الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abou Bekr Belkaid
Tlemcen Algérie

جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire

Université Abou BekrBelkaid – Tlemcen

Faculté des Sciences

Département d'Informatique

Rapport de Projet

Pour l'obtention du diplôme de Master en Informatique

Option : Réseaux et Système Distribuer (RSD)

Thème

Méthode exacte pour le problème du sac à dos

Réalisé par :

- BENTIFOUR Khayreddine
- MEZOUAR Ahmed

Présenté le 24 septembre 2022 devant la commission composée :

- Mme AMRAOUI Asma (Président)
- Mme BENMAHDI Bouchra (Examinateur)
- Mr BELHOCINE Amine (Encadreur)

Année universitaire : 2021-2022

REMERCIEMENTS

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

Nous tenons à remercier vivement notre encadreur Mr BELHOCINE qui nous a dirigé dans ce travail ainsi que pour ses précieux conseils et ses encouragements.

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

Enfin, nous remercions tous nos amis qui nous ont aidés, encouragés, et toute personne ayant contribué à ce travail, par un conseil, ou même par un sourire.

DÉDICACE

« Nous dédions ce travail

Aux meilleurs des pères

A nos très chères mères

Qu'ils trouvent on nous la source de leur fierté à qui nous devons tout

A nos chers frères et sœurs, pour leur appui et leurs encouragements

A nos amis qui nous ont toujours encouragés et à qui nous souhaitons plus de succès

A toutes nos familles pour leur soutien tout au long de notre parcours universitaire

A tous ceux qui nous sont chers

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible

Merci d'être toujours là pour nous. »

BentifourKhayreddine&Mezouar Ahmed

Table des matières

Introduction Générale

Découvrir le plus court chemin pour se rendre à un endroit précis, concevoir un circuit électronique, placer des antennes pour diffuser des chaînes de télévision, ou des relais pour les réseaux de téléphonie mobile, reconstruire le code génétique des êtres humains, décider de l'implantation d'un site de production, …

Cette liste peut paraître hétérogène et provenant de domaines très différents, mais elle contient des problèmes qui ont tous une structure combinatoire. En effet, chacun d'entre eux revient à choisir la meilleure combinaison parmi une quantité bien connue mais souvent exponentiellement grande de possibilités.

Parmi les problèmes traités par la recherche opérationnelle on trouve le problème du sac à dos. Informellement, ce problème est défini de la manière suivante : durant un cambriolage un voleur possède un sac dont la capacité est limitée. Il se trouve face à un ensemble d'objets qu'il peut dérober. Chacun de ces objets est caractérisé par sa valeur et son poids. Le voleur souhaite optimiser la valeur totale des objets qu'il va dérober tout en ne dépassant pas le poids maximal supporté par son sac.

Ce problème est une abstraction pour un grand nombre d'autres problèmes d'optimisation.

Il est aussi utilisé en cryptographie comme une base pour différents schémas de chiffrement. Il faut cependant noter que la plupart de ces schémas de chiffrement ne sont plus actuellement considérés comme sûrs.

L'énoncé de ce problème est simple : « Étant donné plusieurs objets possédants chacun un poids et une valeur et étant donné un poids maximum pour le sac, quels objets faut-il mettre dans le sac de manière à maximiser la valeur totale sans dépasser le poids maximal autorisé pour le sac ? ».

La résolution des problèmes d'optimisation combinatoires (POC) nécessite une modélisation efficace et adaptée.

Dans le premier chapitre nous présenterons les concepts généraux relatifs à la recherche opérationnelle. Nous détaillerons dans le deuxième chapitre le problème du sac à dos et la solution utilisée à savoir la programmation dynamique. Puis nous présenterons l'application, les études menées ainsi que les résultats obtenus dans le troisième chapitre.

Chapitre 01 :

La Recherche Opérationnelle

1- La Recherche opérationnelle

1.1- Introduction :

La recherche opérationnelle (aussi appelée aide à la décision) peut être définie comme l'ensemble des méthodes et techniques rationnelles d'analyse et de synthèse des phénomènes d'organisation utilisables pour élaborer de meilleures décisions.La recherche opérationnelle (RO) propose des modèles conceptuels pour analyser des situations complexes et permet aux décideurs de faire les choix les plus efficaces. Le domaine est fortement lié à l'ingénierie des systèmes.

Dans ce chapitre, on présente c'est quoi la recherche opérationnelle avec les différents types de ce dernier en plus de ça les méthodes utilisées par cette recherche et les domaines d'application.

1.2- Définition :

La recherche opérationnelle (RO) est une discipline de mathématiques appliquées qui traite de l'utilisation optimale des ressources dans l'industrie et le secteur public.

Au cours de la dernière décennie, la portée de la RO s'est élargie pour inclure l'économie, la finance, le marketing et la planification d'entreprise. Plus récemment, la RO a été utilisée dans la gestion des systèmes de santé et d'éducation, dans la résolution de problèmes environnementaux et dans d'autres domaines d'intérêt public.[1]

1.3- Historique :

La recherche opérationnelle est née pendant la Seconde Guerre mondiale en tant qu'effort conjoint d'éminents mathématiciens (dont von Neumann, Danziger, Blackett) à qui on a demandé de fournir des techniques d'optimisation des ressources militaires. Le premier succès de cette approche a été obtenu on 1940 par le prix Nobel de physique Patrick Blackett, qui a résolu le problème de la localisation optimale des radars de surveillance. Le qualificatif « opérabilité » vient du fait que la première application de la discipline est liée aux opérations militaires. Le nom est resté depuis, même si le domaine militaire n'est plus le champ d'application principal de la discipline, et le mot "opérationnel" tend à aller dans le sens d'"efficace". Ce sont donc ces mathématiciens qui ont créé une nouvelle méthode qui propose la modélisation et l'optimisation des problèmes pratiques. Dès les années 1950, la recherche opérationnelle fait son entrée dans l'entreprise. en France, des sociétés comme EDF, Air France et la SNCF créent à cette époque des services de recherche opérationnelle (qui existent toujours). La matière a commencé à être enseignée dans les universités et les grandes écoles. Puis, au milieu des années 1970, la discipline a sans doute perdu de son élan en raison de l'enthousiasme initial et des ressources informatiques insuffisantes pour appliquer l'approche RO. A partir du milieu des années 1990, on assiste à un retour en force de la RO, et les outils informatiques sont désormais comparables aux méthodes proposées par la recherche

opérationnelle. Depuis lors, nous avons assisté à une explosion du nombre de logiciels commerciaux et de nombreuses boîtes à suggestions.[2]

1.4- Types de problèmes résolus par la RO

La recherche opérationnelle peut aider les décideurs face à des problèmes combinatoires, stochastiques ou concurrentiels.

Un problème est dit combinatoire lorsqu'il contient un grand nombre de solutions acceptables, parmi lesquelles on cherche une solution optimale ou quasi optimale. Exemple type : déterminer où installer 5 centres de distribution sur 30 emplacements d'installation possibles afin de minimiser les coûts de transport entre ces centres et les clients. Ce problème ne peut pas être résolu par le cerveau humain enénumérant simplement les solutions possibles, car il y a 30 x 29 x 28 x 27 x 26 / (5x4x3x2) = 142 506 (!) possibilités. Même si des problèmes de cette ampleur peuvent être résolus par énumération informatique, les décideurs sont souvent confrontés à des problèmes d'une complexité infinie, où le nombre de solutions acceptables se chiffre en milliards (explosion combinatoire).

Un problème est dit aléatoire s'il consiste à trouver une solution optimale à un problème posé en termes incertains. Exemple type : Connaissant la distribution aléatoire du nombre de personnes entrant dans la commune enune minute et la distribution aléatoire du temps de traitement de dossier d'une personne, déterminer le nombre minimum d'armoires ouvertes de sorte qu'il y ait moins de 5 % de chances qu'une personne attende plus de 15 minutes.

Un problème est dit compétitif s'il s'agit de trouver la solution optimale à un problème dont les conditions dépendent des interrelations entre ses propres actions et celles des autres décideurs. Exemple : Établir une politique de prix de vente, sachant que le résultat de cette politique dépend de ce que fera un concurrent. [3]

1.5- Méthodes utilisées par la RO

- **Algorithmes polynomiaux**

Certains problèmes de recherche opérationnelle ne sont pas NP-complets. Dans ce cas, on utilise un algorithme polynomial pour les résoudre, si le polynôme est de degré raisonnable.

- **Programmation dynamique**

La programmation dynamique a été initiée par Bellman on 1957, elle s’appuie sur un principe simple, appelé le principe d’optimalité de Bellman : « Une politique optimale est formée de sous-politiques optimales ». Le mot politique désigne une séquence de décisions définissant une solution d’un problème d’optimisation. Concrètement, la programmation dynamique est une méthode utilisée pour résoudre des problèmes où une séquence de décisions optimale doit être trouvée. L’idée de base est que l’on peut déduire une solution optimale d’un problème en combinant des solutions optimales d’une série de sous problèmes consistant à choisir des séquences plus courtes de décisions. Les solutions des problèmes sont calculées de manière ascendante, c’est-à-dire qu’on débute par les solutions des sous-problèmes les plus petits pour

ensuite déduire progressivement les solutions de l'ensemble. La programmation dynamique est une approche d'optimisation qui transforme un problème complexe en une séquence de problèmes plus simples, sa caractéristique essentielle est la nature à plusieurs stades de la procédure d'optimisation. Enplus des techniques d'optimisation décrites précédemment, la programmation dynamique fournit un cadre général pour analyser de nombreux types de problèmes. Dans ce cadre, une variété de techniques d'optimisation peut être utilisée pour résoudre des aspects particuliers d'une formulation plus générale. Habituellement, la créativité est nécessaire avant de pouvoir reconnaître qu'un problème particulier peut être efficacement lancé en tant que programme dynamique, et souvent des idées subtiles sont nécessaires pour restructurer la formulation afin qu'elle puisse être résolue efficacement. La programmation dynamique est connue pour résoudre les problèmes les plus répandus comme le problème de sac à dos, la tour de Hanoi, le chemin le plus court par Dijkstra. La programmation dynamique peut être utilisée à la fois de haut en bas et de bas en haut. Et bien sûr, la plupart du temps, on se référant à la sortie de la solution précédente, il est moins coûteux que de recalculer en termes de cycles CPU. [4]

- **Processus stochastiques**

Les processus stochastiques concernent tous les problèmes aléatoires, en particulier des problèmes de fiabilité (de systèmes, de composants électroniques…) et des phénomènes d'attente.

