de combinaison des solutions

Dans cette étape,on utilise les solutions combinées dans le but de créer d'autres nouvelles solutions. On définit un score pour chaque variable en se basant sur les solutions d'un sous- ensemble E et leurs valeurs objectifs correspondantes comme suit :

score de la variable i correspondant à la solution du sous ensemble est :

$$score(i) = \frac{\sum\limits_{x \in E} Z(x) * x_i}{\sum\limits_{x \in E} Z(x)}$$

$Z(x)$: la valeur de la fonction objectif de la solution x.

x_i : la valeur de la $i^{ème}$ variable dans la solution x.

La solution sera construite de la manière suivante :

$$x_i^{'} = \begin{cases} 1 & si\ le\ score(i) > 0.5 \\ 0 & si\ le\ score(i) \leq 0.5 \end{cases}$$

Type 01 :

$\underline{(S_{14}, S_1)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.50	0.50	0.50	0.50	0	0	0
S_1	0	0.49	0.49	0.49	0	0.49	0	0.49
Total	0	0.99	0.99	0.99	0.50	0.49	0	0.49

D'où, $S_1^{'} = (0, 1, 1, 1, 0, 0, 0, 0)$ $avec$ $Z = 250$

$\underline{(S_{14}, S_{10})}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.54	0.54	0.54	0.54	0	0	0
S_{10}	0	0.46	0.46	0	0.46	0	0	0.46
Total	0	1	1	0.54	1	0	0	0.46

D'où, $S_2^{'} = (0, 1, 1, 1, 1, 0, 0, 0)$ $avec$ $Z = 290$

$\underline{(S_{14}, S_6)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.72	0.72	0.72	0.72	0	0	0
S_6	0.28	0	0.28	0	0	0	0.28	0
Total	0.28	0.72	1	0.72	0.72	0	0.28	0

D'où, $S_3' = (0,1,1,1,1,0,0,0)$ *avec* $Z = 290$

$\underline{(S_{14}, S_7)}$

$S_{14} = (0,1,1,1,1,0,0,0) \Rightarrow Z = 290$

$S_7 = (1,0,0,0,0,0,1,1) \Rightarrow Z = 45$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.86	0.86	0.68	0.86	0	0	0
S_7	0.13	0	0	0	0	0	0.13	0.13
Total	0.13	0.86	0.86	0.86	0.86	0	0.13	0.13

D'où, $S_4' = (0,1,1,1,1,0,0,0)$ *avec* $Z = 290$

$\underline{(S_1, S_{10})}$

$S_1 = (0,1,1,1,0,1,0,1) \Rightarrow Z = 285$

$S_{10} = (0,1,1,0,1,0,0,1) \Rightarrow Z = 250$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_1	0	0.53	0.53	0.53	0	0.53	0	0.53
S_{10}	0	0.47	0.47	0	0.47	0	0	0.47
Total	0	1	1	0.53	0.47	0.53	0	1

D'où, $S_5' = (0,1,1,1,0,1,0,1)$ *avec* $Z = 285$

$\underline{(S_1, S_6)}$

$S_1 = (0,1,1,1,0,1,0,1) \Rightarrow Z = 285$

$S_6 = (1,0,1,0,0,0,1,0) \Rightarrow Z = 115$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_1	0	0.71	0.71	0.71	0	0.71	0	0.71
S_6	0.29	0	0.29	0	0	0	0.29	0
Total	0.29	0.71	1	0.71	0	0.71	0.29	0.71

D'où, $S'_6 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$\underline{(S_1, S_7)}$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_1	0	0.86	0.86	0.86	0	0.86	0	0.86
S_7	0.13	0	0	0	0	0	0.13	0.13
Total	0.13	0.86	0.86	0.86	0	0.86	0.13	0.99

