Résolution d'un problème combinatoire fractionnaire par la programmation linéaire mixte

Alain Billionnet et Karima Djebali
CEDRIC-IIE, 18 allée Jean Rostand, 91025 Evry cedex, France
billionnet,djebali@iie.cnam.fr

Résumé

Nous considérons dans ce papier un problème d'optimisation combinatoire fractionnaire : la maximisation de la somme de plusieurs ratios hyperboliques en 0/1. Contrairement à la maximisation d'un seul ratio, très peu d'auteurs se sont intéressés à ce problème. Nous proposons trois formulations par des programmes linéaires en variables mixtes et trois stratégies pour les résoudre :

– une résolution directe en utilisant un outil général de programmation linéaire en variables mixtes (Cplex) ;
– une résolution heuristique également exploitée pour améliorer la première stratégie en fournissant au solveur une bonne solution admissible initiale ;
– une résolution par un algorithme branch-and-bound spécifique, fondé sur une réécriture plus précise des programmes en chaque nœud de l'arbre de recherche, cette réécriture consistant à ajuster les coefficients des programmes linéaires en variables mixtes.

Nous présentons des résultats expérimentaux permettant de comparer les différentes stratégies.

Mots-clés : Optimisation combinatoire, Somme de ratios hyperboliques, Programmation linéaire en variables mixtes, Branch-and-bound.

1 Introduction

Les programmes fractionnaires linéaires ou non linéaires, en continu, en nombres entiers ou en variables 0-1 apparaissent dans plusieurs domaines tels que les bases de données, l'optimisation combinatoire, la programmation stochastique et l'économie [Stancu-Minasian, 1997]. De nombreuses applications des programmes fractionnaires ont été décrites dans la littérature [Schaible, 1983], [Schaible, 1995], [Hansen et al., 1991], [Falk et al., 1992], [Radzik, 1998], [Freund et al., 2001]. Le problème considéré ici est un problème en variables 0-1 qui consiste à optimiser la somme de plusieurs ratios hyperboliques. Il s'écrit comme suit :

$$(SRH)\left\{ Max \sum_{i=1}^{m} \frac{a_i + \sum_{j=1}^{n} a_{ij} x_j}{b_i + \sum_{j=1}^{n} b_{ij} x_j} : x \in \{0,1\}^n \right\}$$

où $a_i \in \mathbb{R}$, $b_i \in \mathbb{R}$, $a_{ij} \in \mathbb{R}$, $b_{ij} \in \mathbb{R}$. Nous prenons, de plus, l'hypothèse classique $(b_i + \sum_{j=1}^{n} b_{ij} x_j) > 0 \, \forall x \in \{0,1\}^n, \forall i \in \{1, ..., m\}$, et nous posons $N = \{1, ..., n\}$ et $M = \{1, ..., m\}$.

Diverses applications peuvent se modéliser sous la forme (SRH) (voir par exemple [Arora et al., 1977], [Gilmore et al., 1963], [Saipe, 1975]). Contrairement à la maximisation d'un seul ratio hyperbolique, peu d'études concernent ce problème. Saipe [Saipe, 1975] s'est intéressé à ce problème avec une contrainte de cardinalité et a proposé de le résoudre par un algorithme branch-and-bound. Plus récemment, Li [Li, 1994] et Wu [Wu, 1997] ont formulé le problème par un programme linéaire en variables mixtes (PLVM) et ont proposée de résoudre ce programme par un algorithme branch-and-bound standard. Cette technique de modélisation d'un programme non linéaire en variables 0-1 par un PLVM est bien connue et éprouvée mais il convient de la mettre en œuvre avec prudence car certaines formulations peuvent nécessiter des temps de calcul importants. Dans ce travail, nous présentons la formulation de [Wu, 1997] ainsi que trois nouvelles formulations. Pour la résolution des différentes linéarisations considerées, nous proposons trois approches. La première consiste à utiliser directement un outil standard de programmation linéaire mixte. La deuxième consiste en la résolution par une méthode heuristique. La solution approchée obtenue sera également exploitée pour améliorer la première approche. La troisième utilise un algorithme branch-and-bound spécifique fondé sur une réécriture plus précise des programmes en chaque nœud de l'arbre de recherche. Des résultats expérimentaux seront présentés afin de comparer les différentes linéarisations et approches de résolution.

2 Différentes linéarisations du problème

Dans cette partie, nous présentons différentes techniques de linéarisation qui transforment le problème SRH en un PLVM. Nous exposons tout d'abord la formulation de Wu [Wu, 1997] puis nous proposons trois nouvelles formulations.