- **Simulation informatique**

La simulation est souvent employée pour résoudre des problèmes de RO, notamment dans le milieu non académique.

Ces simulations informatiques sont rapidement devenues incontournables pour la modélisation des systèmes naturels on physique, chimie et biologie, mais également des systèmes humains on économie et on science sociale. Elles permettent de limiter le risque et d'éviter le coût d'une série d'épreuves réelles (ex : essais de véhicules). Elles peuvent offrir un aperçu sur le développement d'un système trop complexe pour simuler avec de simples formules mathématiques (ex : ouragan).

La simulation numérique est utilisée pour :

- Prévoir l'état final d'un système connaissant son état initial (problème direct) ;
- Déterminer les paramètres d'un système connaissant un ou plusieurs couples (état initial - état final) (problème inverse) ;
- Préparer des opérateurs à des conditions plus ou moins rares dans leur interaction avec un système complexe (simulation d'entraînement).

En sciences sociales, la simulation informatique fait partie d'un des cinq angles de collecte de données dans la méthode plus générale dite de percolation de données, qui couvre aussi conjointement les méthodes quantitatives et qualitatives, la revue des écrits (y compris les écrits scientifiques), et les interviews d'experts. À ce chapitre, la percolation des données offre une vision et une méthode plus complètes que la triangulation des données lors de l'analyse des phénomènes sous observation. [5]

- **Programmation linéaire et non linéaire**

La programmation linéaire est très souvent utilisée pour résoudre des problèmes combinatoires. Elle permet de résoudre très efficacement les problèmes dans lesquels les

variables sont continués. Lorsqu'il y a des variables discrètes, programmation linéaire et méthodes arborescentes (voir ci-après) peuvent être combinées.

La programmation non linéaire peut aussi être utilisée. La possibilité d'utiliser des contraintes ou des fonctions objectifs non linéaires offre une puissance de modélisation très importante mais les algorithmes de résolution des programmes non linéaires sont significativement moins efficaces que ceux de la programmation linéaire.

- **Méthodes arborescentes**

La première procédure a été mise au point par GILMORE on 1962 et depuis plusieurs améliorations et d'autres versions ont été développées. Elle est basée sur une recherche arborescente.

Le Branch&Bound est une technique d'énumération implicite fondée sur un principe de décomposition du problème en sous problème, à chaque père correspond deux problème fils dont une variable a été fixée à 1 pour l'un et 0 pour l'autre (dans le cas de la Programmation Linéaire en Nombres Entiers à variable 0/1). L'évaluation d'un nœud s'effectue à l'aide d'une relaxation (continue ou lagrangienne). Ce principe permet de trouver plus rapidement la solution optimale (élimination de nœud pour lesquelles la relaxation est moins bonne que la meilleure solution trouvée). Ainsi de suite jusqu'à ne plus avoir que des problèmes faciles à résoudre ou qui ne peuvent pas contenir de solution optimale ou réalisable. Elle repose sur deux approches; qui sont: l'arbre d'énumération et les bornes duales définies par l'arbre de l'énumération. L'élaguation est le fait d'éliminer les ensembles de solutions pour lesquels il n'existe pas d'optimum. Et le cheminement représente le fait de choisir le prochain ensemble de solutions à étudier. Le terme de nœud définit un sous ensemble de solutions à résoudre. On crée un arbre de recherche et d'énumération. A chaque nœud, nous résolvons un programme linéaire. Le résultat de cette méthode dépend des valeurs qui doivent être entières ; nous aurons alors deux principaux choix:

- Couper le nœud (à cause des bornes), dans ce cas nous avons trois raisons pour lesquelles nous pourrons couper un nœud:
  a. La relaxation linéaire n'est pas une solution réalisable
  b. La valeur objective de la relaxation linéaire n'est pas meilleure que la valeur objective de la meilleure solution entière déjà trouvée.
  c. La solution optimale de la relaxation linéaire est déjà entière

- Diviser le nœud, si la coupe du nœud n'est pas considérée, donc le nœud est divisé en deux nœuds; il faut alors créer deux programmes linéaires nouveaux. Il s'agit essentiellement de diviser (divide and conquer) l'ensemble de toutes les solutions réalisables (problème initial) on sous-ensembles plus petits (sous problèmes) et mutuellement exclusifs. C'est la phase « séparation » (Branch). Puis divers critères sont utilisés pour identifier les sous-ensembles qui peuvent contenir la solution optimale et les sous-ensembles qui ne doivent pas être explorés plus à fond, car ils ne peuvent pas contenir la solution optimale. C'est la phase « évaluation » (Bound) [4].

- **Heuristiques et métaheuristiques**

A. Heuristique :

Face au caractère intraitable de certains problèmes, et vu la nécessité de fournir des solutions raisonnablement bonnes de problèmes pratiques dans des temps raisonnables, se sont

développées des méthodes approchées appelées heuristiques. Les heuristiques classiques apportent une solution à un problème d'optimisation combinatoire enun temps raisonnable sans garantir l'optimalité; car souvent le résultat obtenu n'est pas forcément la solution optimale. Il y a l'heuristique de construction et celle de l'amélioration, ou plus récemment étudiée: l'heuristique d'insertion séquentielle proposée par Mole et Jameson. Ce sont des heuristiques conçues pour un problème particulier ens'appuyant sur sa structure propre. Cependant, on retrouve on général des principes de base utilisés dans ces heuristiques tels que:

- Principe glouton: ce principe repose sur le choix définitif des valeurs de la solution; ce qui interdit toute modification ultérieure.

- Principe de construction progressive: c'est une extension du principe glouton dans la mesure où l'on s'autorise, cette fois-ci, à modifier des valeurs déjà assignées. On retrouve par exemple l'algorithme de Ford pour la recherche des chemins optimaux.

- Principe de partitionnement: ce principe repose sur le fait que résoudre le problème global se révèle souvent plus complexe que résoudre la somme des sous problèmes qui le composent. Toute la difficulté réside alors dans le fusionnement des solutions de chaque sous problème. [6]

B. Métaheuristiques :

La métaheuristique a sa place à un niveau plus général encore, et intervient dans toutes les situations où l'ingénieur ne connaît pas d'heuristique efficace pour résoudre un problème donné, ou lorsqu'il estime qu'il ne dispose pas du temps nécessaire pour on déterminer une. En1996, I.H. Osman et G. Laporte définissaient la métaheuristique comme « un processus itératif qui subordonne et qui guide une heuristique, on combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales.on2006, le réseau Métaheuristiques (metaheuristics.org) définit les métaheuristiques comme « un ensemble de concepts utilisés pour définir des méthodes heuristiques, pouvant être appliqués à une grande variété de problèmes. On peut voir une métaheuristique comme une « boîte à outils » algorithmique, utilisable pour résoudre différents problèmes d'optimisation, et ne nécessitant que peu de modifications pour qu'elle puisse s'adapter à un problème particulier ». Elle a donc pour objectif de pouvoir être programmée et testée rapidement sur un problème [14]. Comme l'heuristique, la métaheuristique n'offre généralement pas de garantie d'optimalité, bien qu'on ait pu démontrer la convergence de certaines d'entre elles. Non déterministe, elle incorpore souvent un principe stochastique pour surmonter l'explosion combinatoire. Elle fait parfois usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche. Ces dernières années, la plupart des méta-heuristiques présentées pour résoudre des problèmes combinatoires sont d'inspiration biologique. Parmi ces métaheuristiques, nous pouvons citer les algorithmes génétiques, le recuit simulé, les réseaux de neurones, les algorithmes à estimation de distribution, et l'optimisation par colonie de fourmis. [7]

1.6- Domaines d'application de la RO

Peu d'entreprises embauchent des chercheurs opérationnels pour aider les décideurs à résoudre les problèmes. Lorsque de telles questions se posent, les références sont souvent faites à de grands cabinets de conseil ou au département de recherche opérationnelle d'une université (bien que la tendance actuelle soit d'externaliser cette expertise académique par le biais de petites entreprises privées, appelées spin-offs). Des problèmes simples peuvent être résolus on interne, et la plupart des universités intègrent des cours d'introduction à la recherche opérationnelle dans leurs cours pour ingénieurs, mathématiciens, informaticiens, comptables on management et économistes.

Malgré son importance inhérente, la RO est encore peu utilisée dans le monde industriel, soit par manque de formation des décideurs, soit par manque de pertinence de l'outil ou des difficultés de mise enœuvre. Les principales préoccupations exprimées par les décideurs politiques concernant l'application du modèle R.O. dans leurs entreprises sont :

Une prise on compte limitée des facteurs :
Pour les problèmes stratégiques, les réponses « pures et parfaites » aux solutions mathématiques semblent rarement s'appliquer on fait. Même si la recherche opérationnelle intègre de nombreux facteurs, si certains aspects sont relativement faciles à modéliser au sens mathématique (ex : coût, rentabilité, distance, durée, rythme), on revanche, d'autres éléments sont plus difficiles à modéliser : contraintes légales, dissuasions des aspirations commerciales des concurrents, l'importance des relations avec les élus, l'ethos social, etc. Or, l'importance de ces facteurs dans la prise de décision est importante et parfois déterminante.

