RAIRO Operations Research
RAIRO Oper. Res. **35** (2001) 401-414

UN ALGORITHME POUR LA BIPARTITION D'UN GRAPHE EN SOUS-GRAPHES DE CARDINALITÉ FIXÉE

Philippe Michelon[1], Stéphanie Ripeau[2] et Nelson Maculan[3]

Communiqué par Gérard Plateau

Abstract. A branch-and-bound method for solving the min cut with size constraint problem is presented. At each node of the branch-and-bound tree the feasible set is approximated by an ellipsoid and a lower bound is computed by minimizing the quadratic objective function over this ellipsoid. An upper bound is also obtained by a Tabu search method. Numerical results will be presented.

Résumé. Nous présentons une méthode de Séparation et Évaluation Progressive pour la bipartition d'un graphe en 2 sous-ensembles ayant une cardinalité fixée. À chaque nœud de l'arbre de recherche, nous calculons une borne inférieure en dualisant les contraintes d'intégralité et en approximant le domaine réalisable par un ellipsoïde. Une borne supérieure est également calculée par la méthode Tabou. Des résultats numériques sont présentés et commentés.

1. Introduction

Nous nous intéressons ici à la bipartition d'un graphe sous contrainte de taille : étant donné un graphe $G = (V, E)$ comportant n sommets et un entier p ($p \leq n$),

Reçu en décembre 1998.

[1] Laboratoire d'Informatique d'Avignon, UAPV, BP. 1228, 84911 Avignon Cedex 9, France.
[2] Université de Montréal, DIRO, CP. 6128, Succursale Centre-Ville, Montréal (Québec) H3C 3J7 Canada.
[3] Universidade Federal do Rio de Janeiro, COPPE/Sistemas, P.O. Box 68511, Rio de Janeiro, RJ 21945–790, Brésil.

© EDP Sciences 2002

le problème consiste à trouver une partition de V en deux sous-ensembles V_1 et V_2 tels que V_1 comporte exactement p sommets (et, donc, V_2 doit comporter $n-p$ sommets). Plus précisément, il faut trouver la partition de capacité (ou poids) minimale ; la capacité d'une partition étant définie par la somme des capacités des arcs qui sont "coupés" par la partition, c'est-à-dire qu'il faut trouver la partition minimisant

$$\sum_{i \in V_1} \sum_{j \in V_2} c_{ij}$$

où c_{ij} est la capacité de l'arc $(i, j) \in E$.

En posant

$$x_i = \begin{cases} 1 & \text{si le sommet } i \text{ appartient à } V_1 \\ 0 & \text{sinon} \end{cases}$$

le problème de bipartition se modélise comme suit :

$$\begin{aligned} (P) \quad \text{Min} \quad & \sum_{(i,j) \in E} c_{ij}(x_i(1-x_j) + x_j(1-x_i)) \\ \text{s-à :} \quad & \sum_{i=1}^{n} x_i = p \\ & x \in \{0,1\}^n. \end{aligned}$$

Il s'agit donc d'un problème quadratique en variables binaires. Bien évidemment, ce problème de bipartition est un cas particulier du problème consistant à partitionner les sommets d'un graphe en K sous-ensembles, chacun ayant une cardinalité fixée.

Il semble que tous ces problèmes de partitionnement possèdent de nombreuses applications pratiques en conception de circuits intégrés, dans la répartition d'équipements, etc. [7,12]. Par ailleurs, ce problème est reconnu pour être NP-Dur [17]. C'est sans doute ce qui explique que de nombreuses méthodes heuristiques, toutes plus ou moins basées sur des procédures d'échanges, aient été proposées [1,2,10,11]. Signalons que cette dernière méthode utilise le récuit simulé et que les auteurs (Johnson *et al.*) sont parvenus à partitionner des graphes de 1000 nœuds.

