Light-MILPopt: Solving Large-scale Mixed Integer Linear Programs with Lightweight Optimizer and Small-scale Training Dataset

Huigen Ye, Hua Xu*, Hongyan Wang

State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

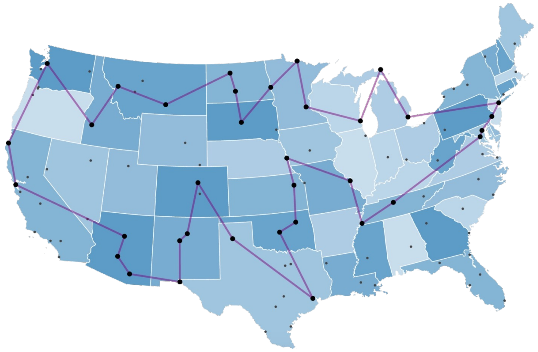
Contents

- Introduction
- Method
- Experiments
- Conclusion

Background

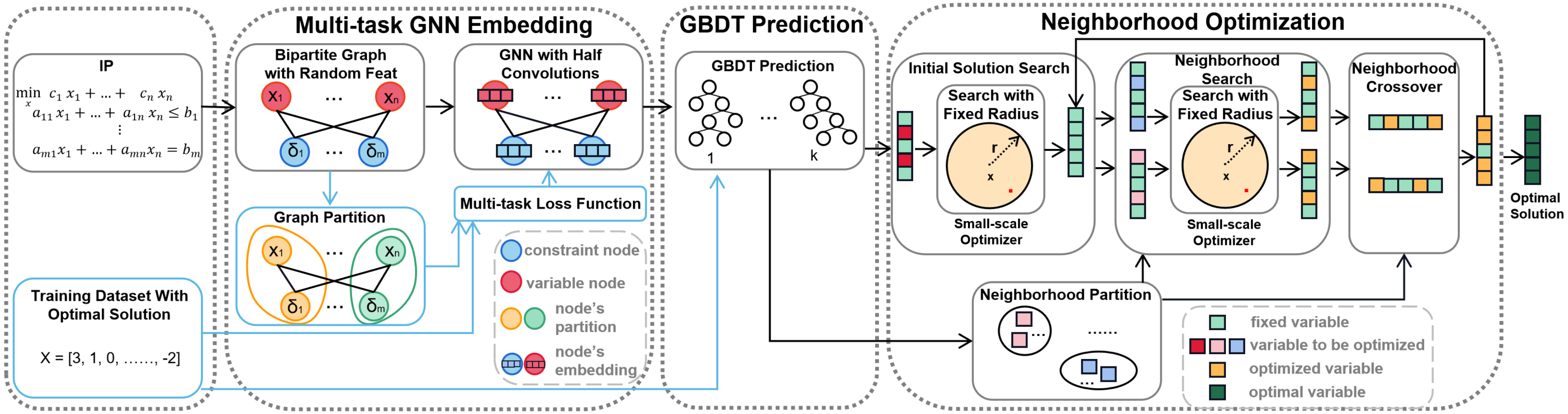
- Many real-world optimization problems in the real world can be abstracted as **mixed integer linear programming problems （MILPs)**
  - **Routing**[1]
  - **Scheduling**[2]
  - **Timetabling**[3]
- Formally, the **MILPs** can be defined as follows

$$
\begin{aligned}
&\min_{x} c^{T}x, \\
&\text{subject to } Ax \leq b,\, l \leq x \leq u,\, x_i \in \mathbb{Z},\, i \in \mathbb{I},
\end{aligned}
\tag{1}
$$



