REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS DE MOSTAGANEM

FACULTE DES SCIENCES EXACTES
ET SCIENCES DE LA NATURE ET DE LA VIE
DEPARTEMENT DE MATHEMATIQUES

Mémoire de Master en Mathématiques

PrésentéPar

AZIRIA lakhdar

Option : **Modélisation**, **Contrôle et Optimisation**

intitulé

le problème de sac à dos

Soutenue le 26/06/2012

Devant les membres de jury :

Président :Monsieur BELHAMITI	, Maître de Conférences à Université de Mostaganem.
Examinateur :Monsieur AMIR	, Maître de Conférences à Université de Mostaganem.
Encadreur :Monsieur ABLAOUI	, Maître de Conférences à Université de Mostaganem.

Table des matières

Remerciements

Tout d'abord, le grand Merci à notre « DIEU» le tout puissant, qui nous a donné la force et la volonté pour finir ce projet

je remercie mes parents dont le rêve a toujours était de nous voir réussir, qu'ils sachent que leur places est dans nos coeurs.

je remercie tout particulièrement Monsieur *Ablaoui* pour avoir accepter de nous encadrer et de mes suivre tout au long de la réalisation de ce mémoire.

je remercie *Belhamiti* pour avoir accepté de présider notre mémoire.

Aussi, je remercie*Amir* pour avoir examiner notre mémoire.

je n'oublie pas mes très chers amis :bouhana

je remercie aussi mes collègues de promotion master Math 2011/2012.

Sans oublier tous les membres de mes famille aziria

En fin je remercie tous ceux qui de loin ou de prés, ont apportés une gracieuse collaboration pour la confection de ce travail.

dédicaces

A mes chers parents qui m'ont soutenu durant ma scolarité
A mes frères et mes soeurs
A tous mes proffesseurs
A tous mes amis
A tous ceux que j'aime
J'exprime mes sentiments les plus amicaux et leur dédie ce modeste mémoire

Introduction

Le problème du sac à dos est l'un des 21 problèmes NP-complets de Richard Karp, exposés dans son article de 1972. Il est intensivement étudié depuis le milieu du XXe siècle et on trouve des références dès 1897, dans un article de George Ballard Mathews1. La formulation du problème est fort simple, mais sa résolution est plus complexe. Les algorithmes existants peuvent résoudre des instances pratiques de taille importante. Cependant, la structure singulière du problème, et le fait qu'il soit présent en tant que sous-problème d'autres problèmes plus généraux, en font un sujet de choix pour la recherche.

Le problème du Sac à Dos est un problème générique de l'informatique théorique au même titre que le voyageur de commerce (TSP). Ces deux problèmes ont été très largement étudié soit au niveau de la complexité (NP-complétude, polynomialité pour certaines instances), soit au niveau de l'approximation (approximation avec garantie de performance, schéma d'approximation, schéma d'approximation totalement polynomiale), soit de manière exacte (programmation dynamique et branch and bound). Ce problème a servi de "benchmark" pour comparer les méthodes. Le problème du sac à dos intervient dans de nombreuses situation (navette spatiale,....). Le problème du sac à dos intervient également dans la cryptographie comme une méthode concurrente de la méthode RSA (en fait nous verrons plus loin que ce n'est pas le cas).

Le problème du Sac à Dos aussi noté KP(Knapsack Problem en anglais) qui est de la forme :

$$(KP)\begin{cases} \max z(x) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i x_i \leq w \\ x_i \in \{0,1\}, \forall i \in \{1,...,n\} \end{cases}$$

où n : l'objets

w_i :poids du l'objet

c_i : l'utilité de l'objet

Selon qu'un objet i est sélectionné ou non, la variable associée x_i prend, respectivement, la valeur 1 ou 0.

Si on utilise l'enumération complète pour résoudre le problème du sac à dos nous avons à considerer 2^n possibilités. Pour n assez grand le nombre de possibilités devient trop important pour utiliser cette methode.

Chapitre 1

Introduction à la théorie des graphes

1.1 Définitions et concepts de base

1.1.1 graphes :concepts orientés

un graphe $G = (X.U)$ est déterminé par la donnée :

$\star$d'un ensemble X dont les élements sont appelés des *sommets* ou des *noeuds.*

si $N = |X|$ est le nombre de *sommets* (de *noeuds*),on dit que le graphe G est d'ordre N

on supposera que les *sommets* sont numérotés $i = 1, ..., N$

$\star$d$'$unensemble U dont les élements $u \in U$ sont des couples ordonnés de sommets appelés des *arcs.* si $u = (i, j)$ est un arc de G, iest l'éxtrémité initiale de u et j est l'éxtrémité terminale de u.On notera $|U| = M$ le nombre d'arcs.

Graphiquement, les sommets peuvent être representés par des points et $u = (i, j)$ sera représenté par une flèche joingnant les deux points $i et j$ (j corespondant à la pointe de la flèche).

un arc $u = (i, j)$ dont les éxtrémités coïncide est appelé une *boucle.*

un *p-graphe* est un graphe tel qu'il n'existe jamais plus d'un arc de la forme

$$(i, j)(\forall i et \forall j)$$

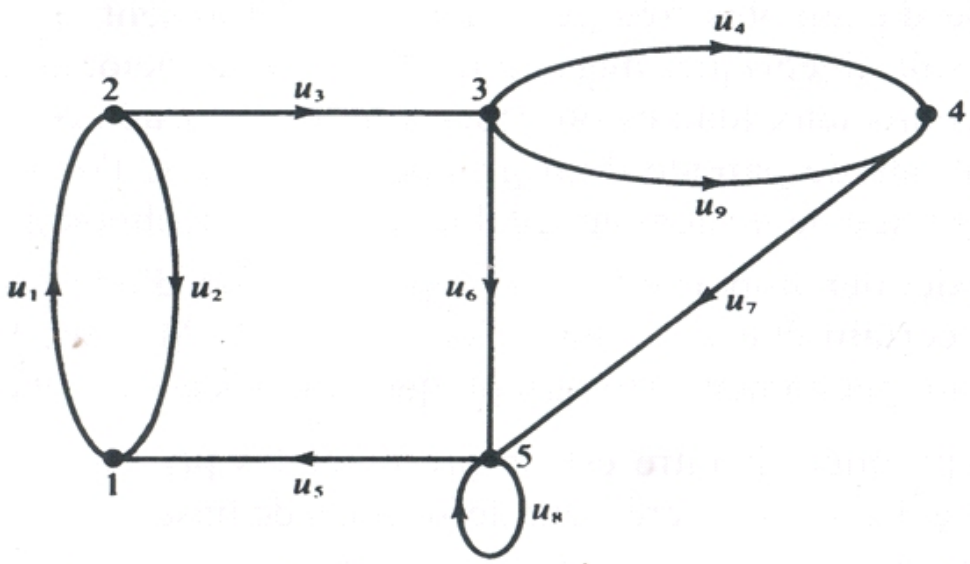


*figure*1 : *exemple d'un*2 − *graphes*

$u_8 = (5, 5)$ est une boucle.il s'agit d'un 2-graphe car $u_4 = (3, 4), u_9 = (3, 4)$

1.1.2 Graphes et applications multivoques

on dit que j est un *successeur* de i s'il existe un arc ayant i comme extrémité initiale et j comme extrémité terminale. l'ensemble des successeurs d'un sommet $i \in X$ est noté :Γ_i.l'application Γ qui à tout élément de X, fait correspondre une partie de$X$ (un élément de $\rho(X)$),est appelée une *application multivoque.*

on dit que j est un prédécesseur de$i$, s'il existe un arc de la forme :(i, j).

l'ensemble des prédécesseurs de $i \in X$ peut alors être noté :Γ_i^{-1},ou Γ^{-1} est l'application (multivoque) réciproque de :Γ.

si le graphe G est un 1-graphe, on voit qu'il est parfaitement déterminé par la donné de l'ensemble Xet de l'application multivoque Γ de $X \to \rho(X)$. un tel graphe peut donc être noté : $G = [X.U]$.

en élimine l'arc u_9 du graphe de la figure 1, on obtient un 1-graphe qui peut donc être représenté par une application multivoque Γ. pour ce graphe l'application Γ sera :

$$\Gamma_1 = \{2\}, \Gamma_2 = \{1, 3\}, \Gamma_3 = \{4, 5\}, \Gamma_4 = \{5\}, \Gamma_5 = \{1, 5\}$$

1.1.3 graphes :concepts non orientés :

dans l'etude de certaines propriétés des graphes, il arrive que l'orientation des arcs, c'est -à-dire la distinction entre extrémité initiale et extrémité terminale, ne joue aucun rôle. on

s'intéresse simplement à l'existence ou à la non-existence d'un (ou de plusieurs) arcs entre deux sommets (sans en préciser l'ordre).

à tout arc $(i, j)-$ à tout couple ordonné $(i, j)-$ on associe le couple non ordonné(i, j), qui est appelé l'arête(i, j).on peut dire de façon équivalente, qu'une arête est un arc sur lequel on a "oublié" l'orientation .Graphiquement, l'arête (i, j) sera représentée par un *segment* (sans flèche) joignant les deux points représentatifs des sommets $i$et j.

Dans l'étude des propriétés (non orientées) d'un graphe $G = (X.U)$, il suffit donc de considérer l'ensemble U comme un ensemble d'*arêtes,*c'est -à-dire, une famille finie de parties à deux éléments (distincts ou non) de$X$.

