Résolution efficace de problèmes en variables 0-1 avec contraintes en égalité

L. Gely, M. Elkihel, D. El Baz *

LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse CEDEX 4, France
lgely@laas.fr elkihel@laas.fr elbaz@laas.fr

Abstract. On considère des problèmes en variables 0-1 avec contraintes en égalité. Généralement les méthodes de branch and bound et la programmation dynamique sont difficiles à adapter à cette classe de problèmes. Afin de faire face à ces difficultés on donne une formulation équivalente du problème basée sur l'utilisation de contraintes en inégalité et on présente des méthodes de relaxation Lagrangienne qui sont particulièrement adaptées à ce contexte. Des résultats originaux qui concernent l'existence d'une solution admissible sont donnés. On propose aussi un algorithme itératif simple et performant conçu afin d'obtenir une solution optimale du problème ou de conclure qu'il n'existe pas de solution admissible. Finalement, une série préliminaire d'expérimentations numeriques est présentée et analysée.

Mots-Clefs. Programmation entière, relaxation, admissibilité.

1 Introduction

De nombreux problèmes de recherche opérationnelle peuvent être écrits sous la forme de problèmes en variables 0-1 avec contraintes en égalité. On peut citer par exemple des problèmes d'affectation, de voyageur de commerce, de localisation de dépots et des problèmes de flots dans les réseaux à charge fixe, cf. [10], [12] et [13]. Bien que de nombreuses méthodes comme la méthode de branch and bound, la programmation dynamique et les méthodes heuristiques aient été conçues pour résoudre des problèmes de sac à dos en variables 0-1 avec des contraintes en inégalité ; à notre connaissance, il existe très peu de méthodes spécifiques dédiées à la résolution de problèmes avec contraintes en égalité, on citera principalement l'approche basée sur les méthodes de plan de coupe conçue initialement pour des problèmes de programmation entière, cf. [2] et [13].

En général, les problèmes de sac à dos en variables 0-1 avec contraintes en inégalité sont plus simples à résoudre que les problèmes avec contraintes en égalité ; par conséquent, un des objectifs principaux de cette étude est de déduire une formulation équivalente de ces problèmes qui soit basée sur l'utilisation de

* correspondant: Didier El Baz, LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse CEDEX 4, France, email: elbaz@laas.fr

contraintes en inégalité. Etant donné que chaque contrainte en égalité peut être réécrite comme deux inégalités, nous proposons d'utiliser une approche basée sur des méthodes de relaxation Lagrangienne. On obtient alors un problème relaxé avec contraintes en inégalité et l'on montre qu'il existe des multiplicateurs pour lesquels on peut déduire soit qu'une solution optimale du problème relaxé est solution optimale du problème d'origine, soit que le problème d'origine n'admet pas de solution admissible. Ce résultat étend les travaux de Plateau et Elkihel effectués dans le cas monocontrainte, cf. [1] et [11] et aussi [8]. L'utilisation de méthodes de pénalité est originale et particulièrement adaptée au cas des problèmes en variables 0-1 avec contraintes en égalité. Toutefois, plus les multiplicateurs sont grands, plus le problème est difficile En tenant compte de cette dernière remarque, nous proposons une méthode itérative exacte et performante basée sur la relaxation Lagrangienne ; cette méthode qui converge en temps fini nous permet de tirer des conclusions en matière d'admissibilité concernant le problème d'origine ; elle fournit aussi une solution optimale du problème d'origine si celui-ci admet au moins une solution admissible.

Le paragraphe 2 traite de la formulation du problème. Une formulation équivalente avec contraintes en inégalité est donnée à la section 3, où sont aussi présentés les principaux résultats de cet article qui portent sur l'existence d'une solution admissible. Le paragraphe 4 présente des exemples simples qui illustrent la théorie. Un algorithme itératif original qui permet de déduire une solution optimale du problème d'origine ou de conclure qu'il n'existe pas de solution admissible est proposé à la section 5. Une série préliminaire de résultats expérimentaux est présentée et analysée au paragraphe 6.