On trouve également dans la littérature un certain nombre de techniques permettant de calculer des bornes inférieures : Donath et Hoffman [6], Boppana [3] et Falker *et al.* [7] utilisent les valeurs et vecteurs propres de la matrice apparaissant dans la forme quadratique associée au problème (voir section suivante) alors que Billionnet et Faye [4] étendent au problème la théorie de la dualité du toît ("roof duality"). Cependant, aucun de ces auteurs ne proposent de méthode pour

combler l'éventuelle différence entre la borne inférieure calculée et la valeur optimale du problème. Ainsi, à notre connaissance, il n'existe que deux algorithmes résolvant le problème de la bipartition avec contrainte de taille de manière exacte. Le premier est dû à Christofides et Brooker [5], le second à Roucairol et Hansen [17]. Tous deux sont en fait des schémas de séparation et évaluation progressive ("branch and bound"). Christofides et Brooker basent leur algorithme sur le calcul d'une borne inférieure obtenue en résolvant successivement plusieurs problèmes de flot maximum. Roucairol et Hansen dualisent la contrainte de taille, se ramenant ainsi à la résolution de plusieurs problèmes de coupe minimale dans un graphe où toutes les capacités sont positives. Les meilleurs résultats semblent avoir été obtenus par cette dernière méthode mais ceux-ci restent tout de même modestes : pour des graphes de pleine densité et $p = n/2$, la méthode n'a pu solutionner que des problèmes de taille (n) inférieure ou égale à 20 alors que pour des problèmes à très faible densité et $p = n/2$, la méthode n'a pu résoudre que des problèmes de taille inférieure ou égale à 50.

Notre but ici est de présenter un nouveau schéma de séparation et évaluation progressive basé sur une borne inférieure obtenue en dualisant les contraintes d'intégralité et en approximant le domaine réalisable par un ellipsoïde. Nous avons déjà utilisé une technique similaire pour la minimisation de formes quadratiques en variables binaires mais sans contrainte [14]. À la section suivante, nous allons expliciter cette borne inférieure. Nous montrerons comment résoudre le sous problème correspondant en section 3. Dans la quatrième section, nous définirons les différentes étapes que comporte notre schéma de séparation et évaluation progressive. Enfin, en section 5, nous reporterons et commenterons des résultats numériques.

2. Une nouvelle borne inférieure

Rappelons que notre problème consiste à partitionner les n sommets d'un graphe $G = (V, E)$ en deux sous-ensembles V_1 et V_2 tels que $|V_1| = p$ et $|V_2| = n - p$. Désignant par c le vecteur des capacités associées aux arcs de E, le problème de bipartition se modélise comme suit :

$$
\begin{aligned}
(P) \quad \text{Min} \quad & \sum_{(i,j)\in E} c_{ij}(x_i(1-x_j) + x_j(1-x_i)) \\
\text{s-à :} \quad & \sum_{i=1}^{n} x_i = p \\
& x \in \{0,1\}^n.
\end{aligned}
$$

Sous une forme matricielle (P) peut encore s'écrire :

$$\begin{array}{lll} (P) \quad \text{Min} & \frac{1}{2}\langle Qx, x\rangle + \langle q, x\rangle \\ \text{s-à :} & \langle e, x\rangle = p \\ & x \in \{0,1\}^n \end{array}$$

où : $e = (1, 1, ..., 1) \in R^n$

Q est une matrice symétrique (n, n) définie par : $Q_{ij} = \begin{cases} 0 & \text{si } (i,j) \notin E \\ -2c_{ij} & \text{si } (i,j) \in E \\ 0 & \text{si } i = j \end{cases}$

et $q \in R^n$ est défini par : $q_i = \sum_{j \text{ tel que } (i,j)\in E} c_{ij}$

Avant d'introduire notre borne inférieure, nous devons écrire (P) sous une forme équivalente :

$$\begin{array}{llrll} (P) & \text{Min} & \frac{1}{2}\langle Qx, x\rangle & +\langle q, x\rangle & & \\ & \text{s-à :} & \langle e, x\rangle & = p & & (1) \\ & & \|x\|^2 & = p & & (2) \\ & & x_i^2 - x_i & = 0 & i = 1, 2, ..., n. & (3) \end{array}$$

La contrainte (3) est trivialement équivalente à $x_i \in \{0, 1\}$ alors que la contrainte (2) est redondante : elle a été obtenue en remplaçant x_i par x_i^2 dans (1).

