Minimisation de la date d'achèvement et du nombre de travaux en retard pour l'ordonnancement multiagent non-disjoint

Tran VAN UT(1), Ameur SOUKHAL(1), Nguyen HUYNH TUONG(2)

(1) Laboratoire d'Informatique de Tours EA 6300
Equipe Recherche Opérationnelle, Ordonnancement et Transport ERL-CNRS 6305
64 av. Jean Portalis, 37200 Tours, France
(2) Faculty of Computer Science & Engineering,
Ho Chi Minh City University of Technology, VNU-HCM, 268 Ly Thuong Kiet Street,Ho Chi Minh City 740500, Vietnam
vanut.tran@etu.univ-tours.fr, ameur.soukhal@univ-tours.fr, htnguyen@hcmut.edu.vn

Mots-clés : *Recherche Opérationnelle, Ordonnancement multiagent, Algorithmes gloutons, Programme dynamique, Programmation Linéaire.*

1 Introduction

Les problèmes d'ordonnancement multiagent sont des problèmes d'optimisation combinatoire. Il s'agit de trouver un meilleur compromis entre différents utilisateurs sollicitant des ressources communes pour la réalisation de leurs activités ou leurs travaux respectifs en respectant leurs contraintes d'exécution. Ainsi, chaque utilisateur (agent) souhaite optimiser un critère appliqué sur ses travaux [1]. Ces problèmes se rencontrent dans tout système de production de bien ou de service.

Selon l'intersection des sous-ensembles des travaux des agents, Agnetis et al. ont identifié quatre classes de problèmes d'ordonnancement multiagent [1]. Celle qui nous intéresse dans cette étude est définie comme suit. Les sous-ensembles des travaux sont non-disjoints : $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K$. $\mathcal{N}_k$ peut correspondre à une commande du client k ou d'un utilisateur k qui soumit ses travaux sur un data center. Ainsi nous avons : $\cup_{k=1}^{K} \mathcal{N}_k = \mathcal{N}$ et $\mathcal{N}_k \cap \mathcal{N}_{k'} \neq \emptyset$. Ces travaux sont à exécuter sur machines parallèles identiques. Dans cette étude, deux types de fonctions objectif sont considérés : minimisation de la date d'achêvement globale ($C_{\max}$) et la minimisation du nombre de travaux en retard ($\sum U_i$). Pour calculer une solution optimale au sens de Pareto, nous utilisons l'approche ε-contrainte.

Notre démarche est illustrée au cas de deux agents ($K = 2$). Nous montrons comment certains résultats peuvent (ou pas) se génfraliser aux cas d"un nombre K quelconque d'agents. Selon la notation des problèmes d'ordonnancement multiagent introduite dans [1], le problème étudié est noté : $P2|ND, d^B|(C^A_{max}, U^B_j)$, où ND indique que les ensembles des travaux sont non-disjoints. Quand l'approche ε-contrainte est considérée, nous notons le problème étudié par $P2|ND, d^B, \sum U^B_j \leq Q_B|C^A_{max}$. Dans ce dernier cas, le problème consiste à trouver un ordonnancement minimisant l'objectif de l'agent A tout en conservant la valeur de l'objectif de l'agent B inférieure à un seuil donné Q_B.

Le problème étudié est NP-difficile [2]. Cependant, nous montrons la proposition suivante indiquant la structure d'une solution optimale.

Proposition Si le problème $Pm|ND, d^B, \sum U^B_j \leq Q_B|C^A_{max}$ accepte une solution faisable, alors il est possible de construire une solution optimale telle que sur chaque machine nous avons :

1. Les travaux de l'agent B sont ordonnancés selon l'ordre SPT.

2. Le travail en retard J_j de $\mathcal{N}^B \backslash \{\mathcal{N}^A \cap \mathcal{N}^B\}$ est ordonnancé après les travaux de l'agent A.

2 Programmes linéaires en nombres entiers

Pour résoudre le problème à l'optimum, nous proposons deux programmes linéaires en nombres entiers : indexé machine et indexé temps. Ces deux modèles sont également utilisés pour calculer le front de Pareto. Les résultats expérimentaux montrent que le modèle indexé temps est plus performant, permettant ainsi de générer le front de Pareto optimal pour des instances allant jusqu'à 70 travaux en moins d'une heure.

3 Deux heuristiques gloutonnes

Par la première, nous cherchons à minimiser le makepan de l'agent A par la règle *LPT-FAM* (performante pour résoudre le problème $Pm||C_{max}$). On suppose que les travaux de l'agent B sont numérotés selon l'ordre SPT. L'heuristique est définie comme suit : Ordonnancer les travaux $n_B - Q_B$ premiers travaux de B (ensemble E), puis les travaux de $\mathcal{N}^A \backslash \{\mathcal{N}^A \cap E\}$ et terminer par les travaux de B non encore exécutés. La complexité de cette heuristique est de $O(n_A log(n_A) + n_B log(n_B))$.

La deuxième n'est autre qu'une modification de la précédente où certains travaux seront réaffectés. Itérativement, les travaux de $E \cup \mathcal{N}^A$ sont ordonnancés selon la règle *LPT-FAM*, et si l'exécution du travail suivant de E le conduit à être en retard, alors nous cherchons la meilleure position pour qu'il soit à l'heure. La complexité reste la même en $O(n_A log(n_A) + n_B log(n_B))$.

4 Deux heuristiques pseudo-polynomiales

La première heuristique commence par l'ordonnancement des $n_B - Q_B$ premiers travaux de l'agent B par le programme dynamique (PD) introduit pour résoudre le $Pm||C_{max}$. Si la valeur optimale du makepan obtenu est supérieure à d^B alors pas de solution réalisable. Sinon, ordonnancer les travaux de A suivis de B restant, en appliquant à chaque fois le PD. Cet algorithme s'exécute en $O(n^2 + nUB^2)$ où UB est la borne supérieure du makespan.

La deuxième est inspirée de l'heuristic 2, mais au lieu d'utiliser la règle *LPT-FAM* pour l'affectation des travaux aux machines, elle utilise le programme dynamique PD. Cet algorithme s'exécute en $O(nlogn + nUB^2)$.

5 Résultats expérimentaux

Trois métriques sont utilisées pour comparer les front de Pareto et ainsi analyser les performances des méthodes proposées : i) le nombre de solutions strictement non-dominées trouvées par chaque algorithme ; ii) la distance générationnelle qui indique la moyenne des distances euclidiennes entre les solutions approximatives et les solutions exactes ; iii) l'hypervolume.

Les résultats expérimentaux montrent ainsi la dominance de la deuxième et quatrième heuristiques par rapport aux deux autres.

Références

[1] A. Agnetis, J-C Billaut, D. Pacciarelli, A. Soukhal. Multiagent Scheduling : Models and Algorithms. Springer (2014).

[2] F. Sadi, A. Soukhal, J.-C. Billaut, Solving multi-agent scheduling problems on parallel machines with a global objective function, RAIRO - Operations Research, 48 (2) (2014), 255–269.