UNIVERSITE MOHAMED BOUDIAFDE M'SILA

Faculté des Mathématiques et de l'Informatique

Département de Mathématiques

MEMOIRE DE FIN D'ETUDE

Présenté pour l'obtention du Diplôme de **MASTER**

Domaine : Mathématiques et Informatique

Filière : Mathématiques

Option : Analyse mathématique et numérique

Par

Nasri Samira

Sujet

Représentation de quelques algorithmes pour les problèmes d'optimisation combinatoire

Devant le jury :

Mr. Selt Omar	M.C.A. Univ de M'sila	Encadreur
Mr. Mustapha Dilmi	M.C.B. Univ de M'sila	Président
Mr. Bachir Gagui	M .C.A. Univ de M'sila	Examinateur

Promotion : 2019 / 2020

Remerciements

Je remercie Dieu le tout puissant pour m'avoir donné toute cette force et ce courage pour faire aboutir ce travail.

Je tiens à remercier ma encadreur de mémoire Mr Omar Selt. Pour m'avoir soutenue et encouragée tout au long de la préparation de cette mémoire. Et pour m'avoir inspirée et guidée durant le cheminement de ce travail. Cette mémoire n'aurait pas vu le jour sans sa détermination à mener à bien ce projet.

Ma sincère reconnaissance à tous les membres du jury pour l'honneur qu'ils me font en acceptant de présider et examiner ce travail.

Mr Gagui Bachir

Mr Dilmi Mustapha

Je remercie également ceux qui m'ont aidé de près ou loin à réaliser ce travail.

DÉDICACES

Je dédie ce travail à mon mari qui m'a accompagné et soutenu durant ces années, ma famille qui m'a vraiment encouragé pour terminer ce travail, mon collègue. Je les remercie pour leurs encouragements durant toute la période d'élaboration de ce travail.

Table des matières

Introduction

L'optimisation combinatoire occupe une place trés 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 [PAP 82] et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire [RIB 94]. 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 des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [GAR 79]. Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA).

Ce mémoire est construit de trois chapitres:

Dans le premier chapitre, On commence par une rappels sur Problèmes d'optimisation Combinatoire: Optimisation combinatoire et aectation sous contraintes; Programmation mathématique.

Chapiter 2 : Présente Programme linéaire Programmation Linéaire en Nombres Entiers(PLNE); La Méthode de Dakin pour le PLNE général; Les algorithemes heuristiques.

Chapiter 3 C'est une partie purement pratique, elle met en oeuvre une technique de résolution La méthode du simplexe et un exemple.

Notations

PL	Programmation Linéaire
$PLNE$	Programmation Linéaire en Nombres Entiers
OC	*Optimisation* Combinatoir
$PASC$	*Probléme* d'affectation sous contraintes
$CSOP$	*Probléme* d'*Optimisation* sous contraintes
PM	*Probléme* Mathématique
$MCSP$	*Probléme* de contisfaction partielles(maximales)
IA	*Intelligen* Aritificielle
RO	*Recherche* Opérationalle
PMD	Programme Mathématique Discrette
CSP	*Contraint* de datisfaction *Probléme*

Chapitre 1

Optimisation Combinatoire et modélisations

1.1 Introduction

La programmation linéaire en nombres entiers est un outil puissant de modélisation : en et, de nombreux problèmes d'optimisation combinatoire peuvent être formulés comme des PLNE particuliers. Ainsi, un algorithme permettant de résoudre un PLNE permet de résoudre beaucoup de problèmes d'optimisation combinatoire. Il existe d'ailleurs de nombreux logiciels, appelés solveurs entiers, utilisés pour la résolution de problèmes d'optimisation combinatoire pratiques (industrie, transport,...). Malheureusement, les seuls algorithmes ($B\&B$) connus pour résoudre tous les PLNE sont exponentiels car ils consistent à énumérer un grand nombre de solutions. Nous verrons dans cette partie, comment l'algorithme $B\&B$ peut-être enrichi par un algorithme de coupes. D'autre part, il est souvent di-cile de trouver une formulation PLNE e-cace pour traiter problème d'optimisation combinatoire. Nous verrons qu'il est possible d'utiliser des formulation non compactes, c'est-à-dire possédant un nombre exponentiel de contraintes (Résolution par algorithme de coupes ou $B\&C$) ou de variables (Résolution par Génération de colonnes ou $B\&C$).