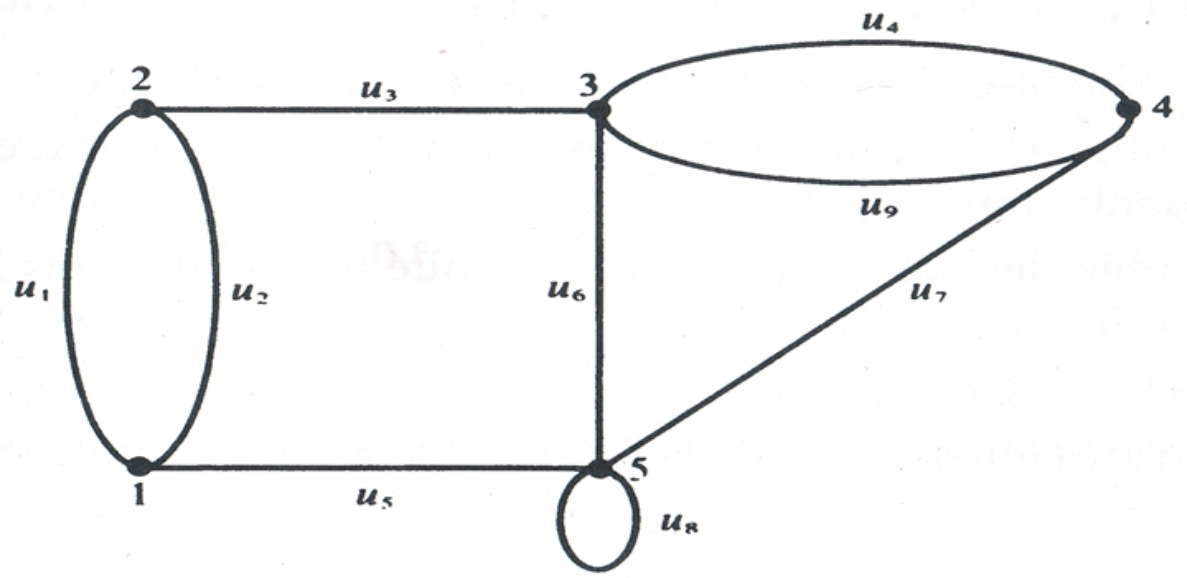


*figure*2 : *graphe non orienté correspondant au graphe de la figure*1

Dans toute la suite ,*U sera toujours implicitement concidéré comme un ensemble d'arcs.*Chaque fois que U devra être considéré comme un ensemble d'*arêtes,*nous le mentionnerons explicitement.Nous dirons alors que le Graphe G considéré est *non orieté.*

Un *multigraphe* est un graphe pour iequel il peut exister plusieurs arêtes entre deux sommets i et$j$ donnés.

Un graphe est dit *simple* si :

– il est sans boucles ;

– il n'y a jamais plus d'une arête entre deux sommets quelconques.

1.1.4 principales définitions

Arcs adjacents.Arêtes adjacentes

Deux arcs (deux arête) sont dits adjacents s'ils ont au moins une extrémité commune.

Degrés et demi-degrés

le demi-degré *extérieur* du sommet i,noté $d_G^+(i)$ (ou $d^+(i)$ lorsqu'il n'y a pas d'ambiguîté), est le nombre d'arcs ayant i comme extrémité initiale.

le demi-degré *intérieur* du sommet i,noté $d_G^-(i)$ (ou $d^-(i)$ lorsqu'il n'y a pas d'ambiguîté), est le nombre d'arcs ayant i comme extrémité terminale.

le *degré* du sommet i,noté $d_G(i)$ (ou $d(i)$ lorsqu'il n'y a pas d'ambiguîté), est le nombre d'arcs (ou d'arête) ayant i comme extrémité ,et on a :

$$\boxed{d_G(i) = d_G^+(i) + d_G^-(i)}$$

par exemple, pour le graphe de la figure 1,on a :
$d^+(2) = 2, d^-(2) = 1, d(2) = 3, etc,$

graphes symétriques

un graphe G est dit symétrique si pour toute paire de sommets (i, j), il existe autant d'arcs de la forme (i, j) que de la forme (j, i).

graphes antisymétriques

un 1-graphe $G = (X.U)$ est dit *antisymétrique* si :

$$(i, j) \in U \Rightarrow (j, i) \notin U$$

.

graphe complet (clique)

un graphe $G = [X.U]$ est dit *complet* si pour toute paire de sommets (i, j) il existe au moins un arc de la forme (i, j) où (j, i).

un 1-graphe est complet si et seulement si :

$$(i, j) \notin U \Rightarrow (j, i) \in U$$

un graphe simple complet d'ordre N est noté K_N. un sous ensemble de sommets$C \subset X$ tel que deux sommets quelconques de C sont reliés par une arête est appelé une *clique*.c'est donc l'ensemble des sommets d'un sous graphe complet de G.

Sous-graphe induit par un sous-ensemble de sommets

étant donné $A \subset X$, le *sous-graphe* induit par A est le graphe G_A dont les sommets sont les éléments de A et dont les arcs de$G$ ayant leurs deux extrémités dans A.

Graphe partiel engendré par un sous ensemble d'arcs

soient un graphe $G = (X.U)$ et $V \subset U$.le graphe partiel engendré par $V \subset U$ est le graphe ayant le même ensemble$X$ de sommets que G,et dont les arcs sont les arcs de V (on élimine de G les arcs de $U \backslash V$).

Sous-graphe partiel

étant donné un graphe $G = (X.U)$ et $A \subset X, V \subset U$,le sous graphe partiel engendré par A et V est le graphe partiel de G_A engendré par V.

1.2 Matrices associées à un graphe

1.2.1 Matrice d'incidence sommets_arcs

la matrice d'incidence sommets-arcs d'un graphe $G = (X, U)$ est une matrice

$$A = (a_{iu}), i = 1, ...N, u = 1...M$$

à coefficients entiers 0,+1,-1 telle que chaque colonne corespond à un *arc* de G,et chaque ligne à un *sommet* de G;

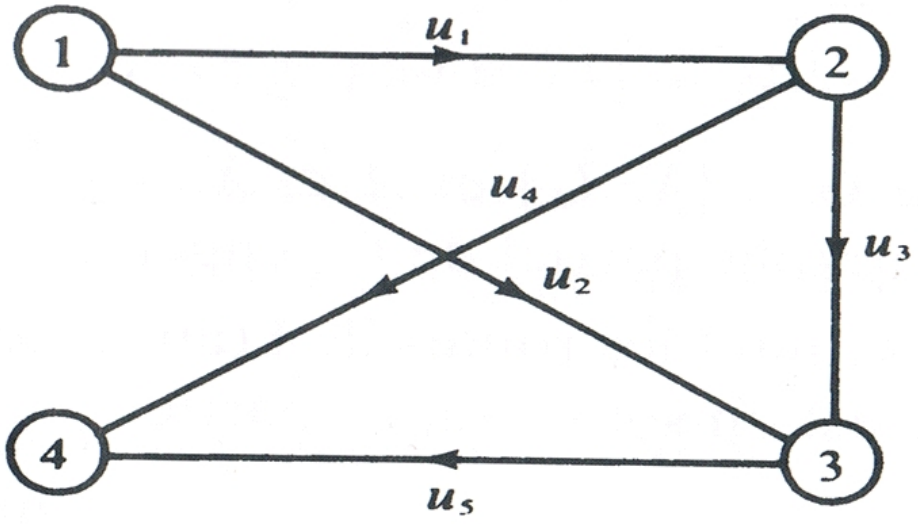
si $u = (i, j) \in U$, la colonne u a tous ses termes nuls,sauf :

$$a_{iu} = +1$$

$$a_{ju} = -1$$

exemple : la matrice d'incidence sommets-arcs du graphe de la figure 3 est :

$$\begin{bmatrix} +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}$$



$figure3$

une matrice carrée $n * n$ est dite *unimodulaire* si son déterminant est soit +1,soit -1.

une matrice A rectangulaire $m * n$ est dite *totalement unimodulaire* si et seulement si toutes les sous matrices carrées régulières extraites de A sont unimodulaires

Proposition 1.1 *la matrice A d'incidence sommets-arcs d'un graphe $G = (X.U)$ est totalement unimodulaire*

1.2.2 Matrice d'incidence sommets-arêtes

soit un graphe $G = (X, U)$ où U est un ensemble d'arêtes

la matrice d'incidence sommets-arêtes de G est une matrice à coefficients 0 ou 1,où chaque ligne i correspond à un sommet i de G,et chaque colonne à une arête $u = (i, j)$ de G;de plus ,si $u = (i, j)$,alors la colonne u a tous ses éléments nuls sauf : $a_{iu} = +1$et $a_{ju} = +1$

*exemple :*la matrice d'incidence sommets-arêtes du graphe de la figure 3 est :

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

la matrice d'incidence sommets-arêtes n'est pas en général *totalement unimodulaire*. dans l'exemple ci-dessus,la sous-matrice $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ a pour déterminant :-2

on dit qu'un graphe $G = (X, U)$ est *biparti* si l'ensemble des sommets Xpeut être partionné en deux sous ensemble X_1 et X_2de telle sorte que, pour toute arête $(i, j) \in U$:

$i \in X_1 \Longrightarrow j \in X_2$

$i \in X_2 \Longrightarrow j \in X_1$

Proposition 1.2 *la matrice d'incidence sommets-arêtes d'un graphe $G = [X, U]$ est* totalement unimodulaire *si et seulement si G est un graphe biparti*

1.2.3 Matrice d'adjacence(matrice d'incidence sommets- sommets)

soit $G = (X, U)$ un 1-graphe(c.à.d qu'il nexiste pas plus d'un arc)

la matrice d'adjacence est une matrice A à coefficients 0 ou 1 : $A = (A_{ij})_{i,j=1,...N}$

où chaque ligne correspond à un *sommet* de$G$.où chaque colonne correspond à un *sommet* et où : $A_{ij} = +1$ si et seulement si $(i, j) \in U$ $(A_{ij} = 0$si non)

dans le cas non orinté, on peut aussi definir la matrice d'adjacence d'un graphe simple en considérant qu'à chaque arête(i, j),correspondent deux arcs $(i, j)et(j, i)$ dans ce cas, la matrice d'adjacence est symétrique

1.2.4 les différents représentation d'un graphe

pour décrire un graphe G diverses représentation peuvent être utilisées. en pratique , le choix d'une représentation plutôt qu'une autre a souvent un impact sur l'efficacité des algorithmes.

il existe deux grande familles de representation :la première utilise la matrice d'adjacence ou ses dérivées, la seconde d'incidence ou ses dérivées.