2.1 Linéarisation de Wu [Wu, 1997] $(LIN1)$

Proposée par [Wu, 1997], cette linéarisation consiste tout d'abord à remplacer l'inverse des dénominateurs des ratios par des variables continues $y_i = 1/(b_i + \sum_{j=1}^{n} b_{ij}x_j)$ puis à linéariser le programme obtenu. On obtient ainsi, après changement de variables, le programme quadratique suivant :

$$(SRH') \begin{cases} Max \sum_{i=1}^{m}(a_i y_i + \sum_{j=1}^{n} a_{ij} y_i x_j) & \\ b_i y_i + \sum_{j=1}^{n} b_{ij} y_i x_j = 1 & i \in M \\ x_j \in \{0,\ 1\} & j \in N \end{cases}$$

Une façon classique de linéariser ce programme est de remplacer le produit de deux variables (une variable continue et une variable bivalente) par une seule variable tout en ajoutant des contraintes appelées contraintes de linéarisation.

Lemme 1 **([Wu, 1997]) :** Soit $x \in \{0,\ 1\}$, $y \geq 0$ et K une borne supérieure de y. Le produit xy peut être remplacé par la variable z soumise à l'ensemble de contraintes suivant :

$$\begin{cases} y - z \leq K(1-x) & (1) \\ z \leq y & (2) \\ z \leq Kx & (3) \\ z \geq 0 & (4) \end{cases}$$

La preuve est évidente en considérant successivement les deux valeurs possibles de la variable x.

Appliquons le lemme 1 au programme SRH', en remplaçant le produit $y_i x_j$ par z_{ij} pour tout $i \in M$ et pour tout $j \in N$. On obtient le theorème 1 où $\overline{y}_i$ une borne supérieure de y_i.

Théorème 1 : Soit $(x^*,\ y^*,\ z^*)$ une solution optimale de $LIN1$. x^* est une solution optimale de SRH.

$$(LIN1) \begin{cases} Max \sum_{i=1}^{m}(a_i y_i + \sum_{j=1}^{n} a_{ij} z_{ij}) & \\ b_i y_i + \sum_{j=1}^{n} b_{ij} z_{ij} = 1 & i \in M \\ z_{ij} \geq y_i - \overline{y}_i(1 - x_j) & i \in M,\ j \in N \\ z_{ij} \leq y_i & i \in M,\ j \in N \\ z_{ij} \leq \overline{y}_i x_j & i \in M,\ j \in N \\ x_j \in \{0,\ 1\},\ z_{ij} \geq 0 & i \in M,\ j \in N \end{cases}$$

La preuve est donnée dans [Djebali, 2003].

2.2 Première linéarisation proposée $(LIN2)$

L'idée de cette linéarisation est d'écrire la fonction objectif comme la somme de m variables r_i, $i \in M$ où $r_i = \left(a_i + \sum_{j=1}^{n} a_{ij}x_j\right) \Big/ \left(b_i + \sum_{j=1}^{n} b_{ij}x_j\right)$.

Le programme SRH après changement de variables devient :

$$(SRH'') \begin{cases} Max \sum_{i=1}^{m} r_i & \\ a_i + \sum_{j=1}^{n} a_{ij}x_j = b_i r_i + \sum_{j=1}^{n} b_{ij} r_i x_j & i \in M \\ x_j \in \{0,\ 1\} & j \in N \end{cases}$$

En remplaçant le produit $r_i x_j$ par la variable z_{ij} et en appliquant le lemme 1, nous obtenons le PLVM suivant :

$$\begin{cases} Max \sum_{i=1}^{m} r_i & \\ a_i + \sum_{j=1}^{n} a_{ij}x_j = b_i r_i + \sum_{j=1}^{n} b_{ij} z_{ij} & i \in M \\ z_{ij} \geq r_i - \overline{r}_i(1 - x_j) & i \in M,\ j \in N\ (1) \\ z_{ij} \leq r_i & i \in M,\ j \in N\ (2) \\ z_{ij} \leq \overline{r}_i x_j & i \in M,\ j \in N\ (3) \\ x_j \in \{0,\ 1\},\ z_{ij} \geq 0 & i \in M,\ j \in N \end{cases}$$

où $\overline{r}_i$ est une borne supérieure de r_i.

Remarquons que les contraintes (1)-(3) ne sont pas toutes nécessaires pour exprimer le fait que la variable z_{ij} est égale au produit $r_i x_j$. On distinguera 2 cas : $b_{ij} > 0$ et $b_{ij} < 0$, ce qui donne le théorème suivant.