1.2 Problèmes d'optimisation Combinatoire

Un problème d'optimisation combinatoire (OC) consiste à déterminer un plus grand (petit) élément dans un ensemble n_i valué. En d'autres termes, étant donnés une famille F de sousensembles d'un ensemble n_i $E = \{e_1, ...e_n\}$ et un système de poids $w = (w(e_1), ..., w(e_n))$ associé aux éléments de E, un problème d'optimisation consiste à trouver un ensemble $F \in F$ de poids $w(F) = Pe \in Fw(e)$ maximum (ou minimum), $i.e$, max ou min $\{w(F)|F \in F\}$. La famille F représente donc les solutions du problème. Elle peut correspondre à un ensemble de très grande taille que l'on ne connaît que par des descriptions ou des propriétés théoriques qui ne permettent pas facilement son énumération. Une première formulation PLNE naïve est alors possible : il su-t de prendre une variable $0-1$ associée à chaque éléments de F et de poser le PLNE associé à ces variables. Le seul algorithme adaptée est alors l'énumération une à une des solutions. Comme ceci n'est pas envisageable, on abandonnera cette formulation. Cette idée néanmoins prouve que tout problème d'OC peut être formulé comme un PLNE. A l'opposée de cette idée, on peut vouloir associer une variable $0-1$ à chaque élément E : pour pouvoir alors dénir une formulation associée, il faut pouvoir dénir par des inégalités le fait que les solutions décrites par la variable doivent être dans l'ensemble F. On dit dans ce cas que la formulation associée est naturelle. Même si elle existe toujours théoriquemen, il n'est pas toujours simple de déterminer une telle formulation. Nous allons voir que cette description peut mener à plusieurs formulations PLNE bien diérentes qui méneront à des résolutions diérentes d'un même problème.

1.3 Les problèmes combinatoires

Parmi les défis au sens commun, il y a donc les problèmes combinatoires pour lesquels il est, a priori, impossible d'énumérer. On distingue toutefois les problèmes faciles des problèmes difficiles.

1.3.1 Problèmes faciles

Il en est ainsi pour les problèmes de toute petite taille (il faut énumérer !), pour les problèmes fortement contraints (il suffit d'énumérer), pour les problèmes faiblement contraints (une

heuristique type recuit simulé ou méthode tabou sera parfaite surtout dans un contexte industriel) et enfin pour les problèmes résolus polynomialement, encore faut-il le savoir !

1.3.2 Problèmes difficiles

Mon expérience pratique me permet d'affirmer qu'ils sont fréquents et que pour les résoudre il faut utiliser de multiples outils (méthodes sérielles, recuit simulé, méthodes arborescentes, programmation dynamique, algorithme A*....). Il faut ajouter que pour certains d'entre eux la méthode actuelle la plus efficace est de reproduire l'expertise humaine.

1.3.3 Les méthodes automatiques

(ça n'existe pas encore !) Il y a 40 ans, GOMORY découvrait ses fameuses coupes pour la Programmation Linéaire en Nombres Entiers (PLNE). Or, de trés nombreux problèmes combinatoires peuvent se modéliser par un PLNE. A l'époque, on a cru pouvoir résoudre les problèmes combinatoires. L'espoir a été déçu ! 40 ans après, on ne sait pas résoudre automatiquement les problèmes combinatoires. Mais il n'est pas exclu théoriquement que cela soit possible dans l'avenir. Cela revient à dire que les problèmes NP-difficiles pourraient être résolus "pratiquement" de façon polynomiale. Il faudrait une grande avancée algorithmique.

1.3.4 Les méthodes semi-automatiques

Modéliser un problème combinatoire ne sert à rien (1), si on ne décrit pas, en plus, son algorithme de résolution et plus particulièrement de bonnes évaluations, certains disent de bonnes contraintes. D'où l'idée de faire des langages de programmation adaptés au combinatoire et incluant des parcours arborescents avec des choix automatiques. L'avantage est de pouvoir, au moins pour le spécialiste d'un tel langage, programmer très vite. A mon avis, un tel outil reste ambitieux car on a des surcoûts liés à la rigidité d'un tel système et à la faiblesse de leurs structures de données. (1)En fait, tout modèle est une simplification de la réalité et, quand on modélise, il faut chercher à obtenir le modèle le plus représentatif que l'on puisse espérer résoudre de façon satisfaisante !

1.3.5 Les programmes spécifiques

Je citerai les méthodes arborescentes, les méthodes polyédrales et la programmation dynamique, mais aussi les heuristiques. L'expérience de résolution des problèmes combinatoires montre l'importance cruciale pour construire une méthodes efficaces des points suivants:

1. Une modélisation adapte.
2. La complexité des algorithmes et des structures de donnes.
3. La proximité de la machine (langage procédural, par exemple).
4. La proximité du problème (si on n'a pas d'excellentes évaluations, celles-ci sont inutiles).