La borne inférieure que nous proposons consiste à dualiser la contrainte d'intégralité (3) après lui avoir associé un multiplicateur u_i $(\in R)$ pour obtenir la fonction lagrangienne l.

$$\begin{array}{llrl} \text{l(u)=} & \text{Min} & \frac{1}{2}\langle Qx, x\rangle & +\langle q, x\rangle + \frac{1}{2}\sum_{i=1}^{n} u_i(x_i^2 - x_i) \\ & \text{s-à :} & \langle e, x\rangle & = p \\ & & \|x\|^2 & = p \\ = & \text{Min} & \frac{1}{2}\langle Q(u)x, x\rangle & +\langle q(u), x\rangle \\ & \text{s-à :} & \langle e, x\rangle & = p \\ & & \|x\|^2 & = p \end{array}$$

avec $Q(u) = Q + \text{Diag}(u)$ et $q(u) = q - \frac{1}{2}u$.

Il est notoire que l est une fonction concave, continue, non-différentiable et telle que :

$$\forall u \in R^n \; l(u) \leq v(P)$$

($v(P)$ désignant la valeur optimale du problème (P)).

Afin d'obtenir la meilleure borne inférieure possible, nous devons maximiser l. Ce problème de maximisation est le problème dual (D)

$$(D) \text{ Max } \{l(u)/u \in R^n\}\cdot$$

Le calcul de cette borne inférieure peut donc être réalisé par l'algorithme classique du sous-gradient [9,16]. Pour ce faire, il faut bien sûr être capable de résoudre le problème d'optimisation non-linéaire et non-convexe associé à l. La section suivante est justement dévolue à ce problème.

3. Résolution du problème lagrangien

Pour calculer $l(u)$ (et trouver un sous-gradient de l au point u), nous allons nous ramener à la minimisation d'une forme quadratique sur une sphère.

Posons d'abord : $y = x - \dfrac{p}{n}e$

$$\begin{aligned} l(u) = \quad & \text{Min} && \frac{1}{2}\langle Q(u)x, x\rangle + \langle q(u), x\rangle \\ & \text{s-à :} && \langle e, x\rangle = p \\ & && \|x\|^2 = p \end{aligned}$$

$$= \quad \text{Min} \quad \frac{1}{2}\left\langle Q(u)\left(y + \frac{p}{n}e\right), y + \frac{p}{n}e\right\rangle + \left\langle q(u), y + \frac{p}{n}e\right\rangle$$

$$\text{s-à :} \quad \left\langle e, y + \frac{p}{n}e\right\rangle = p$$

$$\left\|y + \frac{p}{n}e\right\|^2 = p$$

$$\begin{array}{rlrl} = & \text{Min} & \frac{1}{2}\langle Q(u)y, y\rangle & + \left\langle q(u) + \frac{p}{n}Q(u)e, y \right\rangle + K \\ & \text{s-à :} & \langle e, y\rangle & = 0 \\ & & \|y\|^2 & = \frac{p(n-p)}{n} \end{array}$$

avec $K = \frac{p}{n}\langle q(u), e\rangle + \frac{p^2}{2n^2}\langle Q(u)e, e\rangle$.

À ce stade-ci, il est intéressant de noter la parfaite symétrie, existant dans le calcul de $l(u)$, entre p et $n-p$. En effet, nous avons modélisé le problème en posant $x_i = 1$ si $i \in V_1$ et 0 sinon, mais nous aurions aussi bien pu modéliser le même problème en posant $x_i = 1$ si $i \in V_2$ et 0 sinon. Nous aurions alors la contrainte :

$$\sum_{i=1}^{n} x_i = n - p.$$

Il faut noter qu'en appliquant le même procédé à cette seconde modélisation, nous aurions obtenu exactement la même fonction lagrangienne $l(u)$ justement à cause de la symétrie entre p et $n-p$ apparaissant dans son expression.