Challenge

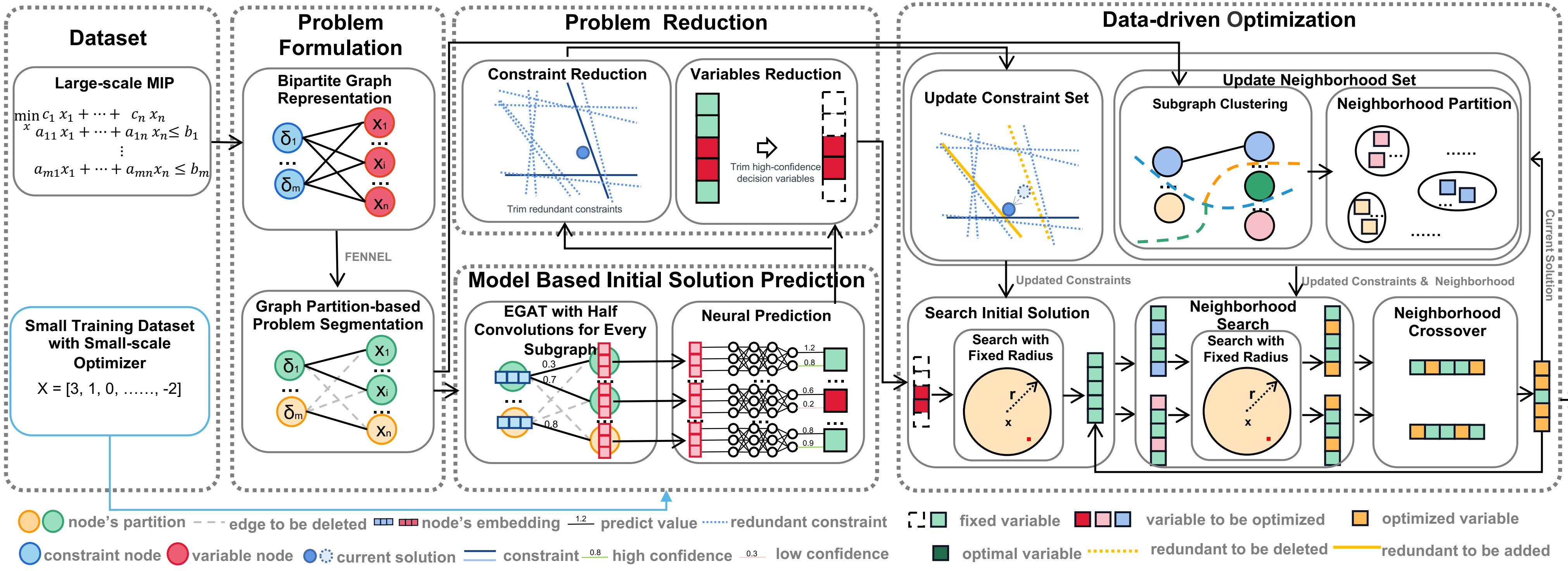
- **GNN&GBDT-guided optimizing framework（SOTA）**：
  - Firstly, representing MILPs as an entire graph poses **challenges in terms of model training and computational resources**, particularly when tackling large-scale MILPs
  - Secondly, GNN requires large-scale MILP instances of similar size as training data, leading to **significant computational and storage resource demands** during the training phase
  - Thirdly, the problem reduction is exclusively applied at the decision variable level, neglecting potential synergy with constraint reduction, thereby **resulting in limited effectiveness in problem reduction**