Une soixantaine d'heures (cours et exercices inclus). Nous ne le prétendrons pas. Nous insisterons sur les idées principales et la présentation des algorithmes les plus fondamentaux. En effet ces algorithmes sont les outils de bases pour des méthodes plus élaborées. Nous objectifs sont de faire prendre conscience de la complexité des problèmes, du danger du combinatoire et de l'utilité des graphes pour modéliser. Espérons que cela vous évitera sur le terrain de concevoir de belles maquettes parfaites pour des exemples d'écoles de petites tailles mais inutilisables sur des problèmes réel.

1.4 Optimisation combinatoire et affectation sous contraintes

1.4.1 Optimisation combinatoire

Un problème d'optimisation combinatoire est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution $s \in X$ le nombre réel (ou entier) $f(s)$. Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une

solution $s^* \in X$ optimisant la valeur de la fonction de coût f. Une telle solution s^* s'appelle une solution optimale ou un optimum global.

Définition 1.4.1 *[PAP 82]. Une instance I d'un problème de minimisation est un couple (X, f) où $X \subseteq S$ est un ensemble fini de solutions admissibles, et f une fonction de coût (ou objectif) à minimiser $f : X \rightarrow R$. Le problème est de trouver $s^* \in X$ tel que $f(s^*) \leq f(s)$ pour tout éléments $\in X$. Notons que d'une manière similaire, on peut également définir les problèmes de maximisation en remplaçant simplement $\leq$ par $\geq$. L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique elle-même.*

1.4.2 Problème d'affectation sous contraintes

La définition générale de l'optimisation ne précise ni la forme des solutions de S, ni la façon de générer ces solutions (admissibles ou non admissibles). Nous nous intéressons en particulier à une classe importante de problèmes dont une solution peut être décrite explicitement par une affectation de valeurs à l'ensemble des variables du problème. Etant donné un ensemble fini $V = \{V_1, ..., V_n\}$ de variables et un ensemble $D = \{D_1, ..., D_n\}$ de domaines finis associés, une solution potentielle du problème (affectation) consiste à choisir pour chaque variable V_i $(1 \leq i \leq n)$ une valeur choisie dans son domaine D_i. L'ensemble S des solutions potentielles est donc représenté par le produit cartésien $D_1 \times ... \times D_n$ des domaines. On dispose en outre d'un ensemble $C = \{C_1, ..., C_p\}$ de contraintes : chaque contrainte C_j $(1 \leq j \leq p)$ est une relation sur un sous-ensemble V_j de V qui spécifie quelles combinaisons de valeurs sont compatibles pour les variables de V_j. Nous utilisons le terme informel de « problèmes d'affectation sous contraintes » (PASC) pour qualifier la classe des problèmes d'affectation utilisant la notion de contrainte (en étendant éventuellement la définition de contrainte présentée plus haut). Cette classe de problèmes inclut notamment les problèmes de satisfaction de contraintes (CSP pour Contraint Satisfaction Problèmes) [MAC 77, 87], les problèmes de satisfaction partielles (maximales) (MCSP) [FRE 92] et les problèmes d'optimisation sous contraintes (CSOP) [TSA 93]. Etant donné un triplet $< V, D, C >$, le problème CSP consiste à trouver une assignation qui satisfait toutes les

contraintes et le problème MCSP une assignation qui satisfait un nombre maximum de contraintes. Etant donné un quadruplé $< V, D, C, f >$, le problème CSOP consiste à trouver une assignation qui satisfait toutes les contraintes et qui minimise la fonction $f : S \rightarrow R$. Il existe de nombreux autres modèles comme les CSP flous, possibilistes et probabilistes [FAR 95] ainsi que les CSP values [SCH 97] qui peuvent être éventuellement inclus dans la classe PASC. Les problèmes d'affectation sous contraintes possèdent de très nombreuses applications pratiques concernant l'affectation de ressources, le groupement, la classification, la planification, l'emploi du temps et l'ordonnancement, dans des domaines très variés. Les PASC permettent également de modéliser facilement des problèmes de référence comme par exemple la k-coloration et la satisfiabilité.

