Accelerating Linear Programming Solving by Exploiting the Performance Variability via Reinforcement Learning

Xijun Li1,2, Qingyu Qu3, Fangzhou Zhu2, Mingxuan Yuan2, Jia Zeng2, and Jie Wang1

1 University of Science and Technology of China

2 Huawei Noah's Ark Lab

3 Beihang University

AAAI
Association for the Advancement of Artificial Intelligence

ABSTRACT

The trend of using machine learning techniques to improve the mathematical programming solvers has recently drawn lots of attention. Most previous work focuses on replacing key components within the solvers using machine learning techniques. Empirically, practitioners observed that the solving efficiency of the solver is highly sensitive to the formulation of inputted mathematical programming models, such as the appearing order of variables (one of performance variability of solvers). In other words, an inappropriate formulation might harm the performance robustness of the solver. Instead, we exploit this type of performance variability, proposing a novel approach for accelerating linear programming solving via reinforcement learning-based reformulation. We implemented the proposed approach with three reputable solvers, i.e., Gurobi, SCIP, and CLP. We conducted extensive experiments over two public LP datasets from NeurIPS 2021 ML4CO competition and one large-scale LP dataset collected from a real-life production planning scenario. Experimental results demonstrate that the proposed approach effectively reduces the solving iteration number (20% on average) and solving time (15% on average) over the above datasets, compared to directly solving the original linear programming models. This work can inspire the future research for better exploiting the performance variability of solvers with machine learning techniques.

PROBLEM DESCRIPTION AND MOTIVATION

An easily overlooked phenomenon: the appearing order of variables in a given LP instance indeed affects the solver's performance.

Motivation: using machine learning technique to infer better formulation for solvers.

Three challenges to introduce ML techniques:

① How to appropriately represent an LP instance
② What machine learning model is suitable
③ How to efficiently train above inferring model

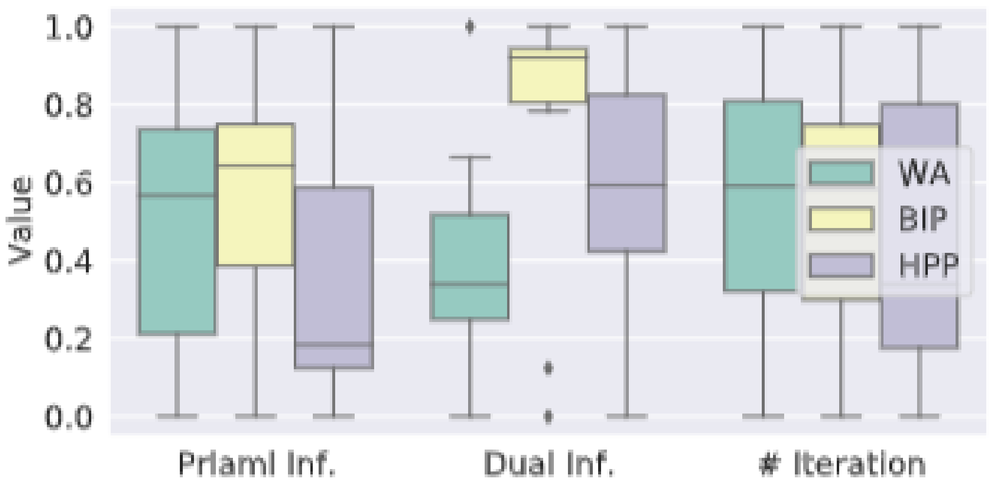
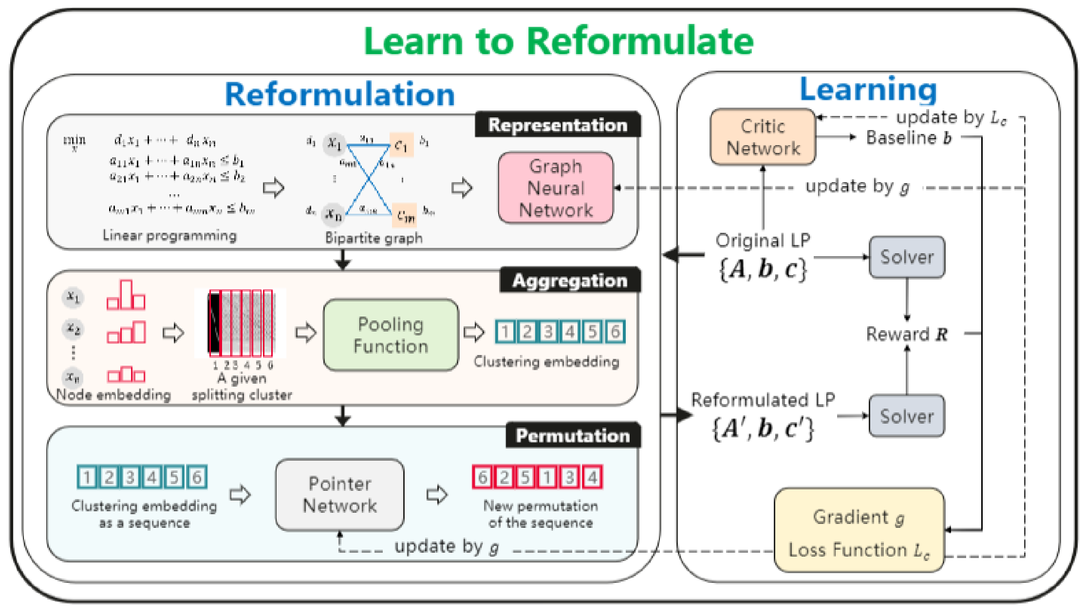