Un investissement important :
Les outils mathématiques eux-mêmes nécessitent un haut niveau de connaissances mathématiques, une bonne capacité à modéliser des problèmes et à décrire des facteurs ; ces contraintes coûtent du temps et de l'argent (que ce soit par un développement interne qui consomme des ressources ; ou un développement externe qui coûte de l'argent). Ensuite, il faut trouver un équilibre entre l'investissement nécessaire et le rendement attendu.

Pour des événements peu fréquents :
L'entreprise ne bénéficie pas de l'effet d'expérience : de temps on temps, le problème concerne un service différent, ou les responsables ont changé entre deux études. Il est donc difficile d'entretenir les compétences R.O. à l'intérieur de l'entreprise.

Le décideur devra prendre ces différents aspects on compte lorsqu'il décidera ou non de mettre enœuvre des modèles de recherche opérationnelle dans son entreprise.

1.7- Conclusion :

Nous avons pu voir que la recherche opérationnelle est un domaine très vaste. Elle fait appel à des connaissances mathématiques variées et est utile dans beaucoup de cas : l'optimisation de la productivité, la gestion des stocks, la taille optimale d'un convoi militaire, etc.

De plus, certaines de ces matières font appels à plusieurs méthodes de résolution et plusieurs algorithmes qui ont chacun leurs avantages et inconvénients.

Chapitre 02 :

Le problème du sac à dos et la Programmation Dynamique

1- Problème du sac à dos

1.1- Introduction

Le problème du sac à dos, également appelé KP (en anglais, KnapsackProblem) est un problème d'optimisation combinatoire. Il simule une situation similaire au rembourrage du sac à dos, où le sac à dos ne peut pas supporter plus d'un certain poids, et tout ou partie d'un groupe d'objets a un poids et une valeur. Les articles placés dans le sac à dos doivent maximiser la valeur totale et ne pas dépasser le poids maximum.

Dans ce chapitre, premièrement on présente c'est quoi le problème de sac à dos avec la formulation mathématique de ce dernier en plus de ça les méthodes de résolution, Ensuite on présente la programmation dynamique avec leur principe et algorithme et on parle des avantages et des inconvénients.

1.2- Définition :

le problème du knapsack (sac à dos) est l'un des problèmes d'optimisation combinatoire les mieux étudiés, ainsi que ses variantes dérivées. Notons que c'est un problème appartenant à la classe des problèmes NP-difficiles. [8]

1.3- Formulation mathématique :

Toute formule commence par une déclaration de données. Dans notre exemple, nous avons un sac à dos avec un poids maximum de W et n objets. Pour chaque objet i, nous avons un poids wi et une valeur pi. Pour quatre objets (n = 4) et un sac à dos d'un poids maximum de 30 kg (W = 30), nous avons les données suivantes :

Objets	1	2	3	4
p_i	7	4	3	3
w_i	13	12	8	10

Table 01 : Exemple d'une instance du problème du sac à dos

Ensuite, nous devons définir des variables qui représentent on quelque sorte les actions ou les décisions qui mèneront à la recherche d'une solution. On définit la variable xi associée à l'objet i comme suit : xi = 1 si l'objet i est mis dans le sac, xi = 0 si l'objet i n'est pas mis dans le sac.

Dans notre exemple, une solution possible est de mettre tous les objets dans le sac à dos sauf le premier, nous avons donc :

$$x_1 = 0,\ x_2 = 1,\ x_3 = 1, \text{ et } x_4 = 1.$$

Ensuite, nous devons définir les contraintes du problème. Il n'y on a qu'une ici : la somme des poids de tous les éléments du sac doit être inférieure ou égale au poids maximum du sac à dos.

Cela s'écrit : $x_1.w_1 + x_2.w_2 + x_3.w_3 + x_4.w_4 \leq W$
ou plus généralement pour n objets :

$$\boldsymbol{valeur} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{n}} \boldsymbol{x_i} * \boldsymbol{w_i} < \boldsymbol{W_{sac}}$$

Pour vérifier que les contraintes sont respectées dans notre exemple, il suffit de calculer la somme suivante : 0 × 13 + 1 × 12 + 1 × 8 + 1 × 10 = 30, ce qui est bien inférieur ou égal à 30, donc les contraintes sont respectées. Ensuite, nous parlons d'une solution possible. Mais ce n'est pas forcément la meilleure solution. Enfin, nous devons exprimer la fonction qui transforme notre objectif : maximiser la valeur totale des articles dans le sac.

Pour *n* objets, cela s'écrit :

$$\boldsymbol{Max(valeur)} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{n}} \boldsymbol{x_i} * \boldsymbol{p_i}$$

Dans notre exemple, la valeur totale contenue dans le sac est égale à 10. Cette solution n'est pas la meilleure, car il existe aussi une solution pour les valeurs supérieures à 10 : il faut prendre uniquement les objets 1 et 2, leur valeur totale sera de 11. Il n'y a pas de meilleure solution que cette dernière, alors cette solution est optimale.

1.4- Méthodes de résolution :

Il existe deux principaux types de méthodes pour résoudre les problèmes d'optimisation combinatoire : les méthodes exactes et les méthodes approchées. La méthode exacte peut aboutir à chaque fois à la solution optimale, mais si le problème est complexe, le temps de calcul peut être très long. Les méthodes approchées, également appelées heuristiques, permettent d'obtenir rapidement des solutions approchées et ne sont donc pas nécessairement optimales. Nous détaillerons des exemples d'algorithmes pour parser, pour chaque catégorie.

Méthode approchée :

Les méthodes d'approximation visent à trouver des solutions avec un bon compromis entre la qualité de la solution et le temps de calcul. Pour le problème du sac à dos, voici un exemple d'un tel algorithme :

DEBUT

Calculer la valeur (p_i / w_i) pour chaque objet *i*,

Trier tous les objets par ordre décroissant de cette valeur,

Sélectionner les objets un à un dans l'ordre du tri et ajouter l'objet sélectionné dans le sac si le poids maximal reste respecté.

FIN

Déroulons cet algorithme sur notre exemple :

- Première étape :

Objets	1	2	3	4
p_i / w_i	0,54	0,33	0,37	0,30

Table 02 : Premier étape de Déroulement

- Deuxième étape : l'ordre des objets est donc le suivant : 1, 3, 2, et 4.
- Troisième étape :
    - Objet 1 : on le met dans le sac (qui est vide au départ).
    - Objet 3 : on le met dans le sac car l'addition du poids de l'objet 1 et de l'objet 3 est inférieure à 30.
    - Objet 2 : on ne le met pas dans le sac car le poids total (33) dépasse le poids maximal du sac.
    - Objet 4 : on ne le met pas dans le sac pour la même raison que pour l'objet 2.

Donc la solution trouvée est de mettre les objets 1 et 3 dans le sac avec une valeur de 10. Cette solution n'est pas optimale car il existe des solutions avec une valeur totale de 11. Pourtant, l'algorithme reste rapide même avec une augmentation significative du nombre d'objets.

Ce type d'algorithme est aussi appelé algorithme glouton car il ne remet jamais on cause les décisions prises précédemment. Ici, lorsque l'objet 2 ne peut pas être mis dans le sac, l'algorithme n'essaie pas de retirer l'objet 3 du sac et de remettre l'objet 2 à sa place.

En plus des algorithmes gourmands, il existe de nombreuses approximations : méthodes de recherche **taboues**, **algorithmes génétiques**, **algorithmes de descendance locale**, et même un algorithme basé sur des **fourmis artificielles**. [9]

Méthode exacte :

Afin de trouver la solution optimale, et d'être sûr qu'il n'y a pas de meilleure solution, il faut utiliser une méthode exacte, qui prend plus de temps de calcul (si le problème est difficile à résoudre). Aucune méthode exacte n'est plus rapide que toutes les autres. Chaque problème a une approche plus appropriée que les autres. Nous présenterons un exemple de ce type d'algorithme, appelé processus de séparation et d'évaluation (PSE), ou branch and bound en anglais. Nous ne couvrirons ici qu'une version simplifiée du PSE. PSE est un algorithme qui énumère intelligemment toutes les solutions possibles. Enpratique, seules les solutions potentielles de haute qualité sont répertoriées et les solutions qui n'améliorent pas les solutions actuelles ne sont pas explorées. Pour représenter le PSE, nous utilisons un « arbre de recherche » composé :

- de nœuds ou sommets, où un nœud représente une étape de construction de la solution

- d’arcs pour indiquer certains choix faits pour construire la solution.