1.5 Programmation mathématique

Définition 1.5.1 *Un Programme Mathématique (mathématical program), noté PM, est un problème d'optimisation sous contrainte (p) qui peut s'écrire de la façon suivante:*
Maximiser $f(x)$ *sous les contraintes* $g_i(x) \leq 0,\ \ i = 1, ..., m,\ \ x \in S$, *où*

- S *est une partie de* $\mathbb{R}^n$ *et est un vecteur appelé variable, ces composantes sont dites les inconnues du problème.*
- *la fonction* $f : S \rightarrow \mathbb{R}$ *est appelée fonction objective ou objectif (objective function).*
- *les fonctions*$g_i : S \rightarrow \mathbb{R}$, $i = 1, ..., m$ *forment des inégalités qui sont appelées les contraintes (constraint) du problème.*

On peut remarquer qu'un PM peut être une maximisation ou une minimisation (il suffit de poser la fonction $f' = -f$.

On appelle inégalités une contrainte $g_i(x) \leq 0$ ou $g_i(x) \geq 0$: en cas de présences des deux contraints $g_i(x) \leq 0$ et $g_i(x) \geq 0$, on parle alors d'égalité $g_i(x) = 0$.

Un vecteur $\bar{x}$ vérifiant les contraintes d'un PM est dit solution ou solution réalisable du PM. L'ensemble des solutions d'un PM forme un domaine de dénition. Le domaine de dénition d'un PM peut être : vide (dans ce cas, le problème n'admet pas de solutions), dans le cas contraire, le PM admet des solutions. Sous certaines conditions, il peut exister

des solutons x^* dites optimales, c'est-à-dire qui maximisent la fonction $f(x)$ sur toutes les solutions du PM.

Plusieurs cas de PM sont à mettre en évidence:

- – si l'ensemble S est continu, on parle de programme mathématique continu (continuous).
  - si l'ensemble S est discret (c'est-à-dire isomorphe à $\mathbb{N}^n$), on parle de programme mathématique discret (discrete) que l'on notera ici (PMD); on le dit également entier (integer program) si $S \subset \mathbb{N}^n$ ou même binaire si $S \subset \{0,1\}^n$. En fait tout PMD peut se ramener au cas d'un programme entier, voir même d'un programme binaire.
  - si certaines composantes du vecteur x solution prennent leurs valeurs dans un ensemble discret et les autres dans un ensemble continu, on le dit programme mathématique mixte (Mixed Integer Program ou MIP).

Dans le cas des programmes entiers (donc discrets également), on peut noter alors un (PMD) de la façon suivante :

Maximiser $f(x)$ sous les contraintes $g_i(x) \leq 0,\ \ i = 1, ..., m,\ x \in \mathbb{Z}^n$.

On désigne alors $x \in \mathbb{Z}^n$ comme étant la contrainte d'intégrité (ou d'entiéreté en Belgique ou d'intégralité au Québec) (integrity or integrality constraint).

On appelle relaxation le fait de "relâcher ", c'est-à-dire supprimer une contrainte du problème. Ainsi, un programme relaxé désignera un programme où l'on aura supprimé une ou plusieurs contraintes. On appelle relaxation continue le fait de "relâcher" les contraintes d'intégrité du problème. Par abus de langage, on appelle aussi souvent relaxation continue le fait de résoudre le programme que où l'on a relâché les contraintes d'intégrité (l'expression désigne même parfois la solution optimale obtenue).

Chapitre 2

Programation linéaire et heuristique

2.1 Programme linéaire

Si la fonction objective f et les fonctions contraintes g sont toutes linéaires : on parle de Programme linéaire discret (ou entier ou mixte) (on notera PLNE). Ce cas étant largement le plus utilisé et le plus étudié, on l'appelle parfois même simplement Programme entier (ou mixte) (Integer Program ou Mixed Integer Program) (MIP).

Maximiser $c_1^T x_1 + c_2^T x_2$sous les contraintes $A_1 x_1 + A_2 x_2 \leq b$, $x_1 \in \mathbb{R}^{n_2}$, $x_2 \in \mathbb{Z}^{n_2}$; où c_1, c_2 sont des vecteurs et A_1et A_2 des matrices avec x_1 partie continue de la solution et x_2 partie entière de la solution.

Un problème linéaire continu peut être résolu en temps polynomial (Khachiyan 1979). Il existe des algorithmes polynomiaux e-caces pour résoudre un programme linéaire comme ceux dits de points intérieurs initiés par Karmarkar (1984). Néanmoins l'algorithme du simplexe (Dantzig 1947) est le plus célèbre (et le plus efficace dans le cas général) des algorithmes de résolution, bien qu'il ne soit pas polynomial! L'algorithme du simplexe repose sur le fait qu'une solution optimale d'un programme linéaire peut être prise parmi les sommets du polyèdre de $\mathbb{R}^n$ déterminé par $Ax \leq b$.

En revanche, la PLNE est un problème NP-difficile. Il est facile de montrer que la PLNE est un problème NP-di-cile car de nombreux problèmes NP-difficiles peuvent être exprimés comme des PLNE.