D'où, $S'_7 = (0, 1, 1, 1, 0, 1, 0, 1)$ *avec* $Z = 285$

$\underline{(S_{10}, S_6)}$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{10}	0	0.68	0.68	0	0.68	0	0	0.68
S_6	0.31	0	0.31	0	0	0	0.31	0
Total	0.31	0.68	0.99	0	0.68	0	0.31	0.68

D'où, $S'_8 = (0, 1, 1, 0, 1, 0, 0, 1)$ *avec* $Z = 250$

$\underline{(S_{10}, S_7)}$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{10}	0	0.84	0.84	0	0.84	0	0	0.84
S_7	0.15	0	0	0	0	0	0.15	0.15
Total	0.15	0.84	0.84	0	0.84	0	0.13	0.99

D'où, $S'_9 = (0, 1, 1, 0, 1, 0, 0, 1)$ *avec* $Z = 250$

$\underline{(S_6, S_7)}$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 1) \Rightarrow Z = 115$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_6	0.72	0	0.72	0	0	0	0.72	0.72
S_7	0.28	0	0	0	0	0	0.28	0.28
Total	1	0	0.72	0	0	0	1	1

D'où, $S'_{10} = (1, 0, 1, 0, 0, 0, 1, 1)$ *avec* $Z = 135$

Type 02 :

$\underline{(S_{14}, S_1, S_{10})}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.35	0.35	0.35	0.35	0	0	0
S_1	0	0.34	0.34	0.34	0	0.34	0	0.34
S_{10}	0	0.30	0.30	0	0.30	0	0	0.30
Total	0	0.99	0.99	0.69	0.65	0.34	0	0.64

D'où, $S'_{11} = (0, 1, 1, 1, 1, 0, 0, 1)$ *avec* $Z = 310$

$\underline{(S_{14}, S_6, S_1)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.42	0.42	0.42	0.42	0	0	0
S_6	0.16	0	0.16	0	0	0	0.16	0
S_1	0	0.41	0.41	0.41	0	0.41	0	0.41
Total	0.16	0.83	0.99	0.83	0.42	0.41	0.16	0.41

D'où, $S'_{12} = (0, 1, 1, 1, 0, 0, 0, 0)$ *avec* $Z = 250$

$\underline{(S_{14}, S_7, S_1)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.46	0.46	0.46	0.46	0	0	0
S_7	0.07	0	0	0	0	0	0.07	0.07
S_1	0	0.46	0.46	0.46	0	0.46	0	0.46
Total	0.07	0.92	0.92	0.92	0.46	0.46	0.07	0.53

D'où, $S'_{13} = (0, 1, 1, 1, 0, 0, 0, 1)$ *avec* $Z = 270$

$\underline{(S_{10}, S_6, S_{14})}$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{10}	0	0.38	0.38	0	0.38	0	0	0.38
S_6	0.17	0	0.17	0	0	0	0.17	0
S_{14}	0	0.44	0.44	0.44	0.44	0	0	0
Total	0.17	0.82	0.99	0.44	0.82	0	0.17	0

D'où, $S'_{14} = (0, 1, 1, 0, 1, 0, 0, 0)$ *avec* $Z = 230$

$\underline{(S_{10}, S_7, S_{14})}$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{10}	0	0.43	0.43	0.43	0.43	0	0	0
S_7	0.07	0	0	0	0	0	0.07	0.07
S_{14}	0	0.49	0.49	0.49	0.49	0	0	0.49
Total	0.07	0.92	0.92	0.92	0.92	0	0.07	0.55

D'où, $S'_{15} = (0, 1, 1, 1, 1, 0, 0, 1)$ *avec* $Z = 310$

$\underline{(S_6, S_7, S_{14})}$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_6	0.25	0	0.25	0	0	0	0.25	0
S_7	0.10	0	0	0	0	0	0.10	0.10
S_{14}	0	0.64	0.64	0.64	0.64	0	0	0
Total	0.35	0.64	0.89	0.64	0.64	0	0.35	0.10