à partir de la matrice d'adjacence

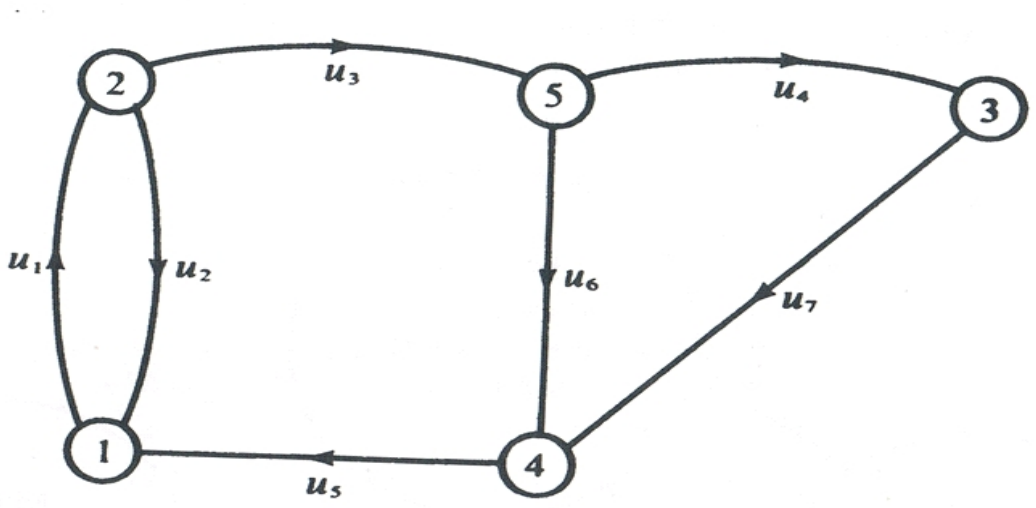
à partir de la matrice d'incidence sommets-arcs ou sommets-arêtes

1.3 Connexité

1.3.1 Chaîne.Chaîne élémentaire.Cycle.Cycle élémentaire

une *chaîne* de longueur q (de cardinalité q) est une séquence de q arcs : $L = \{u_1, u_2, ..., u_q\}$ telle que chaque arc u_r de séquence($2 \leq r \leq q-1$) ait une extrémité commune avec l'arc u_{r-1} ($u_{r-1} \neq u_r$) et l'autre extrémité commune avec l'arc u_{r+1}($u_{r+1} \neq u_r$)

l'extrémité i de u_1 non adjacente à u_2, et l'extrémité j de u_q non adjacente à u_{q-1} sont appelées les extrémités de la chaîne L. on dit auussi que la chaîne L joint les sommets$i$ et j.



$figure9$

ainsi sur le qraphe de la figure 9,

$$L = \{u_2, u_5, u_6, u_4\}$$

est une chaîne allant du sommet 2 au sommet 3

on appelle chaîne élémentaire une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet. de façon équivalente, on peut définir une chaîne élémentaire comme une chaîne dont les sommets sont tous de degré 2 au plus.

un cycle est une chaîne dont les extrémités coîncident.
un cycle élémentaire est un cycle minimal

1.3.2 Chemin.Chemin élémentaire.Circuit.Circuit élémentaire

un chemin de longeur q (de cardinalité q) est une séquence de q arcs : $P = \{u_1, u_2, ..., u_q\}$ avec

$$u_1 = (i_0, i_1)$$

$$u_2 = (i_1, i_2)$$

$$u_q = (i_{q-1}, i_q)$$

(autrement dit, un chemin est une chaîne dont tous les arcs sont orientés dans le même sens).

le sommet i_0 est l'extrémité initiale du chemin P ;le sommet i_q est l'extrémité terminale du chemin P.

Ainsi pour le 1-graphe de la figure 9,

$$P = \{u_1, u_3, u_4, u_7\}$$

est un chemin allant du sommet 1 au sommet 4 ; on peut le représenter par la séquence des sommets $\{1, 2, 5, 3, 4\}$.

un *chemin élémentaire* est un chemin tel qu'en le parcourant ,on ne rencontre pas deux fois le même sommet .Dans un chemin élémentaire, tous les sommets sont de degré 2 au plus.

un *circuit* est un chemin dont les éxtrémités coïncident. un circuit élémentaire est un circuit dont tous les sommets sont de degré 2.

1.4 Plus court chemin

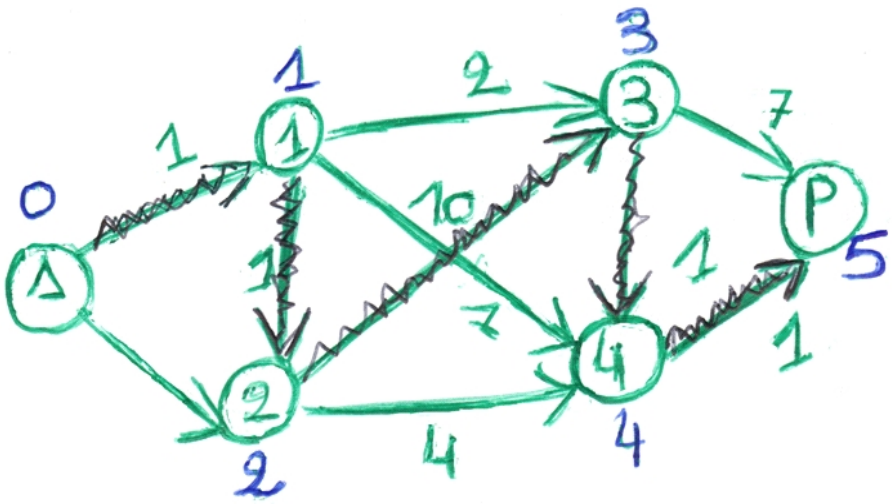
1.4.1 Algorithmes de PCC

Il existe deux cas importants en pratique quand le graphe n'a pas de circuit absorbant :
a) Lorsque tous les poids sont positifs ou nuls, on dispose de l'algorithme de *Dijkstra,*
b) Lorsque le graphe est sans circuit, on dispose de l'algorithme de*Bellman*.

Algorithm 1.1 *BELMAN 1958*
G(x,u,d) ne possède pas de circuit
$\pi(s) \longleftarrow 0$
$A(s) \longleftarrow \epsilon$
$S \longleftarrow \{s\}$
tant que ($\exists x \notin S$ *dont tous ses prédesseur sont dans* S
$S \longleftarrow S \cup \{x\}$
$\pi(x) = \min_{u/T(x)=x} (\pi(I(u)) + d(u))$
soit $\tilde{u}/\pi(x) = \pi(I(\tilde{u}) + d(\tilde{u}))$

$A(x) \longleftarrow \tilde{u}$
fin tant que
fin l"algorithme

Exemple 1.1 *soit le graphe dans la figure suivante :*



$\pi(s) = 0$;
$S = \{s, 1, 2, 3, 4, p\}$
$\pi(1) = 1$
$A(1) = (s, 1)$
$x = 2$
$\pi(2) = \min(1 + 1, 0 + 6) = 2$
$A(2) = (1, 2)$
$x = 3$
$\pi(3) = \min(2 + 1, 1 + 2) = 3$
$A(3) = (2, 3)$
$x = 4$
$\pi(4) = \min(3 + 1, 1 + 10, 2 + 4) = 4$
$A(4) = (3, 4)$
$x = p$
$\pi(p) = \min(4 + 1, 3 + 7) = 5$
$A(p) = (4, p)$
le plus court chemin du sommet s au sommet dans un graphe est $s \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow p$

Chapitre 2

Le problème de sac à dos

2.1 le problème de sac à dos

Le problème de sac à dos est un problème d'optimisation combinatoire modélisant une situation dans laquelle un décideur dispose d'un ensemble d'objets parmi lesquels il doit faire une sélection, en respectant une contrainte de capacité. L'objectif est de maximiser le profit apporté par les objets choisis. Dans sa forme la plus simple, dénomée « sac à dos unidimensionnel en variables binaires », le problème se formule ainsi :

$$(KP)\begin{cases} \max z(x) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i x_i \leq w \\ x_i \in \{0,1\}, \forall i \in \{1,...,n\} \end{cases}$$

où c_i est le profit apporté par la sélection de l'objet i et w_i est son poids. La contrainte exprime le fait que les objets sélectionnés doivent tenir dans la capacité w du sac à dos. Selon qu'un objet i est sélectionné ou non, la variable associée x_i prend, respectivement, la valeur 1 ou 0.