2 Formulation du problème

Nous considérons le problème en variables 0-1 avec contraintes en égalité suivant qui est noté noté (PCE),

$$\begin{aligned} & z = \max cx, \\ \text{s.c.} \quad & Ax = b, \\ & x \in \{0,1\}^n, \end{aligned} \tag{PCE}$$

où A est une matrice (m, n) dont les coefficients sont des entiers non négatifs et b et c sont des vecteurs de taille m et n respectivement, dont les composantes sont des entiers positifs. De plus, on fait les hypothèses suivantes,

$$\max_{1 \leq j \leq n} a_{ij} \leq b_i < \sum_{j=1}^{n} a_{ij}, \ \forall i \in \{1, \ldots, m\}, \tag{1}$$

ainsi, on n'aura pas de décision triviale.

3 Formulation avec contraintes en inégalité

Une formulation équivalente à (PCE) peut étre donnée comme suit,

$$\begin{aligned} & z = \max cx, \\ \text{s.c.}\quad & Ax \leq b, \\ & Ax \geq b, \\ & x \in \{0,1\}^n. \end{aligned} \tag{E}$$

Nous introduisons l'ensemble S défini de la manière suivante.

$$S = \{x \in \{0,1\}^n \mid Ax \leq b\}. \tag{2}$$

Considérons la relaxation Lagrangienne de (E) par rapport à $Ax \geq b$, cf. [10],

$$v(u) = \max_{x \in S}\{cx + u(Ax - b)\}, \tag{Ru}$$

où u est un vecteur non négatif de dimension m. Nous rappelons maintenant un résultat bien connu de la littérature, cf. [4].

Théorème 1. Si pour un vecteur $u \in R^m_+$, le vecteur $x \in S$ satisfait les conditions suivantes :
(a) x est optimal pour (Ru),
(b) $Ax \geq b$,
alors, x est une solution optimale du problème (PCE).

Lemme 1. La fonction v est linéaire par morceaux non croissante et convexe.

Preuve. Notons tout d'abord que S est un ensemble fini. Pour tout $x \in S$, on considère la fonction $f_x : R^m \to R$, définie comme suit,

$$f_x(u) = cx + u(Ax - b). \tag{3}$$

De manière claire, f_x est une fonction linéaire non croissante. Puisque v correspond à un maximum sur un ensemble fini de fonctions linéaires non croissantes, il s'en suit que la fonction v est linéaire par morceaux, de plus v est aussi non croissante et convexe.

Nous présentons maintenant un des résultats principaux de cet article. Ce résultat généralise un résultat de Plateau et Elkihel pour des problèmes en variables 0-1 à un seule contrainte, cf. [1] et [11], voir aussi [8].

Théorème 2. Le problème (PCE) ne possède pas de solution admissible si et seulement s'il existe un vecteur non negatif $u \in R^n$ tel que $v(u) \leq 0$.

Preuve. supposons qu'il existe un vecteur $u \geq 0$ tel que $v(u) \leq 0$ et x est une solution admissible de (PCE). Alors, $cx + u(Ax - b) \leq v(u) \leq 0$. Toutefois, nous avons $Ax = b$, puisque x est une solution admissible de (PCE). Il s'en suit que $cx \leq 0$, ce qui conduit à une contradiction puisque $c > 0$, $x \in \{0,1\}^n$ et $b > 0$. On montre maintenant la réciproque. Supposons que le problème (PCE) n'a pas de solution admissible. Alors, on peut réécrire S comme suit.

$$S = \{x \in \{0,1\}^n \mid Ax \leq b \text{ et } \exists\, l \in \{1,\ldots,m\} \text{ avec } A_l x < b_l\}, \tag{4}$$

où A_l représente la l-ème ligne de la matrice A et b_l la l-ème composante du vecteur b.