D'où, $S'_{16} = (0, 1, 1, 1, 1, 0, 0, 0)$ *avec* $Z = 290$

Type 03 :

$\underline{(S_{14}, S_1, S_{10}, S_6)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.31	0.31	0.31	0.31	0	0	0
S_1	0	0.30	0.30	0.30	0	0.30	0	0.30
S_{10}	0	0.27	0.27	0	0.27	0	0	0.27
S_6	0.12	0	0.12	0	0	0	0.12	0
Total	0.12	0.88	1	0.61	0.58	0.30	0.12	0.57

D'où, $S'_{17} = (0, 1, 1, 1, 1, 0, 0, 1)$ *avec* $Z = 310$

$\underline{(S_{14}, S_7, S_1, S_{10})}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.33	0.33	0.33	0.33	0	0	0
S_7	0.05	0	0	0	0	0	0.05	0.05
S_1	0	0.32	0.32	0.32	0	0.32	0	0.32
S_{10}	0	0.28	0.28	0	0.28	0	0	0.28
Total	0.05	0.93	0.93	0.55	0.61	0.32	0.05	0.65

D'où, $S'_{18} = (0, 1, 1, 1, 1, 0, 0, 1)$ *avec* $Z = 310$

$\underline{(S_6, S_7, S_{14}, S_1)}$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_6	0.15	0	0.15	0	0	0	0.15	0
S_7	0.06	0	0	0	0	0	0.06	0.06
S_{14}	0	0.39	0.39	0.39	0.39	0	0	0
S_1	0	0.39	0.39	0.39	0	0.39	0	0.39
Total	0.21	0.78	0.93	0.78	0.39	0.39	0.21	0.45

D'où, $S'_{19} = (0, 1, 1, 1, 0, 0, 0, 0)$ *avec* $Z = 250$

Type 04 :

$\underline{(S_{14}, S_1, S_{10}, S_6, S_7)}$

$S_{14} = (0, 1, 1, 1, 1, 0, 0, 0) \Rightarrow Z = 290$

$S_1 = (0, 1, 1, 1, 0, 1, 0, 1) \Rightarrow Z = 285$

$S_{10} = (0, 1, 1, 0, 1, 0, 0, 1) \Rightarrow Z = 250$

$S_6 = (1, 0, 1, 0, 0, 0, 1, 0) \Rightarrow Z = 115$

$S_7 = (1, 0, 0, 0, 0, 0, 1, 1) \Rightarrow Z = 45$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
S_{14}	0	0.29	0.29	0.29	0.29	0	0	0
S_1	0	0.28	0.28	0.28	0	0.28	0	0.28
S_{10}	0	0.25	0.25	0	0.25	0	0	0.25
S_6	0.12	0	0.12	0	0	0	0.12	0
S_7	0.04	0	0	0	0	0	0.04	0.04
Total	0.16	0.82	0.94	0.57	0.54	0.28	0.26	0.57

D'où, $S'_{20} = (0, 1, 1, 1, 1, 0, 0, 1)$ *avec* $Z = 310$

Après la combinaison des solutions, on trouve les nouvelles solutions suivantes :

$F = \{S'_{10}, S'_{11}, S'_{12}, S'_{13}, S'_{14}\}$, avec :