Dans notre exemple, un nœud représente une étape où certains objets vont être mis dans le sac, d'autres sont exclus du sac, et d'autres pour lesquels une décision n'a pas encore été prise. Chaque arc représente l'action de placer un objet dans le sac ou à l'inverse pas. Le schéma suivant représente une partie d'un arbre de recherche

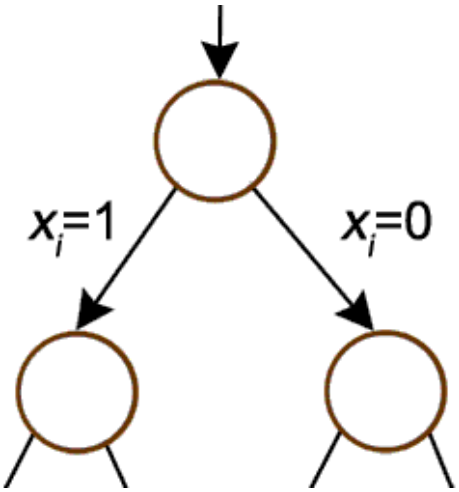

Figure 01 : Représenter une partie de l’arbre de recherche

A partir de l'étape N, l'algorithme construit deux nouvelles étapes : à gauche, on garde toutes les décisions prises à l'étape N et on ajoute la décision de mettre un nouvel objet (ici l'objet i) dans le sac. À droite, on garde aussi toutes les décisions prises à l'étape N et on ajoute la décision de ne pas placer l'objet i. Un arbre de recherche commence par un seul nœud où aucune décision n'est prise pour un objet. Une fois tous les objets sélectionnés, les nœuds finaux, également appelés nœuds feuille, représentent chacun la solution finale. La figure suivante prend l'exemple ci-dessus comme exemple pour représenter l'arbre de recherche induit. [10]

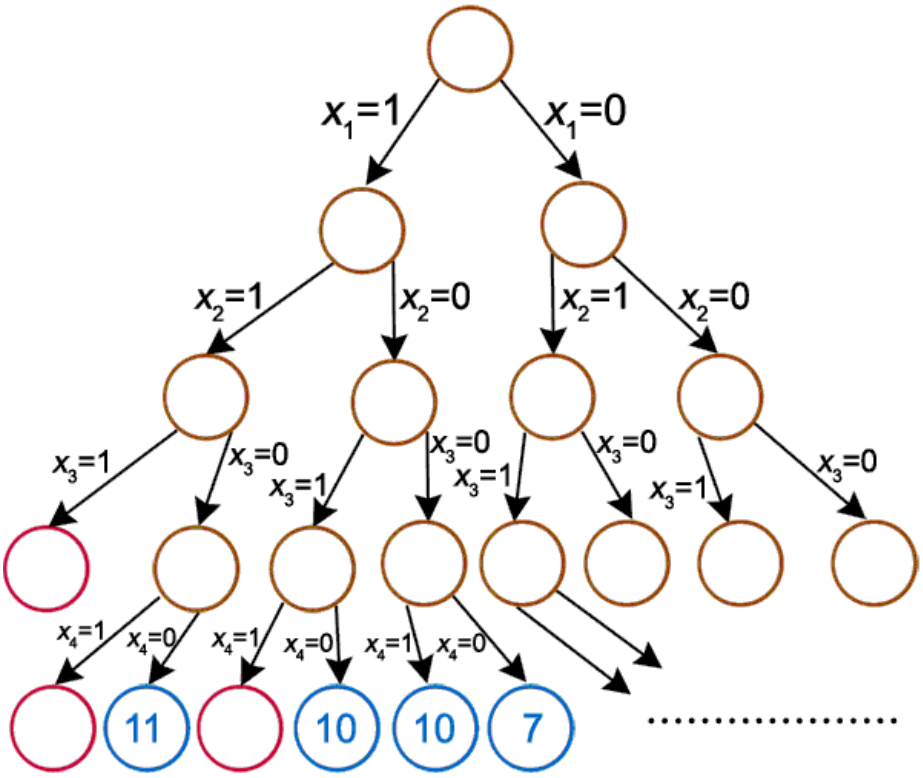

Figure 02 : Reprend l'exemple précédent et représente l'arbre de recherche induit.

Le nœud rouge représente une solution impossible. Par exemple, pour le nœud rouge le plus à gauche, il est impossible de prendre les trois premiers objets car le poids maximum ne peut pas être dépassé. Par conséquent, il n'est pas nécessaire de développer l'étape suivante avec le dernier objet. Les nœuds bleus sont des nœuds feuille correspondant à des solutions réalisables. À la fin de l'algorithme, calculez simplement la valeur du sac de chaque nœud feuille et prenez la solution avec la valeur maximale. Le graphique n'expose pas tous les nœuds feuilles car beaucoup d'entre eux n'ont que 4 objets. Et plus le nombre d'objets augmente, plus le nombre de feuilles augmente rapidement. On parle de croissance exponentielle. [11]

Il existe de nombreuses techniques pour améliorer ce processus d'arborescence. Ces techniques sont conçues pour réduire la taille des arbres et augmenter la vitesse.

Un élément essentiel dans une PSE est le calcul de bornes, inférieures et supérieures, de la fonction objectif.

Une borne inférieure est une valeur minimum de la fonction objectif. Autrement dit, c'est une valeur qui est nécessairement inférieure à la valeur de la meilleure solution possible. Dans notre cas, une configuration du sac réalisable quelconque fournit une borne inférieure.

Une borne supérieure est une valeur maximale de la fonction objectif. Autrement dit, nous savons que la meilleure solution a nécessairement une valeur plus petite. Dans notre cas, nous

savons que la valeur de la meilleure solution est nécessairement inférieure à la somme des valeurs de tous les objets (comme si on pouvait tous les mettre dans le sac).

Enfin, comme note finale, lorsqu'une solution réalisable avec une valeur plus grande est trouvée, la valeur de la borne inférieure peut être mise à jour.

Ce système informatique de frontière nécessite un faible coût en temps de calcul et peut augmenter la vitesse de PSE car il coupe les branches afin de ne pas perdre de temps à les explorer.

D'autres techniques sont utilisées pour réduire la taille des arbres et augmenter la vitesse. Par exemple, ils sont basés sur l'ordre dans lequel les décisions sont prises sur les objets, ou sur l'évaluation de chaque nœud, ou sur les attributs du problème qui permettent de tirer des conclusions sur certains objets.

Une autre méthode précise qui convient bien à ce problème de sac à dos est la programmation dynamique. Ce type d'algorithme est basé sur le fait qu'un problème peut être décomposé on sous-problèmes de même nature. L'idée est de fournir une expression de la valeur de la solution optimale sous la forme d'une fonction récursive basée sur la valeur de la solution optimale du sous-problème.

Dans le cas du sac à dos, en ne considérant que les i premiers objets, le problème peut être résolu de manière optimale grâce à la solution optimale obtenue enutilisant les i premiers objets. En effet, en faisant attention à la valeur du sac obtenu par H(i) et en ne considérant que les i premiers objets, les deux solutions pour H(i) sont :

Tout objet i est placé dans le sac, donc H(i) = H(i-1) + pi

Aucun objet i n'est mis dans le sac, donc H(i) = H(i-1)

Ceci suppose que la résolution du sous-problème correspondant à H(i-1) est connue. Enfin, pour n objets, à partir de H(0)=0, la solution optimale à ce problème de sac à dos correspondra à H(n). Dans le processus de calcul de la solution optimale, il est évidemment nécessaire de considérer la capacité maximale du colis, qui ne peut être dépassée. [12]

2- Programmation Dynamique

2.1- Introduction

La programmation dynamique est une technique mathématique conçue pour aider à prendre des décisions séquentielles indépendantes les unes des autres. Contrairement à la programmation linéaire, il n'y a pas de forme mathématique standard. Il s'agit d'une méthode de résolution dans laquelle les équations doivent être spécifiées en fonction du problème à résoudre.

2.2- Définition

La programmation dynamique est une technique algorithmique pour optimiser des sommes monotones croissantes de fonctions contraintes. Elle a été mentionnée pour la première fois

par Richard Bellman dans les années 1940. Elle convient aux problèmes d'optimisation où la fonction objectif est décrite comme une "somme de fonctions monotones croissantes de ressources".[13]

2.3- Historique

Dans les années 1940, Richard Bellman a utilisé le terme de programmation dynamique pour décrire le processus de résolution d'un problème, dans lequel les meilleures décisions sont trouvées une par une.on1953, il donne la définition moderne, dans laquelle les décisions à prendre sont ordonnées par sous-problèmes, et le domaine est alors reconnu par l'IEEE comme le sujet de l'analyse et de l'ingénierie des systèmes. La contribution de Bellman est connue sous le nom d'équation de Bellman, qui formule un problème d'optimisation sous forme récursive. Bellman a expliqué le concept du terme programmation dynamique dans son autobiographie, The Eye of the Hurricane : Autobiography 7. Ildit:

« *I spent the Fall quarter (of 1950) at RAND. My first taskwas to find a name for multistagedecisionprocesses. An interesting question is, Wheredid the name, dynamicprogramming, come from ? The 1950s were not good years for mathematicalresearch. Wehad a veryinterestinggentleman in Washington named Wilson. He wasSecretary of Defense, and heactuallyhad a pathologicalfear and hatred of the wordresearch. I'm not using the termlightly;I'musingitprecisely. His face wouldsuffuse, hewouldturnred, and hewouldget violent if people used the termresearch in hispresence. You can imagine how hefelt, then, about the termmathematical. The RAND Corporation wasemployed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the factthat I wasreallydoingmathematicsinside the RAND Corporation. Whattitle, whatname, could I choose? In the first place I wasinterested in planning, in decisionmaking, in thinking. But planning, is not a good word for variousreasons. I decidedtherefore to use the word "programming". I wanted to getacross the ideathatthiswasdynamic, thiswasmultistage, thiswastime-varying. I thought,let'skilltwobirdswith one stone. Let'stake a wordthat has an absolutelyprecisemeaning, namelydynamic, in the classicalphysicalsense. It also has a veryinterestingproperty as an adjective, and thatit's impossible to use the worddynamic in a pejorativesense. Try thinking of some combination thatwillpossiblygiveit a pejorativemeaning. It's impossible. Thus, I thoughtdynamicprogrammingwas a good name. It wassomething not even a Congressmancouldobject to. So I usedit as an umbrella for myactivities.* »

Par conséquent, Bellman a choisi l'adjectif dynamique pour souligner l'aspect temporel du problème, et parce que le mot était impressionnant. Le terme dispatch fait référence à l'utilisation d'une méthode pour trouver la meilleure dispatch au sens militaire : dispatch ou logistique. De plus, il est utilisé au même titre que les termes programmation linéaire et programmation mathématique, et est synonyme d'optimisation mathématique. Mais cette interprétation est controversée. Selon le livre de Russell et Norvig "ArtificialIntelligence: A Modern Approach" , l'explication ci-dessus n'est pas tout à fait correcte, car le premier article de Bellman utilisant le terme programmation dynamique est apparu on 1952, avant que Wilson ne devienne secrétaire à la Défense on 1953. Dansuneconférence de Harold J.