Il existe de nombreux solveurs de PL: des solveurs commerciaux Cplex (IBM), Xpress, Gurobi (microsoft), et même Matlab ou Excel; des solveurs académiques Lp de COIN-OR, Soplex de la ZIB, et des solveurs libres comme Glpk (gnu).

Les meilleurs d'entre eux peuvent résoudre des PL jusqu'à 200000 variables et 200000 contraintes en quelques secondes.

En revanche, les solveurs entiers performants sont beaucoup moins performants: Ils sont en général liés aux solveurs PL: Glpk par exemple ne dépassent pas quelques 100 aine de variables et contraintes; les solveurs commerciaux Cplex ou Gurobi sont les plus performants (Xpress est un peu en-dessous) pouvant réussir parfois quelques milliers de variables/contraintes, un solveur "universitaire" les rattrape : SCIP de la ZIB.

Un des objectifs de ce cours est de comprendre comment et dans quels cas ces solveurs atteignent de telles capacités.

2.2 Programmation Linéaire en Nombres Entiers(PLNE)

Toutes les variables sont obligées de prendre des valeurs entières.

En général les coefficients $a_{i,j}$ sont aussi entiers, et donc, on peut se limiter à des constantes bj entières. Si les variables sont aussi contraintes d'être entre 0 et 1, elles n'ont queles deux valeurs possibles 0 et 1 et on parle de PL en $0-1$.

Maintenant on peut exprimer des contraintes telles $x_1 \geq 1$ ou $x_2 \geq 1$ par la contrainte $x_1 + x_2 \geq 1$ parceque des "solutions" comme $x_1 = x_2 = 0.5$ sont inter dites.

Le PL ordinaire avec les mêmes variables, contraintes et fonction objective s'appelle le PL relaxée du PLNE. Sa solution optimale donne une borne (supérieure pour un problème de maximisation) sur la solution optimale du PLNE. (Mais la solution optimale du PLNE peut être très différente ou même ne pas exister.)

2.2.1 Aspects Théoriques

Chercher une solution réalisable du PLNE équivaut à chercher un point dans le poly tope de solutions réalisables avec tous ses coordonnés entiers. Si le poly tope est longe té troi t il n'est pas évident qu'il contient de tels points. Décider si un PLNE possède des solutions

réalisables est un exemple d'un problème NP-complet une classe de problèmes pour les quels personne ne sait s'il existe ou non un algorithme efficace (opérant en temps polynomial).

Les meilleurs algorithmes connus sont capables de traiter des programmes avec quelques dizaines de variables (par comparaison ,on peut traiter des PL de plusieurs milliers de variables).Méthodes heuristiques.

2.2.2 Des cas où les deux programmes sont équivalents

Pour quelques types de PL ,on sait que(s'il existe une solution optimale), il existe une solution optimale entière, donc, le PLNE et le PL ont la même valeur optimale et c'est beaucoup plus vite de résoudre le PL, essentiellement des problèmes sur des réseaux des programmes où chaque variable est présente dans deux contraintes au maximum et avec des coefficients de ± 1.

2.2.3 Une classe de méthodes Séparation et Evaluation

On choisit une variable qui a une fourchette de valeurs possibles ,disons $[min, max]$, on construit deux nouveaux problèmes, un avec$[min, mi]$, et l'autre avec $[mi + 1, max]$ séparation.

On utilise la borne sur chaque problème fournie par son programme relaxé évaluation.

Si cette borne est inférieure (pour un problème de maximisation) à une solution déjà trouvée, terminer cette branche et bien sûr teminer la branche si le programme relaxé n'est pas faisable sinon continuer.

A chaque moment il ya un arbre de problèmes à résoudre et la solution optimale est ou la meilleure solution déjà trouvée ou la meilleure de celles des problèmes contenus dans l'arbre

2.2.4 La Méthode de Dakin pour le PLNE général

- – Parcourir l'arbre en profondeur (ce qui a une bonne chance de produire une solution réalisable assez vite).
  - – Choisir à chaque itération le problème possédant la meilleure solution de son programme relaxé.

– Séparer sur une variable dont la valeur (dans la solution optimale du programme relaxé) est la plus proche d'un entier min est cette valeur (arrondie)

2.2.5 La Méthode de Balas pour les PLNE en 0-1

•Considérer les cas où z (à minimiser) n'a pas de coefficients négatif set les variables sont triée sen ordre croissant de ces coefficients (pas de restriction tout programme peut être exprimé de cette

façon). Ici séparation consiste à fixer une variable soit à 0 soit à 1, Ce qui permet à simplifier énormément les sous-problèmes. Si un problème n'a pas de b_j négatif on a déjà une solution réalisable (en fixant toutes les variables restant à 0) et c'est la solution optimale de ce (sous-)problème. Parcourir l'arbre en profondeur.