Théorème 2 : Soit $(x^*,\ y^*,\ z^*)$ une solution optimale de $LIN2$. x^* est une solution optimale de SRH.

$$(LIN2) \begin{cases} Max \sum_{i=1}^{m} r_i & \\ a_i + \sum_{j=1}^{n} a_{ij}x_j = b_i r_i + \sum_{j=1}^{n} b_{ij} z_{ij} & i \in M \\ z_{ij} \geq r_i - \overline{r}_i(1 - x_j) \mid si\ b_{ij} > 0 & i \in M,\ j \in N \\ \left.\begin{array}{l} z_{ij} \leq r_i \\ z_{ij} \leq \overline{r}_i x_j \end{array}\right| si\ b_{ij} < 0 & i \in M,\ j \in N \\ x_j \in \{0,\ 1\},\ z_{ij} \geq 0 & i \in M,\ j \in N \end{cases}$$

La preuve est donnée dans [Djebali, 2003].

2.3 Amélioration de $LIN1$ $(LIN3)$

Pour améliorer la linéarisation $LIN1$, nous considérons à la fois une borne supérieure $\overline{y}_i$ et une borne inférieure $\underline{y}_i$ de y_i. Ainsi, au lieu de remplacer le produit $y_i x_j$ par z_{ij}, on remplacera le produit $(y_i - \underline{y}_i)x_j$ par z_{ij}. De cette façon, nous dégagons une partie linéaire $(\underline{y}_i x_j)$ dans la fonction objectif et dans les containtes. Nous obtenons :

$$(LIN3) \begin{cases} Max \sum_{i=1}^{m}(a_i y_i + \sum_{j=1}^{n} a_{ij} z_{ij} + \sum_{j=1}^{n} a_{ij} \underline{y}_i x_j) & \\ b_i y_i + \sum_{j=1}^{n} b_{ij} z_{ij} + \sum_{j=1}^{n} b_{ij} \underline{y}_i x_j = 1 & i \in M \\ z_{ij} \geq y_i - \overline{y}_i(1 - x_j) - \underline{y}_i x_j & i \in M,\ j \in N \\ z_{ij} \leq y_i - \underline{y}_i & i \in M,\ j \in N \\ z_{ij} \leq (\overline{y}_i - \underline{y}_i)x_j & i \in M,\ j \in N \\ x_j \in \{0,\ 1\},\ z_{ij} \geq 0 & i \in M,\ j \in N \end{cases}$$

La formulation $LIN3$ comprend le même nombre de variables et de contraintes que $LIN1$. Cependant, elle nécessite un pré-traitement supplementaire, le calcul des coefficients $\underline{y}_i$ pour tout $i \in M$.

Proposition 1 : La valeur optimale de $\overline{LIN3}$, la relaxation continue de $LIN3$, est inférieure ou égale à la valeur optimale $\overline{LIN1}$, la relaxation continue de $LIN1$.

La preuve est donnée dans [Djebali, 2003].

2.4 Amélioration de $LIN2$ $(LIN4)$

L'idée de cette amélioration est similaire à celle que nous avons utilisée pour construire $LIN3$. Nous exploitons la

connaissance d'une borne inférieure $\underline{r}_i$ de r_i à l'optimum. On obtient :

$$(LIN4)\begin{cases} Max \sum_{i=1}^{m} r_i \\ a_i + \sum_{j=1}^{n} a_{ij}x_j = b_i r_i + \sum_{j=1}^{n} b_{ij} z_{ij} + \sum_{j=1}^{n} b_{ij}\underline{r}_i x_j & i \in M \\ z_{ij} \geq r_i - \overline{r}_i(1-x_j) - \underline{r}_i x_j \,|\, b_{ij} > 0 & i \in M,\, j \in N \\ \left.\begin{array}{l} z_{ij} \leq r_i - \underline{r}_i \\ z_{ij} \leq (\overline{r}_i - \underline{r}_i)x_j \end{array}\right| b_{ij} < 0 & i \in M,\, j \in N \\ x_j \in \{0,\,1\},\, z_{ij} \geq 0 & i \in M,\, j \in N \end{cases}$$

$LIN4$ comprend le même nombre de variables et de contraintes que $LIN2$ mais nécessite le calcul de $\underline{r}_i$.

Proposition 2 :La valeur optimale de $\overline{LIN4}$, la relaxation continue de $LIN4$, est inférieure ou égale à la valeur optimale de $(\overline{LIN2})$, la relaxation continue de $(LIN2)$.

La preuve est donnée dans [Djebali, 2003].

3 Élaboration des bornes supérieures et inférieures $\underline{y}_i$, $\overline{y}_i$, $\underline{r}_i$ et $\overline{r}_i$

Les différents modèles proposés nécessite la connaissance de bornes inférieures et supérieures de certaines variables. Pour $LIN1$ et $LIN3$, on dispose de bornes evidentes pour $y_i : \overline{y}_i = 1/b_i$ et $\underline{y}_i = 1/(b_i + \sum_{j=1}^{n} b_{ij})$.

