Graph Neural Networks as Value Heuristic in Scheduling Problems

Tim Luchterhand, Emmanuel Hebrard, Sylvie Thiébaux

Laboratoire d'analyse et d'architecture des systèmes (LAAS) Toulouse, France
{tim.luchterhand, hebrard, sylvie.thiebaux}@laas.fr

Keywords : *scheduling, graph neural networks, heuristics, tree-search*

1 Introduction

Scheduling problems frequently occur in real-life applications such as manufacturing tasks or time tabling problems. The goal is to find a schedule with minimum execution time (the so called makespan) while respecting precedence constraints imposed by the user and / or resource constraints. Algorithms that solve these kind of problems are usually based on a tree search or more specifically a branch and bound search. These methods guarantee to find the optimal solution by intelligently exploring an exponentially large search space. Since in many real-world problems this search space becomes prohibitively large, the performance of such algorithms is heavily dependent on a good search strategy that dictates which parts of the search space to explore first. This is where heuristics come in.

Graph neural networks (GNNs) are a special type of neural network that operate on graph structured data. In their most general form they take a graph annotated with arbitrary features, e.g. real valued feature vectors on the nodes and edges, and transform those features through learnable functions and a message passing scheme [1]. Thus, they can be used to predict scores for nodes in the graph or relationships between them. In particular, GNNs seem perfectly adapted to make predictions about combinatorial optimization problems which often exhibit a graph structure. Recently there have been multiple publications on GNNs used in conjunction with classical solvers for discrete optimization problems like SAT or (mixed) integer linear programming (MILP) [3, 4]. We propose to extend these ideas and to apply them to scheduling problems like the Job-Shop or resource constrained project scheduling problems (RCPSP). Specifically, we seek to implement a GNN based value branching heuristic for a SAT based scheduler, i.e. a heuristic that takes binary branching decisions during the search and seeks to quickly guide the solver to good solutions.

2 General Framework and GNN Architecture

We represent an RCPSP as a graph $G = (V, E, \mathbf{V}, \mathbf{E}, \mathbf{U})$ consisting of task nodes V with corresponding features $\mathbf{t} \in \mathbf{V}$, edges E with features $\mathbf{e} \in \mathbf{E}$ and additionally resources $r \in R$ annotated with features $\mathbf{u} \in \mathbf{U}$. The graph G then consists of multiple fully connected sub-graphs, one for each resource r. These sub-graphs are connected to each other if there exist tasks that consume more than one resource. An example graph is given in fig. 1. The GNN then transforms a graph G to a graph $G' = (V, E, \mathbf{V}', \mathbf{E}', \mathbf{U}')$, i.e. the topology of G remains the same while the features are transformed via a message passing scheme inspired by [1]. The input features of G describe the problem instance. For example, task features could include the minimum and maximum execution time of a task, edge features could contain precedence information specified by the user and finally, resource features could represent the capacity of a resource. But in general any type of real valued feature can be added to the feature vectors.

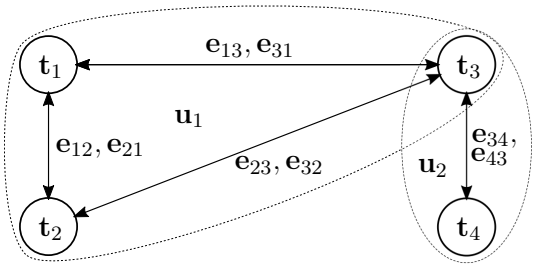

FIG. 1: Representation of an RCPSP as graph. Tasks that share a resource form a fully connected sub-graph. Tasks, edges and resource "bubbles" are annotated with task, edge and resource features respectively.

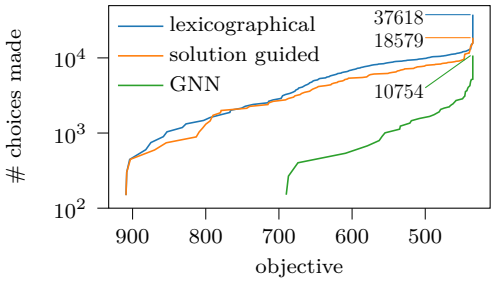

FIG. 2: Heuristic performance on an OpenShop problem instance. The solver needs to make significantly less choices to arrive at the optimal solution when using the GNN based heuristic.

The features of the output graph G' on the other hand can then be used to derive the actual heuristic. In our case, the solver needs to take binary decisions for pairs of edges between two tasks t_1 and t_2, i.e. whether t_1 should be scheduled before t_2 or vice-versa. Consequently, we are currently predicting a Bernoulli distribution for edge pairs between tasks t_1 and t_2 that indicates which ordering is more likely. But the flexibility of the proposed GNN architecture allows for all kinds of outputs. For example, the GNN could also predict a scalar value for each task node that is then used to directly construct an ordering.

Training of the GNN is done in a supervised manner. We solve a number of scheduling problems using the solver and then extract the edge directions (i.e. the labels for the training) from the optimal solution. Additionally, we also include critical intermediate states of the search, i.e. subproblems encountered during the search, in the dataset. The label generation for the latter is more involved since the best solution corresponding to a given subproblem is not necessarily the global optimum. During evaluation, the solver can then query the GNN by constructing the graph G from the current state of the search and then extract the edge probabilities from the output.

3 Perspectives

First results on Job-Shop problems seem promising: the solver using the GNN expands significantly fewer nodes during search than solution guided search [2] (see fig. 2). However, this is only the first step as a couple of open points remain. The inference is currently too slow for the GNN to be queried at every decision of the solver. Additionally, we want to experiment with different input features, different input graph representations as well as different GNN outputs and thus heuristic types.

References

[1] Peter W. Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. In: (June 2018). arXiv: 1806.01261 [cs.LG].

[2] J. C. Beck. “Solution-Guided Multi-Point Constructive Search for Job Shop Scheduling”. In: *Journal of Artificial Intelligence Research* 29 (May 2007), pp. 49–77. ISSN: 1076-9757.

[3] Maxime Gasse et al. “Exact Combinatorial Optimization with Graph Convolutional Neural Networks”. In: (June 2019). arXiv: 1906.01629 [cs.LG].

[4] Elias B. Khalil, Christopher Morris, and Andrea Lodi. “MIP-GNN: A Data-Driven Framework for Guiding Combinatorial Solvers”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.9 (June 2022), pp. 10219–10227. DOI: 10.1609/aaai.v36i9.21262.