Figure: Three distinct LP instances (WA, BIP, and HPP) are selected to perform the preliminary experiment. For each original LP instance, we randomly changed the appearing order of variables to get many reformulated but mathematical identical LP instances. Next, we called Gurobi to solve the above LP instances. We recorded the primal infeasibility (`Primal Inf.'), dual infeasibility (`Dual Inf.') of the first iteration of Gurobi solving process, and the total iteration number (`# Iteration') to solve the instances. The significant variance of metrics shows that solver performance is quite sensitive to the different formulations for a given LP instance.

PROPOSED SOLUTION



Reformulation

a) **Representation:** The inputting LP instance is represented by a bipartite graph, and then the embedding of variables is obtained via a GNN;
b) **Aggregation:** The embedding of variables will be further aggregated with a given group of variable clusters
c) **Permutation:** Taking as input the previous embeddings, a pointer network (PN) is used to output a new permutation of variables

Learning

Update the policy parameter: the learning part interacts with the reformulation part to update the parameters of GNN and PN, trained with REINFORCE algorithm.

① **Reward function** $R(\pi|l) = 1 - \frac{\mathcal{S}_{\mathcal{M}}(l|\pi)}{\mathcal{S}_{\mathcal{M}}(l)}$

② **Loss function**

$$\mathcal{L}(\theta_c) = \mathbb{E}_{l\sim\mathcal{S},\pi\sim p_{\theta_G},\theta_P(.|l)}[b_{\theta_c}(l) - R(\pi|l)]^2$$

EVALUATION AND CONCLUSION

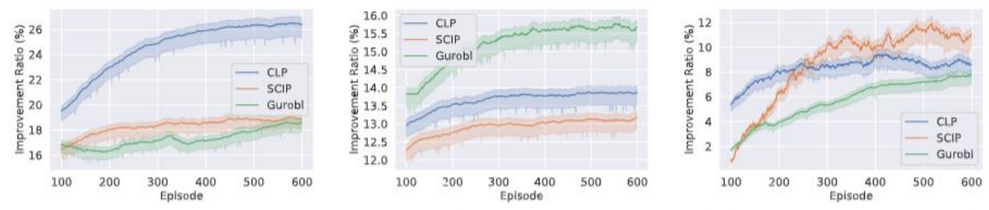


Figure: Learning convergence and improvement of the iteration number over three datasets' training (seen) instances.

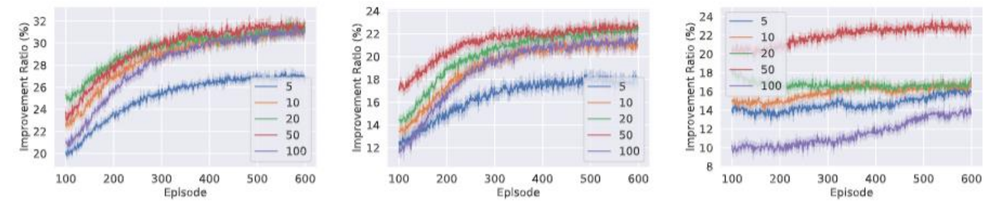


Figure: How different number of clustering block impacts our method's performance

Conclusion:

- The first work that exploits the performance variability of modern solvers via ML techniques to gain performance
- This work can inspire the future research to better exploit the performance variability of solver, such as pricing, variable selection, cut selection, etc.