$S'_{10} = (1, 0, 1, 0, 0, 0, 1, 1), Z = 135$

$S'_{11} = (0, 1, 1, 1, 1, 0, 0, 1), Z = 310$

$S'_{12} = (0, 1, 1, 1, 0, 0, 0, 0), Z = 250$

$S'_{13} = (0, 1, 1, 1, 0, 0, 0, 1), Z = 270$

$S'_{14} = (0, 1, 1, 0, 1, 0, 0, 0), Z = 230$

Amélioration de nouvelles solutions

Solution	Z	Poids Total	$\frac{w_i}{p_i}$
$S'_{10} = (1, 0, 1, 0, 0, 0, 1, 1)$	135	$131 > 115$	$x_1 = 0, 42$ $x_2 = 3, 57$ $x_3 = 4, 5$ $x_4 = 2, 5$ $x_5 = 1, 05$ $x_6 = 0, 6$ $x_7 = 0, 16$ $x_8 = 1, 25$ On change la valeur de x_i ayant la plus petite valeur de $\frac{w_i}{p_i}$ de 1 vers 0. $x_7 = 0$
$S'_{10} = (1, 0, 1, 0, 0, 0, 0, 1)$	125	$71 < 115$	On calcule : 115-71=44 La valeur dont le poids est ≤ 44, on la change de 0 vers 1 $x_5 = 1$
$S'_{10} = (1, 0, 1, 0, 1, 0, 0, 1)$	165	$109 < 115$	On calcule : 115-109=06 Il n'y a pas de valeur dont le poids est ≤ 06, alors on arrête
$S'_{11} = (0, 1, 1, 1, 1, 0, 0, 1)$	310	$126 > 115$	$x_5 = 0$
$S'_{11} = (0, 1, 1, 1, 0, 0, 0, 1)$	270	$88 < 115$	On calcule : 115-88=27 La valeur dont le poids est ≤ 27, on la change de 0 vers 1 $x_6 = 1$
$S'_{11} = (0, 1, 1, 1, 0, 1, 0, 1)$	285	$113 < 115$	On calcule : 115-113=02 Il n'y a pas de valeur dont le poids est ≤ 02, alors on arrête
$S'_{12} = (0, 1, 1, 1, 0, 0, 0, 0)$	250	$72 < 115$	On calcule : 115-72=43 La valeur dont le poids est ≤ 43, on la change de 0 vers 1 $x_5 = 1$
$S'_{12} = (0, 1, 1, 1, 1, 0, 0, 0)$	290	$110 < 115$	On calcule : 115-110=05 Il n'y a pas de valeur dont le poids est ≤ 05, alors on arrête
$S'_{13} = (0, 1, 1, 1, 0, 0, 0, 1)$	270	$88 < 115$	On calcule : 115-88=27 La valeur dont le poids est ≤ 27, on la change de 0 vers 1 $x_6 = 1$
$S'_{13} = (0, 1, 1, 1, 0, 1, 0, 1)$	285	$113 < 115$	On calcule : 115-113=02 Il n'y a pas de valeur dont le poids est ≤ 02, alors on arrête
$S'_{14} = (0, 1, 1, 0, 1, 0, 0, 0)$	230	$86 < 115$	On calcule : 115-86=29 La valeur dont le poids est ≤ 29, on la change de 0 vers 1 $x_6 = 1$
$S'_{14} = (0, 1, 1, 0, 1, 1, 0, 0)$	245	$111 < 115$	On calcule : 115-111=04 Il n'y a pas de valeur dont le poids est ≤ 04, alors on arrête

Le tableau des nouvelles solutions améliorées :

Nouvelle solution	Z	Après amelioration	Z
$S'_{10} = (1,0,1,0,0,0,1,1)$	135	$S'_{10} = (1,0,1,0,1,0,0,1)$	$Z = 165 = Z(S_2)$
$S'_{11} = (0,1,1,1,1,0,0,1)$	310	$S'_{11} = (0,1,1,1,0,1,0,1)$	$Z = 285 = Z(S_1)$
$S'_{12} = (0,1,1,1,0,0,0,0)$	250	$S'_{12} = (0,1,1,1,1,0,0,0)$	$Z = 290 = Z(S_{14})$
$S'_{13} = (0,1,1,1,0,0,0,1)$	270	$S'_{13} = (0,1,1,1,0,1,0,1)$	$Z = 285 = Z(S_1)$
$S'_{14} = (0,1,1,0,1,0,0,0)$	230	$S'_{14} = (0,1,1,0,1,1,0,0)$	Z=245

$S'_{14} = (0,1,1,0,1,0,0,0)$ est une nouvelle solution.

