Un Branch and Bound utilisant la programmation semidéfinie pour un problème de placement de tâches et un problème de partition des sommets d'un graphe

M.-C. Costa, N. Derhy, et F. Roupin

CEDRIC, Conservatoire National des Arts et Métiers, 292 rue Saint-Martin, 75141 Paris
{costa,nicolas.derhy,roupin}@cnam.fr

1 Introduction

La programmation semidéfinie est aujourd'hui très utilisée notamment dans le domaine de l'approximation. Elle a notamment été rendue populaire par Goemans et Williamson [3] pour l'obtention d'un meilleur ratio d'approximation pour plusieurs problèmes classiques d'optimisation comme la recherche d'une coupe de poids maximum dans un graphe.

Nous étudions ici l'intérêt de l'utilisation de la programmation semidéfinie concernant une approche de résolution exacte pour un problème de partition de graphe ainsi que pour un problème de placement de tâches sur un système multi-processeurs.

2 Présentation des problèmes et de la méthode utilisée

2.1 Les problèmes Min K-cut et CMAP

Le premier problème étudié est un problème classique de théorie des graphes. Soit $G = (V, E)$ un graphe non orienté dont les arêtes (ij) sont pondérées par des entiers strictement positifs w_{ij} et soit un entier K donné. Le problème Min K-cut consiste à partitionner les sommets de G dans K paquets de façon à minimiser la somme des poids des arêtes reliant deux paquets distincts. Pour $K = 2$, on retrouve le problème classique de recherche d'une coupe minimum résolu en temps polynomial par exemple avec l'algorithme de Ford et Fulkerson. Le problème est cependant NP-difficile dans le cas général [4].

Le second problème étudié consiste à placer un ensemble T de tâches sur un ensemble P de processeurs. Chaque tâche possède un coût d'exécution dépendant du processeur sur lequel elle est exécutée. De plus, il existe un coût de communication entre deux tâches lorsque celles-ci ne sont pas exécutées sur le même processeur. Enfin, l'exécution de chaque tâche nécessite une certaine quantité de ressource et chaque processeur ne dispose que d'une quantité limitée de ressource. Le problème CMAP consiste à assigner chaque tâche à un processeur de manière à ce que la somme des ressources consommées par les tâches placées sur un même processeur ne dépasse pas la capacité du processeur. On recherche ainsi une solution qui minimise la somme des coûts d'exécution et des coûts de communication. Ce problème est NP-difficile et à moins que $P = NP$, il ne peut exister aucun algorithme polynomial approché à ratio de performance constant [6].

2.2 Description de la méthode de résolution utilisée

L'objectif est de résoudre de manière exacte ces deux problèmes. Pour cela, nous allons utiliser une méthode de type Branch and Bound ($B\&B$). Cependant, à la différence des méthodes classiques de résolution, la borne en chaque noeud de l'arbre de recherche sera calculée par une relaxation semidéfinie et non pas par une relaxation continue d'un modèle linéaire.

Cette relaxation semidéfinie s'inspire principalement des travaux de Frieze et Jerrum dans [2] où ils généralisent le travail de Goemans et Williamson dans [3] afin d'obtenir un algorithme polynomial approché pour Max K-cut (le problème de maximisation correspondant à Min K-cut). D'autres relaxations semidéfinies plus efficaces existent, notamment pour CMAP [5], mais leur taille permet difficilement leur utilisation pour un algorithme de type $B\&B$.

Prenons l'exemple de la formulation utilisée pour le problème Min K-cut :

$$\text{(Min K-cut)} \begin{cases} Min\ \dfrac{K-1}{K} \displaystyle\sum_{ij \in E} w_{ij}(1 - \ v_i^T v_j) & & \\ s.c \quad v_i^T v_i = 1 & \forall i \in V & (1) \\ \quad e_k^T e_k = 1 & \forall k \in \{1, \ldots, K\} & (2) \\ \quad e_k^T e_{k'} = -\frac{1}{K-1} & \forall k \neq k' & (3) \\ \quad \displaystyle\sum_{i \in V} v_i^T e_k \ \geq \dfrac{K-n}{K-1} & \forall k \in \{1, \ldots, K\} & (4) \\ \quad v_i \in \{e_1, \ldots, e_K\} & \forall i \in V & (5) \end{cases}$$

Toutes les variables de ce problème sont des vecteurs et nous souhaitons connaître la valeur du produit scalaire de ces vecteurs entre eux. v_i représente le vecteur associé au sommet i et e_k le vecteur associé au paquet k. Les contraintes (1) et (2) forcent les vecteurs à être normés et la contrainte (4) force chaque paquet à contenir au moins un sommet. Un sommet i appartiendra à un paquet k si $v_i = e_k$, soit $v_i^T e_k = 1$ puisque tous les vecteurs sont normés. La contrainte (5) force donc chaque sommet à appartenir à l'un des paquets.

A partir de cette formulation du problème, nous obtenons une relaxation semidéfinie en relâchant la contrainte (5) et en la remplaçant par celle indiquant que la matrice carrée formée par tous les produits scalaires entre les vecteurs doit être semidéfinie. C'est cette relaxation qui sera résolue en chaque noeud de l'arbre de recherche. On peut écrire un modèle similaire pour CMAP où les v_i représenteront les tâches et les e_k les différents processeurs.

3 Conclusion

Nous n'avons pas choisi ces deux problèmes au hasard pour tester l'efficacité de l'utilisation d'une relaxation semidéfinie dans un $B\&B$. En effet, les relaxations linéaires continues de ces problèmes ne donnent pas des bornes assez intéressantes pour permettre une résolution exacte efficace sur des instances de grande taille [1].

Il faut également être conscient du fait que la relaxation semidéfinie doit permettre d'obtenir une borne bien meilleure que celle obtenue par la relaxation linéaire continue. En effet, résoudre une relaxation semidéfinie nécessite un temps de calcul beaucoup plus important qu'une relaxation linéaire continue de même taille.

Nous comparons les résultats obtenus par CPLEX 9.0 à partir des modèles linéaires et ceux obtenus par un $B\&B$ utilisant le solveur de programmes semidéfinis CSDP pour résoudre la relaxation décrite dans le paragraphe précédent. Les instances sont générées aléatoirement et permettent de comparer le temps de résolution ainsi que le nombre de noeuds développés dans l'arbre de recherche. Si notre méthode ne s'avère pas systématiquement meilleure que CPLEX, elle semble néanmoins plus rapide sur environ la moitié des instances testées, notamment pour Min K-cut. La résolution de programmes semidéfinis étant un domaine relativement récent, on peut espérer que les prochaines évolutions des solveurs de programmes semidéfinis permettront sur certains problèmes spécifiques d'obtenir des résultats meilleurs que ceux obtenus avec la programmation linéaire classique.

Références

1. S. Elloumi, F. Roupin et E. Soutif. Comparison of Different Lower Bounds for the Constrained Module Allocation Problem. Rapport scientifique CEDRIC No 473 (2003)
2. A. Frieze et M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. Algorithmica 18, 67-81 (1997)
3. M. X. Goemans et D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42, 1115-1145 (1995)
4. O. Goldschmidt et D. S. Hochbaum. A Polynomial algorithm for the k-cut problem for fixed k. Mathematics of Operations Research 19, 24-37 (1994)
5. F. Roupin. From Linear to Semidefinite Programming : an Algorithm to obtain Semidefinite Relaxations for Bivalent Quadratic Problems. Journal of Combinatorial Optimization 8(4), 469-493 (2004)
6. F. Roupin. On approximating the memory-Constrained Module Allocation Problem. Information Processing Letters 61(4), 205-208 (1997)