Résolution du problème SALB3PM grâce à une approche SAT

Matthieu Py, Arnauld Tuyaba

Université Clermont Auvergne, Mines Saint-Etienne, CNRS, LIMOS, F-63000 Clermont-Ferrand, France

{matthieu.py, arnauld.tuyaba}@uca.fr

Mots-clés : *SALB3PM, programmation par contraintes, problème SAT.*

1 Présentation du problème

Le problème SALB3PM (*Simple Assembly Line Balancing Problem with Power Peak Minimization*) est un problème d'optimisation qui consiste, étant donné un ensemble de tâches et un ensemble de machines, à construire l'ordonnancement des tâches sur les différentes machines de manière à minimiser le pic d'énergie de l'ordonnancement. Au niveau des données, chacune des n tâches a une durée de traitement et une consommation énergétique (à prendre en compte à chaque instant de son exécution). On doit décider sur quelle machines (les machines sont numérotées de 1 à m) on affecte chaque tâche et à quelle date elle débute (elle doit débuter et finir entre la date 1 et le temps de cycle c). Il y a aussi des précédences entre certaines tâches : dire qu'une tâche est le prédecesseur d'une autre tâche signifie que soit on doit affecter la première tâche sur une machine mise en amont de celle de la seconde tâche, soit qu'on doit les mettre sur la même machine mais que la première tâche doit être réalisée avant la seconde. Le but de cet article est de proposer une méthode issue de la programmation par contraintes pour ce problème et de vérifier sa pertinence sur quelques instances, par comparaison avec l'implémentation d'un modèle de programme linéaire en nombres entiers (ILP) existant [2].

2 Approche SAT

On modélise le problème SALB3PM sous forme d'un problème de satisfaisabilité booléenne (problème SAT). Comme le problème SAT est un problème de décision, et non un problème d'optimisation, on modélise initialement le problème de faisabilité qui consiste à trouver une solution qui respecte toutes les contraintes du problème. Les décisions à prendre sont représentées par trois jeux de variables de décision binaires pour décider : (1) si la tâche i est affectée à la machine j, (2) si la tâche i débute à la date t et enfin (3) si la tâche i est active à la date t. Ce troisième jeu de variables peut être déduit à partir du deuxième mais aide à la modélisation du problème et en particulier à sa composante d'optimisation que l'on verra ensuite.

Les contraintes du problème de faisabilité sont représentées sous forme de clauses (disjonctions de variables ou de leurs opposées) et sont listées ci-dessous : (1) Chaque tâche débute sur une et une seule machine. (2) Chaque tâche débute à une et une seule date. (3) Chaque tâche ne peut débuter que si elle peut terminer avant la fin du temps de cycle. (4) A un instant donné et pour une machine donnée, il ne peut y avoir au maximum qu'une tâche en cours d'exécution. (5) Si deux tâches i et j sont soumises à une relation de précédence, soit la première tâche se déroule sur une machine qui précède la machine sur laquelle se déroule la seconde tâche, soit les deux tâches se déroulent sur la même machine mais dans ce cas la première tâche est exécutée avant la seconde. (6) Les tâches dont la durée dépasse la moitié du temps de cycle sont obligatoirement actives aux dates médianes. (7) On interdit d'affecter une tâche sur une machine si cette affectation ne permet pas aux séquences de tâches qui la précèdent (respectivement qui la suivent) d'être placées en amont (respectivement en aval) de cette tâche. (8) On interdit d'affecter une tâche à une date donnée sur une machine donnée si

cette affectation ne permet pas à une suite de tâches qui la précèdent (respectivement qui la suivent) d'être placées en amont (respectivement en aval) de cette tâche.

Les contraintes (5) à (8) servent à simplifier le modèle, à la fois en fixant la valeur de certaines variables mais aussi en allégeant le modèle (si la valeur d'une variable est fixée, les contraintes impliquant cette variable avec la même polarité ne sont pas intégrées au modèle).