Plusieurs règles d'élagage permettent des avoir qu'un problème donné est infaisable ou ne peut donner une solution meilleure d'une solution déjà trouvée.

2.3 Les algorithemes heuristiques

Un algorithme heuristique, ou plus simplement une heuristique, produit aussi une solution approxime la solution optimale toute fois, contrairement à un algorithme d'aproximation il n'est pas nécessairement possible de borner de façon exacte la qualité de la solution produite relativement à la solution optimale.

Ainsi, il peut meme tout à fait incapable de produire une solution.

Chapitre 3

La méthode du simplexe

3.1 Principe

Lorsque nous sommes en présence de plus de deux produits, la méthode du simplexe est la seule méthode permettant de trouver la combinaison de produits qui rend optimal la fonction économique.

Le principe de résolution nécessite un certain nombre d'étapes contenu au travers de l'algorithme du simplexe dont la démarche est la suivante : (voir schéma).

3.2 Application

La résolution par l'algorithme du simplex se déroule selon 8 étapes avant un nouveau passage.

1^{er} étape : Ecrire le systeme sous forme standard

ll s'agit convertir le programme établi sous forme canonique (systeéme d'inéquation) sous la forme standard (systeme d'équation avec variable d'écarts). Les variables d'écart introduites au cours de cette transformation représentent les contraintes techniques et commerciales

disponibles qu'il convient de saturer.

Forme canonique

$$\begin{cases} 3x + 2y \leq 1800 \\ x \leq 400 \\ y \leq 600 \\ x \geq 0 \text{ et } y \geq 0 \\ MaxB = 30x + 50y \end{cases}$$

Forme standard

e_1, e_2, e_3 représentant les variables d'écart

$$\begin{cases} 3x + 2y + e_1 = 1800 \\ x + e_2 = 400 \\ y + e_3 = 600 \\ \\ \\ 30x + 50y \end{cases}$$

2ème étape : Construire le premier tableau correspondant a la forme standard

	x	y	e_1	e_2	e_3	
e_1	3	2	1	0	0	1800
e_2	1	0	0	1	0	400
e_3	0	1	0	0	1	600
max	30	50	0	0	0	0

où $\{e_1, e_2, e_3\}$ sont les valeur de base; Coefficient E_{ij} =$\{3, 2, 1, 0, 0, ...0, 1\}$; max=$\{30, 50, 0, 0, 0,\}$ fonction économique; Valeur solutions $\{1800, 400, 600, 0\}$.

3eme étape: Choisir les variables a introduire dans la base. Pour cela choisir le coefficient le plus fort de la fonction économique

Le coefficient de la fonction économique (MAX) est 50. Ainsi il s'agit de la variable y qui rentre en base.

4ème étape: Choisir la variable a enlever de la base (rapport : second membres / coefficient de la variable choisie). Retenir le plus faible.

Le second membre, nous retenons la valeur la plus faible du rapport second membre/coefficient de la variable choisie. Ainsi la variable e; est la variable a enlever de la base.

5ème étape : Encadrer le pivot.

6ème étape : Multiplier la ligne du pivot par le rapport : 1/ valeur du pivot (ou diviser la ligne du pivot par le pivot)

7ème étape : Calculer les valeurs des autres lignes

$$E'_{ij} = E_{ij} - [(A_{ij}/\text{Pivot}) \times \text{Ligne du pivot}]$$

où $E'_{ij} = (1/\text{Pivot})$.

Cette opération consiste a transformer E_{ij} des autres lignes en E'_{ij}, nous effectuons un calcul maitriciel.

$1^{ère}$ ligne
$3 = 3 - [(2/1) \times 0]$
$0 = 2 - [(2/1) \times 1]$
$1 = 1 - [(2/1) \times 0]$
$0 = 0 - [(2/1) \times 0]$
$-2 = 0 - [(2/1) \times 1]$
$600 = 1800 - [(2/1 \times 600]$

$2^{ème}$ ligne
$1 = 1 - [(0/1 \times 0]$
$0 = 0 - [(0/1 \times 1]$
$0 = 0 - [(0/1 \times 0]$
$1 = 1 - [(0/1 \times 0]$
$0 = 0 - [(0/1 \times 1]$
$400 = 400 - [(0/1 \times 600]$