Considérons maintenant $(v^1, v^2, ..., v^n)$ une base orthonormale de R^n telle que $v^n = \frac{1}{\sqrt{n}}e$ et V la matrice dont les lignes sont les v^i. V est évidemment une matrice orthonormale, que l'on peut calculer soit par le procédé de Graham–Schmidt, soit en calculant les vecteurs propres de la matrice carrée de n lignes et n colonnes dont tous les coefficients sont égaux à 1. Posons maintenant $z = Vy$. Alors :

$$\begin{array}{rlrl} \text{l(u)=} & \text{Min} & \frac{1}{2}\langle VQ(u)V^t z, z\rangle & + \left\langle V(q(u) + \frac{p}{n}Q(u)e), z \right\rangle + K \\ & \text{s-à :} & z_n & = 0 \\ & & \|z\|^2 & = \frac{p(n-p)}{n}. \end{array}$$

Par conséquent, nous pouvons éliminer la dernière composante de z (puisque celle-ci est fixée à 0) et le calcul de $l(u)$ revient à résoudre un problème de la forme :

$$\begin{array}{rlc} \text{l(u)} = & \text{Min} & \frac{1}{2}\langle Q'(u)z, z\rangle + \langle q'(u), z\rangle + K \\ & \text{s-à :} & \sum_{i=1}^{n-1} z_i^2 = \frac{p(n-p)}{n} \\ & & z \in R^{n-1} \end{array}$$

où $Q'(u)$ est la matrice constituée des $n-1$ premières lignes et colonnes de $VQ(u)V^t$ et $q'(u)$ est le vecteur constitué par les $(n-1)$ premières composantes de $V(q(u)+\frac{p}{n}Q(u)e)$.

Le calcul de $l(u)$ est donc effectué en minimisant une forme quadratique (de R^{n-1}) sur une sphère. Ce problème a fait l'objet de nombreuses études [8,13–15,18] car il apparaît comme sous-problème dans les algorithmes d'optimisation non-linéaire basés sur les régions de confiance ("trust region method") mais aussi dans certains problèmes statistiques d'évaluation de paramètres. Il a ainsi été démontré que, une fois connus les vecteurs et valeurs propres de $Q'(u)$, on obtient sa (ou l'ensemble de ses) solution(s) optimale(s) en résolvant une équation à une inconnue (par exemple, par la méthode de Newton). Il s'agit donc d'un problème particulièrement facile à résoudre.

Explicitons maintenant de quelle façon nous avons introduit cette borne inférieure dans un schéma de séparation et évaluation progressive.

4. Le schéma de séparation et évaluation progressive

La conception de notre méthode de séparation et évaluation progressive a été guidée par une constatation : la borne inférieure que nous préconisons nécessite de calculer les vecteurs et valeurs propres d'une matrice. En particulier, dans l'algorithme du sous-gradient, les variables duales sont modifiées à chaque itération et, par suite, un tel calcul de valeurs et vecteurs propres doit être réalisé autant de fois que l'on va évaluer la fonction lagrangienne. Nous avons donc choisi de résoudre le problème dual une fois seulement, à la racine de l'arbre de séparation et évaluation progressive, et d'utiliser les variables duales optimales ainsi déterminées pour calculer une borne inférieure aux nœuds subséquents. En procédant ainsi, nous ne calculons pas une aussi bonne borne inférieure que nous le pourrions (aux nœuds subséquents), mais nous espérons gagner en efficacité : la borne inférieure se calcule alors, à chaque nœud, de façon extrêmement rapide. Par ailleurs, rien n'indique que si nous calculions la meilleure borne possible, le gain ainsi réalisé permettrait de couper significativement l'énumération.

Toujours en raison de ce calcul de valeurs et vecteurs propres, nous avons choisi de déterminer dès la racine sur quelle variable nous allions baser la séparation à chacun des nœuds subséquents (ordre d'arbitrage statique des variables). En effet, dans ce cas tous les nœuds d'un même niveau (par niveau, nous entendons nombre de variables fixées) ont exactement les mêmes variables fixées et, par conséquent, tous les nœuds d'un même niveau ont la même matrice $Q'(u)$. Ainsi, nous n'aurons à calculer les valeurs et vecteurs propres de cette matrice $Q'(u)$ qu'une seule fois par niveau, améliorant d'autant l'efficacité de la méthode. Bien évidemment, cela n'est vrai que parce que le "u" hérité de la racine est le même à chacun des nœuds.