Kushner 11, il a parlé de Berman : « On the other hand, when I askedhim the same question, herepliedthathewastrying to upstageDantzig'slinearprogramming by adding dynamic. Perhapsboth motivations weretrue. » [14]

2.4- Principe

La programmation dynamique (dynamic programing on anglais ou encore optimisation dynamique) est une technique d'optimisation d'algorithmes qui vise à éviter de recalculer des sous-problèmes enstockant les résultats en mémoire. L'idée est simple, mais le gain en complexité temporelle est considérable, et cette technique est très largement utilisée dans de nombreux algorithmes. Cette technique de programmation suit le principe d'optimalité de Bellman, selon lequel la solution optimale à un problème peut être calculée à partir des solutions optimales aux sous-problèmes (Richard Bellman est l'un des fondateurs de la programmation dynamique). Enfait, vous utilisez souvent cette optimisation sans même vous on rendre compte. Si je vous demande de calculer le résultat de "1 + 1 + 1 + 1 + 1 + 1 + 1 + 1", il vous faudra un certain temps pour trouver la somme des termes, mais si j'ajoute "+ 1" à la fin de ce calcul, vous pouvez me répondre instantanément le résultat de la nouvelle expression. Vous avez naturellement stocké en mémoire le résultat de l'expression originale, que vous avez ensuite réutilisé pour trouver le résultat d'une nouvelle expression sans avoir besoin de refaire le calcul en entier.

2.5- Algorithme Programmation dynamique

```
INITIALISATION :
                    f(d,i)==0,d=0,…… ,c, i=0,……..,nfd,i:=0,d=0,…… ,c, i=0,……..,n
VARIABLE : OBJ ensemble des objets ordonné de taille N contienne
                                                    pi ωiPi ωi
DEBUT
 POUR (TOUTS LES OBJETS I = 0 ….. N) FAIRE
 DEBUT
  POUR (D = 0,… …,C) FAIRE
  DEBUT
                                    δ[d]:=d .piωi−f(d,i−1)δd:=d .piωi−f(d,i−1)
     ……
                δ[d] ≥0 ∀d apers l'orde d'ensemble OBJδd ≥0 ∀d apers l'orde d'ensemble OBJ
  FIN POUR
  POUR (R = 0,……,
                                                    ωi−1ωi−1
  ) FAIRE
  DEBUT
                d0:=r d0 capacité initiale de séquenced0:=r d0 capacité initiale de séquence
   MISE A JOUR :
   d :=r+ ωi          d la capacité apers la mise a jourd :=r+ ωi          d la capacité apers la mise a jour
      k :=1              k la taille d'ensemble OBJk :=1              k la taille d'ensemble OBJ
   SUPPRIMER S
   TANTQUE
                                                    (d ≤c)(d ≤c)
   FAIRE
   DEBUT
      SI (
                                    f(d,i−1)<f(d0,i−1)+k pifd,i−1<fd0,i−1+k pi
      ) ALORS
      DEBUT
          f(d,i):=f(d0,i−1)+k pi  ajouter nouveaux objetfd,i:=fd0,i−1+k pi  ajouter nouveaux objet
          INSERER D DANS S
                                                    d :=d+ ωid :=d+ ωi
          SI
                                                    (k < bi)(k < bi)
          ALORS RETOUR OU PAIRE PEUT ETRE SOLUTION POSSIBLE
      FIN SI
  FIN POUR
FIN.
```

Figure 03 : Algorithmique: Algorithmique de Programmation Dynamique [15]

2.6- Exemple

Le problème du plus court chemin (algorithme de Bellman-Ford et algorithme de Floyd-Warshall);

Problème d'allocation des ressources. Cela comprend (par exemple) l'attribution de m skis à n skieurs (m > n), tout enminimisant la différence de taille entre les skis et les skieurs. L'optimalité de la sous-structure (si la distribution est optimale, alors tout sous-ensemble de surfeurs et de skieurs est optimal) la rend facile à manipuler par programmation dynamique ;

Le problème du sac à dos est un problème de recherche opérationnelle classique, qui est NP-difficile, mais qui est résolu de manière pseudo-polynomiale à l'aide d'un algorithme de programmation dynamique ;

changement on général;

Trouver la plus longue sous-suite strictement croissante dans une suite de nombres ;

Problèmes d'algorithme de texte, tels que le calcul de la plus longue sous-séquence commune entre deux chaînes, le calcul de la distance de Levenshtein ou l'alignement de séquences (par exemple enutilisant l'algorithme Smith-Waterman) ;

L'algorithme CYK est un algorithme analytique ;

La méthode la plus largement utilisée pour résoudre le problème de détection des points d'arrêt est la programmation dynamique.

2.7- Avantages

L'un des principaux avantages de l'utilisation de la programmation dynamique est qu'elle accélère le traitement car une référence précédemment calculée est utilisée. Comme il s'agit d'une technique de programmation récursive, elle réduit le nombre de lignes de code dans un programme.

2.8- Inconvénients

Si vous avez une mémoire limitée pour exécuter votre code et que la vitesse de traitement n'est pas un problème, vous pouvez utiliser la récursivité. Par exemple, si vous développez une application mobile, la mémoire est très limitée pour exécuter l'application.

Si vous souhaitez que le programme s'exécute plus rapidement et n'ait aucune restriction de mémoire, il est préférable d'utiliser la programmation dynamique.

3- Conclusion

La programmation dynamique est séduisante car son formalisme est assez générique, ce qui laisse libre cours à de multiples variantes et à la résolution de problèmes assez divers, qu'ils soient déterministes ou non. Cependant, seule la catégorie des problèmes d'optimisations séquentielles est concernée et la vérification de l'applicabilité du principe d'optimalité est indispensable avant toute utilisation.

Chapitre 03 :
Implémentation et Evaluation des Résultats

CHAPITR 03: Implémentation et Evaluation des résultats

1- Implémentation et Evaluation des Résultats

1.1- Introduction

La programmation dynamique est l'une des plus anciennes méthodes d'optimisation combinatoire. Elle a été utilisée très tôt notamment pour des problèmes de type sac à dos. Après avoir été longtemps négligée, plusieurs nouveaux aspects sont récemment apparus dans cedomaine.

Dans ce chapitre nous réalisons une évaluation de cette technique.Pour appliquer notre travail nous utilisons l'environnement suivant :

1.2- Matériel physique

Nous utilisons une machine de type Dell inspirons 15 avec laconfiguration suivante:

CPU : Intel Core ™ i3 1.72GHZ 1.76GHZ

RAM : 4GB SODIMM Vitesse de 1600 MHZ

Stockage : HDD 500gb.

Carte graphique : Intel (R) graphics 4400

1.3- Les Outils logiciels

Java est un langage de programmation orienté objet, développé par Sun Microsystems en 1995, et racheté depuis par Oracle. Le principal avantage de Java est son interopérabilité : la technologie fonctionne aussi bien sur Windows que sur Mac ou bien Linux. Ainsi que sur une panoplie d'appareils : centres de données, ordinateur, téléphone mobile, lecteur Blu-ray, périphériques TV, consoles de jeux, appareils connectés... Un autre avantage est son caractère universel : le même système peut être utilisé pour une grande variété d'applications. Le langage Java est basé sur le C++, mais avec une approche simplifiée et des fonctionnalités plus avancées. [16]

JavaFX est une plate-forme de développement logiciel conçue pour créer et déployer des applications de bureau et Web accessibles via une variété d'appareils, tels que des ordinateurs de bureau et des navigateurs sur macOS, Linux et Microsoft Windows, ainsi que des appareils Android et iOS.

La plate-forme, dans son ensemble, permet aux développeurs de logiciels de concevoir, créer, tester et déboguer leurs applications sur diverses plates-formes virtuelles avant de les déployer pour garantir leur compatibilité.

Le langage de script de JavaFX fonctionne parfaitement pour les développeurs qui cherchent à gérer des graphiques lourds et des interfaces utilisateur riches car il utilise le pipeline graphique Prism pour rendre le travail. Il contient des bibliothèques graphiques riches qui permettent aux développeurs de créer des applications d'interface utilisateur graphique (GUI) et des graphestrès organisés. [17]

NetBeans est un environnement de développement intégré (EDI), placé on open source par Sun on juin 2000 sous licence CDDL et GPLv2 (Common Développement and Distribution License). Enplus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur on couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçuen Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac Os X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développement Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java). L'IDE NetBeans s'appuie sur cette plateforme.

L'IDE Netbeans s'enrichit à l'aide de plugins. [18]

1.4- Scénarios appliqués

Le problème du sac à dos unidimensionnel zéro-un est défini par :

L'objectif

$$\max(V) = \sum_{i=1}^{n} V_i X_i \quad (1)$$

$$\sum_{i=1}^{n} P_i X_i \leq P_{sac} \quad (2)$$

Tel que :

- $X_i = 0,1\ (i = 1, \ldots . n)$une valeur binaire égale à 0 si l'objet n'est pas dans le sac et inversement.
- V_i la valeur de chaque objet
- P_i le Poids de chaque objet

1.5- Présentation de scénario

Pour l'évaluation de cette approche, on doit définir des scénarios bien précis comme suit :

Etude N°1

Dans cette étude, nous comparons la programmation dynamique avec l'heuristique présentée précédemment.

Scénario N°1

Dans ce scénario, la saisie des objets se fait manuellement. Nous prenons la situation suivante comme exemple: Un campeur, dont le sac a une capacité w=20kg, doit choisir parmi ces objets :

Objet	Poids	utilité
Tente	11	20
Gourde	7	10
Duvet	5	25
Vêtement	5	11
Réchaud	4	5
Nourriture	3	50
Boite de pharmacie	3	15
Anti-moustique	2	12
Lampe	2	6
Cape	2	5
Gamelle	2	4
Couteau	1	30

Table 03 : Exemple Objet, Poids et Utilité

Dans le scénario ci-dessus, le campeur prendrait le couteau et le duvet et la tente ce qui représente un poids de 20 pour une utilité de 125 avec la nourriture.

