Apprentissage de précédences pour des problèmes de planification avec des réseaux de neurones en graphes

Hélène Verhaeghe[1,2], Quentin Cappart[3], Gilles Pesant[3], Claude-Guy Quimper[4]

[1] KULeuven, Louvain, Belgique
[2] UCLouvain, Louvain-la-Neuve, Belgique
[3] Polytechnique Montréal, Montréal, Canada
[4] Université Laval, Quebec, Canada

helene.verhaeghe@uclouvain.be,quentin.cappart@polymtl.ca,gilles.pesant@polymtl.ca,
claude-guy.quimper@ift.ulaval.ca

Résumé

Le problème de gestion de projet à contraintes de ressources consiste à planifier un ensemble de tâches consommant des ressources durant un horizon temporel sujet à certaines capacités de ressources et relations de précédences entre des paires de tâches. Le problème est naturellement représenté par un graphe dirigé, généralement appelé graphe de précédences, où des paires de tâches soumises à une précédence sont liées. Dans ce papier, nous proposons d'utiliser les capacités d'un réseau de neurones en graphes pour extraire de la connaissance du graphe de précédence. Ce papier est un résumé de l'article "Learning Precedences for Scheduling Problems with Graph Neural Networks"[3] publié à CP2024.

Mots-clés

Planification, graphe de précédence, réseau de neurones en graphes.

Abstract

The resource constrained project scheduling problem (RCPSP) consists of scheduling a finite set of resource-consuming tasks within a temporal horizon subject to resource capacities and precedence relations between pairs of tasks. The problem is naturally represented as a directed graph, commonly referred to as the precedence graph, by linking pairs of tasks subject to a precedence. In this paper, we propose to leverage the ability of graph neural networks to extract knowledge from precedence graphs. This paper is a summary of the paper "Learning Precedences for Scheduling Problems with Graph Neural Networks"[3] published at CP2024.

Keywords

Scheduling, Precedence graph, Graph neural network.

1 Introduction

Les problèmes de planification existent dans de nombreux domaines, de l'assemblage d'avions à la planification de tâches de maintenance. La Programmation par Contraintes (PPC) a déjà été utilisée de manière fructueuse pour résoudre différents types de problèmes de planification. Le succès de cette technique provient entre autres de la combinaison des contraintes globales et des heuristiques efficaces dédiées à ce type de problèmes. Les problèmes de planification sont souvent NP-difficiles quand ils sont sujets à des contraintes de précédence et de ressources. Cependant, celles-ci peuvent également aider à améliorer les inférences faites par les contraintes globales. Les précédences peuvent être naturellement représentées par un graphe dirigé en liant deux tâches sujettes à précédence. Les réseaux de neurones en graphes (RNG) sont conçus pour apprendre de données structurées en graphes, permettant, entre autre, d'apprendre à prédire si une arête est présente ou pas. Notre papier vise à répondre aux questions suivantes : un RNG peut-il nous aider à identifier de nouvelles précédences, et si oui, à quel point peuvent-elles être utiles ?

2 Méthodologie

Pour apprendre des précédences présentes dans les solutions, nous avons décidé d'utiliser un RNG à trois couches *GraphSAGE* [1], suivi d'un perceptron multicouche de deux couches (réseau entièrement connecté).

En pratique, pour obtenir une prédiction une fois le modèle entrainé, l'utilisateur crée le graphe d'entrée. Celui-ci a comme arêtes toutes les précédences de l'instance ainsi que la fermeture transitive de celles-ci. De plus, les noeuds sont annotés d'un vecteur contenant les caractéristiques de la tâche : la durée et la consommation normalisées de chaque ressource. De ce graphe, on peut automatiquement déduire les arêtes candidates (celle qui ne font pas partie de la fermeture transitive, ni de l'inverse de celle-ci). Le graphe est ensuite entré dans le RNG afin de calculer pour chaque noeud les projections des caractéristiques. Pour chaque arête candidate, les projections des deux extrémités sont ensuite entrées dans un perceptron multicouche, dont la tâche est de prédire la présence ou non de cette arête. Les prédictions sur les candidats sont ensuite classées par prédictions décroissantes et sélectionnées une à une si elles

ne créent pas de cycle et que leur prédiction a sa valeur au-dessus d'un certain seuil de sélection.

L'utilisateur peut ensuite faire ce qu'il désire des arêtes prédites. Dans notre cas, nous avons testé deux stratégies. Premièrement, nous créons une restriction du problème en ajoutant des contraintes pour toutes précédences prédites. Notre hypothèse est que cela peut diminuer la taille de l'arbre de recherche à explorer, mais pour chaque précédence mal prédite, il y a un risque d'enlever la partie de l'arbre contenant la solution optimale. Deuxièmement, à partir du graphe de précédence étendu des précédences prédites, nous pouvons créer un ordre topologique des tâches, qui peut être utilisé comme guide à une heuristique de recherche (telle que la combinaison sbps/vsids). Notre hypothèse dans cette situation est que si cet ordre est proche de celui de la solution optimale, alors celle-ci sera parmi les premières à être découverte. De plus, avec une heuristique, nous sommes sûrs de ne pas perdre la solution optimale.