2.1.1 EXEMPLE

Pour illustrer le problème nous allons prendre la situation suivante. Un campeur, dont le sac a pour capacité w=20, 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

Dans l'exemple ci-dessus, le campeur prendrait alors la nourriture, le couteau,le duvet et la tente ce qui représente un poids de 20 pour une utilité de 125. Une autre méthode serait de prendre les objets les moins lourds, le campeur prendrait alors 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. Une dernière méthode un peu moins naïve consisterait à prendre les objets dont le rapport utilité/poids est le plus important. Dans ce cas le campeur prendrait le couteau, la nourriture, l'anti-moustique, le duvet, la boite pharmacie, la lampe et la cape ce qui fait un poids de 18 pour une utilité de 143. On a donc 3 résultats différents ce qui montre que le problème n'est pas si facile que ce qu'il semblait. De plus aucune de ces méthodes ne donne la solution optimale qui a un poids total de 20 et une utilité totale de 147.

2.2 P et NP

Considérons le problème de décision associé au problème du Sac à Dos qui détermine s'il existe un sous-ensemble d'objets dont la somme des utilités est au moins égale à une constante K.

Ce problème n'appartient pas à la classe P, qui est la classe des problèmes résolus par des algorithmes dont la complexité est polynomiale (par rapport à la taille des données).

Par contre il appartient à la classe NP, qui est la classe des problèmes dont les solutions sont vérifiables en temps polynomial. En effet si l'on a un sousensemble d'objets, il suffit de de vérifier si$\sum_{i=1}^{n} w_i x_i \leq w$ et si :$\sum_{i=1}^{n} c_i x_i \leq K$ ce qui se fait en temps polynomial.

Définition 2.1 *(Tightness ratio). On appelle tightness ratio le ratio de la somme des poids des objets sur la capacité du sac :* $\boxed{r = \frac{\sum_{i=1}^{n} w_i}{w}}$ *Il est généralement admis que les instances pour lesquelles ce ratio est proche de 0,5 font intervenir une combinatoire plus grande.*

2.2.1 Variantes autour du problème

Il existe de nombreuses variantes du problème de sac à dos, selon le domaine des variables (valeurs binaires, entières ou réelles), le nombre de contraintes (unidimensionnel, bi-dimensionnel ou multidimensionnel), le nombre de profits associés à chaque objet (mono-objectif ou multi-objectif), le nombre de sacs, etc. Du fait de la quantité des paramètres intervenant dans la formulation, les variantes sont nombreuses. Cette section présente quelques unes d'entre elles. Le lecteur désireux de connaître plus de détails sur ces variantes .

Variables continues

Le problème de sac à dos en variables continues (LKP) est une variante dans laquelle il est possible de ne prendre qu'une fraction des objets. Sa résolution s'appuie sur les concepts d'efficacité d'un objet et d'élément bloquant, définis ci-dessous.

Définition 2.2 *(Efficacité d'un objet). On appelle efficacité d'un objet le rapport de son coût sur son poids, noté*

$$\boxed{e_i = \frac{c_i}{w_i}}$$

Remarque 2.1 *La notion d'efficacité d'un objet est à distinguer de la notion d'efficacité d'une solution dans le cadre multi-objectif.*

Définition 2.3 *(Élément bloquant). On appelle élément bloquant le premier objet ne pouvant tenir dans le sac lorsque les objets sont ajoutés par ordre décroissant d'efficacité. Son indice sera noté :*

$$s = \min\left\{k : \sum_{i=1}^{k} w_i \succ w\right\}$$

pour les e_i triés en orde décroissant

La valeur optimale z^* (LKP)d'une instance est obtenue en prenant les objets dans l'ordre décroissant de leur efficacité, jusqu'à l'élément bloquant puis en ajoutant la fraction de cet élément permettant de saturer le sac

$$z^*(LKP) = \sum_{i=1}^{s-1} c_i + (w - \sum_{i=1}^{s-1} w_i) * \frac{c_s}{w_s}$$

Ce problème correspond à la relaxation linéaire de (KP). Ainsi, si I est une instance de ce dernier et I^{LP}est l'instance obtenue à partir de I en remplaçant la contrainte $x_i \in \{0,1\}, \forall i \in \{1,...,n\}$ par $x_i \in [0;1], \forall i \in \{1,...,n\}$ alors l'inégalité $z^*(I) \leq z^*(I^{LP})$ se vérifie, où $z^*(I)$ est la valeur optimale des solutions d'une instance I. De plus,LKP appartient à la classe de complexité P. Pour cela, il est souvent utilisé dans la résolution des variantes à variables entières, dans l'optique d'obtenir une borne supérieure de la solution optimale en un temps raisonnable

Sac à dos multi-dimensionnel

Le problème de sac à dos multi-dimensionnel (d-KP), est une variante du problème ayant plusieurs contraintes de capacité. Il est à la frontière entre l'optimisation combinatoire et la programmation linéaire en nombres entiers. En effet, comme l'atteste sa formulation ci-dessous, il peut être considéré comme un programme linéaire en nombres entiers (ILP) classique sous la seule restriction que les coefficients soient positifs et les variables binaires. Son champ d'application est large, ce qui a grandement contribué à sa popularité.

$$(mKP)\begin{cases} \max z(x) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i^j x_i \leq w_j; \\ x_i \in \{0,1\}, \forall i \in \{1,...,n\} \end{cases}$$

Problème de set packing

Le problème de set packing est une variante du problème de sac à dos multi-dimensionnel dans laquelle les poids sont binaires et les capacités sont toutes égales à 1.

$$(spp)\begin{cases} \max z(x) = \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} w_i^j x_i \leq 1; w_i^j \in \{0,1\}, j \in \{1,...,d\} \\ x_i \in \{0,1\}, \forall i \in \{1,...,n\} \end{cases}$$

Très peu de méthodes exactes existent pour ce problème , en particulier parce que sa relaxation linéaire ne donne pas de bonnes bornes. Ainsi, des heuristiques et méta-heuristiques sont en général utilisées pour résoudre des instances de ce problème avec un grand nombre de variables et de contraintes.

Sac à dos multi-objectif

Le problème de sac à dos multi-objectif (MOKP), est une variante du problème où plusieurs objectifs sont à maximiser simultanément.

$$(MOKP)\begin{cases} \max z_j(x) = \sum_{i=1}^{n} c_i^j x_i, j \in \{1,...,p\} \\ \sum_{i=1}^{n} w_i x_i \leq w \\ x_i \in \{0,1\}, \forall i \in \{1,...,n\} \end{cases}$$

Chapitre 3

méthodes de résolutions

Il existe deux grandes catégories de méthodes de résolution de optimisation combinatoire : les méthodes exactes et les méthodes approchées. Les méthodes exactes permettent d'obtenir la solution optimale à chaque fois, mais le temps de calcul peut être long si le problème est compliqué à résoudre. Les méthodes approchées, encore appelées heuristiques, permettent d'obtenir rapidement une solution approchée, donc pas nécessairement optimale. Nous allons détailler un exemple d'algorithme de résolution de chaque catégorie.

3.1 Méthodes approchées

3.1.1 principe des algorithmes de voisinage :l'algorithme du meilleur voisin

supposons que le problème à resoudre se pose sous la forme $\min_{x \in X} f(x)$ oùX :l'ensemble des solutions réalisables du problème

l'idée consiste à définir pour tout solution $x \in X$ un sous ensemble de solutions dites *voisines* $v(x) \subset X$

si X est l'ensemble $\{0,1\}^n$ de tous n vecteurs à composantes 0 où 1, on poura $\lambda \in N$ le voisinage de $x \in X$ par : $v_\lambda(x) = \{y \backslash |x-y|_H \leq \lambda\}$ où $|.|_H$ designe la distance de HAMMING, $v_\lambda(x)$ est l'ensemble des vecteurs $y \in \{0,1\}^n$

Algorithm 3.1 *(a) soit $x^0 \in X$ la solution départ choisie*
$k \longleftarrow 0$
(b) répéter (itération courante k)
x^k *solution courante*
déterminer $x^{k+1} \in X$ *telque* $f(x^{k+1}) = \min_{x \in v(x^k)} \{f(x)\}$
$k+1 \longleftarrow k$
tant que $f(x^k) \prec f(x^{k-1})$
(c) x^k *est une solution*

on dit que obtenu un *optimum local*

le défaut de cet algorithme est le qualité de solution produite dépend beaucoup du choix du point de départ

3.1.2 méthodes de recherche à profondeur variable (variable depth search)où V.D.S

L'idée de départ de ces méthode est simple. tant qu'un optimum local n'a pas été atteint, chaque itération consiste à se déplacer (comme dans l'algorithme du meilleur voisin) vers la meilleure solution contenue dans le voisinage de la solution courante

Algorithm 3.2 *recherche à profondeur variable V.D.S*

(a) soit $\bar{x} \in X$ *la solution de départ choisie*

$k \longleftarrow 1$

(b) répéter :

$x^0 \longleftarrow \bar{x}$;

$\hat{x} \longleftarrow \bar{x}$;*(N.B :*$\hat{x}$ *représente la meilleure solution obtenue au cours de la phase k)*

$L \longleftarrow \phi$

$CA \longleftarrow VRAI$ *; (CA représente la condition d'arrêt)*

pour $(j = 0 \grave{a} n-1)$ *faire :*

déterminer x^{j+1} *tel que :*

$f(x^{j+1}) = \min_{x \in v_L(x^j)} f(x)$

si $f(x^{j+1}) \prec f(\hat{x})$ *alors :*

$\hat{x} \longleftarrow x^{j+1}$;

fsi

$L \longleftarrow L \cup \{\delta(x^j, x^{j+1})\}$ *(rappelons que* $\delta(x^j, x^{j+1})$ *l'indice de composante la quelle* $x^j \neq x^{j+1}$*)*

fpur

si $f(\hat{x}) \prec f(\bar{x})$ *alors :*

$\bar{x} \longleftarrow \hat{x}$;

$k \longleftarrow k+1$;

$CA \longleftarrow FAUX$;

fsi

tant que $(CA = FAUX)$

(c) la solution fournie par lalgorithme est $\hat{x}$*, de coût* $f(\hat{x})$

3.1.3 méthode du recuit simulé "simulated annealing"

La méthode du "recuit simulé", suggérée par *kirkpatrick* et *co-auteurs (1983)* repose sur une analogie avec l'évolution dans le temps des systèmes physiques à grand nombre de constituants (atomes, particules, etc) vers un état d'énergie minimale lorsque la température décroît lentement.