Ou alors le campeur prendrait les objets les moins lourds. Comme le couteau, l'anti-moustique, la lampe, la cape, la gamelle, la nourriture, la pharmacie et le réchaud ce qui représente un poids de 19 pour une utilité de 127.

Ce scénario nous permet de voir l'efficacité de cette approche pour faire les bons choix.

Scénario N°2

Dans ce scénario, nous utilisons un jeu de données plus large généré aléatoirement. L'utilisateur doit préciser la capacité du sac, le nombre d'objets ainsi que le maximum et le minimum des poids des objets et leurs valeurs.

Etude N°2

Dans cette étude, nous comparons la programmation dynamique avec un benchmark et l'heuristique. Le scénario appliqué est choisi par l'utilisateur parmi les scénarios du benchmark.

1.6- Présentation de pseudo Code de l'algorithme.

Algorithme Programmation dynamique pour le Problème de Sac a dos

Enter : PO Poids d'objets Type Tableaux
VO valeurs d'objets Type Tableaux
M poids de sac Type entier
N nombre d'objetsType entier
Sortie : ValMax valeur maximal Type entier et les Objets Sélectionné

```
1   variable : B  Type Tableaux de deux dimensions de Taille M+1,N+1
2   Début
3      init B avec 0
4      pour (i :0 ,i<=N, i = i+1)
5      Début
6                  pour (j:0 ,j<=M, j = j+1)
7                  Début
8                     B[i][j] = B[i - 1][j];
9                     Si ((j>=W[i-1]) et (B[i][j] < B[i - 1][j - W[i - 1]] + V[i - 1]))
10                    Début
11                          B[i][j] = B[i - 1][j - PO[i - 1]] + VO[i - 1];
12                    Fin Si
13                Fin Pour
14     Fin Pour
15  ValMax=  B[N,M]
16  écrire( 'la valeur maximal : ',ValMax)
17  écrire( 'les objets sélectionné sont : ')
18  Tantque (n !=0)
19  Début
20             Si (B[n][M] !=  B[n - 1][M])
21             Début
23                écrire( 'objet : ', N , 'poids de ',PO[N], 'valeur de ',VO[N]) ;
24                M = M - PO[n-1];
25             Fin Si
26  Fin TantQue
27  Fin.
```

1.7- Présentation de l'application.

Voici l'interface principale de notre application programmée en Java. Cette interface est réalisée à l'aide de la bibliothèque JavaFX.

Figure 04 : La première interface de l'application

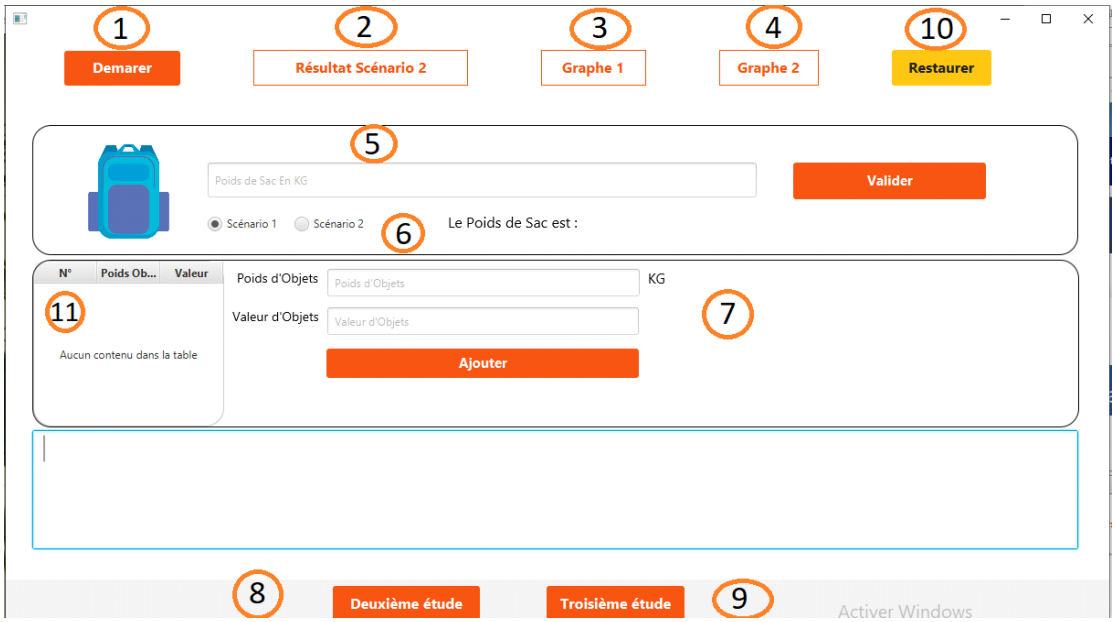

Figure 05 : L'interface principale de l'application

1)-Bouton pour démarrer l'opération.

2)-Bouton pour voir les résultats de scénario 2.

3)-pour voir le graphe de Temps d'exécution par rapport le nombre d'objets.

4)-pour voir le graphe de valeur Obtenu par rapport le nombre d'objets.

5)- Zone de Texte pour préciser la taille de sac à dos et le Bouton pour de valider.

6)-Case à cocher scénario.

7)-deux Zone de Texte, un pour le poids d'objet et l'autre pour la valeur et le Bouton pour ajouter objets a liste et le tableau

8)-Bouton pour aller à Etude 2.

9)-Bouton pour aller à Etude 3.

10)-Bouton pour remettre à Zéro l'application pour l'utiliser une autre fois.

11)- Un tableau pour organiser les objets

i	M=1		M=2		M=3		M=4		M=5		M=6		M=7		M=8		M=9		M=10		M=11	
	P	V	P	V	P	V	P	V	P	V	P	V	P	V	P	V	P	V	P	V	P	V
1	7	10	5	25	5	25	4	5	3	50	3	50	2	12	2	12	2	12	2	12	1	30
2			12	35	10	36	5	25	7	55	6	65	3	50	3	50	3	50	3	50	3	50
3					17	46	9	30	8	75	8	75	5	62	5	62	5	62	5	62	4	80
4							10	36	12	80	11	90	6	65	6	65	6	65	6	65	6	92
5							14	41	13	86	15	95	8	77	7	68	7	68	7	68	7	95
6							17	46	17	91	16	101	10	87	8	77	8	77	8	77	8	98
7									20	96	20	106	11	90	10	87	10	87	10	87	9	107
8													13	102	11	90	11	90	11	90	11	117
9													17	107	12	93	12	93	12	93	12	120
10													18	113	13	102	13	102	13	102	13	123
11															15	108	15	108	15	108	14	132
12															18	113	17	113	17	113	16	138
13																	18	113	18	113	18	143
14																	20	119	19	117	19	143
15																			20	119	20	147
16																						
17																						

Figure 06 : Résultat obtenu par Scénario N°1

1.8- Résultats obtenus

Résultat Scénario N°1

Le résultat obtenu pour Scénario N°1 est comme suit :

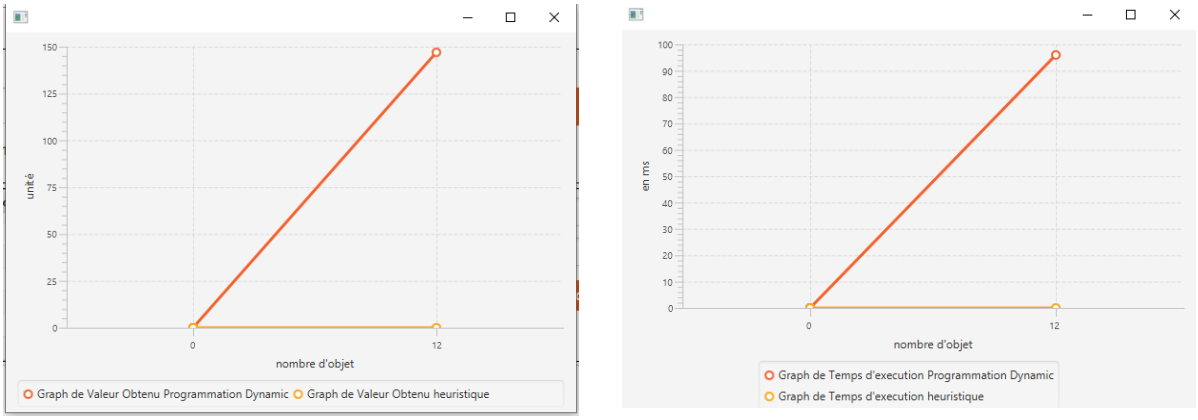

Figure 07 : Les Graphes de valeur Obtenu et temps d'exécution de la programmation dynamique

Les objets qui ont été sélectionnés par la solution optimale sont comme suit :

- Objet N°1 Poids= 11 et Valeur=20 qui est la **tente**
- Objet N°2 Poids= 7 et Valeur=10 qui est la **gourde**
- Objet N°3 Poids= 5 et Valeur=25 qui est le ***duvet***
- Objet N°4 Poids= 5 et Valeur=25 qui est le ***réchaud***
- Objet N°5 Poids = 3 et Valeur = 50 qui est la **nourriture**
- Objet N°6 Poids= 3 et Valeur = 15 qui est la **boite de pharmacie**
- Objet N°7 Poids= 2 Valeur = 12 qui est **l'anti-moustique**
- Objet N°8 Poids= 2 et Valeur = 6 qui est **la lampe**
- Objet N°9 Poids= 2 et Valeur = 5 qui est **la cape**
- Objet N°10 Poids = 2 et Valeur = 4 qui est **la gamelle**
- Objet N°11 Poids = 1 et Valeur = 30 qui est **le couteau**

La solution trouvée est la meilleure possible.

Résultat Scénario N°2
Dans ce scénario nous prenons la taille du sac égale à 80 Kg.