Contents

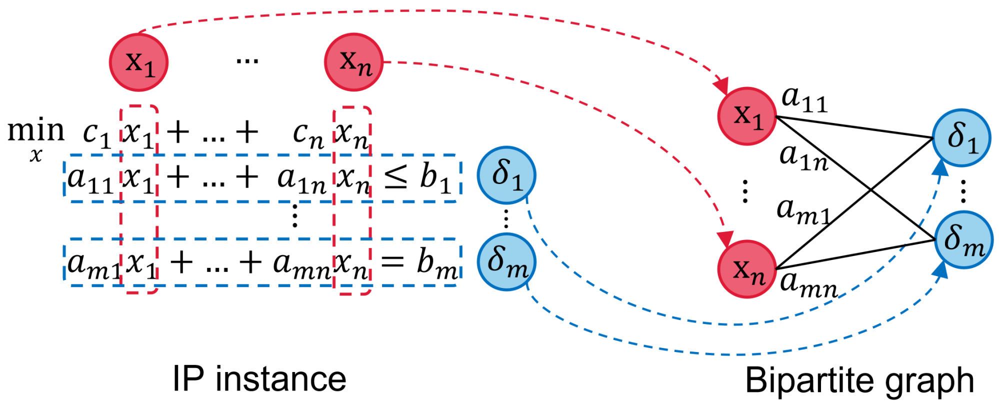
Overview

- Light-MILPopt is divided into four stages: **Problem Formulation, Model-based Initial Solution Prediction, Problem Reduction, Data-driven Optimization.** The proposed lightweight optimization framework can solve large-scale MILPs with only small-scale optimizer and small training dataset.

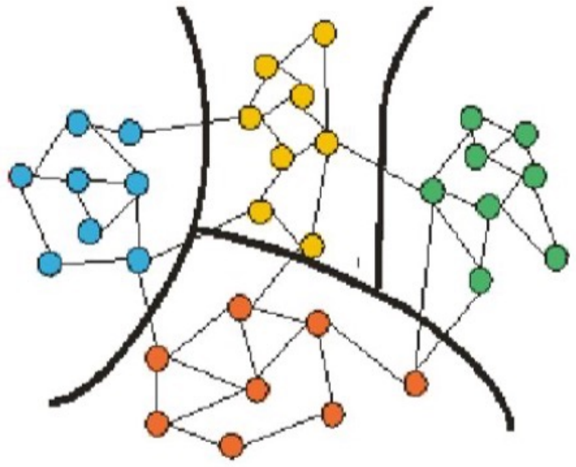


Problem Formulation

- **Problem Formulation**: problem division to reduce model computational costs
  - Initially, the MILP to be solved is represented as a **bipartite graph**
  - Then the **FENNEL graph partition algorithm** combined with the idea of Graph-Bert subgraph partition is used for problem division to reduce computational cost
  - Based on the above steps, all the subgraphs obtained from the graph partition form the inputs for feature-embedding neural networks