De surcroît, choisir dès la racine l'ordre de branchement a aussi comme avantage important de faciliter la gestion de l'arbre de recherche : il nous suffit de renuméroter les variables de telle sorte que x_n soit maintenant la variable sur laquelle on veut effectuer la première séparation, x_{n-1} la seconde, etc. Dès lors, la création de la matrice $Q'(u)$ au niveau suivant sera triviale : nous n'aurons qu'à transmettre la matrice du niveau courant avec une dimension de moins (en "supprimant" la dernière ligne et la dernière colonne de la matrice). Sinon, il faudrait soit créer une nouvelle matrice (ne comportant pas la ligne et la colonne i sur laquelle on veut brancher) qu'il faudrait remplir, soit permuter lignes et colonnes (de sorte que la ligne et la colonne i soit à l'extrémité droite de la matrice pour pouvoir l'enlever) et ce à chaque nœud.

En pratique, nous avons choisi de baser notre ordre de branchement selon x^* solution optimale du problème associé à $l(0)$. Ce problème ne fournit pas nécessairement la meilleure borne inférieure (*i.e.* o n'est pas, généralement, solution optimale du problème dual), mais il constitue, en terme de fonction objectif (puisque celle-ci est inchangée pour $u = 0$), la meilleure approximation du problème initial. Ainsi, si x^* possède une composante très grande, on peut penser que, "probablement", la composante correspondante sera à 1 dans la solution optimale du problème initial. De même, si x^* possède une composante négative mais très grande en valeur absolue, il est vraisemblable que la composante sera à 0 dans la solution optimale du problème initial. Nous avons donc renuméroté les variables de telle sorte que les $|x_i^* - \frac{1}{2}|$ soit en ordre décroissant et choisi d'explorer d'abord la branche la moins "probable", c'est-à-dire celle où x_i a été fixée à la valeur la plus éloignée de x_i^*. Précisons aussi que l'arbre sera parcouru selon la stratégie "profondeur d'abord".

Enfin, nous avons initialisé notre schéma de séparation et évaluation progressive par le calcul d'une borne supérieure à la valeur optimale de (P). Pour cela, nous avons appliqué une méthode Tabou relativement simple : le voisinage de la solution courante est généré en échangeant chacun des sommets de V_1 avec chacun des sommets de V_2, un échange est décrété Tabou si il invoque un sommet qui a été changé d'ensemble au cours des $\dfrac{n}{2}$ dernières itérations et la méthode s'arrête après $n \times p$ itérations sans amélioration de la meilleure solution. En pratique, cette méthode Tabou a toujours trouvé la solution optimale du problème initial (à l'exception de 5 instances sur 300 traitées). Il est vrai qu'il s'agit de problèmes de petites tailles pour cette méthode heuristique réputée pour sa capacité à trouver de bonnes solutions à des problèmes difficiles.

5. Résultats numériques

Nous avons implanté ce schéma de séparation et évaluation progressive en PASCAL (à l'exception de la procédure calculant les valeurs et vecteurs propres qui vient de la librairie NAG en FORTRAN) sur une SUN SPARC STATION 20 et nous l'avons testé sur des problèmes générés aléatoirement.

Nous avons considéré différentes classes de problèmes selon la taille du graphe, sa densité et la valeur de p. En effet, la difficulté du problème de bipartition augmente avec la valeur de chacun de ces trois paramètres. Cela semble à peu près évident en ce qui concerne la taille. La densité influence directement le "degré de non-linéarité" du problème quadratique correspondant : les graphes à faible densité vont donner des problèmes qui seront "presque" linéaires et donc d'autant plus faciles. Les graphes à forte densité seront, au contraire, difficiles à partitionner. Quant à la taille de la partition (p), elle augmente la difficulté du problème à mesure que p se rapproche de $\frac{n}{2}$ puisque le nombre de points réalisables pour (P) est égal au nombre de combinaisons de p objets pris parmi n.

En pratique, nous avons généré aléatoirement des graphes de n=10,20,...,70 sommets avec une densité variant de 0,1 à 1 et considéré 3 valeurs possibles pour la taille de V_1 : $p = \lceil \frac{n}{8} \rceil, \lceil \frac{n}{4} \rceil$et $\lceil \frac{n}{2} \rceil$. Les capacités ont été générées aléatoirement, selon une loi uniforme dans $[0, 50]$. Pour chaque classe de problèmes ainsi crée, nous avons résolu 10 instances. Les tableaux suivants reportent la moyenne obtenue sur les 10 problèmes pour chaque classe que nous avons été capable de résoudre en moins de 10 minutes de CPU.