L'analogie consiste à considérer chaque solution réalisable $x \in X$ comme l'état d'un système physique, auquel est attachée une distribution de probabilité suivant la loi de *Gibbs :*

$G_T(x) = \frac{1}{Z_T} e^{\frac{-f(x)}{KT}}$ où $Z_T = \sum_{x \in X} e^{\frac{-f(x)}{KT}}$

où $f(x)$ est analogue à l'énergie du système, où T est la température du système et K la constante de *Boltzmann*. L'objectif est alors d'atteindre l'état le plus probable, c'est-à-dire l'état d'énergie minimale ($f(x)$ minimum)

Algorithm 3.3 *(a) soit $x^0 \in X$ la solution départ choisie*
$k \longleftarrow 0$;
$\overset{*}{x} \longleftarrow \overset{0}{x}$ *(meilleure solution rencontrée)*
choisir une suite décroissante de nombre $\theta_k \succ 0$
(b) répéter (itération courante)
choisir aléatoirement $y \in v(x^k)$;
calculer $p = \min\left\{1; e^{\frac{-f(y)+f(x^k)}{\theta_k}}\right\}$ *puis définir* x^{k+1} *par :*
$x^{k+1} \longleftarrow y$ *avec probabilité p ;*
$x^{k+1} \longleftarrow x^k$ *avec probabilité 1-p ;*
si($f(x^{k+1}) \prec f(x^*)$*) alors :*
$x^* \longleftarrow x^{k+1}$;
fsi
$k \longleftarrow k+1$;
tant que (condition d'arrêt non vérifieé)
(c) la solution fournie par l'algorithme est $\overset{*}{x}$ *de coût* $f(x^*)$

3.1.4 la méthode de recherche Tabou

l'idée de cette méthode est trés simple, supposons que partant d'une solution initiale $\overset{0}{x}$, on applique une méthode classique de descente du type meilleur voisin. on engendre ainsi une suite de solutions $x^0, x^1, ..., x^p$ de valeurs strictement decroissantes et on s'arrête en un minimum local x^p

Algorithm 3.4 *recherche Tabou*
(a) soit $x^0 \in X$ *la solution départ choisie*
$k \longleftarrow 0$;
$L \longleftarrow \phi$; *(initialisation la liste Tabou)*
$x^* \longleftarrow x^0$;*(meilleure solution rencontré jusqu'à présent)*
(b) répéter (itération courante k)
déterminer x^{k+1} *tel que :*
$f(x^{k+1}) = \min\limits_{x \in v_L(x^k)} \{f(x)\}$ *($v_L(x^k)$ designe l'ensemble des solutions voisines de x^k aprés élimination des solutions interdites telles que spécifiées par la liste Tabou L)*
si($f(x^{k+1}) \prec f(x^*)$*) alors :*
$x^* \longleftarrow x^{k+1}$;
fsi
si($|L| = T$*) alors :*
eliminer de L l'information la plus ancienne
fsi
ajouter dans L l'information concerant la nouvelle solution x^{k+1}*, ou la transition entre* x^k *et* x^{k+1};
$k \longleftarrow k+1$;
tant que (condition d'arrêt non vérifieé)
(c) la solution fournie par l'algorithme est x^* *de coût* $f(x^*)$

de nombreuses conditions d'arrêt sont possibles par exemple :
$v_L(x^k) = \phi$;
k a atteint la valeur maximale autorisés du nombre d'iterations ;
le nombre d'iterations effectuées depuis la dernière mise à jour de x^* est supérieure a une valeur fixée

3.1.5 algorithmes génétiques

le principe est de faire évoluer une *population de solutions* en sélectionnant les meilleures solutions à chaque étape l'évolution sera obtenue par croisement des solutions selectionnées, qui pourront également subir des perturbation aléatoires. ce principe général autorise evidemment une trés grande variété de mises en oeuvre possibles, suivant les choix effectués pour la représentation (codage) des solutions à considérer les opérations de selection ,croisement et mutation les solutions étaient supposés representées (codées) sous forme de séquences de bits (0,-1) de taille $n$déterminée

Algorithm 3.5 *génétique (principe général)*
(a) constituer une population initiale de m solutions :$p^0 = \{x^1, x^2, ..., x^m\}$ *à chaque solution* x^i *de* p^0 *est associée sa valeur* $f(x^i)$.$k \longleftarrow 0$;*(numéro de la génération)*
(b) répeter (passage de la génération k à la génération $k+1$ *)*
appliquer un opérateur de selection pour selectionnées des paires de solutions de la population courantep^k
soient $(y^1, z^1), (y^2, z^2), ..., (y^p, z^p)$ *les paires de solutions selectionnées (une même solution de* p^k *pouvant apparaître dans plusieur cos paires)*
initialiser $p^{k+1} \longleftarrow \phi$;
pour$(j = 1$ *à* $p)$*faire :*
appliquer l'opérateur de croisement à (y^j, z^j) *pour obtenir* $\varkappa(y^j, z^j)$
pour chaque solution $x \in \varkappa(y^j, z^j)$*, appliquer l'opérateur de mutation pour obtenir* $\mu(x)$ *et faire :*
$p^{k+1} \longleftarrow p^{k+1} \cup \{\mu(x)\}$
fpour
tantque (condition d'arrêt non vérifiée)

3.1.6 algorithmes gloutons "greedy algorithms"

le principe des algorithmes dits "gloutons" est simple :
on part d'une *solution partielle non réalisable du problème et on fixe à chaque étape une (ou plusieurs) variables* jusqu'a determiner une solution réalisable
exemple (le problème du sac à dos)

$$\begin{cases} \max z = \sum\limits_{j=1,n} c_j x_j \\ \sum\limits_{j=1,n} a_j x_j \leq b \\ x_j = 0 o\grave{u} 1 \forall j \end{cases}$$

on peut considérer $\frac{c_j}{a_j}$ comme le coût moyen de la variable x_j.on ordonne donc les variablex_j par coût moyen décroissant puis on fixe les x_jà 1 successivement dans cet ordre, tantque la contrainte de capacité peut être satisfaite

Algorithm 3.6 *glouuton pour le problème de sac à dos*
(a) initialisation :
b, les c_j *et* a_j *reéls positifs sont données*
on suppose $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq ... \geq \frac{cn}{an}$
(b) pour $(j = 1 à n)$ *faire :*
$x_j \longleftarrow 0$;
si $a_j \prec b$, *faire*
$x_j \longleftarrow 1$;
$b \longleftarrow b - a_j$;
fsi
fpour

le temps pris par l'étape (b) est en O(n). l'algorithme précédent est donc de complexité O(n) si les $\frac{c_j}{a_j}$ sont déja ordonnés, en O(nlogn) s'il faut commencer par les trier

Exemple 3.1 $\begin{cases} \max z = 10x_1 + 8x_2 + 5x_3 \\ 6x_1 + 5x_2 + 4x_3 \leq 9 \\ x_i \in \{0, 1\}\,; i \in \{1, 2, 3\} \end{cases}$

Dans l'exemple maximiser $10x_1 + 8x_2 + 5x_3$ *avec les contraintes de positivité (sous-entendues) et la contrainte* $6x_1 + 5x_2 + 4x_3 \leq 9$,

On a comme rapports respectivement $\frac{10}{6}$, $\frac{8}{5}$ *et* $\frac{5}{4}$, *déjà classées par ordre décroissant. Une solution rationnelle est ici* $x_1 = 1,5$, $x_2 = x_3 = 0$, *d'utilité totale 15.Une solution entière approchée est* $x_1 = 1$, $x_2 = x_3 = 0$ *d'utilité totale 10 On peut déjà dire que, pour une solution entière optimale, l'utilité est de 10 au moins et de 15 au plus.*

3.2 résolution exacte

3.2.1 les algorithmes d'énumération par separation évaluation et propagation decontraintes

3.2.2 l'énumération :

Principe de l'énmération des solutions :

dans un problème de programation en nombres entiers, les solutions sont ,en général non seulement dénombrables mais finies.

on peut donc envisager d'énumérer, c'est à dire de considérer toutes les solutions et pour chacune évaluer si elle appartient au domaine défini par les contraintes et dans l'affirmative, calculer la valeur correspondante de la fonction économique

Aprés avoir énumeré toutes les solutions, on garderait parmi celles qui satisfont à toutes les contraintes, la (ou les) solution (s) qui optimise la fonction économique.

cependant, des connaissances partielles et a priori concernant la solution optimale permettant d'éviter d'énumérer explicitement toutes les solutions ;l'énumeration ne sera pas totale mais seulement *implicite*

Principe de l'énumération implicite

le principe des méthodes d'énumération implicite est de ne faire des calculs que sur un sous ensemble des2^n solutions pour cela :