On a : $Refset_1 = \{S_{14}, S_1, S_{10}\}$

$Z(S'_{14}) = 245 < Z(S_{14}) < Z(S_1) < Z(S_{10})$ donc,

La solution S'_{14} n'est pas meilleure que les solutions de $Refset_1$, donc, on ne l'ajoute pas à cet ensemble.

On la compare maintenant avec les solutions de $Refset_2$:

Solution candidate	Distance			Distance minimale
	S_{14}	S_1	S_{10}	
S'_{14}	2	3	2	$2 < 5$
S_2	4	6	2	2
S_4	3	6	3	3
S_3	4	5	4	4
S_6	5	6	5	5
S_5	6	5	4	4
S_8	5	6	3	3
S_7	7	6	5	5

Donc on ne l'ajoute pas aussi à $Refset_2$. D'où, l'ensemble de référence reste inchangé.

La meilleure solution trouvée est $S'_{12} = S_{14}(0,1,1,1,1,0,0,0)$ avec Z=290.

Chapitre 5

Implémentation et résultat

5.1 Introduction

Dans ce chapitre, nous avons implémenté l'instance du problème du sac à dos que nous avons résolue par scatter search en utilisant la programmation dynamique qui est une méthode exacte pour comparer les deux solutions exacte et approchée. Pour ce faire, nous avons utilisé le langage Python sous windows.

5.2 Présentation du langage Python

Python est un langage de programmation interprété, multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Le langage Python est placé sous une licence libre proche de la licence BSD9 et fonctionne sur la plupart des plates-formes informatiques, des smartphones aux ordinateurs centraux, de Windows à Unix, et peut aussi être traduit en Java ou NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

Python est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant parti-

culièrement utilisé comme langage de script pour automatiser des tâches simples. On l'utilise également comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses bibliothèques optimisées destinées au calcul numérique.

5.2.1 Programmation dynamique

La programmation dynamique est une méthode exacte de résolution d'un problème d'optimisation introduite par Richard Bellman dans les années 50. Elle est similaire à la méthode de branche and bound. Elle divise le problème en sous problèmes plus petits puis elle utilise une expression récursive de la solution en fonction de celle des sous problèmes qu'on appelle principe d'optimalité de Bellman.

5.3 Exemple d'application

$$Z = 15x_1 + 100x_2 + 90x_3 + 60x_4 + 40x_5 + 15x_6 + 10x_7 + 20x_8 \rightarrow max$$