En prenant en compte l'inégalité (1) et (4), il existe toujours un vecteur $x(i) \in S$ tel que

$$x(i) = \arg\min_{x \in S}\{b_i - A_i x \mid b_i - A_i x > 0\},\ i \in \{1,...,m\}. \tag{5}$$

En utilisant les notations suivantes :

$$p = \arg \min_{i \in \{1,\ldots,m\}}\{b_i - A_i x(i)\}, \tag{6}$$

et

$$\underline{x} = x(p), \tag{7}$$

on a

$$b_p - A_p \underline{x} > 0. \tag{8}$$

On introduit le vecteur $\hat{u}$, dont les composantes $\hat{u}_i$ sont définies de la manière suivante :

$$\hat{u}_i = \frac{d}{b_p - A_p \underline{x}} > 0, \forall i \in \{1,\ldots,m\}, \tag{9}$$

où $d = \sum_{i=1}^{n} c_i$. On a alors

$$v(\hat{u}) = c\hat{x} + \sum_{i=1}^{m} \hat{u}_i (A_i \hat{x} - b_i), \tag{10}$$

où $\hat{x}$ est une solution optimale du problème $(R\hat{u})$. Par conséquent, on a

$$\begin{aligned} v(\hat{u}) &= c\hat{x} + \sum_{i=1}^{m} \frac{d}{b_p - A_p \underline{x}} (A_i \hat{x} - b_i), \\ &= c\hat{x} - d \sum_{i=1}^{m} \frac{b_i - A_i \hat{x}}{b_p - A_p \underline{x}}, \end{aligned} \tag{11}$$

Il résulte de (8) et de la formulation du problème (Ru), que pour tout $i \in \{1,\ldots,m\}$, on a

$$\frac{b_i - A_i \hat{x}}{b_p - A_p \underline{x}} \geq 0. \tag{12}$$

Il résulte de (4), (7) et (8) qu'il existe un indice $l \in \{1, \ldots, m\}$ tel que

$$\frac{b_l - A_l \hat{x}}{b_p - A_p \underline{x}} \geq 1. \tag{13}$$

De plus, il résulte de (1) que l'on a

$$c\hat{x} < d. \tag{14}$$

Par conséquent, $v(\hat{u}) < 0$. Il résulte de la continuité de la fonction linéaire par morceaux v, qu'il existe un $u \in R_+^m$ tel que $v(u) = 0$. Ce qui termine la preuve.

Corollaire 1. Si le problème (PCE) possède une solution admissible, alors il existe un $u^\star$ tel que $v(u)$ est une fonction constante pour tout $u \geq u^\star$. De plus, on a $v(u^\star) = cx^\star > 0$, où $x^\star$ est solution optimale du problème (PCE).

Preuve. Il résulte du Théorème 2 que si le problème (PCE) possède une solution admissible, alors il n'existe pas de u tel que $v(u) \leq 0$. De plus, il résulte du Lemme 1 qu'il existe un vecteur $u^\star$ tel que $v(u) = v(u^\star) > 0$ pour tout $u \geq u^\star$, puisque S est un ensemble fini. Enfin, il résulte de (Ru) que $v(u^\star) = cx^\star > 0$.

Remarque 1. Il résulte de la preuve du Théorème 2, que le vecteur $\overline{u}$ de composantes $\overline{u}_i = d = \sum_{i=1}^n c_i$, pour tout $i \in \{1, ..., m\}$, est suffisamment grand afin de permettre de déduire une solution optimale du problème (PCE) si elle existe, ou de conclure que le problème (PCE) ne possède pas de solution admissible. De manière claire, si le problème (PCE) possède une solution admissible, alors $v(u) = v(\overline{u})$ pour tout $u \geq \overline{u}$.