Pour calculer $\overline{r}_i$ et $\underline{r}_i$, les bornes de r_i utilisées dans $LIN2$ et $LIN4$, nous avons résolu pour tout $i \in M$, les 2 problèmes hyperboliques en 0/1 sans contraintes : $Max/Min \left\{ \dfrac{a_i + \sum_{j=1}^{n} a_{ij}x_j}{b_i + \sum_{j=1}^{n} b_{ij}x_j} : x \in \{0,\,1\}^n \right\}$ pour lesquels, il existe plusieurs méthodes de résolution. Nous avons choisi d'utiliser une approche par paramétrisation, l'algorithme de Isbell et Marlow [Isbell et al., 1956]. Il s'agit d'un algorithme itératif qui demande, à chaque itération, l'optimisation d'une fonction linéaire sans contraintes.

4 Expérimentations

Tous les programmes associés aux différentes linéraisations présentées ont éte résolus en utilisant CPLEX 6.0 [Cplex 6.0, 1998] sur un ordinateur de type Pentium II à 300 MHz. Les expérimentations ont été réalisées sur des problèmes engendrés aléatoirement. Les coefficients b_i, a_{ij} *et* b_{ij} de la fonction objectif sont des entiers uniformément distribués entre 10 et 20 et les coefficients a_i sont des entiers uniformément distribués entre 1 et 10.

Le tableau 1 compare les valeurs optimales des relaxations continues des programmes $LIN1$, $LIN2$, $LIN3$ et $LIN4$. Pour chaque relaxation nous donnons la valeur de la solution (val) et le temps de résolution (Tps).

D'après le tableau 1, il est clair que la relaxation du programme $LIN3$ est la plus précise et que la relaxation de $LIN1$ est la moins précise. Néanmoins, $LIN3$ est-elle la linéarisation la plus efficace et $LIN1$ la moins efficace pour résoudre le problème ?

Prob		$\overline{LIN1}$		$\overline{LIN2}$		$\overline{LIN3}$		$\overline{LIN4}$	
m	**n**	**Val**	**Tps(s)**	**Val**	**Tps(s)**	**Val**	**Tps(s)**	**Val**	**Tps(s)**
10	30	13.2	0.3	12.8	0.2	**12.3**	**0.28**	12.8	0.254
10	30	12.7	0.28	12.5	0.196	**11.9**	**0.3**	12.3	0.22
10	30	12.8	0.2	12.4	0.23	**11.9**	**0.27**	12.3	0.21
20	50	27.5	3.43	26.8	2.19	**25.6**	**3.91**	26.7	1.95
20	50	27.9	3.72	27.3	1.76	**26.1**	**3.53**	27.2	1.72
20	50	27.3	2.49	26.7	1.96	**25.6**	**3.36**	26.6	2.73
30	50	41.8	12.52	40.9	3.91	**39.0**	**6.86**	40.8	4.5
30	50	40.6	7.34	39.6	4.01	**37.2**	**9.4**	39.5	4.59
30	50	41.0	4.45	40.0	3.8	**37.9**	**8.24**	39.9	4.42

TAB. 1 – Résultats numériques associés aux relaxations continues des 4 linéarisations en variables mixtes du problème (SRH)

Nous présentons dans le tableau 2 les résultats de la résolution exacte pour les quatre linéarisations. Chaque ligne du tableau représente la moyenne des résultats obtenus sur 5 instances du problème. Lorsque les 5 instances n'ont pas pu être toutes résolues dans la limite de temps fixée à 10000 secondes par instance, la moyenne porte sur les seules instances résolues exactement. Pour chaque type de problème, nous indiquons la durée totale de résolution (Tps) et le nombre d'instances résolues de façon exacte (#).

Prob		$LIN1$		$LIN2$		$LIN3$		$LIN4$	
m	**n**	**Tps(s)**	**#**	**Tps(s)**	**#**	**Tps(s)**	**#**	**Tps(s)**	**#**
10	10	0.963	5	0.684	5	**0.512**	**5**	0.74	5
10	20	101.1	5	687.4	5	**39.09**	**5**	400.1	5
10	30	4322.6	1	-	0	**5223.3**	**3**	-	0
20	10	2.8	5	2.49	5	**1.61**	**5**	2.26	5
20	20	1343.4	5	-	0	**774.7**	**5**	2463	2
30	10	5.11	5	4.42	5	**3.59**	**5**	4.04	5
30	20	3755.1	4	-	0	**3232.5**	**5**	-	0