*on exclut immédiatement de la suite des solutions suscptibles d'être examinées, toute solution déja examinée antérieument ;

*on évite l'examen de solutions non examinées (mais descendantes de solutions examinées) dont on peut montrer (par certains critères) qu'elles ne peuvent être intéressantes (elles donnent un Z moins bon ou sont irréalisables).

les méthodes d'énumeration implicite ont été conçues pour certaines problèmes de $PLTE$ où toutes les variables sont entières et ne peuvent prendre que les valeurs 0 ou 1

Divers algorithmes ont été proposés et ceux ci différent

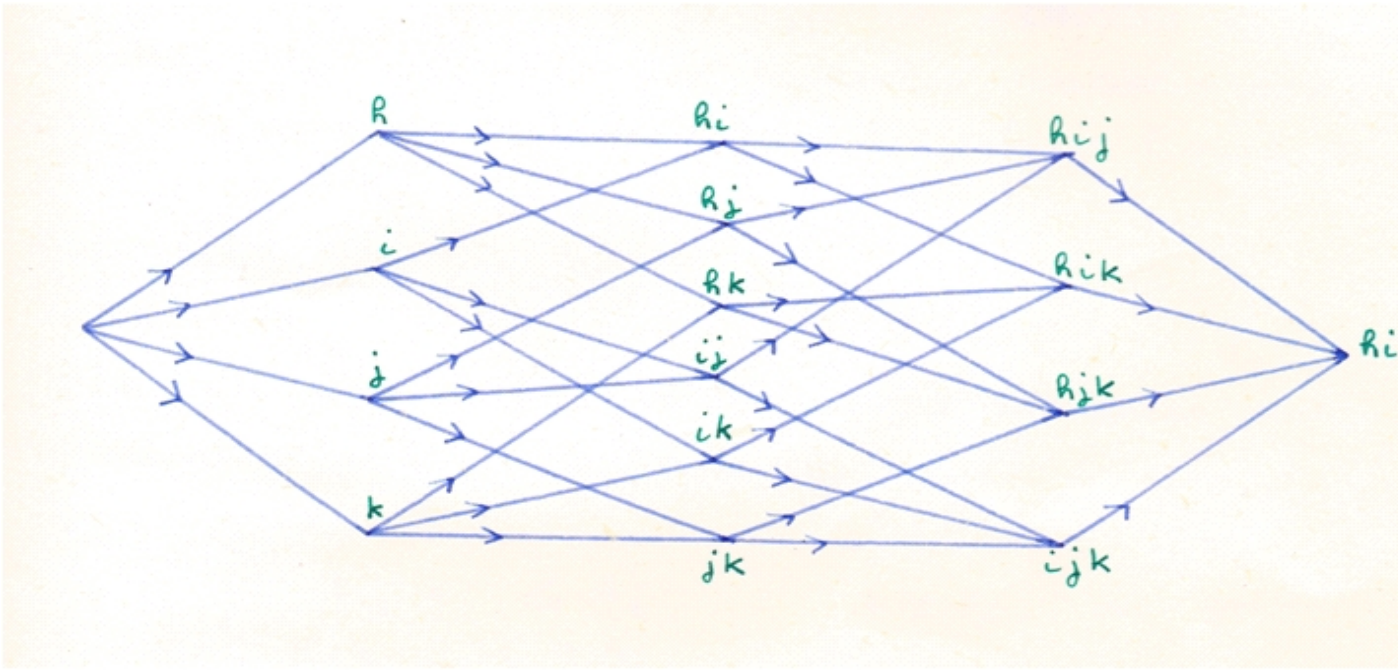
*par la façon dont on établit si une solution est implicitement évaluée ;

*par le choix des variables que l'on fixe à 1,pour passer d'une solution à une solution descendantes, non implicitement évaluée ;

*par la façon dont on retourne en arrière.

présentation du processus par un graphe

chaque sommet corespond à une solution , et chaque arc représente le passage d'une sulution s_i à une autre ,descendante de $\mathbf{s}_i$ (obtenue en donnant la valeur 1 à une variable).



3.2.3 Programmation dynamique

On introduit un paramètre supplémentaire tel que le problème à résoudre corresponde à la dernière valeur du paramètre. Ce paramètre est choisi de façon que la résolution du problème pour une valeur quelconque ne dépende que de la solution pour les valeurs plus petites.

Exemple : le problème du sac à dos
P : Maximiser $\sum_i u_i x_i$ sous les $n+1$ contraintes $x_i \geq 0$ et $\sum_i v_i x_i \leq V$.
On choisit le paramètre k et les problèmes
$P(k, v)$:Maximiser $\sum_i u_i x_i$ sous les $k+1$ contraintes $x_i \geq 0$ et $\sum_i v_i x_i \leq v$.
où les sommes ne portent que sur les indices de 1 à k. $P(1, v)$ est facile à résoudre et pour $P(k, v)$, on essaie les diverses valeurs possibles de x_k, qui à chaque fois ramènent à un problème de paramètre $k-1$. Plus précisément, si $z(k, v)$ est le maximum pour $P(k, v)$,

$$z(k, v) = \max\{z(k-1, v - v_k x_k) + u_k x_k\}$$

pris sur les valeurs possibles de x_k
Maximiser $10x_1 + 8x_2 + 5x_3$ avec les contraintes de positivité (sous-entendues) et la contrainte $6x_1 + 5x_2 + 4x_3 \leq 9$,
On choisit $k = 1, 2$ ou 3 et on s'intéresse seulement aux k premières variables. v peut prendre toutes les valeurs de 0 à 9 (en fait de 4 à 9).
Pour $k = 1$, on doit maximiser $10x_1$ avec $6x_1 \leq v$.
Pour $k = 2$, on a $z(2, v) = \max\{z(1, v - 5x_2) + 8x_2\}$, avec $v - 5x_2 \geq 0$.
Pour $k = 3$, on a $z(3, v) = \max\{z(2, v - 4x_3) + 5x_3\}$, avec $v - 4x_3 \geq 0$.

v/k	0	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	10	10	10	10
2	0	0	0	0	0	8	10	10	10	10
3	0	0	0	0	5	8	10	10	10	13

On retrouve le maximum 13, correspondant à $(0, 1, 1)$

3.2.4 Algorithme specifique au problème du sac à dos

rappel :

$$(p)\begin{cases} \max \sum_{j=1}^{n} u_j x_j \\ \sum_{j=1,n} a_j x_j \leq b \\ x_j entier \geq 0 \,; a_j \succ 0 \\ on \text{ suppose } u_j, a_j \text{ entiers et } u_j \geq 0, \forall j \end{cases}$$

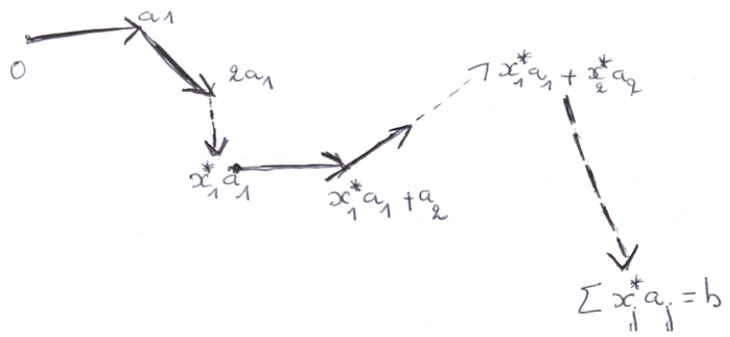
$$(\bar{p})\begin{cases} \min \sum_{j=1}^{n} - u_j x_j \\ \sum_{j=1}^{n} a_j x_j + x_{n+1} = b \\ x_j \geq 0 \,; x_j \text{ entier} \\ u_j \succ 0 \end{cases}$$

soit G un graphe à $(b+1)$ sommets $\{0,1,2,...,b\}$
$E = \{ij / j - i = a_k, \text{pour } k \in \{1,...,n\}\}$
et donc G acyclique :on attachera une valeur $-u_k$ au sommet $ij / j - i = a_k$

Théorème 3.1 *il ya une correspondance entre les chemins de "0" à "b" dans le graphe G est les solutions faisables dans le programme $(\bar{p})$ et tel que si un chemin p^* correspond à une solution $(x_1^*, x_2^*, ..., x_{n+1}^*)$ alors :*

$$l(p^*) = -\sum_{j=1}^{n+1} u_j x_j^*$$

Preuve. $(\Longrightarrow)$ *soit x^* sol de $(\bar{p})$*
$x^* = (x_1^*, x_2^*, ..., x_{n+1}^*)$
$\circ a_1 \in E \qquad a_1 2a_1 \in E.....(x_1^* - 1)a_1 x_1^* a_1 \in E$