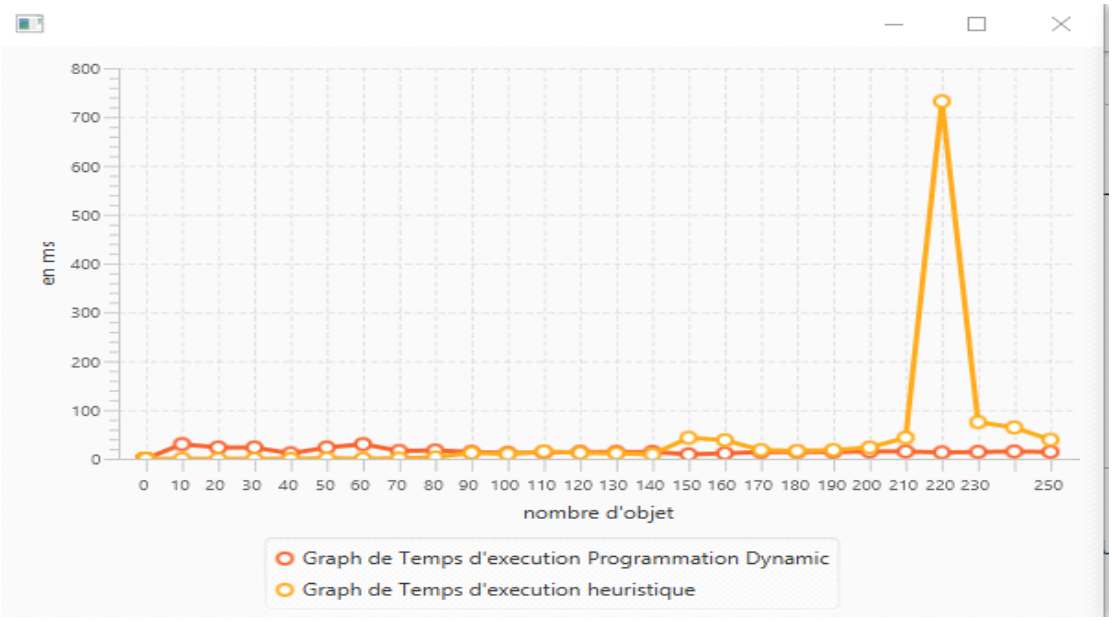

Figure 08 : Graph de temps d'exécution Programmation Dynamique

On peut constater que le temps d'exécution est très court. Il est légèrement supérieur à celui de l'heuristique avant 90 objets, puis les deux sont égaux jusqu'à 200 objets et finalement la programmation dynamique prend moins de temps d'exécution que l'heuristique jusqu'à 250 objets.

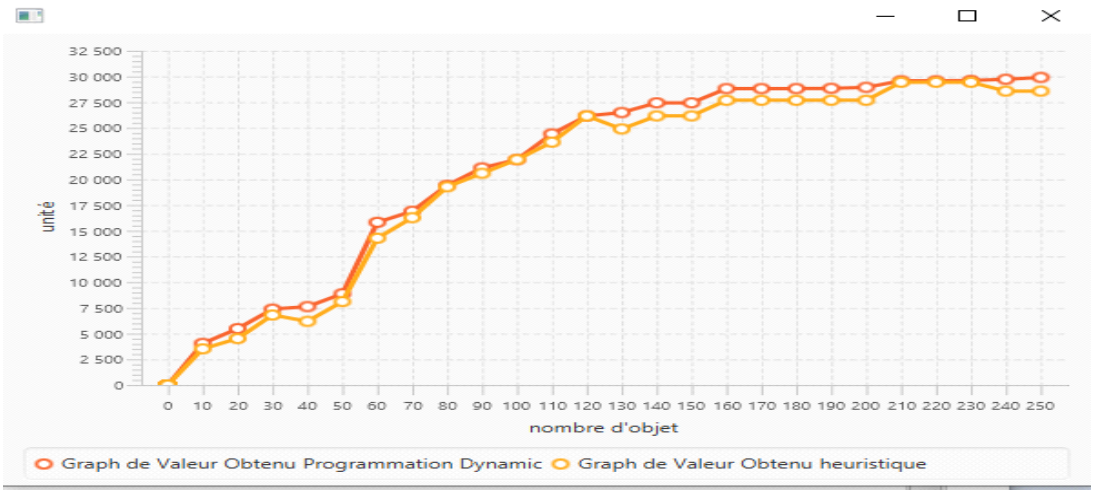

Figure 09 : Graph de valeur Obtenu Programmation Dynamique

Cette figure montre l'efficacité de la programmation dynamique qui donne un résultat meilleur que celui de l'heuristique.

Etude N°2

Dans cette étude, nous proposons un benchmark pour des instances petites et larges du problème de sac à dos. L'utilisateur choisi le scénario qu'il souhaite appliquer. [19]

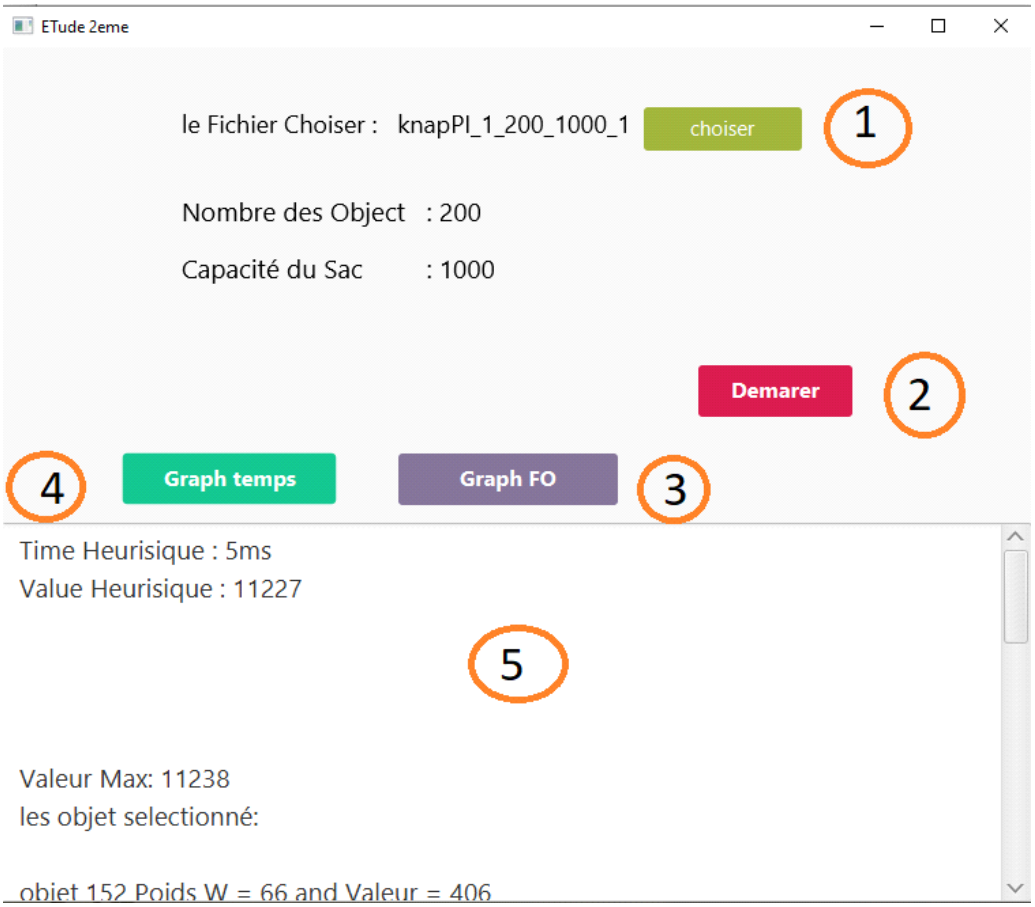

Figure 10 : L'interface de deuxième étude

1)-Bouton pour chois le cas existe (petites et larges instances).

2)-Bouton pour démarrer l'opération.

3)- Bouton pour voir le graphe de Temps d'exécution de la programmation dynamique et l'heuristique.

4)-pour voir le graphe de valeur Obtenu de la programmation dynamique et l'heuristique et Benchmark.

5)-Espace pour voir les objets sélectionnés par programmation dynamique.

La Figure ci-dessous, nous montre que le temps d'exécution la programmation dynamique et plus grand que celui de l'heuristique car l'algorithme de la programmation dynamique est plus

complexe par rapport l'heuristique.