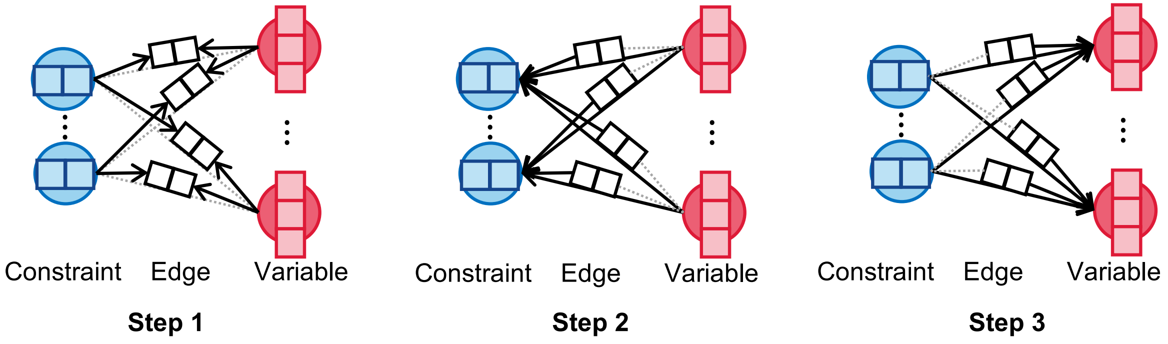
Bipartite graph representation



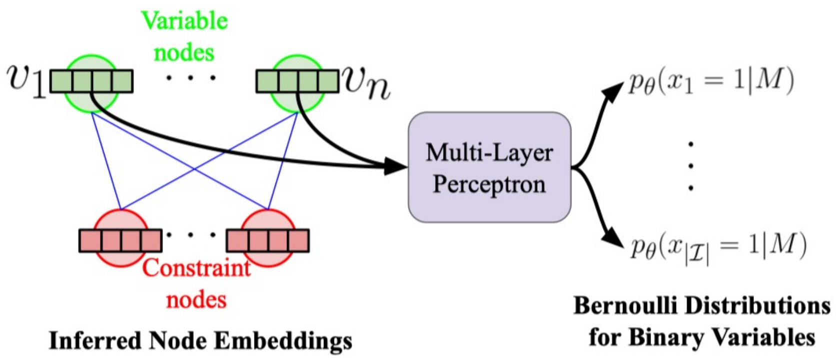
FENNEL graph partition algorithm

Model-based Initial Solution Prediction

- **Model-based Initial Solution Prediction:** predicting and constructing the initial solution using a small-scale training dataset
  - Given the graph representation with multiple small-scale subgraphs for the large-scale MILP, **EGAT with Half-convolutions** learns the neural embedding for the decision variables
  - Then the **Neural Prediction** network with Multi-Layer Perceptron (MLP) structure predicts the initial value of the corresponding decision variable in the subgraph through the neural embedding
  - Finally, the predicted initial solution will guide the subsequent problem reduction



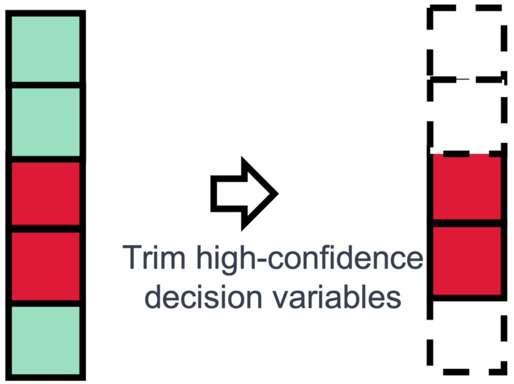
EGAT with Half-convolutions



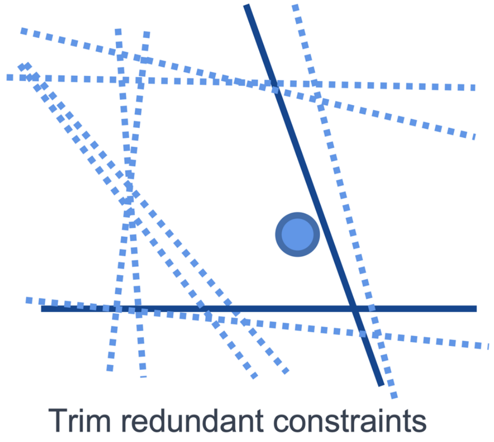
Neural Prediction

Problem Reduction

- **Problem Reduction**: both variable and constraint reduction
  - Given the predicted initial solution of the MILP, the generalized confidence threshold method adaptively fixes the high-confidence decision variable to achieve **Variables Reduction**
  - Then, KNN strategy is used for **Constraint Reduction** to identify active constraints.
  - Finally, the reduction of decision variables and constraints can jointly guide the initial solution search and iterative optimization