Les deux premières séries de tableaux reportent respectivement les temps (en secondes) de résolution et le nombre de noeuds visités dans l'arbre de recherche. Comme on peut s'y attendre lorsqu'on traite des problèmes NP-complet, ceux-ci croissent de façon exponentielle avec la taille du problème. En revanche, notre méthode semble assez peu sensible à la densité et à la valeur de p, ce qui constitue une caractéristique intéressante (nous avons obtenu le même genre de résultat dans [14]). Une raison pour cela réside dans la méthode elle-même : il ne s'agit en rien d'une technique de linéarisation qui se dégraderait avec le "degré" de non linéarité de la fonction objectif, la densité de la matrice reflétant au moins en partie ce degré de non linéarité. Au contraire, la méthode, en se ramenant dans l'espace engendré par l'ensemble des vecteurs propres de la matrice, prend totalement en compte l'aspect quadratique de la fonction objectif et, par conséquent, est peu sensible à la densité du problème.

La troisième série de tableaux rend compte de la qualité de la borne inférieure mesurée en pourcentage (les tables indiquent la moyenne sur les 10 problèmes de chaque classe de $|\frac{\text{valeur opt. } - \text{ borne inférieure}}{\text{valeur optimale}}|$). Il est très surprenant de constater que l'écart relatif entre la borne inférieure et la valeur optimale diminue avec la difficulté des problèmes : le meilleur pourcentage (3,7 %) est obtenu pour un graphe de 60 sommets, pleine densité, et $p = \frac{n}{2} = 30$ alors que les plus mauvais sont atteints pour des problèmes de petites tailles, de faible densité et $p = \frac{n}{8}$! Cela indique tout de même que la borne inférieure est relativement bonne.

Comparés aux résultats de Roucairol–Hansen [17] et Christofides–Brooker [5], nous avons donc multiplié par 3 la taille des problèmes à pleine densité solubles dans un temps raisonnable. Il ne faut cependant pas tirer de conclusions trop

TABLEAU 1. Temps d'exécutions en secondes.

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	0,0	0,1	0,1	0,0	0,0	0,0	0,0	0,0	0,1	0,0
20	0,4	0,4	0,5	0,4	0,4	0,3	0,4	0,5	0,5	0,4
30	1,7	1,7	1,9	1,9	1,8	1,9	1,8	1,9	2,0	1,9
40	5,3	5,5	5,6	5,6	5,7	5,7	5,8	5,6	5,7	5,8
50	12,9	13,1	13,6	13,7	13,9	13,8	14,2	14,0	13,9	13,4
60	26,4	27,6	28,0	28,7	29,3	29,3	28,5	29,0	30,7	31,1
70	50,8	52,7	51,9	54,6	61,1	60,5	56,9	61,0	54,3	58,8

$p = n/8$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	0,1	0,0	0,1	0,1	0,0	0,1	0,0	0,1	0,0	0,0
20	0,4	0,5	0,4	0,5	0,4	0,4	0,4	0,5	0,3	0,4
30	1,8	1,9	2,0	2,0	2,0	2,1	2,2	2,1	1,9	2,1
40	5,5	6,5	6,8	7,5	8,1	6,6	7,5	8,0	7,8	7,6
50	13,2	16,9	21,9	25,8	29,1	26,0	22,5	24,4	23,0	27,3
60	42,2	65,6	73,4	152,1	173,7	202,3	171,5	182,6	218,7	248,4
70	75,9	293,1	351,8	565,5						

$p = n/4$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	0,0	0,0	0,1	0,0	0,0	0,1	0,0	0,1	0,1	0,0
20	0,4	0,4	0,4	0,5	0,4	0,5	0,4	0,5	0,5	0,5
30	1,9	2,0	2,1	2,1	2,1	2,1	2,1	2,1	2,2	2,2
40	5,7	6,7	7,7	8,3	7,0	7,9	7,1	8,3	9,0	8,5
50	15,3	31,1	37,7	37,7	33,8	48,5	33,2	48,2	73,5	47,6
60	47,0	195,2	341,9	273,0	327,6	300,7	466,1	419,0	368,3	433,2
70	507,9									

$p = n/2$

TABLEAU 2. Nombre de nœuds parcourus.