4 Exemples

Nous illustrons la théorie exposée ci-dessus au moyen de deux exemples avec deux contraintes en égalité. Le premier problème qui ne possède aucune solution admissible est le suivant.

$$\begin{array}{ll}
\max & 7x_1 + 2x_2 + 3x_3 + 2x_4 + 6x_5 + 4x_6, \\
\text{s.c.} & 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 = 20, \\
& 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 = 36, \\
& x \in \{0,1\}^6.
\end{array} \tag{15}$$

Le problème (Ru) associé à (15) peut être écrit de la manière suivante,

$$\begin{array}{ll}
v(u) = \max & (7 + 12u_1 + 3u_2)x_1 + (2 + 2u_1 + 8u_2)x_2 + (3 + 3u_1 + 12u_2)x_3 \\
& +(2 + 2u_1 + 13u_2)x_4 + (6 + 4u_1 + 20u_2)x_5 + (4 + 3u_1 + 14u_2)x_6 \\
& -(20u_1 + 36u_2), \\
\text{s.c.} & 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 \leq 20, \\
& 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 \leq 36, \\
& x \in \{0,1\}^6.
\end{array} \tag{16}$$

En tenant compte de la Remarque 1 on considère le cas particulier où les composantes du vecteur u sont données de la manière suivante,

$$u_1 = u_2 = \sum_{i=1}^{6} c_i = 24. \tag{17}$$

On a

$$c + uA = (367, 242, 363, 362, 582, 412),\ ub = 1344. \tag{18}$$

Un calcul simple, effectué par exemple par une méthode de programmation dynamique, donne :

$$v(u) = -10, \tag{19}$$

pour

$$x = (1, 1, 1, 1, 0, 0). \tag{20}$$

Ce qui implique que le problème ne possède pas de solution admissible. Notons que dans ce cas on a :

$$\begin{gathered} 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 = 19 < 20, \\ 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 = 36. \end{gathered} \tag{21}$$

Le second problème diffère du premier seulement par le terme de droite de la première égalité. Ce problème possède une solution admissible.

$$\begin{array}{ll} \max & 7x_1 + 2x_2 + 3x_3 + 2x_4 + 6x_5 + 4x_6, \\ \text{s.c.} & 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 = 19, \\ & 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 = 36, \\ & x \in \{0,1\}^6. \end{array} \tag{22}$$

Le problème (Ru) associé à (22) s'écrit comme suit.

$$\begin{array}{ll} v(u) = \max & (7 + 12u_1 + 3u_2)x_1 + (2 + 2u_1 + 8u_2)x_2 + (3 + 3u_1 + 12u_2)x_3 \\ & +(2 + 2u_1 + 13u_2)x_4 + (6 + 4u_1 + 20u_2)x_5 + (4 + 3u_1 + 14u_2)x_6 \\ & -(19u_1 + 36u_2), \\ \text{s.c.} & 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 \leq 19, \\ & 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 \leq 36, \\ & x \in \{0,1\}^6. \end{array} \tag{23}$$

Considérons de nouveau le cas où les composantes du vecteur u sont données de la manière suivante,

$$u_1 = u_2 = \sum_{i=1}^{6} c_i = 24. \tag{24}$$

En utilisant de nouveau la programmation dynamique on obtient cette fois :

$$v(u) = 14, \tag{25}$$

pour

$$x = (1, 1, 1, 1, 0, 0). \tag{26}$$

De plus, on a dans ce cas

$$\begin{aligned} 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 = 19, \\ 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 = 36, \end{aligned} \tag{27}$$

ce qui signifie que les conditions du Théorème 1 sont satisfaites et que x est solution optimale du problème (PCE). De plus on a

$$cx = v(u) = 14. \tag{28}$$

Remarque 2. En considérant des vecteurs u de la forme $u = \lambda e$, où e est le vecteur de dimension m dont toutes les composantes sont égales à un, la plus petite valeur entière de λ pour laquelle on peut déduire une solution optimale pour le second exemple est $\lambda = 3$. Ce résultat a été obtenu expérimentalement.

5 Algorithmes itératifs

Le problème (Ru) peut être résolu au moyen de méthodes classiques comme la programmation dynamique, cf. [3] ou la méthode de branch and bound, cf. [9], ainsi que par des techniques hybrides, comme par exemple les méthodes de branch and cut, cf. [13]. L'utilisation de codes industriels, comme Cplex, cf. [5] et [7] ou Xpress, cf. [6] et [14], qui sont basés sur certaines techniques précédemment citées, peut s'avérer très attractive afin de résoudre ce type de problèmes, comme on le verra par la suite.