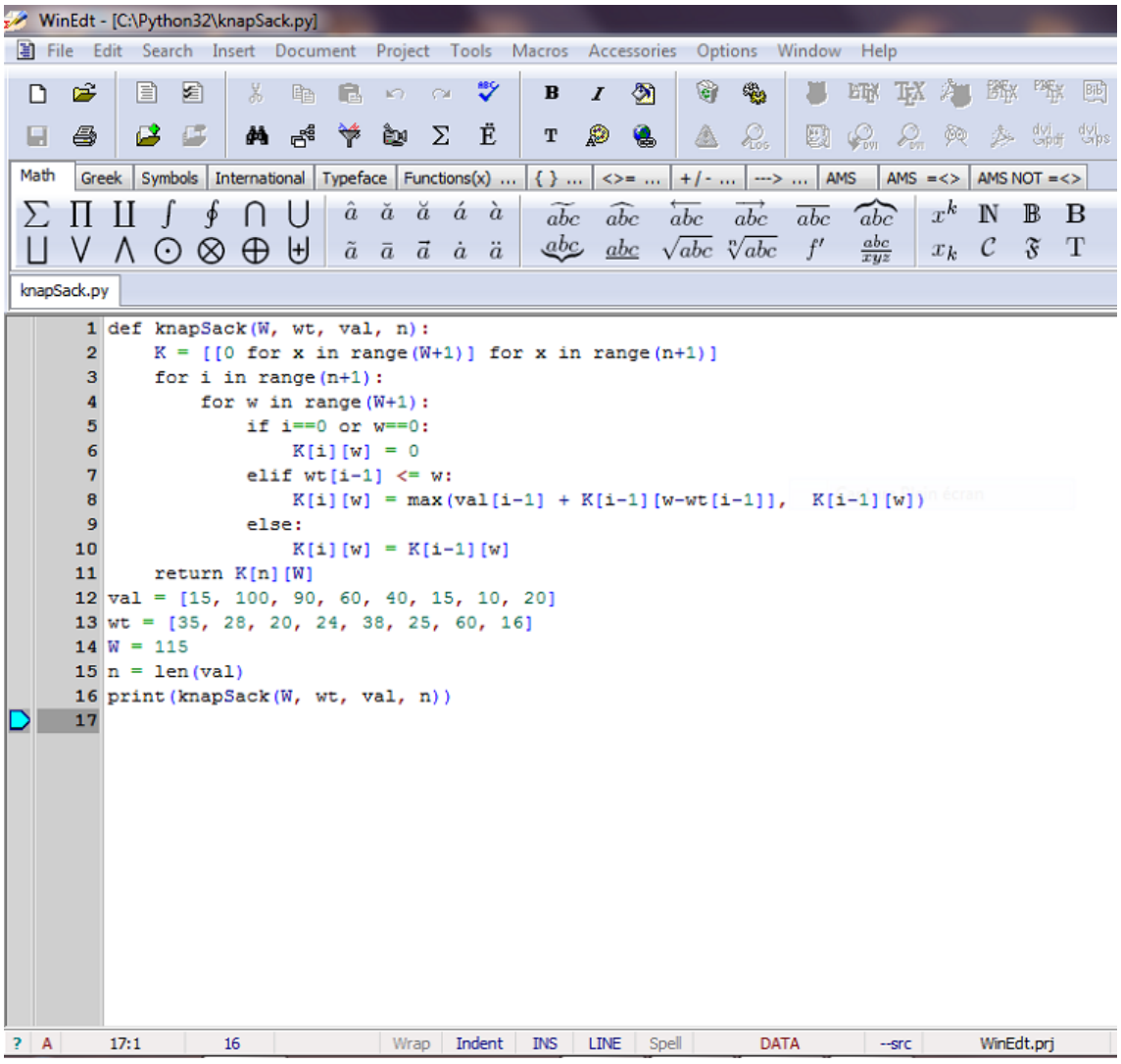
$$35x_1 + 28x_2 + 20x_3 + 24x_4 + 38x_5 + 25x_6 + 60x_7 + 16x_8 \leq 115$$