TAB. 2 – Comparaison numérique de la résolution exacte du problème SRH par les 4 linéarisations proposées

Le tableau 2 montre que la linéarisation $LIN3$ est la plus efficace, ce qui confirme les résultats obtenus pour la relaxation continue. En revanche, ce sont les linéarisations $LIN2$ et $LIN4$ qui sont les moins efficaces et non pas $LIN1$. Pour $LIN2$ et $LIN4$ nous avons constaté que la convergence vers la solution optimale etait très lente. Il est clair également que le temps de résolution croît considérablement avec la taille des problèmes. Cela peut s'expliquer en partie par le fait que, plus l'instance est grande, plus l'écart relatif entre la solution exacte et la borne supérieure donnée par la relaxation continue est important.

Nous proposons ci-après une approche heuristique permettant de résoudre rapidement, de manière approchée, de

grandes instances de SRH. Cette solution heuristique sera également exploitée pour raffiner les bornes des variables y_i et r_i afin d'améliorer la résolution exacte.

5 Une méthode heuristique

5.1 Description de la méthode

Il s'agit d'un algorithme de recherche locale. Partant de la solution admissible $x = (0, ..., 0)$, on essaye à chaque itération d'améliorer le plus possible la solution en modifiant la valeur d'une seule variable. On s'arrête lorsqu'on ne peut plus améliorer la solution de cette façon.

Les expérimentations ont montré que la solution admissible obtenue par cette heuristique était de très bonne qualité (égale à la solution optimale dans la plupart des cas). Elle est également très utile pour la résolution exacte. En fournissant la borne inférieure correspondant à cette solution admissible au logiciel de programmation linéaire en variables mixtes, nous obtenons une réduction de l'arbre d'exploration avec une diminution du temps de calcul.

5.2 Résultats expérimentaux

Afin de déterminer une solution approchée, nous avons implanté en langage C sur un Pentium II 300 Mhz l'heuristique présentée ci-dessus. Les tests ont été réalisés sur des problèmes engendrés aléatoirement comme dans le paragraphe 4. Ces tests concernent uniquement l'heuristique proposée et son influence sur la résolution du problème par $LIN3$. Le tableau 3 présente les résultats correspondants. Chaque ligne du tableau représente la moyenne des résultats obtenus sur 5 instances du problème. Nous indiquons, pour l'heuristique, le nombre d'instances pour lesquelles on obtient la solution optimale si cette dernière est connue et le temps de calcul (Tps). Pour $LIN3$ avec et sans l'utilisation de la valeur de la solution heuristique, nous précisons le nombre de nœuds (Nb_nd), le temps CPU (Tps) et le nombre d'instances que nous avons pu résoudre (#).

		cplex sans sol_initiale					cplex avec sol_initiale		
Prob		$LIN3$			Heuristique		$LIN3$		
m	n	Nb_nd	Tps(s)	#	Tps	#	Nb_nd	Tps(s)	#
5	10	41.2	0.1	5	**0.0003**	**5**	31.8	0.082	5
5	20	1087.8	3.87	5	**0.0005**	**5**	920.6	3.1	5
5	30	67177	324.5	5	**0.0009**	**5**	56631	236.2	5
10	10	141.2	0.646	5	**0.0009**	**5**	131.2	0.58	5
10	20	20574	147.2	5	**0.0019**	**5**	18302	127	5
10	30	340451	4379.1	3	**0.0016**	**4**	400511	4374	4
20	10	95.2	1.39	5	**0.0017**	**5**	91	1.4	5
20	20	37281	880.6	5	**0.0048**	**5**	33851	762.2	5

TAB. 3 – Comparaison numérique entre $LIN3$, l'heuristique proposée et $LIN3$ avec la valeur de la solution heuristique

Le tableau 3 montre que notre heuristique est très efficace. Elle donne dans la plupart des cas la solution optimale en un temps excessivement court, moins d'une seconde quelle que soit la taille du problème. On constate également qu'en donnant la valeur de la solution obtenue par l'heuristique au solveur, on améliore la résolution exacte.

Nous proposons dans le paragraphe suivant une méthode de raffinement des bornes des variables y_i et r_i permettant d'accélérer la résolution exacte. Cette méthode exploite la solution heuristique.

6 Raffinement des coefficients des modélisations et résultats expérimentaux

Nous montrons dans cette partie comment obtenir des bornes inférieures et supérieures plus précises pour les variables y_i et r_i, $i \in M$. Le paragraphe 6.1 décrit la démarche adoptée. Les résultats numériques prouvant l'intêret de la méthode sont présentés au paragraphe 6.2.