Plus le vecteur u est grand, plus on peut déduire de renseignements utiles sur le problème (PCE), c'est à dire : soit $v(u)$ n'est pas positif et par conséquent (PCE) n'admet pas de solution admissible ; soit $v(u') = v(u) > 0$, pour tout $u' \geq u$ (dans ce cas, on a $Ax - b = 0$, où x est solution optimale du problème (Ru)) et l'on peut déduire une solution optimale du problème (PCE). Toutefois, il convient de noter que plus le vecteur u est grand, plus le problème est difficile et par conséquent, plus le temps de résolution est important. En effet, les ratios coûts sur poids sont très proches les uns des autres lorsque le vecteur u est très grand ; d'où une grande difficulté à trouver une solution optimale, puisque les différents objets qui candidatent à l'entrée dans le sac à dos présentent des caractéristiques forts similaires et les méthodes de résolution ont par voie de conséquence d'importantes difficultés à discriminer les solutions.

Dans cette section, on propose un algorithme itératif exact et efficace qui est conçu afin de fournir un vecteur u dont les composantes, idéalement aussi petites que possible, sont suffisamment grandes afin de déduire soit la valeur d'une solution optimale, soit que le problème (PCE) n'admet pas de solution admissible. Notre algorithme est assimilable à une méthode de descente basée sur les résultats des Lemme 1, Théorème 2 et Corollaire 1, qui impliquent principalement que $v(u)$ est une fonction convexe, non croissante, linéaire par morceaux telle qu'il existe un vecteur u avec $v(u) \leq 0$ ou $v(u') = v(u) > 0$, pour tout $u' \geq u$; c'est à dire : $Ax - b = 0$, où x est solution optimale du problème (Ru). L'algorithme que nous proposons est basé sur une méthode itérative qui engendre une suite $\{u_k\}$ commençant par un vecteur u petit, noté u_0 et donnant une approximation du plus petit u qui permet soit de déduire une solution optimale du problème soit de conclure que le problème ne possède pas de solution admissible.

Par souçi de simplicité et sans perte de généralité, nous proposons une méthode exacte basée sur l'utilisation de multiplicateurs entiers, dont on peut montrer la convergence vers une solution optimale en temps fini. De manière plus précise, notre algorithme utilise des vecteurs u de la forme suivante :

$$u = \lambda e, \tag{29}$$

où e, nous le rappelons, est le vecteur de dimension m dont toutes les composantes sont égales à un. L'algorithme commence en utilisant une valeur de λ petite ; cette valeur est notée λ_0. Par exemple, on peut prendre : $\lambda_0 = 0,1$ ou $\min_{i \in \{1,\ldots,n\}} c_i$. Il résulte de (Ru) :

$$v(\lambda e) = cx(\lambda) + \lambda e(Ax(\lambda) - b), \tag{30}$$

où $x(\lambda)$ est une solution optimale du problème $(R\lambda e)$. Par simplicité, $v(\lambda e)$ sera aussi noté par la suite $v(\lambda)$.

On définit la suite $\{\lambda_k\}$ de manière récursive, comme suit. Pour $k = 0, 1, \ldots$ si $v(\lambda_k) \leq 0$, l'algorithme se termine en concluant qu'il n'y a pas de solution admissible. Si $e(Ax(\lambda_k) - b) = 0$, l'algorithme se termine et donne une solution optimale $x(\lambda_k)$ du problème (PCE). Sinon (c'est à dire si $v(\lambda_k) > 0$ et $e(Ax(\lambda_k) - b) < 0$), en considérant la droite de pente $a_k = e(Ax(\lambda_k) - b) < 0$ et d'équation $y = a_k z + d_k$, où $d_k = v(\lambda_k) - a_k \lambda_k$, qui inclut le point $(\lambda_k, v(\lambda_k))$, la méthode recherche une valeur λ_{k+1} telle que

$$a_k \lambda_{k+1} + d_k = 0, \tag{31}$$

puisque la fonction $v(\lambda)$ est linéaire par morceaux, continue et non croissante. Afin d'engendrer une suite monotone croissante d'entiers, nous construisons λ_{k+1} de la manière suivante,

$$\lambda_{k+1} = \lambda_k + \lceil \frac{-v(\lambda_k)}{a_k} \rceil, \tag{32}$$

où $\lceil \alpha \rceil$ représente le plus petit entier supérieur ou égal au réel α.