$$x_i \in \{0, 1\},\ i = 1, ..., 8$$

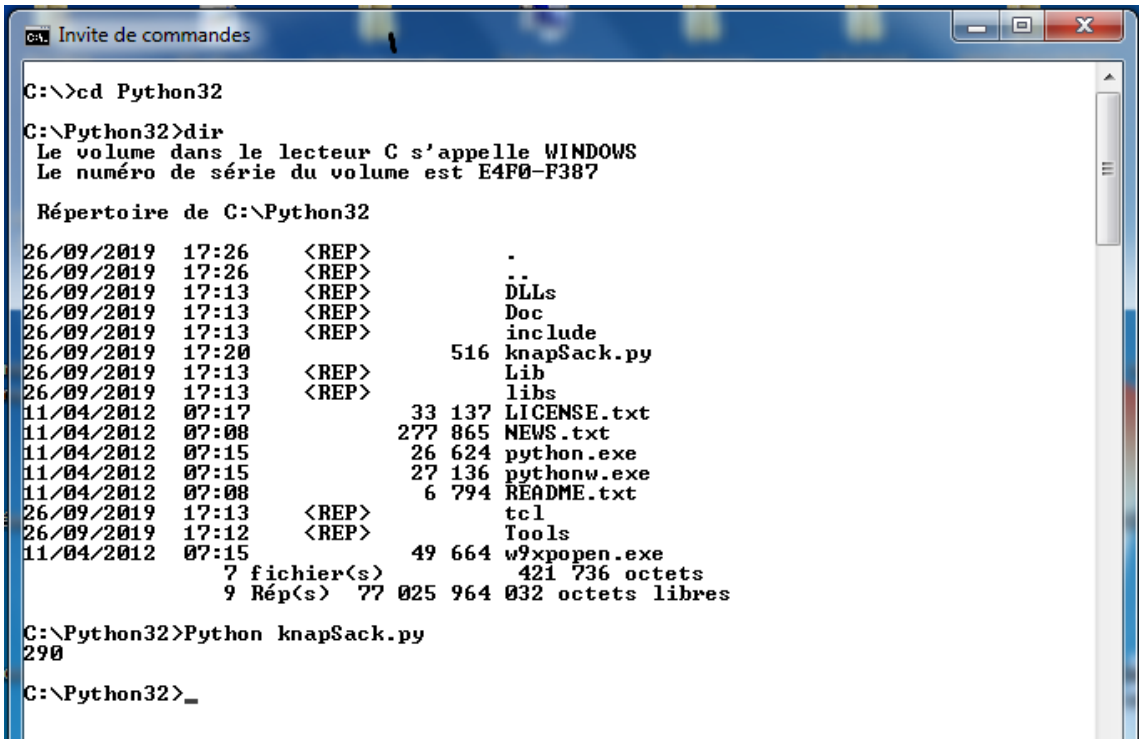
Algorithm 2 Programmation dynamique pour le sac à dos

```
for j from 0 to W do :
m[0, j] := 0
for i from 1 to n do :
for j from 0 to W do :
if w[i] > j then :
m[i, j] := m[i-1, j]
else :
m[i, j] := max(m[i-1, j], m[i-1, j-w[i]] + v[i])
```

Cet algorithme se traduit sous langage Python comme suit :



Nous avons d'abord crée un fichier knapSack sous WinEdit sur lequel nous avons saisi notre programme de la méthode de la programmation dynamique que nous avons enregistré sous l'extension *.py*



La solution optimale obtenue après l'execution du programme est $Z_{opt} = 290$. Nous avons constaté que la solution que nous avons trouvée par la méthode de scatter search est optimale, (identique à la solution exacte).

Conclusion générale

Les métaheuristiques constituent une classe de méthodes approchées adaptables à un très grand nombre de problèmes combinatoires. Elles ont révélé leur grande efficacité pour fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes d'optimisation de grande taille. C'est pourquoi l'étude de ces méthodes est en développement progressif.

L'étude présentée dans ce modeste travail est consacrée à l'application de la méthode de scatter search pour le problème du sac à dos, une métaheuristique opérant sur un ensemble de solutions appelé ensemble de référence, en les combinant pour en créer de nouvelles.

Un des avantages de la méthode de scatter search est aussi qu'elle génère plusieurs solutions de bonne qualité. Cela peut être intéressant dans différents contextes lorsqu'il est demandé de fournir plusieurs solutions élites.

Nous avons résolu une instance du problème à huit variables pour laquelle nous avons trouvé la solution exacte. (Nous avons implémenté l'instance du problème en utilisant la programmation dynamique avec le langage python).

Ce modeste travail nous a permis d'un coté d'acquérir des connaissances dans le domaine d'optimisation combinatoire, en particulier les méthodes de résolutions (exactes et approchées) et plus précisément la méthode de scatter search. D'un autre coté, de toucher un peu au domaine pratique en implémentant l'instance étudiée.

Bibliographie

[1] F. Glover, Tabu search for nonlinear and parametric optimization with links to genetic algorithms, Discrete Applied Mathematics, 49, 231-255, 1994.

[2] B. Mohcene, Parallélisation de la recherche dispersée : Étude du modèle des îles, Application au problème MAX-W-SAT.