$4^{ème}$ ligne
$30 = 30 - [(50/1) \times 0]$
$0 = 50 - [(50/1)x1]$
$0 = 0 - [(50/1) \times 0]$
$0 = 0 - [(50/1) \times 0]$
$-50 = 0 - [(50/1)x1]$
$-30000 = 0 - [(50/1 \times 600]$

		x	y	e_1	e_2	e_3	
$1^{ère}$ ligne	e_1	3	2	1	0	0	1800
$2^{ème}$ ligne	e_2	1	0	0	1	0	400
ligne du Pivot	y	0	1	0	0	1	600
$4^{ème}$ ligne	max	30	50	0	0	0	0

$8^{ème}$ étape : Les coefficients de la fonction économique sont ils tous nuls ou négatifs ? (si oui

Les coefficients de la fonction économique ne sont pas tous nuls ou négatifs (30) il convient d'effectuer un nouveau passage.nous sommes a l'optimum, si non nous effectuons un nouveau passage).

3.2.1 Algorithme du simplexe

1. Ecrire le systeme sous forme standard.
2. Construire le premier tableau correspondant a la forme standard.
   Si Les coefficients de la fonction économique ils sont tous nuls ou négatifs on fin l'algorithme sinon
3. Choisir les variables a introduire dans la base : choisir le coefficient le plus fort de la fonction économique.
4. Choisir les variables a enlever de la base : (Rapport : second membres / coefficient de la variable choisie).
5. Encadrer le pivot.
6. Multiplier la ligne du pivot par le rapport : 1/ valeur du pivot.
7. Calculer les valeurs des autres lignes: $E'_{ij} = E_{ij} - [(A_{ij}/\text{Pivot}) \times \text{Ligne du pivot}]$
   E_{ij} : coefficient a transformer
8. A_{ij} : coefficient de la colonne du pivot (i ligne, j colonne)

Nouveau passage:

Choisir les variables a introduire dans la base. Pour cela choisir le coefficient le plus fort de la fonction économique.

Le coefficient de la fonction économique (MAX) est 30. Ainsi il s'agit de la variable x qui rentre en base.

Choisir la variable a enlever de la base (rapport : second membres/coefficient de la variable choisie). Retenir le plus faible.

Le second membre, nous retenons la valeur la plus faible du rapport second membre /coefficient de la variable choisie. Ainsi la variable e, est la variable a enlever de la base.

- – Le pivot est égal a 3.
  – Multiplier la ligne du pivot par 1/3 (ou diviser la ligne du pivot par le pivot : 3)
  – Calculer les autres de valeur des lignes.

$2^{ère}$ ligne

$0 = 1 - [(1/3) \times 3]$

$0 = 0 - [(1/3) \times 0]$

$-1/3 = 0 - [(1/3) \times 1]$

$1 = 1 - [(1/3) \times 0]$

$-2/3 = 0 - [(1/3) \times -2]$

$200 = 400 - [(1/3 \times 600]$

$3^{ème}$ ligne

$0 = 0 - [(0/3 \times 3]$

$1 = 1 - [(0/3 \times 0]$

$0 = 0 - [(0/3 \times 1]$

$0 = 0 - [(0/3 \times 0]$

$1 = 1 - [(0/3 \times -2]$

$600 = 600 - [(0/3 \times 600]$

$4^{ème}$ ligne

$30 = 30 - [(30/3) \times 3]$

$0 = 0 - [(30/3) \times 0]$

$-10 = 0 - [(30/3) \times 1]$

$0 = 0 - [(30/3) \times 0]$

$- - 30 = -50 - [(30/3) \times -2]$

$-30000 = 0 - [(30/3 \times 600]$

		x	y	e_1	e_2	e_3	$2^{ème}$ membre
ligne du Pivot	x	1	0	1/3	0	$--2/3$	200
$2^{ème}$ ligne	e_2	0	0	$-1/3$	1	2/3	200
$3^{ère}$ ligne	y	0	1	0	0	1	600
$4^{ème}$ ligne	max	0	0	-10	0	$--30$	$--36000$

Les coefficients de la fonction économique sont tous nuls ou négatifs, fin de l'algorithme du simplex. La solution qui rend optimal le programme de production est le suivant:

La marge sur cout variable maximum= 36000€, les quantités produites $x = 200$, $y = 600$, et on constate que la variable d'écart e_2 correspondant a la contrainte de marché de X n'est pas saturée.

Nous aurions pu vendre 200 produits X de plus. Par contre e; la variable d'écart traduisant la contrainte technique et e_3 la variable d'écart correspondant a la contrainte commerciale du marché du

produit y sont saturées.