Théorème 3. La suite $\{\lambda_k\}$ converge en temps fini vers une valeur $\lambda_{\hat{k}}$ telle que $v(\lambda_{\hat{k}}) \leq 0$ ou $a_k = e(Ax(\lambda_{\hat{k}}) - b) = 0$.

Preuve. Si $v(\lambda_0) \leq 0$ ou $a_0 = 0$, alors l'algorithme se termine dès l'itération 0 et l'on déduit clairement qu'il a convergé. Si $v(\lambda_0) > 0$ et $a_0 < 0$, on montre par induction que l'algorithme engendre une suite monotone croissante d'entiers. Tout d'abord, on a $\lambda_1 > \lambda_0$, puisque $a_0 < 0$ et $v(\lambda_0) > 0$ et il résulte de (32) que

λ_1 est un entier. De manière similaire, on montre que si $v(\lambda_k) > 0$ et $a_k < 0$, on a $\lambda_{k+1} > \lambda_k$ et λ_{k+1} est un entier. Montrons maintenant que la suite $\{\lambda_k\}$ converge en temps fini. S'il n'y a pas de solution admissible, Il résulte du Théorème 2 et du Corollaire 1 qu'il existe un vecteur positif u tel que $v(u) \leq 0$. Sinon, il existe un vecteur positif $u^\star$ tel que $v(u) = v(u^\star) \geq 0$, pour tout $u \geq u^\star$ (on a alors, $Ax^\star - b = 0$, où $x^\star$ est solution optimale du problème $(Ru^\star)$). Par conséquent, il existe un $\hat{k}$ fini tel que $\lambda_{\hat{k}} e \geq u$ ou $\lambda_{\hat{k}} e \geq u^\star$.

Remarque 3. Dans le cas où l'on connait une solution admissible du problème (PCE), on peut concevoir un algorithme légèrement différent de celui que nous venons de présenter. En effet, $v(\lambda)$ est alors borné par en dessous par la constante $c\underline{x}$, où $\underline{x}$ est une solution admissible du problème (PCE). On note que le coût optimal satisfait alors $v(u^\star) = cx^\star \geq c\underline{x}$. On peut donc rechercher un λ_{k+1} satisfaisant

$$a_k \lambda_{k+1} + d_k = c\underline{x}. \tag{33}$$

Plus exactement, nous choisirons λ_{k+1} de la manière suivante, afin d'engendrer une suite monotone croissante d'entiers qui converge.
Si $v(\lambda_k) \neq c\underline{x}$ alors

$$\lambda_{k+1} = \lambda_k + \lceil \frac{c\underline{x} - v(\lambda_k)}{a_k} \rceil, \tag{34}$$

sinon

$$\lambda_{k+1} = \lambda_k + 1. \tag{35}$$

Le test mis en œuvre ci-dessus est nécessaire afin d'éviter que l'algorithme ne boucle. Cette situation se produit dans le cas où $\underline{x}$ est une solution optimale du problème (PCE). Notons aussi que l' incrément produit à chaque itération est plus petit que dans le cas de l'algorithme précédent, donné par l'équation (32).

Remarque 4. Les méthodes itératives de descente proposées dans cette section présentent de nombreuses similarités avec des méthodes duales de recherche de la solution du problème d'optimisation suivant,

$$min_{u \geq 0} \; v(u). \tag{36}$$

Nous concluons cette section par quelques remarques générales ayant un rapport moins direct avec le sujet de notre étude.

Remarque 5. L'approche originale développée dans cet article pour des problèmes difficiles avec contraintes en égalité suggère une manière intéressante de créer des problèmes difficiles de sac à dos avec contraintes en inégalité.