6.1 Description de la méthode

Nous avons proposé au paragraphe 3 des valeurs pour $\underline{y}_i$, $\overline{y}_i$, $\underline{r}_i$ et $\overline{r}_i$. Nous proposons ici, des valeurs plus précises. Pour les calculer nous introduisons les 2 contraintes $\sum_{j=1}^{n} x_j \geq K_1$ et $\sum_{j=1}^{n} x_j \leq K_2$ où K_1 et K_2 correspondent respectivement au nombre minimal et maximal de variables x_j non nulles à l'optimum du problème SRH. Pour $k = 0, ..., n$, posons $B_k = \sum_{i=1}^{m} \left(Max \left\{ \frac{a_i + \sum_{j=1}^{n} a_{ij} x_j}{b_i + \sum_{j=1}^{n} b_{ij} x_j} : x \in \{0,1\}^n, \sum_{j=1}^{n} x_j = k \right\} \right)$. B_k est une borne supérieure de SRH sous la contrainte $\sum_{j=1}^{n} x_j = k$. Pour déterminer K_1, on calcule B_k pour $k = 0, 1$, etc. et on s'arrête dès que $B_k \geq \alpha$ où α designe la valeur de la solution heuristique. En d'autres termes $K_1 = min\{k : k \in \{0,1,...,n\}, B_k \geq \alpha\}$. De la même façon $K_2 = max\{k : k \in \{0,1,...,n\}, B_k \geq \alpha\}$. Posons $X = \{x : x \in \{0,1\}^n, \sum_{i=1}^{n} x_i \leq K_1, \sum_{i=1}^{n} x_i \geq K_2\}$. $\underline{y}_i$ et $\overline{y}_i$ sont alors données par la résolution des 2 programmes hyperboliques sous contraintes $Max/Min \left\{ 1 \Big/ \left(b_i + \sum_{i=1}^{n} x_i \right) : x \in X \right\}$, et $\underline{r}_i$ et $\overline{r}_i$, par la résolution des 2 programmes $Max/Min \left\{ \left(a_i + \sum_{j=1}^{n} a_{ij} x_j \right) \Big/ \left(b_i + \sum_{j=1}^{n} b_{ij} x_j \right) : x \in X \right\}$. Pour résoudre ces programmes, nous avons également utilisé la méthode par paramétrisation, qui est très efficace, en résolvant les programmes linéaires en 0-1 associés avec CPLEX.

6.2 Résultats expérimentaux

Nous présentons dans le tableau 4 des résultats numériques permettant de comparer la résolution par $LIN3$ cou-

plée avec l'heuristique avec et sans l'étape de raffinement. Les expérimentations ont été effectuées avec les mêmes instances que celles qui sont décrites au paragraphe 5.2 et également avec des instances de plus grande taille engendrées de la même façon.

Chaque ligne du tableau représente la moyenne des résultats concernant 5 instances. Pour chaque type de résolution, nous indiquons le nombre de noeuds développés dans le branch-and-bound (Nb_nd), la durée de la résolution (Tps) ainsi que le nombre d'instances résolues parmi les 5 considérées (#). Le temps de résolution d'une instance est limité à 10000 secondes.

Problème		$LIN3$ **sans raffinement**			$LIN3$ **avec raffinement**		
m	**n**	**Nb_nd**	**Tps(s)**	#	**Nb_nd**	**Tps(s)**	#
5	10	31.8	0.082	5	**23.4**	**0.07**	**5**
5	20	920.6	3.1	5	**490.4**	**1.89**	**5**
5	30	56631	236.2	5	**31823**	**137.2**	**5**
10	10	131.2	0.58	5	**111.3**	**0.49**	**5**
10	20	18302	127	5	**14490.8**	**98.6**	**5**
10	30	400511	4374	4	**210677.5**	**2463.9**	**4**
20	10	91	1.4	5	**57.25**	**1.12**	**5**
20	20	33851	762.2	5	**25626**	**697.5**	**5**
4	50	216832	1171.5	5	**73694.7**	**401**	**5**
2	100	288901.3	1252.9	4	**181902**	**404.8**	**5**

TAB. 4 – Expérimentations concernant $LIN3$ avec et sans raffinement des coefficients du modèle

Il est clair d'après le tableau 4 que l'étape de raffinement est très efficace. En effet, en améliorant les bornes des variables y_i dans $LIN3$, nous obtenons une réduction de l'arbre d'exploration et une accélération de la résolution. Par exemple, pour les instances (10x30), $LIN3$ sans raffinement permet de résoudre 4 instances en un temps moyen de 4400 secondes, alors que $LIN3$ avec raffinement permet de résoudre 4 instances en moins de 2500 secondes. On note également que plus le nombre de variables est grand, plus l'étape de raffinement est efficace (voir les instances 4x50 et 2x100).