Une fois que l'on a écrit le modèle mathématique précédent, sa résolution par un algorithme de résolution du problème SAT permet de trouver un ordonnancement faisable. Pour ensuite déterminer l'ordonnancement avec un pic d'énergie minimum, on relance, autant de fois que possible, notre algorithme en intégrant au modèle des contraintes supplémentaires. Ces contraintes correspondent à l'interdiction d'exécuter en même temps un ensemble de tâches qui l'étaient dans la dernière solution calculée et dont la somme des énergies est supérieure au pic d'énergie de la meilleure solution rencontrée depuis le début de l'exécution de l'algorithme. Après ajout de ces contraintes, on relance la résolution du problème de faisabilité. Lorsqu'il n'est plus possible de trouver une solution pour le problème de faisabilité, la meilleure solution rencontrée jusqu'à cet instant est une solution optimale pour le problème SALB3PM.

3 Expérimentations

Nous avons expérimenté notre algorithme sur plusieurs instances de la littérature. Les principaux résultats sont listés ci-dessous, avec pour chaque instance les résultats obtenus en implémentant le modèle existant sur CPLEX [2] et les résultats obtenus avec notre algorithme. Le problème SAT sous-jacent est résolu grâce au solveur Sat4j [1]. On donne, pour chaque instance, le nombre de tâches n, de machines m, le temps de cycle c et, pour chaque méthode, le pic d'énergie de la meilleure solution calculée, la durée d'exécution (au plus 1 heure) et son statut (optimal ou non). On constate que l'approche proposée permet de trouver la solution optimale sur plus d'instances (12 instances sur 14) qu'avec l'approche ILP connue (8 sur 14).

Instance	Données			Résultat CPLEX			Résultat SAT		
	n	m	c	Pic	Temps	Statut	Pic	Temps	Statut
mansoor-1	11	4	48	**145**	0.69 s	Optimal	**145**	**0.301 s**	Optimal
mansoor-2	11	2	94	**77**	5.03 s	Optimal	**77**	**0.752 s**	Optimal
mitchell-1	21	8	14	**221**	1.44 s	Optimal	**221**	**0.608 s**	Optimal
mitchell-2	21	3	39	**85**	427.94 s	Optimal	**85**	**11.689 s**	Optimal
roszieg-1	25	10	14	**254**	104.36 s	Optimal	**254**	**9.234 s**	Optimal
roszieg-2	25	4	32	**117**	163.59 s	Optimal	**117**	**4.447 s**	Optimal
heskiaoff-1	28	8	138	≤ 290	≥ 3600 s		**251**	**3489.82 s**	Optimal
heskiaoff-2	28	3	342	≤ 140	$\geq$ **3600 s**		$\leq$ **117**	$\geq$ **3600 s**	
buxey-1	29	14	25	$\leq$ **292**	$\geq$ **3600 s**		$\leq$ **292**	$\geq$ **3600 s**	
buxey-2	29	7	47	≤ 350	≥ 3600 s		**172**	**229.212 s**	Optimal
sawyer-1	30	14	25	**395**	157 s	Optimal	**395**	**4.272 s**	Optimal
sawyer-2	30	7	47	≤ 310	≥ 3600 s		**214**	**1310.82 s**	Optimal
gunther-1	35	14	40	**394**	2297 s	Optimal	**394**	**6.397 s**	Optimal
gunther-2	35	9	54	≤ 450	≥ 3600 s		**295**	**20.589 s**	Optimal

TAB. 1 – Comparaison des approches ILP et SAT sur quelques instances

Références

[1] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7 :59–64, 2010.

[2] Paolo Gianessi, Xavier Delorme, and Oussama Masmoudi. Simple Assembly Line Balancing Problem with Power Peak Minimization. In *IFIP International Conference on Advances in Production Management Systems (APMS)*, pages 239–247, 2019.