Data-driven Policy on Feasibility Determination for the Train Shunting Problem (Extended Abstract) *

Paulo Roberto de Oliveira da Costa[1], Jason Rhuggenaath[1] (✉), Yingqian Zhang[1], Alp Akcay[1], Wan-Jui Lee[2], and Uzay Kaymak[1]

[1] Eindhoven University of Technology
[2] Dutch Railways

Introduction. Parking, matching, scheduling, and routing are common problems in train maintenance. In particular, train units are commonly maintained and cleaned at dedicated shunting yards. The planning problem that results from such situations is referred to as the Train Unit Shunting Problem (TUSP). This problem involves matching arriving train units to service tasks and determining the schedule for departing trains. The TUSP is an important problem as it is used to determine the capacity of shunting yards and arises as a sub-problem of more general scheduling and planning problems. In this paper, we consider the case of the Dutch Railways (NS) TUSP. As the TUSP is complex, NS currently uses a local search (LS) heuristic to determine if an instance of the TUSP has a feasible solution. Given the number of shunting yards and the size of the planning problems, improving the evaluation speed of the LS brings significant computational gain. In this work, we use a machine learning approach that complements the LS and accelerates the search process. We use a Deep Graph Convolutional Neural Network (DGCNN) model to predict the feasibility of solutions obtained during the run of the LS heuristic. We use this model to decide whether to continue or abort the search process. In this way, the computation time is used more efficiently as it is spent on instances that are more likely to be feasible. Using simulations based on real-life instances of TUSP, we show how our approach improves upon the previous method on prediction accuracy and leads to computational gains for the decision-making process.

Methodology. Our methodology consists of three main steps: (i) generation of starting solutions; (ii) a prediction model for feasibility of solutions and (iii) a policy for interation with the LS. A schematic view of our approach can be found in [1]. Each intermediate solution is represented as an activity graph of maintenance activities. We then use a Deep Graph Convolutional Neural Network (DGCNN) [2] as a feature extractor to train a model that predicts the future feasibility of each solution. Afterwards, we combine the predictions with the LS to derive a policy that decides to terminate the LS run based on the sequence of intermediate solutions seen so far.

Main results and insights. We evaluate the classification performance of the graph classification models (Table 1). We consider three types of prediction models: (i) DGCNN-IS, (ii) DGCNN-IS-T and (iii) DGCNN-MS-T. DGCNN-IS

uses only node labels and initial solutions during training. DGCNN-IS-T is the same as DGCNN-IS but with additional temporal features. DGCNN-MS-T uses additional node temporal features and the first 10% of the graphs in each instance of a LS run for training. All models consider as class labels the final feasibility status at the end of an LS run. We perform inference on the initial solutions on the test dataset.

Method	DGCNN-IS	DGCNN-IS-T	DGCNN-MS-T
ACC(%)	54.10 ± 2.37	**60.01 ± 1.67**	59.96 ± 1.68
TPR(%)	52.93 ± 11.47	58.25 ± 9.10	**82.08 ± 2.53**
TNR(%)	55.28 ± 10.27	**60.38 ± 7.37**	37.86 ± 4.99

Table 1. Different models. Adding time features improves performance by 10%.

The main insight is that temporal features add important information for the classification of feasible instances. These results motivate us to consider a variant DGCNN-MS-T-W of DGCNN-MS-T that can be used alongside the local search procedure. DGCNN-MS-T-W is similar to DGCNN-MS-T except that it predicts feasibility of solutions that are W iterations ahead. Given an instance of TUSP, we use the following threshold policy (see [1] for more details):

1. Start the LS with the initial solution and run it for K iterations.
2. For each iteration i where $0 \leq i \leq K$, we pass the current solution G_i to DGCNN-MS-T-W to get a feasibility score $\phi(G_i)$.
3. If the average of the feasibility scores $\phi(G_i)$ seen up to iteration K falls below some threshold $0 < \alpha_{IF} < 1$, we stop the LS and classify the instance as infeasible.

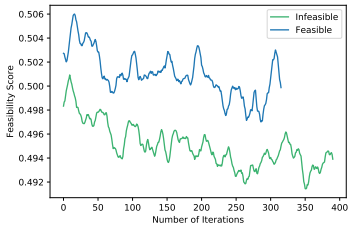

Fig. 1. Moving average over the last 10 iterations of the feasibility scores $\phi(G_i)$ averaged over cross-validation runs of the LS procedure

		Prediction Outcome		
		0: infeasible	**1: feasible**	**Correct**
				Incorrect
Actual Value	**0**	543 33.9%	250 15.6%	68.4% 31.6%
	1	284 17.7%	523 32.7%	64.8% 35.2%
Correct		65.6%	67.6%	**66.6%**
Incorrect		34.4%	32.4%	33.4%

Table 2. Confusion matrix of one of the folds for the final classification of DGCNN-MS-T-W, with $W = 150$, $K = 200$, $\alpha_{IF} = 0.5$.

Our results (see also Fig. 1 and Table 2) indicate that we can achieve a 8% reduction in running time for a single instance, which can account for roughly 20 hours in total real time.

References

1. de O. da Costa, P.R., Rhuggenaath, J., Zhang, Y., Akcay, A., Lee, W., Kaymak, U.: Data-driven policy on feasibility determination for the train shunting problem. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD (2019), to appear
2. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: AAAI-18. pp. 4438–4445 (2018)