7 Un algorithme branch-and-bound avec réajustement progressif des coefficients des contraintes

Nous avons évoqué au paragraphe précédent l'importance du choix des coefficients $\underline{y}_i$, $\overline{y}_i$, $\underline{r}_i$ et $\overline{r}_i$ dans les modèles proposés pour l'efficacité de la résolution par un logiciel standard. C'est pourquoi nous proposons maintenant un algorithme branch-and-bound avec réajustement progressif de ces coefficients en chaque nœud de l'arbre de recherche.

7.1 Description de l'algorithme

L'idée de base de cet algorithme consiste à réajuster certains coefficients du modèle en chaque nœud de l'arbre de recherche. Dans une procédure branch-and-bound standard, l'unique différence entre le sous-problème à résoudre dans un nœud père et celui à résoudre dans un nœud fils est le remplacement d'une seule contrainte ($0 \leq x_i \leq 1$) par une autre contrainte de fixation ($x_i = 1$ ou $x_i = 0$). Dans notre procédure spécifique, certains coefficients du modèle vont évoluer en tenant compte des variables déjà fixées. La réécriture du sous-problème fils se fait donc non seulement par l'ajout d'une contrainte de fixation d'une variable mais aussi par la modification de certains coefficients.

La seule différence entre notre algorithme et une procédure branch-and-bound standard consiste donc à recalculer les coefficients $\underline{y}_i$, $\overline{y}_i$, $\underline{r}_i$ et $\overline{r}_i$ en chaque nœud de l'arbre de recherche. Ces calculs se font par la même procédure que celle que nous avons vue au paragraphe 3 mais en tenant compte des variables déjà fixées dans la branche correspondante. En ce qui concerne la variable de séparation, nous avons choisi, après plusieurs essais, de prendre la variable dont la valeur est la plus petite .

Après de nombreuses expériences numériques, nous avons constaté que notre procédure arborescente spécifique developpait beaucoup moins de nœuds que le solveur standard. En revanche, l'amélioration en temps de résolution etait faible. En effet, le temps nécessaire pour l'étape d'évaluation en chaque nœud est plus important pour notre procédure. Puisque contrairement à un algorithme branch-and-bound standard, pour résoudre un problème d'un nœud fils, nous ne pouvons pas exploiter la base réalisable du problème résolu à son nœud père. Pour accélérer notre procédure, nous avons choisi de limiter le nombre de réajustements des coefficients. Cela entraîne une augmentation du nombre de nœuds développés mais procure un gain important en temps de résolution puisque, pour certains nœuds, nous utilisons les bases réalisables de leur nœud père. De plus, il n'est pas intéressant de réajuster les coefficients si le nombre de variables fixées est encore petit, la coupe étant peu probable.

7.2 Résultats numériques et comparaisons

Nous indiquons dans le tableau 5 les résultats concernant 4 approches pour la résolution du problème SRH : $LIN3$, $LIN4$ avec un branch-and-bound standard (cplex) couplé avec l'heuristique et l'étape de raffinement, puis $LIN3$ et $LIN4$ avec le branch-and-bound spécifique décrit ci-dessus. Ces résolutions ont été implémentées en langage C sur un ordinateur Pentium II à 300 MHz. Les expérimentations ont été réalisées dans les mêmes conditions que celles qui sont décrites dans le paragraphe 5.2. Pour chaque type d'instances, nous indiquons le nombre total de nœuds et le temps CPU (moyenne sur 5 instances).

Le tableau 5 montre que la résolution de $LIN3$ et $LIN4$ avec la procédure spécifique proposée est très efficace par rapport à la procédure standard. On note également que, contrairement à la résolution standard où le programme $LIN3$ donne toujours les meilleurs résultats, c'est $LIN4$

		Cplex+Sol_ initiale+Raf				Branch&Bound Spécifique			
Prob		$LIN3$		$LIN4$		$LIN3$		$LIN4$	
m	n	Nb	Tps	Nb	Tps	Nb	Tps	Nb	Tps
10	20	5736	46	35142	170	2089	64	**2311**	**29**
10	20	5797	49	29806	153	1843	57	**2709**	**24**
10	20	2106	18	27296	166	1421	38	**1179**	**14**
20	20	24894	705	81460	1088	4045	497	**5337**	**185**
20	20	24715	616	76610	1021	3199	399	**4161**	**142**
20	20	40956	1034	99610	1440	4627	522	**6917**	**228**
30	20	35315	1869	95160	3628	4143	961	**5267**	**356**
30	20	17559	1019	79321	2063	3609	836	**4613**	**310**
30	20	15058	972	26838	876	3001	748	**3183**	**270**

TAB. 5 – Comparaison numérique entre le branch-and-bound standard (cplex) et notre procédure spécifique

qui fournit les meilleurs résultats dans le cas du branch and bound spécifique. En utilisant le branch-and-bound standard, $LIN4$ converge très lentement vers la solution optimale. Par contre, avec notre procédure spécifique et grâce au réajustement progressif des coefficients, la convergence devient nettement plus rapide et meilleure que celle de $LIN3$. Nous pouvons donc conclure intuitivement que $LIN4$ est beaucoup plus sensible aux valeurs des bornes que $LIN3$.