3 Expérimentations

Nous avons utilisé le dépôt PSPLib [2] pour nos expériences. Les instances ont 30, 60, 90 ou 120 tâches. Les instances PSPLib sont des instances générées, avec différents paramètres (une série de 10 instances par ensemble de paramètres). Pour nous assurer de ne pas causer de surapprentissage, nous avons mis de côté certaines séries entières, ainsi que certaines instances des séries restantes. Pour chaque instance servant à l'apprentissage, nous avons calculé la meilleure solution après 1h de temps limite pour deux modèles PPC afin de comparer l'effet de la qualité de l'ensemble d'entrainement sur la méthode. Le premier utilise les heuristiques sbps/vsids, l'autre utilise une simple heuristique de placement de la tâche le plus tôt. Nous avons réalisé une validation croisée en K plis sur les arêtes à apprendre pour également éviter un surapprentissage.

3.1 Impact de la qualité de l'ensemble d'entrainement

Pour commencer, nous avons comparé un modèle entrainé sur les solutions générées avec sbps/vsids (solutions plus proches de l'optimal) et un modèle sur les solutions générées sans (moins proches de l'optimal). Les métriques d'entrainement étaient légèrement meilleures pour le second modèle. Cependant, lorsque nous les utilisions, soit comme contrainte additionnelle, soit comme guide heuristique, les prédictions du premier modèle se rendaient plus utiles. Cette expérience a également confirmé certaines hypothèses. Lorsque les prédictions sont utilisées comme contraintes additionnelles, elles créent une restriction, qui dans certain cas (si les prédictions sont correctes) permet d'améliorer les solutions pour un même temps de calcul. Cependant, dans une majorité des cas, la restriction diminue la qualité des résultats, dû à des prédictions erronées. Si les prédictions sont utilisées pour guider l'heuristique, nous n'avons en effet pas de perte de l'optimal, et, dans une majorité des instances, nous avons une amélioration de l'optimum au temps d'arrêt pour les plus faibles temps d'arrêt, traduisant la découverte de meilleures solutions plus tôt dans la recherche.

3.2 Généralisation

La qualité de l'ensemble d'entrainement étant un facteur important sur la qualité des prédictions, idéalement il ne faudrait entrainer que sur des solutions optimales. Hors, plus il y a de tâches, plus il est difficile d'obtenir l'optimum en temps raisonnable. Nous avons donc mesuré la nécessité d'inclure des grosses instances dans l'ensemble d'entrainement. Pour ce faire, nous avons entrainé un modèle sur les instances de 60 tâches maximum et nous l'avons comparé aux performances du modèle entrainé sur les instances de 120 tâches ou moins. Les métriques des deux modèles étaient similaires.

3.3 Agrégation de solutions

Dans les problèmes de planification, il arrive souvent qu'il y ait plusieurs solutions optimales. Celles-ci diffèrent souvent légèrement l'une de l'autre (planning où des tâches moins contraintes et moins demandeuses en ressources ont un peu de liberté de placement). Pour encapsuler cette tendance, nous avons généré, pour chaque instance, 100 solutions avec le même optimum que le meilleur trouvé. L'agrégation de ces solutions correspond aux arêtes présentes dans au moins 70% de ces solutions. Notre but étant d'apprendre des précédences plus cruciales à l'optimalité. Les métriques d'apprentissage étaient similaires aux précédents modèles. Les métriques d'utilisation en étaient cependant améliorées, montrant une découverte de meilleures solutions plus tôt durant la recherche.

4 Conclusion

Ce papier présente une nouvelle approche basée sur les réseaux de neurones en graphes afin de prédire de nouvelles précédences pour le problème de gestion de projet à contraintes de ressources. Les précédences apprises peuvent soit être ajoutées comme contraintes additionnelles afin de réduire l'espace de recherche, soit comme heuristique afin de guider la recherche. Nos expériences sur les instances de PSPLIB nous confirment que dû à la nature difficile du problème, il est difficile d'obtenir un bon rappel au niveau des prédictions. Cependant, on observe quand même une accélération de la résolution. Une amélioration des solutions a été observée dans nos expériences mais reste difficile à atteindre. La qualité des prédictions dépend entre autres de la qualité de l'ensemble d'entrainement. Utiliser un agrégat de plusieurs solutions a permis un apprentissage de précédences plus cruciales aux solutions. Nos expériences montrent également une bonne généralisation, comme le montre la comparaison entre les modèles entrainés sur des instances de 60 tâches maximum et les modèles entrainés sur des instances de 120 tâches ou moins. Notre méthode est également agnostique du solveur et peut être combinée avec d'autres méta-heuristiques telles que la recherche à base de voisinage large.

Remerciements

Cette recherche a reçu du financement de la part de IVADO et des Fonds d'excellence en recherche Apogée Canada (PostDoc-2022-2378128196) et des fonds NSERC Discovery Grants 218028/2017, ainsi qu'un financement du programme Horizon 2020 de recherche et innovation de l'Union Européenne (No 101070149, projet Tuples).

Références

[1] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[2] Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library : Or software-orsep operations research software exchange program. *European journal of operational research*, 96(1) :205–216, 1997.

[3] Hélène Verhaeghe, Quentin Cappart, Gilles Pesant, and Claude-Guy Quimper. Learning precedences for scheduling problems with graph neural networks. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, pages 30–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.