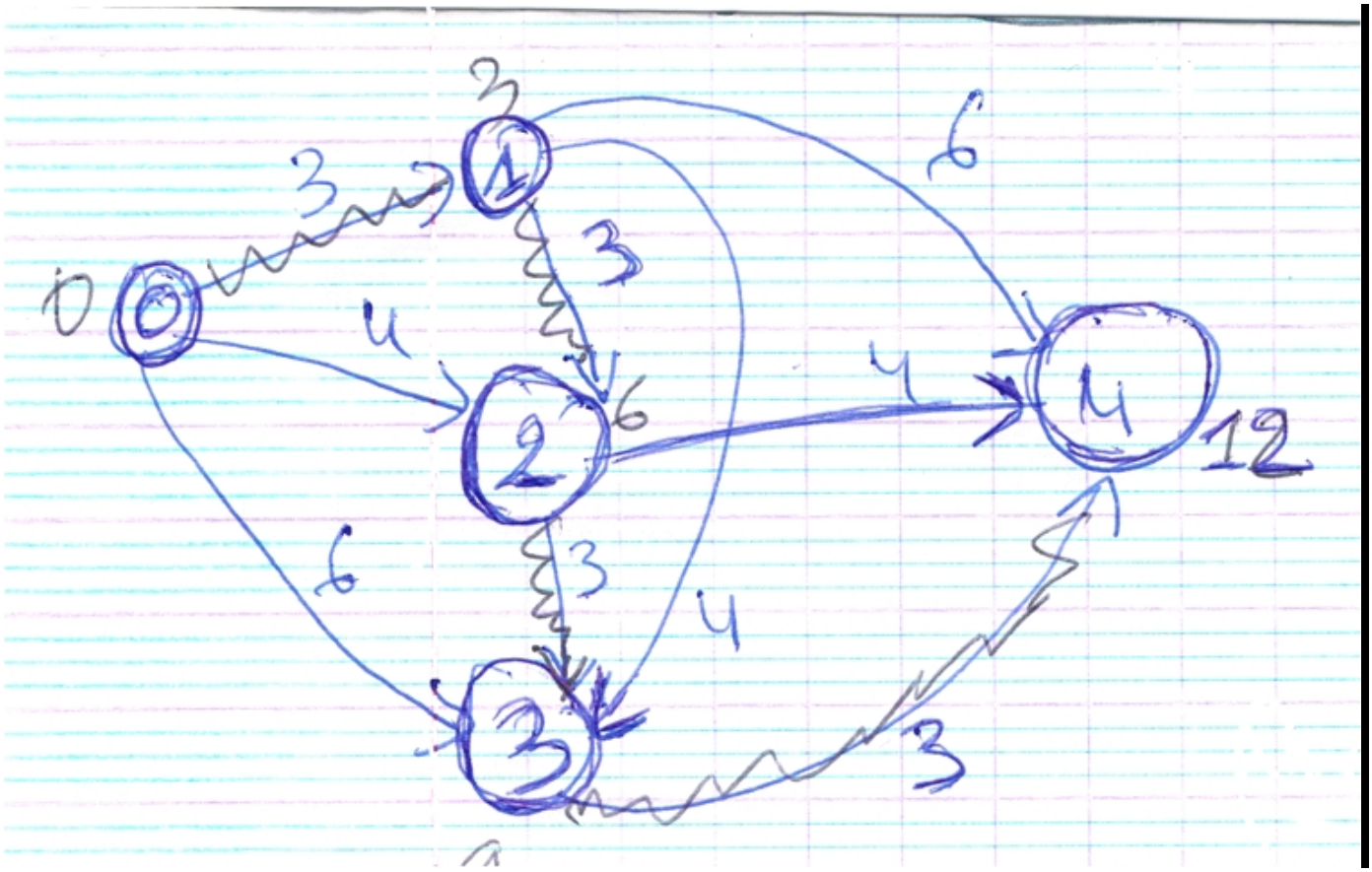


avec $l(p^*) = -\sum_{j=1}^{n+1} u_j x_j^*$; *d'où* $(\Longrightarrow)$ *vrai*
$(\Longleftarrow)$ *soit* $\pi = 0 i_1 i_2 ...., b$
$\Longrightarrow i_1 = a_r$ *:*$i_1 = \sum a_j b_{j_1}$ $\left(\begin{array}{c} b_{r_1} = 1 \\ b_{j_1} = 0 \\ pour\ j \neq r \end{array} \right.$
on suppose $i_{k-1} = \sum_{1}^{n+1} a_j b_{j_{k-1}}$
$i_{k-1} i_k \in E \qquad i_k - \sum_{1}^{n+1} a_j b_{j_{k-1}} = a_q$
$\Longrightarrow i_k = \sum_{1}^{n+1} a_j b_{j_{k-1}} + a_q$
$= \sum_{1}^{n+1} a_j b_{j_k}$ *avec* $b_{j_k} = 1$
$\Longrightarrow b = \sum_{j=1}^{n+1} a_j b_j$
on a $\sum_{j=1}^{n+1} a_j b_j = b \Longrightarrow b = (b_1, b_2, ..., b_{n+1})$ *solution de* $(\bar{p})$ *(ici* $a_{n+1} = 0$*)* ■

3.2.5 Analogie avec le problème du plus long chemin

Exemple 3.2 $\begin{cases} \max z = 6x_1 + 3x_2 + 4x_3 \\ 3x_1 + x_2 + 2x_3 \leq 4 \\ x_i \in N \end{cases}$

on a $b = 4 \Longrightarrow n = 5$



on appliquons l'algorithme de BELMAN pour calculer plus long chemin

$s = \{0\}$
$\pi(0) = 0$
$A(0) \longleftarrow \epsilon$
$s = \{0, 1\}$
$\pi(1) = 3$
$A(1) = (0, 1)$
$s = \{0, 1, 2\}$
$x = 2$
$\pi(2) = \max\{3 + 1, 0 + 4\} = 4$
$A(2) = (0, 2)$
$s = \{0, 1, 2, 3\}$
$x = 3$
$\pi(3) = \max\{3 + 4, 3 + 4, 0 + 6\} = 7$
$A(3) = (2, 3)$
$s = \{0, 1, 2, 3, 4\}$

$x = 4$
$\pi(4) = \max\{4+4, 3+6, 7+3\} = 10$
$A(4) = (3, 4)$
on a : $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$
d'où $z^* = 12$

Chapitre 4

le problème du sac à dos multidimensionnel

4.1 Introduction

Le problème du sac à dos multiple (MKP) à variables bivalentes est une variante du problème du sac à dos (KP) dont la résolution est beaucoup plus difficile. Le fait qu'on rencontre ce problème dans des domaines d'application aussi différents que l'économie, l'industrie, les transports, le chargement de cargaisons et l'informatique répartie, lui confère un grand intérêt pratique. Ce problème d'optimisation combinatoire sous contraintes est NP-complet, les méthodes exactes existantes sont limitées à de petites instances. Aujourd'hui, la résolution des instances très difficiles s'effectue grâce à des approches heuristiques, l'aptitude de ces dernières à fournir des solutions de bonne qualité les rend indispensables dans le domaine pratique et elles s'avèrent aussi très utiles pour le développement de méthodes exactes fondées sur des techniques d'évaluation et de séparation.

4.2 Définition du problème du sac à dos multiple (MKP)

On considère un ensemble $N = \{1, ..., n\}$ d'articles à charger dans m sacs à dos de capacité c_i; $i \in \{1, ..., m\}$. Chaque article$j \in N$ est caractérisé par son poids w_j ,son profit p_j et sa variable de décision x_{ij} qui vaut 1 si l'article j est chargé dans le sac$i$ et 0dans le cas contraire. Il s'agit alors de trouver m sous-ensembles disjoints de N (où chaque sous-ensemble correspond au remplissage d'un sac) qui maximisent le profit total formé par la somme des articles sélectionnés . Le problème du MKP s'écrit de la manière suivante :

$$(MKP)\begin{cases} \max \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_{ij} \\ \text{s.c } \sum_{j=1}^{n} w_j x_{ij} \leq c_i; i \in \{1, ..., m\} \\ x_{ij} \in \{0, 1\}, i \in \{1, ..., m\}, j \in \{1, ..., n\} \end{cases} \quad (1)$$

avec p_j, c_i et w_j des entiers positifs.

4.3 Algorithme de BALAS

4.3.1 Introduction :

l'algorithme de BALAS a été le premier algorithme proposé en énumération implicite

nous avons vu que ce type de méthodes a été conçu pour les problèmes où toutes les variables sont entières et ne peuvent prendre que les valeurs 0 où 1

4.3.2 Formulation du problème

un problème de$PLTE$ à variables bivalentes peut être ainsi formulé, sous forme canonique

$$\begin{cases} \sum a_{ij}x_j \leq d_i \quad , i = 1, ..., m & (1) \\ \left.\begin{array}{r} 0 \leq x_j \leq 1 \\ x_j : "entier" \end{array}\right\} & (2) \\ \min z = \sum_{j=1}^{n} c_j x_l & (3) \\ hypothèse : c_j \geq 0 \quad j = 1, ..., n & (4) \end{cases}$$

4.3.3 Algorithme :

$$\begin{cases} \min c(z) \\ Ax \geq b \\ x \in \{0, 1\} \end{cases}$$

Remarque 4.1 *si on a un problème de maximisation , on se ramène à un problème de minimisation* $[\max z = \min(-z)]$

1èreétape

*on s'arrange pour que les $c_j \geq 0$

si c_k est négatif, on pose :$y_k = 1 - x_k$

*quand tous les $c_j sont \geq 0$,on les classe par ordre croissant :$c_1 \leq c_2 \leq c_3 \leq ... \leq c_n$

*comme on a des y et des x,on les remplace par des z_i (i selon l'ordre du classement)

*on pose :$z_u = +\infty$

z_u :meilleure valeur de z parmi les solutions réalisables antérieures

2èmeétape :

*on explore le sommet où l'on se trouve (stratégie (ifo)), en donnant la valeur 1 à l'une des variables

*calcul de z_l :

$z_l = \sum_{i=1}^{N} c_i x_i$, N : indice de la darnière variable introduite

tests de stérilisation :

1^0)si $z_l \geq z_u \rightarrow$stérilisation

2^0)Faisabilité (respect des contraintes)$\rightarrow$stérilisation $\sum_{j=1}^{N} a_{ij}x_j \geq b_i$, $\forall i$