Remarque 6. Pour finir, notons que notre approche des problèmes avec contraintes en égalité peut aussi être adaptée à des problèmes d'admissibilité de la forme suivante,

$$\text{trouver } x \in \{0,1\}^n \text{ tel que } Ax = b. \tag{37}$$

Dans ce cas, on peut introduire une fonction $v(u)$ comme suit,

$$v(u) = max_{x \in S}\ u(Ax - b), \tag{38}$$

où l'on doit nécessairement avoir $u > 0$ afin de conclure s'il existe une solution admissible. Toutefois, le problème (37) diffère de (PCE) par sa nature. Les hyperplans $f_x(u) = u(Ax - b)$ ont une intersection à l'origine ; par conséquent, la fonction v est, dans ce cas, non croissante et linéaire. Il en résulte que s'il n'y a pas de solution admissible, $v(u) < 0$ pour tout $u > 0$. De plus, s'il existe une solution admissible $v(u) = 0$ pour tout $u > 0$. En conséquence, l'existence d'une solution admissible peut être testée ici pour des u petits.

6 Résultats numériques

Dans ce paragraphe, nous présentons une première série de résultats numériques. Les résultats ont été obtenus en utilisant le code Xpress, cf. [6] et [14]. En particulier, l'algorithme itératif presenté à la section 5 a été mis en œuvre simplement en résolvant de manière répétitive le problème (Ru) avec le logiciel Xpress à partir d'une valeur initiale λ_0 et en modifiant λ_{k+1} suivant (32).

Les Tables 1 à 3 présentent les résultats numériques pour 5 problèmes avec 30 variables et 2 contraintes, c'est à dire : $n = 30$ et $m = 2$. Nous avons considéré des problèmes qui possèdent une solution admissible (pour tout $i \in \{1, ..., n\}, b_i = \sum_{j=1}^{10} a_{ij}$), afin de comparer les algorithmes itératifs présentés à la section 5 avec la procedure de base de Xpress utilisée pour résoudre le problème (PCE). Les poids a_{ij} et les coûts c_j sont distribués uniformément dans l'intervalle $[1, 1000]$. Nous avons considéré la résolution des différents problèmes au moyen de plusieurs approches : la méthode de résolution de base de Xpress, notée Basique, appliquée directement au problème (PCE), la méthode itérative proposée à la section 5, notée IM1 et une version légèrement différente présentée à la Remarque 3 de la Section 5, notée IM2. Les méthodes itératives ont été mises en œuvre pour deux valeurs différentes de λ_0 : $\lambda_0 = 1$ et $\lambda_0 = \min_{i \in \{1,..,n\}} c_i$. La Table 1 présente les temps de calcul en secondes des différentes méthodes. La Table 2 donne le nombre d'iterations des deux méthodes itératives. La Table 3 présente pour IM1 et IM2 la valeur de λ_k lorsque le critère d'arrêt est satisfait. La Table 4 illustre le comportement des algorithmes iteratifs en donnant le temps de calcul et la valeur de λ_k à chaque itération dans le cas particulier du Problème 3.

La Table 4 permet de constater que plus la valeur de λ_k est grande, plus le temps de calcul d'une itération est grand. Par exemple, une itération prend moins d'une seconde pour $\lambda = 1$ et 1521 secondes pour $\lambda = 6705$.

La Table 1 nous permet de noter qu'IM1 et IM2 sont en général plus rapides que la méthode de base de Xpress ; ce qui montre l'intérêt de l'approche développée dans cette étude. En effet, lorsqu'on utilise Xpress, il apparaît que la résolution d'une suite de problèmes avec une formulation basée sur la relaxation Lagrangienne et présentant des contraintes en inégalité est plus rapide que la résolution du problème (PCE) original présentant des contraintes en égalité. Ce résultat nous permet de penser que le temps de calcul pourrait être par ailleurs

