Contraintes globales pour l'ordonnancement des transferts de données pour les missions spatiales

Julien Rouzot[1,2], Christian Artigues[1], Philippe Garnier[2], Emmanuel Hebrard[1], Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{julien.rouzot,christian.artigues,emmanuel.hebrard,pierre.lopez}@laas.fr
[2] IRAP, Université de Toulouse, CNRS, Toulouse, France
{julien.rouzot,philippe.garnier}@irap.omp.eu

Mots-clés : *Programmation par contraintes, contraintes globales, missions spatiales, transfert de données*

1 Introduction

Dans le cadre des missions spatiales où les ressources à bord sont limitées, il est essentiel de transférer les données acquises de manière efficace afin de limiter le risque de saturation des mémoires. Nous nous intéressons ici en particulier au problème du vidage des données de l'orbiteur Rosetta, défini par Chien, Rabideau et al. [1]. Dans l'*overlapping Memory Dumping Problem (oMDP)*, un certain nombre d'activités, planifiées à l'avance, produisent des données temporairement stockées dans la mémoire de leur instrument. Ces données doivent être transférées à la Terre avant que les mémoires des instruments ne soient saturées. De plus, un ordre de priorité doit être établi pour chacune des fenêtres de visibilité entre la sonde et les stations sol. Cet ordre de priorité dicte le partage de bande passante entre les différents instruments lors du transfert des données pendant cette fenêtre. Il s'agit d'un ordre partiel, plus précisément un *classement* : les mémoires de même priorité se partagent la bande passante grâce à un algorithme de type Round-Robin (file d'attente circulaire). Ainsi, les mémoires avec la plus grande priorité peuvent utiliser toute la bande passante initiale, puis la bande passante restante est redistribuée jusqu'à ce qu'il n'en reste plus, ou que les mémoires soient toutes vides. On considère ici que le remplissage des mémoires est complètement déterminé par la planification des expériences scientifiques et donc connu à l'avance. Les priorités ne peuvent être changées qu'en début de fenêtre de visibilité. L'objectif est de minimiser le pic d'utilisation maximal des mémoires afin de garantir un certain niveau de robustesse.

Ce problème a précédemment été résolu avec succès par des méthodes heuristiques [1, 2]. Nous proposons dans cette communication un modèle de programmation par contraintes (PPC) pour le résoudre de manière exacte, en incluant plusieurs nouvelles contraintes globales pour le partage de bande passante en fonction de priorités. Les résultats montrent la pertinence et l'efficacité d'une telle approche exacte pour la résolution de ce problème. Un des buts dans la conception de ces contraintes est leur généricité, afin qu'elles puissent être appliquées à toute mission spatiale nécessitant des transferts de données par priorité.

2 Contraintes globales pour l'oMDP

Dans l'*oMDP*, on considère m fenêtres de vidage telles que pour une fenêtre j, la bande passante à partager entre les mémoires est une constante δ_j. On a n mémoires concurrentes, avec une capacité limitée. On considère les variables de décisions suivantes : $r_{i,j}$ l'usage maximum de la mémoire i par rapport à sa capacité au cours de la fenêtre j. $mem_{i,j}$ l'usage de la mémoire i au début de la fenêtre j et, pour chaque fenêtre, une priorité $p_{i,j}$ à chaque mémoire (1 est la