3^0) si pas de solution faisable$\rightarrow$stérilisation $\sum_{j=1}^{N} a_{ij}x_j + \sum_{j=N+1}^{n} \max\{a_{ij}, 0\} \prec b_i$

-si il y a stérilisation→aller à l'étape 3
sinon : retour à l'étape 2
3èmeétape :
si $z_l \prec z_u \Longrightarrow z_l = z_u$
4èmeétape :retour en arrière
·on remonte jusqu'au dernier sommet précédent et ayant une branche non encore explorée
·retourner à l'étape 2
FIN : on s'arrête quand tous les sommets_feuilles sont stérilisés

4.3.4 Exemple numérique

$$\begin{cases} \max z = 2x_1 - x_2 + 5x_3 - 3x_4 + 4x_5 \\ 3x_1 - 2x_2 + 7x_3 - 5x_4 + 4x_5 \leq 6 \\ x_1 - x_2 + 2x_3 - 4x_4 + 2x_5 \leq 0 \\ x_j \in \{0,1\} \quad \forall j = 1,...,5 \end{cases}$$

on se ramène à un problème de minimisation $maxz = min(-z)$

$$\begin{cases} \min \bar{z} = -2x_1 + x_2 - 5x_3 + 3x_4 - 4x_5 \\ 3x_1 - 2x_2 + 7x_3 - 5x_4 + 4x_5 \leq 6 \\ x_1 - x_2 + 2x_3 - 4x_4 + 2x_5 \leq 0 \\ x_j \in \{0,1\} \quad \forall j = 1,...,5 \end{cases}$$

on a des c_j négatifs donc :
on pose :

$$y_1 = 1 - x_1 \rightarrow x_1 = 1 - y_1$$

$$y_3 = 1 - x_3 \rightarrow x_3 = 1 - y_3$$

$$y_5 = 1 - x_5 \rightarrow x_5 = 1 - y_5$$

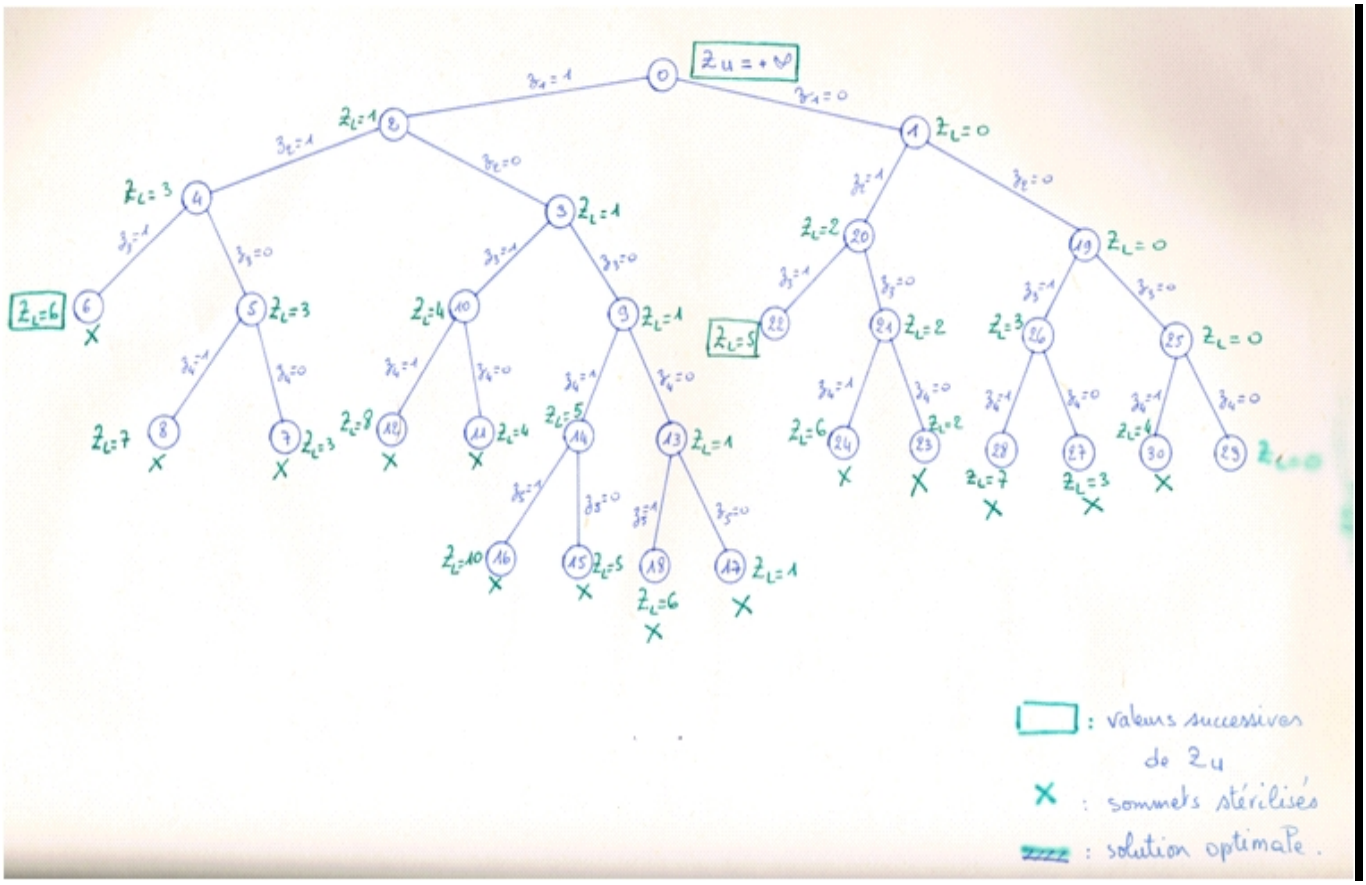
$\bar{z} = -2(1 - y_1) + x_2 - 5(1 - y_3) + 3x_4 - 4(1 - y_5)$
$= 2y_1 + 5y_3 + x_2 + 3x_4 + 4y_5 - 11$
on classe les c_j par ordre croissant :
$\bar{z} = x_2 + 2y_1 + 3x_4 + 4y_5 + 5y_3 - 11$
$\bar{z} = z_1 + 2z_2 + 3z_3 + 4z_4 + 5z_5 - 11$
$z_1 = x_2$
$z_2 = y_1 = 1 - x_1$
$z_3 = x_4$
$z_4 = y_5 = 1 - x_5$
$z_5 = y_3 = 1 - x_3$
on tire les x_i :
$x_1 = 1 - z_2$
$x_2 = z_1$
$x_3 = 1 - z_5$
$x_4 = z_3$

$x_5 = 1 - z_4$
d'où le programme linéaire devient , en fonction de z

$$\begin{cases} \min \bar{z} = z_1 + 2z_2 + 3z_3 + 4z_4 + 5z_5 - 11 \\ 3(1-z_2) - 2z_1 + 7(1-z_5) - 5z_3 + 4(1-z_4) \leq 6 \\ (1-z_2) - z_1 + 2(1-z_5) - 4z_3 + 2(1-z_4) \leq 0 \end{cases}$$

$$\begin{cases} \min \bar{z} = z_1 + 2z_2 + 3z_3 + 4z_4 + 5z_5 - 11 \\ z_1 + 3z_2 + 5z_3 + 4z_4 + 7z_5 \geq 8 \\ z_1 + z_2 + 4z_3 + 2z_4 + 2z_5 \geq 5 \\ z_i \in \{0,1\} \ \ \forall i = \overline{1,5} \end{cases}$$

note : la consante -11 sera prise en compte à l'obtention de la solution optimale



solution optimale : $\bar{z}^* = 5 - 11 = -6$ d'où $z^* = 6$
programme : $(z_1^*, z_2^*, z_3^*, z_4^*, z_5^*) = (0, 1, 1, 0, 0)$
on transforme pour tirer les x_i^* : $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*) = (0, 0, 1, 1, 1)$

Conclusion

Nous avons utilisé plusieurs méthodes classiques en informatique pour résoudre soit de manière approchée, soit de manière exacte. Nous avons focalisé notre étude simplement sur le problème du Sac à Dos unidimensionnel. Une extension possible serait d'étudier le problème du sac à dos multi-dimensionnel afin de comparer la résistance des méthodes employées

au passage aux dimensions supérieures. Une autre extension possible est d'effectuer une analyse de sensibilité des objets : supposons que nous ayons des objets, et une solution S pour le problème associé. Si nous modifions le poids ou/et l'utilité d'un objet ou plusieurs objets, devons-nous tout recommencer ? Pourrions-nous effectuer quelques changements sans trop perturber la solution ? Ces questions méritent que l'on s'y intéresse.

.

Bibliographie

[1] **Gondran michel. et Minoux Michel :**Graphes et algorithmes ; 4^e édition 517-122.3

[2] **Minoux.Michel :***programation mathématique :théori et algorithmes ,*2^{eme} édition 517-79

[3] **AZIEZ MARZAGHA** :l'algorithme de BALAS ;compte_rendu d'éxposé février 1986

[4] **Marc SIBERT** Version du 2 décembre 2004 Recherche Opérationnelle

[5] **Kevin SOL** Directeur de recherche : Rodolphe GIROUDEAU Université de Montpellier 2 Juin 2007

[6] **Julien JORGE** Thèse de Doctorat de l'Université de Nantes 2010

[7] **Hans Kellerer**, Ulright Pferschy et David Pisinger, Knapsack Problems, Springer, 2004