Table 1. Temps de calcul en secondes

Méthode / λ_0 / Problème	Basique	IM1		IM2	
		1	$\min_i c_i$	1	$\min_i c_i$
1	1596	927	904	895	893
2	2018	1192	1206	1184	1151
3	2188	1526	1153	1554	1573
4	14183	11169	10894	15797	15822
5	3717	2942	2941	2741	2774

Table 2. Nombre d'iterations

Méthode / λ_0 / Problème	IM1		IM2	
	1	$\min_i c_i$	1	$\min_i c_i$
1	3	2	3	2
2	3	2	4	2
3	3	2	4	3
4	3	2	4	3
5	3	3	3	3

réduit de manière importante si nous utilisions des algorithmes efficaces dédiés à la résolution de problèmes de sac à dos avec contraintes en inégalité comme, par exemple, les algorithmes présentés dans [3].

La Table 2 nous permet de constater que l'algorithme IM2 requiert en général plus d'itérations qu'IM1. Toutefois, IM2 est parfois plus rapide qu'IM1. La performance réelle d'IM1 et IM2 dépend à la fois du nombre d'itérations et des valeurs associées de λ_k, cf. Tables 2 à 4. La Table 2 nous permet aussi de noter que les algorithmes IM1 et IM2 nécessitent généralement plus d'itérations avec $\lambda_0 = 1$, qu'avec $\lambda_0 = \min_{i \in \{1,..,n\}} c_i$. Toutefois, au vu de la Table 1, il est difficile de déterminer la meilleure valeur de λ_0. De manière générale on peut dire que plus λ_0 est petit et plus il faudra itérer, cependant, il faut remarquer que les problèmes à résoudre à chaque itération seront plus faciles ; ces divers aspects sont mis en évidence dans la Table 4.

References

1. M. Elkihel, Programmation dynamique et rotation de contraintes pour les problèmes d'optimisation entière, *Thèse de Doctorat de l'Université des Sciences et Techniques de Lille*, (1984).
2. R. Garfinkel, G. Nemhauser, *Integer Programming*, John Wiley & Sons, New York (1972).
3. B. Gavish, H. Pirkul, Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality, *Mathematical Programming*, **31**, 78-105 (1985).

Table 3. Valeur de λ_k lorsque le critère d'arrêt est satisfait

Méthode	IM1		IM2	
λ_0 Problème	1	$\min_i c_i$	1	$\min_i c_i$
1	602	602	348	348
2	1128	1170	378	378
3	6705	1163	2537	2537
4	536	536	1460	1460
5	3779	3779	1898	1898

Table 4. Durée de calcul et valeur de λ_k à chaque itération du Problème 3

Méthode	IM1				IM2			
λ_0	1		$\min_i c_i$		1		$\min_i c_i$	
	temps	λ_k	temps	λ_k	temps	λ_k	temps	λ_k
Itération k								
1	0	1	1	23	0	1	1	23
2	5	69	1152	1163	2	29	66	468
3	1521	6705	-	-	64	468	1506	2537
4	-	-	-	-	1488	2537	-	-

4. A. M. Geoffrion, Lagrangian relaxation for integer programming, *Mathematical Programming Study*, **2**, 82-114 (1974).
5. K. Holmström, A. Göran, M. Edvall, *User's guide for Tomlab Cplex v9.0*, (2004).
6. K. Holmström, A. Göran, M. Edvall, *User's guide for Tomlab Xpress R2004*, (2004).
7. *Ilog Ampl Cplex System User's Guide*, (2002).
8. L. Kaufman, F. Plastria, S. Tubeeckx, The zero-one knapsack problem with equality constraint, *European Journal of Operational Research*, **19**, 384-389 (1985).
9. T. L. Morin, R. E. Marsten, Branch and bound strategies for dynamic programming, *Operations Research*, **24**, (1976).
10. G. Nemhauser, L. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York (1988).
11. G. Plateau, M. Elkihel, A hybrid method for the 0-1 knapsack problem, *Methods of Operation Research* **49**, 277-293 (1985).
12. A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, New York (1998).
13. L. Wolsey, *Integer Programming*, John Wiley & Sons, New York (1998).
14. *Xpress-MP user guide and reference manual*, (1997).