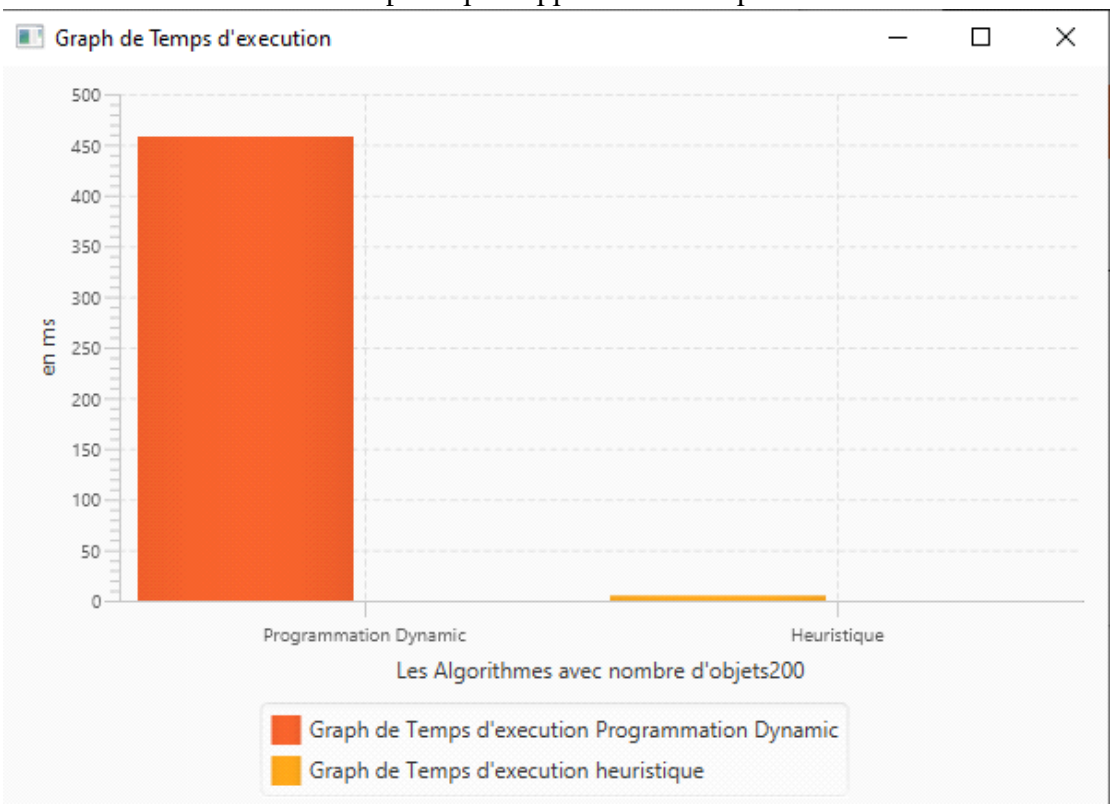

Figure 11 : Graph de temps d'exécution Programmation Dynamique

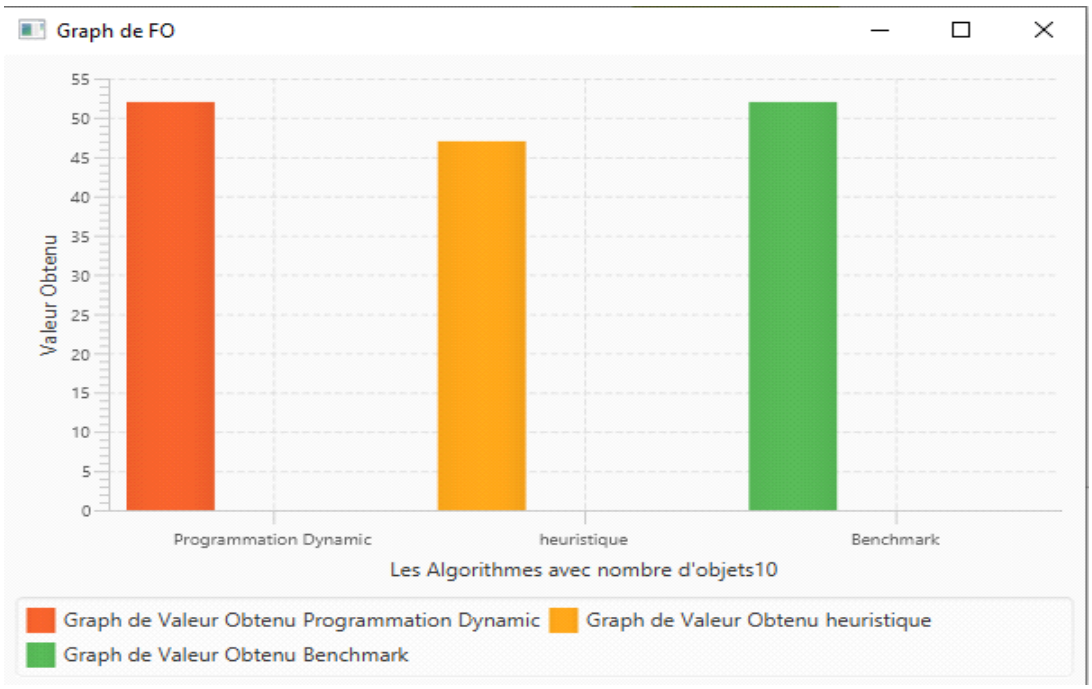

Figure 12 : Graph de Valeur Obtenu Programmation Dynamique

La figure ci-dessus, nous montre que la programmation dynamique et le benchmark ont la solution optimale qui est meilleure que celle de l'heuristique.

2- Conclusion

D’après l'étude menée, la programmation dynamique s'avère meilleure avec une durée d’exécution courte. Cependant, sa consommation de l'espace mémoire dans le cas où la taille du sac est très grande ainsi que le nombre d'objets représente un inconvénient non négligeable.

Conclusion Général

Conclusion Général

La programmation dynamique est séduisante car son formalisme est assez générique, ce qui laisse libre cours à de multiples variantes et à la résolution de problèmes assez divers, qu'ils soient déterministes ou non. Cependant, seule la catégorie des problèmes d'optimisations séquentielles est concernée et la vérification de l'applicabilité du principe d'optimalité est indispensable avant toute utilisation de cette méthode.

L'approche de programmation dynamique des contraintes de sac à dos n'est certainement pas toujours appropriée. Si les instances sont très grandes, alors le temps et les besoins en espace peuvent être prohibitifs.

Comme perspective à ce travail, une étude plus élargie aux différentes variantes du problème de sac à dos peut être envisagée, et ce dans le but d'avoir une évaluation de l'approche programmation dynamique plus complète.

Liste des Tables

Liste des Figures

Résumé

Mot clés : Sac à dos, Méthodes exactes, programmation dynamique, Optimisation.

Le problème du sac à dos est un cas classique des problèmes d'optimisation combinatoire. Plusieurs méthodes sont utilisées pour le résoudre, dont une partie sont approchées et d'autres exactes. La caractéristique d'optimalité qui est spécifique à sa structure lui permet d'être résolu par la programmation dynamique. Cette dernière est une méthode exacte efficace pour des problèmes représentant une telle structure. Ce travail s'intègre dans ce cadre pour évaluer cette efficacité en termes de fonction objectif et de temps.

الملخص

مشكلة حقيبة الظهر هي حالة كلاسيكية لمشاكل التحسين التوافقي. يتم استخدام العديد من الطرق لحلها، يتم التعامل مع بعضها والبعض الآخر دقيق. تسمح خاصية الأمثلية الخاصة بهيكلها بحلها عن طريق البرمجة الديناميكية. هذا الأخير هو طريقة دقيقة فعالة للمشاكل التي تمثل مثل هذا الهيكل. هذا العمل هو جزء من هذا الإطار لتقييم هذه الفعالية من حيث الوظيفة الموضوعية والوقت.

Abstract

The knapsack problem is a classic case of combinatorial optimization problems. Several methods are used to solve it, some of which are approached and others are exact. The optimality characteristic that is specific to its structure allows it to be solved by dynamic programming. The latter is an effective exact method for problems representing such a structure. This work is part of this framework to evaluate this effectiveness in terms of objective function and time.

Liste des abréviations

KP : problème de sac à dos

PD: programmation dynamique

RO : recherche opérationnelle

Références

1- Larousse 3 volumes, Paris 1966.
2- La recherche opérationnelle", Que sais-je ? PUF, Vidal Cohen, 1995.
3- Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright et Paul E. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM Journal on Optimization, 9(1):112–147, 1998.
4- R. E. Bellman, «DynamicProgramming,» Princeton UniversityPress, 1957. S. Jacquin, Hybridation des métaheuristiques et de la programmation dynamique pour les problèmes d'optimisation mono et multi-objectif : Application à la production d'énergie, Lille, 2015

5- John Brockman et Steven Pinker (introduction) (préf. Richard Dawkins), Whatisyourdangerousidea ? : today'sleadingthinkers on the unthinkable, New York, Harper Perennial, 2007, 300 p.
6- A. Sbihi, “Les méthodes hybrides en optimisation combinatoire: algorithmes exacts et heuristiques”, Thèse de doctorat, CERMSEM, LaRIA, 2006.
7- I.Alaya, C.Solnon, &K.Ghedira, “Différentes stratégies phénoménales pour le sac à dos multidimensionnel“. Pages 151-160 of : Méthodologies et Heuristiques pour l'Optimisation des Systèmes Industriels (MHOSI 2005).
8- M.Elhavedh, “ La Contribution à la résolution du sac-à-dos à contraintes disjonctives“, Thèse de doctorat, UNIVERSITÉ DE PICARDIE JULES VERNE, France, 2009.
9- Fayard and G. Plateau. An algorithm for the solution of the 0-1 knapsackproblem. Computing, 28 :269 287, 1982.
10- Dynamic ProgrammingRevisited:ImprovingKnapsackAlgorithmsU. Pferschy, GrazReceived 1999 p 421-422

11- Yifang Li, YannickKergosien& Jean-Charles Billaut
12- S. Martello and P. Toth. Algorithms for knapsackproblems. Annals of DiscreteMathematics, 31 :70 79, 1987
13- John Brockman et Steven Pinker (introduction) (préf. Richard Dawkins), Whatisyourdangerousidea? : today'sleadingthinkers on the unthinkable, New York, Harper Perennial, 2007, 300 p
14- Richard Bellman, Eye of the Hurricane: An Autobiography , 1984, p. 159.
15- Dynamic ProgrammingRevisited:ImprovingKnapsackAlgorithmsU. Pferschy,GrazReceived 1999 p 421-422. M.Elhavedh, “ La Contribution à la résolution du sac-à-dos à contraintes disjonctives“,Thèse de doctorat, UNIVERSITÉ DE PICARDIE JULES VERNE, France, 2009

16- HYPERLINK https://www.futura-sciences.com/tech/definitions/informatique-java-485/ at 14/05/2022 10:45
17- HYPERLINK https://www.techopedia.com/definition/34417/javafx at 15/05/2022 13:22
18- HYPERLINK " https://www.techno-science.net/glossaire-definition/NetBeans.html" at 15/05/2022 15:45
19- Bennchmark :http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/çavient du papier : "Pisinger, D., Where are the hard knapsackproblems? Computers & Operations Research, 2005. 32(9): p. 2271-2284" at 18/05/2022 10:45