8 Conclusion

Nous avons consideré dans ce papier le problème de la maximisation de la somme de plusieurs ratios hyperboliques en 0/1. Il s'agit d'un problème d'optimisation combinatoire fractionnaire pour lequel il existe très peu de travaux. Nous avons proposé différentes techniques de linéarisations qui se distinguent par les expressions linéarisées et les contraintes ajoutées. Ces techniques sont très faciles à mettre en œuvre si l'on dispose d'un outil de programmation linéaire en variables mixtes. De plus, elles pourraient être appliquées au cas de problèmes sous contraintes ainsi qu'à de nombreux autres problèmes classiques de programmation quadratique en variables mixtes.

Nous avons également proposé dans ce travail trois stratégies de résolution. La première consiste à résoudre directement le problème linéaire en utilisant un outil de programmation linéaire mixte. La deuxième stratégie est une résolution heuristique qui est aussi exploitée pour améliorer la première stratégie. Enfin, la troisième consiste à résoudre le problème par un algorithme branch-and-bound spécifique, fondé sur une réécriture plus précise du problème en chaque nœud de l'arbre de recherche. Une comparaison numérique permettant d'évaluer l'efficacité des différentes linéarisations et stratégies est présentée.

Références

[Arora et al., 1977] Arora S.R., Swarup K., Puri M.C., "The set covering problem with linear fractional functional", Indian Journal of Pure and Applied Mathematics, Vol. 8, p. 578-588 (1977).

[Cplex 6.0, 1998] Ilog, Inc., Cplex Division, "Using the callable Library", Version 6.0 (1998).

[Djebali, 2003] Djebali K., "Modélisation et résolution de problèmes d'optimisation combinatoire par la programmation linéaire en variables mixtes", Thèse en cours, CNAM, Paris (2003).

[Falk et al., 1992] Falk J.E., Paloscay S.W., "Optimising the sum of linear fractional functions", Advances in Global Optimisation, Kluwer Academic Publishers, p. 221-258 (1992).

[Freund et al., 2001] Freund R.W., Jarre F., "Solving the sum-of-ratios problem by an interior-point method", Journal of Global Optimization, Vol. 19, p. 83-102 (2001).

[Gilmore et al., 1963] Gilmore P.C., Gomory R.E., "A linear programming approach to the cutting stock problem", Operations Research, Vol. 11, p. 52-53 (1963).

[Hansen et al., 1991] Hansen P., Poggi De Aragao M. V., Ribeiro C.C., "Hyperbolic 0-1 programming and information retrieval", Mathematical Programming, Vol. 52, p. 255-263 (1991).

[Isbell et al., 1956] Isbell J. R., Marlow W. H., "Attribution games", Naval Research Logistics Quarterly, Vol. 3, p. 71-94 (1956).

[Li, 1994] Li H., "A Global approach for general fractional programming", European Journal of Operational Research, Vol. 73, p. 590-59 (1994).

[Radzik, 1998] Radzik T., "Fractional combinatorial optimization", Handbook of Combinatorial Optimization, Edited by Z.-Z. Du and P. Paradaloas, Kluwer Academic Publishers, p. 429-478 (1998).

[Saipe, 1975] Saipe A. L., "Solving a 0-1 hyperbolic program by branch and bound", Naval Research Logistics Quarterly, Vol. 22, p. 397-416 (1975).

[Schaible, 1983] Schaible S., Ibaraki T., "Fractional programming", European Journal of Operational Research, Vol. 12, p. 325-338 (1983).

[Schaible, 1995] Schaible S., "Fractional programming" Handbook of Combinatorial Optimization, Edited by R. Horst and P. Paradalos, Kluwer Academic Publishers, p. 495-608 (1995).

[Stancu-Minasian, 1997] Stancu-Minasian I.M., "Fractional Programming",Kluwer Academic Publishers (1997).

[Wu, 1997] Wu T., "A note on a global approach for general 0-1 fractional programming", European Journal of Operational Research, Vol 101, p. 229-223 (1997).