Bornes inférieures pour le RCPSP : une approche hybride programmation par contraintes – programmation linéaire

S. Demassey, C. Artigues et P. Michelon

Laboratoire Informatique d'Avignon, 339 chemin des Meinajariès,
Agroparc BP 1228, 84911 Avignon Cedex 9
{ *sophie.demassey, christian.artigues, philippe.michelon* }*@lia.univ-avignon.fr*

Mots-clés : Ordonnancement, Méthodes polyédrales, Programmation par contraintes.

1 Présentation

Le RCPSP, ou *problème d'ordonnancement de projets à moyens limités*, consiste à organiser, en un temps minimum, la réalisation de n tâches non préemptives au moyen de m ressources cumulatives et renouvelables. Un ordre partiel E sur l'ensemble des tâches est donné, signifiant des contraintes de précédence entre les tâches. De plus, chaque tâche i nécessite, durant les p_i unités de temps de son exécution, une quantité constante r_{ik} de chaque ressource k. Il en suit une concurrence entre les tâches pour l'utilisation des ressources : à chaque instant, la quantité totale d'une ressource k requise par l'ensemble des tâches en cours d'exécution ne peut excéder la disponibilité limite R_k de la ressource.

Puisqu'il généralise le problème classique du Job-shop, le RCPSP est évidemment un problème d'optimisation combinatoire NP-difficile, pour lequel de petites instances ($n = 60$) sont encore non résolues. Parmi les nombreuses évaluations par défaut existantes pour le RCPSP, certaines sont calculées par programmation linéaire, d'autres par propagation de contraintes. Récemment, Brucker and Knust [1] ont présenté une borne inférieure, la meilleure existante à ce jour sur les instances KSD, basée sur les deux techniques à la fois.

Nous proposons, à notre tour, une méthode couplant programmation par contraintes et programmation linéaire en nombres entiers, encouragés par l'efficacité démontrée de telles approches hybrides pour de nombreux problèmes d'optimisation combinatoire. Notre approche reprend la manière utilisée dans [1] de prétraiter, par des techniques de propagation de contraintes, une formulation du problème en un programme linéaire en nombres entiers, de façon à renforcer une éventuelle relaxation de ce programme. Au-delà, nous proposons une coopération plus poussée entre ces deux phases de résolution. Cette coopération passe par la génération de coupes pour le programme linéaire, basées sur des déductions effectuées par propagation de contraintes.

2 Méthode de coopération PPC–PLNE

La méthode que nous employons se décompose en deux phases successives : programmation par contraintes puis, programmation linéaire entière par génération de coupes.

Programmation par contraintes :

- Le RCPSP est vu comme un problème de satisfaction de contraintes P_0 de variables $S_j - S_i$ et de domaines $[b_{ij}, -b_{ji}]$, où S_i représente la date de début de la tâche i et où la *distance* b_{ij} est un minorant (éventuellement négatif) de $S_j - S_i$ sur l'ensemble des ordonnancements réalisables S.
- Les contraintes initiales du CSP sont : les contraintes de précédence (initialisation de $B = (b_{ij})$ à partir de E et d'une borne supérieure donnée UB) et des contraintes de ressources simples (*disjonctions*).
- Des techniques classiques de propagation des contraintes de ressources (edge-finding primal et dual, triplets symétriques, sélection immédiate) sont appliquées, en vue de réduire les domaines des variables, *i.e.* d'augmenter les b_{ij}.
- Nous implémentons notre propre technique de *shaving*, basée sur la réfutation du séquencement relatif entre deux tâches i et j. EX. : la contrainte $c = i \rightarrow j$ (i précède j) est temporairement propagée dans P_0. Si elle est prouvée inconsistante, la négation de c entre dans la définition de P_0. Sinon, la propagation de c peut éventuellement induire une nouvelle contrainte, par exemple, $c' = h \| l$ (les tâches h et l s'exécutent en parallèle).

La phase PPC se termine si $b_{0(n+1)} = UB$ ($b_{0(n+1)}$ est alors la valeur optimale du problème) ou, quand plus aucune déduction n'est faite. Dans ce cas, la matrice B est fixée et stockée, de même que la relation de disjonction D, les cliques de disjonctions C calculées pour le edge-finding et les déductions intermédiaires du shaving (EX. : $(i \rightarrow j) \Longrightarrow (h \| l)$).

Programmation linéaire :

- Le RCPSP est formulé en un programme linéaire en nombres entiers P_1, renforcé grâce à B et D.
- La borne inférieure calculée est la valeur optimale de la relaxation linéaire $\tilde{P}_1$ augmentée de *coupes*. Elle est, par construction de P_1 au moins égale à $b_0(n+1)$.
- Les coupes générées sont de deux sortes : Les premières expriment certaines règles de edge-finding et sont appliquées aux cliques C. Les secondes transcrivent les déductions intermédiaires du shaving en inégalités linéaires valides pour $\tilde{P}_1$. EX. : pour la formulation disjonctive de Balas [2] pour le RCPSP, incluant des variables booléennes x_{ij} définies par $x_{ij} = 1 \iff i$ précède j, l'inégalité $x_{ij} + x_{hl} + x_{lh} \leq 1$ est valide puisqu'elle traduit l'implication déduite du shaving : $(i \rightarrow j) \Longrightarrow (h \| l)$.

3 Résultats expérimentaux

Nous avons testé cette méthode sur deux modèles linéaires en nombres entiers : la formulation disjonctive de Balas [2], relâchant en plus des contraintes d'intégrité une partie des contraintes de ressources, et la formulation en temps discretisé de Pritsker [3]. Bien que mieux adaptée à la génération de coupes telles que décrites ci-dessus, les résultats de la formulation disjonctive restent largement en deça des résultats obtenus de la formulation en temps discretisé. En revanche, les résultats du modèle en temps discretisé tendent à approcher ceux de Brucker et Knust.

[1] P. Brucker et S. Knust, "A linear programming and constraint propagation-based lower bound for the RCPSP", *European Journal of Operational Research* 127, 2000, 355–362.
[2] E. Balas, "Project scheduling with resource constraints", in E.M.L. Beale (ed.), *Applications of Mathematical Programming Techniques*, American Elsevier, 1970.
[3] A. Pritsker, L. Watters et P. Wolfe, "Multiproject scheduling with limited resources: a zero-one programming approach", *Management Science* 16, 1969, 93–108.