[3] F. Glover, A template for scatter search and path relinking, in :J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers(eds.), Artificial Evolution, Lecture Notes in Computer Science 1363, Springer, p. 3-51, 1998.

[4] F. Glover, Tabu search and adaptive memory programming-advances : Applications and challenges, in : R. Barr, Helgason and Kennigton (Co-eds), Advences in Metaheuristics, Optimization and Stochastic Modeling Techniques, Kluwer Academic Publishers, Boston, USA, 1-175, 1997.

[5] G. Laporte et I.H. Osman, Metaheuristics in combinatorial optimization, Annals of Operations Research 63, J.C. Baltzer Science Bublishers, Basel, Switzerland, 1996.

[6] C.R. Reves(Ed.), Modern heuristic techniques for combinatirial problems, Blackwell Scientific Publications, Oxford, 1993.

[7] G. Vincent et V. Gareux, Optimisation par Essaim Particulaire, Mémoire Adaptative, Recherche Tabou, 4 décembre 2008.

[8] M.G. Lagoudakis, The 0-1 Knapsack Problem, An Introductory Survey, 1996.

[9] U. Pferschy, Dynamic programming revisited : improving knapsack algorithms, Computing, Archives for Scientific Computing, vol. 63, n°4, 419-430, 1999.

[10] R.S. Garfinkel et G.L. Nemhauser, Integer Programming, John Wiley and Sons, New York, 1972.

[11] S. Martello et P. Toth, Knapsack Problems : Algorithms and Computer Implementations, John Wiley and Sons, 1990.

[12] G. Plateau et M. Elkihel, A hybrid method for the 0-1 knapsack problem, Methods Oper, Res., 49, 277-293, 1985.

[13] C. Blum et A. Roli, Metaheuristics in combinatorial optimization : Overview and conceptual comparison, ACM Computing Surveys, vol. 35, n°3, p. 268-308, septembre 2003.

[14] C.H. Papadimitriou et K. Steiglitz, Combinatorial optimization-algorithms and complexity, Prentice Hall, 1982.

[15] W. Dullaret, M. Sevaux, K. Soïresen et J. Springael, Applications of metaheuristics, Numéro spécial du European Journal of Operational Research, n°179, 2007.

[16] C.C. Ribeiro et N. Maculan(Eds), Applications of combinatorial optimization, Annals of Operations Research 50, 1994.

[17] M.R. Garey et D.S. Johnson, Computers and intractibility : a guide to the theory of NP-completeness, W.H. Freeman and Company, New York, 1979.

[18] I.H. Osman et J.P. Kelly(Eds), Metaheuristics : theory and applications, Kluwers Academic Publishers, Boston, 1996.

[19] S.F. Smith, A Learning System Based on Genetic Adaptive Algorithms, PhD dissertation, University of Pittsburgh, 1980.

[20] F. Glover, Future Paths for Integer Programming and Links to Artificial Intelligence, Comput. and Ops, Res., vol. 13, n°5, 533-549, 1986.

[21] P. Moscato, On Evolution, Search Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms, Caltech Concurrent computation Program, C3P Report 826, 1989.

[22] M. Dorigo, Optimization, Learning and Natural Algorithms, Thèse de doctorat, Politecnico di Milano, Italie, 1992.

[23] L. Fogel, A.J. Owens et M.J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley, 1966.

[24] E.L.H. Aarts et J.K. Lenstra(Eds), Local search in combinatorial optimization, John Wiley and Sons, 1997.

[25] H. Kellerer, U. Pferschy et D. Pisinger, Knapsack Problems, Springer, 2004.

[26] T. Bäck et H.P. Schwefel, An overview of evolutionary algorithms for parameter optimization computation, vol. 1, 1-24, 1993.

[27] F. Glover, Sciences de la décision : Heuristics for integer programming using surrogate constrains, Wiley Online Library, 1977.