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	8	9	11	15	17	14	16	17	19	17
20	22	36	51	48	42	63	61	64	67	52
30	62	60	118	172	123	189	131	218	268	106
40	121	328	647	590	991	808	758	851	601	1038
50	407	652	1509	2196	2665	2185	3166	2629	2631	1216
60	1296	2381	2649	3907	3336	4713	3223	4228	7225	7915
70	2351	4492	3207	7139	11 056	14 629	10 052	16 928	6419	12 744

$p = n/8$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	8	10	15	15	17	19	24	20	16	20
20	36	58	89	124	108	104	126	162	110	96
30	91	334	521	644	696	992	643	768	962	1329
40	500	2929	3237	6599	6353	2506	5736	8124	6483	5580
50	1507	9172	19 091	26 827	34 914	28 352	19 851	24 205	20 247	32 096
60	30 791	62 220	76 345	214 864	240 249	293 492	231 229	245 795	286 512	348 604
70	35 361	316 362	397 498	648 988						

$p = n/4$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	10	10	13	14	17	15	15	17	19	15
20	23	104	79	106	146	119	69	143	120	133
30	239	667	750	973	858	820	804	768	1288	1290
40	393	3325	5279	7941	3440	6258	3639	7870	9354	8167
50	4606	37 134	49 358	49 670	40 987	73 264	37 916	62 808	134 097	69 761
60	28 499	258 537	489 322	373 665	449 510	397 662	663 204	556 394	504 586	606 529
70	2 646 089									

$p = n/2$

TABLEAU 3. Saut de dualité en pourcentage.

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	$+\infty^*$	62,5	38,4	34,3	27,6	14,0	18,5	15,8	17,7	16,4
20	127,5	30,6	27,5	24,1	13,8	16,7	16,4	13,7	12,8	10,3
30	60,7	20,3	20,8	17,1	14,2	12,1	10,0	11,0	11,1	7,9
40	38,8	19,2	18,3	12,9	13,3	11,7	9,4	7,5	7,1	7,6
50	33,0	19,6	15,1	13,4	10,5	9,7	8,9	7,8	7,4	6,1
60	29,3	17,1	13,4	11,9	9,5	9,1	6,9	7,0	7,2	6,5
70	26,0	16,6	11,8	10,8	9,1	7,6	7,5	6,5	5,5	5,2

$\boldsymbol{p = n/8}$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	$+\infty^*$	35,8	27,9	16,6	18,5	18,7	16,3	14,6	9,2	9,5
20	73,3	22,8	18,9	18,1	13,1	10,7	11,8	10,4	6,6	6,7
30	38,3	22,5	17,3	12,9	11,6	9,7	9,2	7,9	8,4	7,5
40	33,2	19,2	14,1	12,7	10,3	7,8	8,1	6,8	6,8	5,9
50	29,1	17,5	13,2	10,9	9,3	8,1	7,0	6,5	5,6	5,9
60	26,3	15,4	11,6	10,0	8,5	7,5	6,5	5,6	5,7	5,3
70	21,9	14,5	11,6	9,8						

$\boldsymbol{p = n/4}$

	densité									
n	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
10	$+\infty^*$	18,3	11,7	7,2	13,4	8,3	7,4	7,6	7,3	4,3
20	$+\infty^*$	17,8	11,2	10,0	9,7	6,9	6,7	6,3	5,1	5,4
30	25,2	15,7	11,5	9,2	7,2	6,3	5,6	5,3	5,2	4,4
40	17,9	13,9	9,8	8,7	6,0	5,9	4,7	4,8	4,7	4,1
50	21,0	13,6	9,7	7,0	6,2	5,8	5,0	4,6	4,3	4,1
60	19,1	11,9	8,7	7,4	6,1	5,4	4,8	4,4	4,0	3,7
70	17,7									

$\boldsymbol{p = n/2}$

(*): Une des 10 instances a une valeur optimale de 0.

hâtives à ce sujet puisque la machine utilisée pour ces expéreinces est sans aucun doute bien meilleure que celles utilisées par ces auteurs. Nous pensons tout de même pouvoir affirmer que notre méthode est, pour le moins, compétitive avec les autres méthodes, en particulier pour les problèmes à forte densité.