3.3 Exemple

Un atelier fabrique 2 modèles X et Y, le produit X ne peut être vendu à plus de 400 exemplaires, le produit Y ne peut être vendu à plus de 600 exemplaires. Pour fabriquer X il faut 3 heures de main

d'œuvre, et 2 heures pour Y, en sachant que l'entreprise ne dispose de 1800 heures de main d'œuvre. La marge sur coût variable réalisée sur la vente d'un X est de 30€, de la vente d'un Y est de 50€.

Quelle est la combinaison productive qui permet de maximiser la marge sur coût variable?

La définition du programme linéaire est la suivante:

- – Contraintes techniques : la production d'un X consomme 3 heures de main d'œuvre, la production d'un Y consomme 2 heures. La capacité de cet atelier est limité à 1800 heures d'où inéquation suivante: $3x + 2y \leq 1800$.

- Contraintes de marché: il n'est pas possible de vendre pour le produit X plus de 400 unités et pour le produit Y plus de 600 unités d'où les inéquations suivantes: $x \leq 400$ et $y \leq 600$.
- Contraintes logiques : les quantités produites ne peuvent pas être négatives d'où les inéquations suivantes: $x \geq 0$ et $y \geq 0$.
- Fonction économique à maximiser : $MAXB = 30x + 50y$ c'est l'objectif à atteindre.

La représentation graphique est la suivante:

(1) : $3x + 2y \leq 1800$

(2) : $x \leq 400$

(3) : $y \leq 600$

■ : $y = -30/50x$

Le champ des possibles est délimité par les droites passant par les points $(0, 0)$, $(400, 0)$, $(400, 300)$, $(200, 600)$ et $(0, 600)$

La fonction MAX est représentée par la droite verte (■) permettant de rechercher par translation parallèle le point le plus éloigné du champ des possibles. Le point le plus éloigné de cette droite dans le champ des possibles est le point M $(200, 600)$.

Donc pour atteindre l'optimum, les quantités à produire sont : $x = 200$, $y = 600$. La marge maximum sera de $(30x200) + (50x600) = 36000$€.

Nous observons que la contrainte commerciale du produit X n'est pas saturée, nous aurions puvendre 200 unités de plus ; la contrainte commerciale du produit Y est saturée, le marché était limité à 600 unités. De même, la contrainte technique concernant la capacité productive est saturée $(3x200) + (2x600) = 1800$. Les contraintes logiques sont respectées, à savoir $x \geq 0$ et $y \geq 0$.

Conclusion

Dans ce mémoire nous avons étudier comment maximisé ou bien minimisé un problème d'optimisation avec plusieure méthode en eet la méthode de simplexe comme un exemple, et on a aussi la résolution d'approximation d'algorithmes heuristiques.

Bibliographie

[1] Jin - Kao Hao, philippe Galinier, Michel HAbib. RIA, Revue d' Intelligence Artificielle Méthaheuristiques pour L'optimisation combinatoire et L'affectation sous contraintes, Vol: No, 1999.

[2] pierre fouilhoux pierre, optimisation combinatoire:programation linéair et Algorithmes, unuvecité pierre et Marie Curie, 29septembre 2015.

[3] Jacques CARLIER. Recherche opérationnelle: Optimisation combinatoire.

[4] Internet, Programation Linéaire en Nombes Entiers (PLNE), dept.info_laibri Fr.

[5] GUY, Résolution de problèmes difficiles: algorithmes d'approximation algorithmes probabilistes, heuristiques et métaheuristiques.

[6] Bemard Auge - Alexandre vemhet, Module 06 - Leçon 2 leçon 03: La mèthode du simplexe,Académie de l'execllence-kademiatn,2019

ملخص

في هذه المذكرة استخدمنا برمجة عدد صحيح خطي لحل مشكلة التحسين الاندماجي وقدمنا طريقة البسيط كمثال.

الكلمات المفتاحية :

- مشكلة التحسين الاندماجي.
- برمجة الرياضيات.
- البرمجة الخطية ,برمجة عدد صحيح خطي.
- طريقة البسيط.

Dans cette mémoire ,nous avons utilisé la programmation linéaire en nombre entier pour la résolution de problème d'optimisation combinatoires ,et nous avons fourni la méthode de simplexe comme exemple.

Mots clé:

- Problème d'optimisation combinatoire.
- programmation mathématique.
- programme linéaire en nombre entier.
- la méthode de simplexe.

Abstract

In this memory we have used integer linear programming for solving combinatorial optimization problem , and we have provided the simplex method as an example.

Key words :

- Combinatorial optimization problem.
- mathematical programming.
- linear program, linear programming in integer number.
- the simplex method.