meilleure priorité, n la pire). On veut minimiser $rmax = \max_{i,j}(r_{i,j})$. La principale difficulté est d'exprimer efficacement la relation entre une affectation de priorité et l'évolution de l'utilisation des mémoires résultante. Cette difficulté a motivé le développement de la contrainte globale `PRIORITY TRANSFER` qui, pour chaque fenêtre j, lie les variables $p_{i,j}, mem_{i,j}, mem_{i,j+1}$ et $r_{i,j}$. Cette contrainte utilise l'algorithme `Simulation`, qui permet de calculer l'évolution des mémoires au cours du temps pour une affectation de priorité donnée [2]. Afin d'accélérer la résolution, nous avons aussi développé deux contraintes globales qui permettent d'éliminer les solutions équivalentes. En effet, des affectations de priorité différentes peuvent mener au même classement et par conséquent, au même partage de la bande passante. Par exemple, l'affectation $p_{1,1} = 1,\ p_{1,2} = 2,\ p_{1,3} = 2$ est équivalente à l'affectation $p_{1,1} = 1,\ p_{1,2} = 3,\ p_{1,3} = 3$. Nous forçons donc les variables de priorité à respecter les règles d'un classement *Dense Ranking*. Ce classement impose que le groupe de priorité 1 soit non vide et que si le groupe de priorité $i+1$ est non vide, alors le groupe de priorité i doit être non vide également. Ainsi, $1, 2, 2$ est valide, mais pas $1, 3, 3$. La contrainte globale `DENSE RANKING` élimine les valeurs des domaines des variables de priorité qui mèneraient à une violation du *Dense Ranking*. Un autre type de symétrie est possible, quand un groupe de mémoires prioritaires utilisent toute la bande passante. Dans ce cas, toutes les mémoires moins prioritaires auront un taux de transfert nul. La contrainte `PRIORITY SYMMETRY` fixe la borne supérieure des variables de priorité à la valeur pour laquelle le transfert sera nul. Pour l'exploration de l'arbre de recherche, nous utilisons une stratégie de branchement heuristique qui tend à affecter les meilleures priorités aux mémoires qui saturent en premier, si on interdit le transfert des données. Enfin, nous avons développé une méthode utilisant les solutions des heuristiques décrites dans [2] comme solution initiale pour notre modèle. Une stratégie de recherche à divergences limitées (*Limited Discrepancy Search, LDS*) est implémentée pour améliorer cette solution initiale. Une comparaison des performances d'un modèle PPC naïf (sans les contraintes globales, sans stratégie de branchement), du modèle global, de la version intégrant LDS et de la meilleure heuristique pour *oMDP* (*Repair descent*) est présentée dans le tableau 1 :

	Instances			
	MTP011	MTP012	MTP013	MTP014
	rmax, temps	rmax, temps	rmax, temps	rmax, temps
Modèle naïf	91.1%, 1h	62.6%, 1h	66.6%, 1h	65.4%, 1h
Modèle global	**53.6*%**, <1s	**27.5*%**, 6s	46%, 1h	47.6%, 1h
Modèle global + LDS	**53.6*%**, <1s	**27.5*%**, <1s	**45.2*%**, <1s	**47.1%**, 1h
Repair descent	**53.6*%**, <1s	**27.5*%**, <1s	**45.2*%**, <1s	47.2%, 1h

TAB. 1 – Comparaison de *rmax* pour les deux modèles sur les instances réelles de Rosetta pour un temps limite d'une heure. Les meilleures solutions sont représentées en gras.

Malgré un temps de calcul moyen plus long que la meilleure heuristique, ces résultats montrent la pertinence d'une approche exacte PPC pour résoudre l'*overlapping Memory Dumping Problem* et l'efficacité des contraintes globales présentées. La méthode LDS permet d'améliorer les résultats des méthodes heuristiques précédentes.

Références

[1] Gregg Rabideau, Steve Chien, M. Galer, Federico Nespoli, and Manuel Costa. Managing spacecraft memory buffers with concurrent data collection and downlink. *Journal of Aerospace Information Systems*, 14(12) :637–651, 2017.

[2] Emmanuel Hebrard, Christian Artigues, Pierre Lopez, Arnaud Lusson, Steve Chien, Adrien Maillard, and Gregg Rabideau. An efficient approach to data transfer scheduling for long range space exploration. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4635–4641. International Joint Conferences on Artificial Intelligence Organization, July 2022. Main Track.