6. Conclusion

Ces résultats viennent confirmer ceux que nous avions obtenu dans [14] : pour un problème similaire, nous avions été en mesure de traiter des instances de tailles équivalentes dans des temps sensiblement égaux. Ainsi, selon nous, une des conclusions de ce travail, outre le développement d'une nouvelle technique efficace de bipartition d'un graphe, est que la dualisation des contraintes d'intégralité, mises sous la forme $x_i^2 - x_i = 0$, est une méthode performante pour traiter les problèmes quadratiques en variables binaires. On pourra d'ailleurs l'appliquer probablement avec succès à des problèmes de partitions plus généraux (en K sous-ensembles de cardinalité fixée), au problème d'affectation quadratique, au problème de semi-affectation quadratique, etc. voire à des problèmes linéaires en variables binaires (où l'aspect quadratique ne serait introduit que par la dualisation des contraintes d'intégralité).

Références

[1] E.R. Barnes, An algorithm for partitioning the nodes of a graph. *SIAM J. Algebraic Discrete Math.* **3** (1982) 541-550.

[2] E.R. Barnes, A. Vanelli et J.Q. Walker, A new heuristic for partitioning the nodes of a graph. *SIAM J. Discrete Math.* **1** (1988) 299-305.

[3] R.B. Boppana, Eigenvalues and graph bissection: An average case analysis, in *Proc. of the 28^{th} annual symposium on computer sciences.* IEEE London (1987) 280-285.

[4] A. Billionnet et A. Faye, A lower bound for a constrained quadratic $0-1$ minimization problem. *Discrete Appl. Math.* (soumis).

[5] N. Christofides et P. Brooker, The optimal partitioning of graphs. *SIAM J. Appl. Math.* **30** (1976) 55-69.

[6] W.E. Donath et A.J. Hoffman, Lower bounds for the partitioning of graphs. *IBM J. Res. Developments* **17** (1973) 420-425.

[7] J. Falkner, F. Rendl et H. Wolkowicz, A computational study of graph partitioning. *Math. Programming* **66** (1994) 211-240.

[8] D.M. Gay, Computing optimal locally constrained steps. *SIAM J. Sci. Statist. Comput.* **2** (1981) 186-197.

[9] M. Held, P. Wolfe et H.D. Crowder, Validation of the subgradient optimization. *Math. Programming* **6** (1974) 62-88.

[10] D.S. Johnson, C.R. Aragon, L.A. McGeoch et C. Schevon, Optimization by simulated annealing: An experimental evaluation, Part 1, Graph partitioning. *Oper. Res.* **37** (1989) 865-892.

[11] B.W. Kernighan et S. Lin, An efficient heuristic procedure for partitioning graphs. *The Bell System Technical J.* **49** (1970) 291-307.

[12] T. Lengauer, *Combinatorial algorithms for integrated circuit layout.* Wiley, Chicester (1990).

[13] J.M. Martinez, Local minimizers of quadratic functions on euclidean balls and spheres. *SIAM J. Optim.* **4** (1994) 159-176.

[14] P. Michelon, N. Brossard et N. Maculan, A branch-and-bound scheme for unconstrained $0-1$ quadratic programs, Rapport Technique # 960, DIRO. Université de Montréal. *SIAM J. Optim.* (soumis).
[15] J.J. Moré et D.C. Sorensen, Computing a trust region step. *SIAM J. Sci. Statist. Comput.* **4** (1983) 553-572.
[16] G.L. Nemhauser et L.A. Wolsey, *Integer and Combinatorial Optimization*. Wiley, New York (1988).
[17] C. Roucairol et P. Hansen, Problème de la bipartition minimale d'un graphe. *RAIRO: Oper. Res.* **21** (1987) 325-348.
[18] D.C. Sorensen, Newton's method with a model trust region modification. *SIAM J. Numer. Anal.* **19** (1982) 406-426.

to access this journal online:
www.edpsciences.org