ÉLÉMENT DE PORTFOLIO 03

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : An Adversarial Model for Scheduling with Testing, par Christoph Dürr, Thomas Erlebach, Nicole Megow et Julie Meißner, publié dans la revue Algorithmica (82) 3630–3675, 2020.

URL de l'élément : `https://arxiv.org/pdf/1709.02592`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Si vous avez pu avoir l'occasion de travailler dans un projet industriel, alors vous avez sûrement remarqué qu'une grande partie du projet consiste à obtenir les données précises. Travailler avec des estimations grossières ne peut aboutir qu'à des solution non-optimales. Cette situation a motivé l'étude d'un modèle d'optimisation où les valeurs du problème seraient données sous forme d'un intervalle d'incertitude et l'algorithme a la possibilité d'obtenir la valeur exacte par des requêtes. Les premiers travaux dans ce modèle concernaient le nombre de requêtes minimum pour pouvoir exhiber une solution prouvée optimale. L'article que nous avons sélectionné pour l'évaluation HCERES combine le coût des requêtes avec la valeur objectif de la solution produite. Il a nécessité le développement de nouvelles techniques et ouvert la porte à de nouveaux types de problèmes dans le domaine de l'ordonnancement.

3 PRÉSENTATION DE CET ÉLÉMENT

Cet élément est le fruit d'une collaboration entre Christoph Dürr, Thomas Erlebach, Nicole Megow et Julie Meißner. Il a d'abord été présenté sous le titre "Scheduling with Explorable Uncertainty" à la conférence *The 9th Innovations in Theoretical Computer Science Conference (ITCS)*, 2018, puis publié dans sa version complète à Algorithmica, (82) 3630–3675, 2020.

Pour donner une application à notre modèle, imaginez que vous ayez à envoyer n fichiers à travers une ligne de communication. Chacun des fichiers j a une taille u_j, et le temps de transmission est proportionnel à sa taille. Si vous envoyez les fichiers dans l'ordre croissant des tailles, vous minimisez le temps moyen que les fichiers doivent attendre leur fin de transmission. Maintenant vous avez la possibilité de compresser les fichiers avant leur transmission. Cette compression prend un temps, que nous avons normalisé à 1, pour simplification, et aboutira à une nouvelle taille de fichier p_j. Donc cette opération n'est intéressante que si $1 + p_j < u_j$. Par exemple, pour un fichier déjà compressé, vous auriez $p_j = u_j$ et une unité de temps serait perdue pour rien. Or, à priori vous n'avez pas la possibilité de savoir si la compression est efficace avant de l'avoir faite (voir Figure 1 pour une illustration).

Un algorithme aura alors à tout moment la possibilité de compresser un fichier, ou de transmettre un fichier. Se posent alors les questions de comment mesurer la performance d'un algorithme et de trouver le meilleur algorithme. Nous adoptons le langage de l'ordonnancement pour ce problème, et appelons *tâches* les fichiers et *tester* l'opération de compression.

Une instance I du problème consiste en les durées u_j connues par l'algorithme et les durées p_j, initialement inconnues. On note C_j le temps qu'il a fallu depuis le début pour compléter l'exécution de la tâche j. La valeur objectif d'un algorithme A sur l'instance I est $A(I) = \sum C_j$. Elle est comparée avec OPT(I), qui est la valeur d'une solution optimale pour cette instance. Cette comparaison résulte en un *rapport de compétitivité*.

Une borne inférieure x à ce rapport est une construction qui montre qu'aucune algorithme ne peut atteindre un rapport plus petit que x. Une borne supérieure x est un algorithme avec une preuve qu'il atteint un rapport au plus x. Les résultats que nous avons obtenus sont indiqués dans le tableau 1. Ils concernent à la fois le cas d'algorithmes déterministes (dans le cas général et dans des cas particuliers) et d'algorithmes randomisés - clairement pour ce type de problème, des choix aléatoires aident pour se protéger des instance pire des cas. La borne inférieure déterministe utilise des instances structurées, où les durées u_j sont toutes identiques, ne

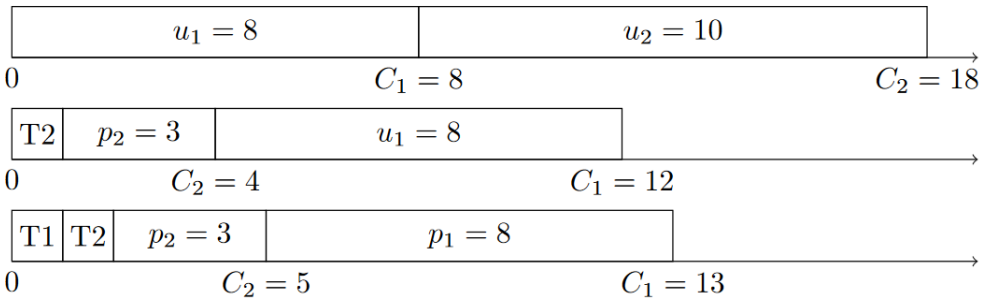

FIGURE 1 – Exemple pour 2 tâches. Le premier ordonnancement exécute les tâches dans l'ordre croissant des durées u_j, et a un coût $8 + 18 = 26$. Le deuxième commence par tester la tâche 2 et apprend la durée $p_2 = 3$. Il décide de l'exécuter avant la tâche 1, ce qui engendre le coût $4 + 12 = 16$. Le troisième ordonanncement teste les deux tâches et les exécute dans l'ordre croissant des durées p_j, pour finir avec un coût $5 + 13 = 18$. Pour cette instance le deuxième ordonnancemet est optimal.

ratio de compétitivité	borne inférieure	bornes supérieures	
algorithmes déterministes	1.8546	2	THRESHOLD
algorithmes randomisés	1.6257	1.7453	RANDOM
ratio dét. quand $\forall j : u_j = p$	1.8546	1.9338*	BEAT
...et $\forall j : p_j \in \{0, p\}$	1.8546	1.8668	UTE
...et $p \approx 1.989$	1.8546	1.8552	UTE

TABLE 1 – Nos résultats pour différents variantes du problème. La dernière colonne indique le nom de notre algorithme proposé. * ratio asymptotique seulement, quand la taille des instances tend vers infini.

fournissant aucune possibilité à l'algorithme de distinguer les tâches, et où les durées après test sont aux extrêmes, soit 0 (meilleur bénéfice du test) soit p (aucun bénéfice du test).

D'autres travaux ont suivi, et considèrent que le temps d'exécution des tâches n'est pas influencé par le test, et que la valeur u_j initialement connue, n'est qu'une borne supérieure sur le temps d'exécution. Différents variantes de modèles ont été étudiées par exemple dans [1–3].

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Susanne Albers and Alexander Eckl. Scheduling with Testing on Multiple Identical Parallel Machines. In Anna Lubiw, Mohammad Salavatipour, and Meng He, editors, *Algorithms and Data Structures*, volume 12808, pages 29–42. Springer International Publishing, Cham, 2021. Series Title : Lecture Notes in Computer Science.

[2] Fanny Dufossé, Christoph Dürr, Noël Nadal, Denis Trystram, and Óscar C. Vásquez. Scheduling with a processing time oracle. *Applied Mathematical Modelling*, 104 :701–720, April 2022.

[3] Retsev Levi, Thomas Magnanti, and Yaron Shaposhnik. Scheduling with Testing. *Management Science*, 65(2), 2018.