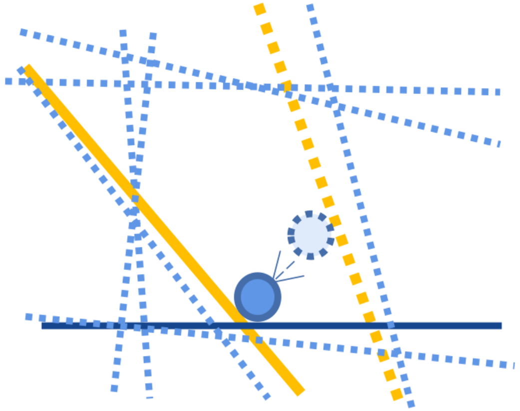
Variables Reduction



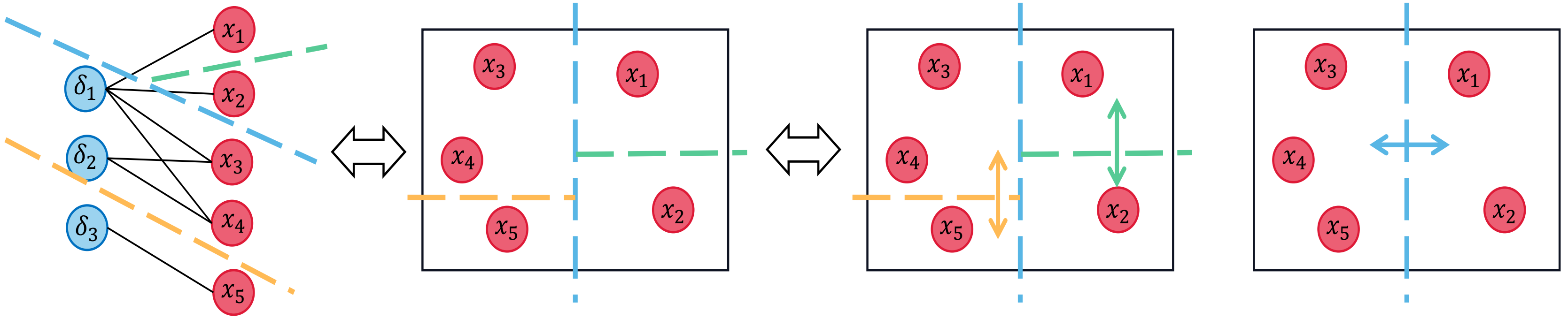
Constraint Reduction

Data-driven Optimization

- **Data-driven Optimization:** current solution improvement employing a lightweight optimizer
  - Based on the predicted initial solution and the problem reduction, we first solve the reduced subproblem to obtain the **Initial Solution** for the complete MILP using lightweight optimizer
  - Then, under the guidance of Neighborhood Updating and the active **Constraint Set Updating**, **neighborhood search and individual crossover** iteratively improve the current solution
  - Finally, when a predetermined wall-clock time or condition is reached, the current solution is output as the final optimization result



Constraint Set Updating



Neighborhood search and individual crossover

Contents

Settings

- Dataset
  - Four widely used NP-hard benchmark MILPs: Set Covering (SC, Minimize), Minimum Vertex Covering (MVC, Minimize), Maximum Independent Set (MIS, Maximize), Mixed Integer Knapsack Set (MIKS, Maximize)
  - One real-world large-scale MILP in the internet domain (Case Study, Maximize)

Problem	Scale	Number of Variables	Number of Constraints
SC (Minimize)	SC_1	200000	200000
	SC_2	2000000	2000000
MVC (Minimize)	MVC_1	100000	300000
	MVC_2	1000000	3000000
MIS (Maximize)	MIS_1	100000	300000
	MIS_2	1000000	3000000
MIKS (Maximize)	$MIKS_1$	200000	200000
	$MIKS_2$	2000000	2000000
Case Study (Maximize)	Case Study	2040000	100003

Comparison of Objective Value

- Compared to the large-scale solvers SCIP and Gurobi, Light-MILPopt obviously outperforms them only using a scale-limited version solver with variable proportion $\alpha = 30\%$
- The proposed framework achieves better results than the GNN&GBDT frameworks in integer programs with the same scale of variable reduction, efficiently solving large-scale MILPs, which cannot be solved by the GNN&GBDT framework

	SC_1	SC_2	MVC_1	MVC_2	MIS_1	MIS_2	$MIKS_1$	$MIKS_2$	Case Study
Ours-30%S	17121.5↑	166756.0↑	27337.8↑	273014.6↑	22621.7↑	227074.5↑	35067.8↑	355887.6↑	944086.4↑
Ours-30%G	**17047.3↑**	**163975.9↑**	**27223.3↑**	**272579.5↑**	**22658.0↑**	**227305.4↑**	**35533.4↑**	**357439.5↑**	**979797.8↑**
GBDT-30%S	17222.2	261174.0	27515.4	276306.9	22389.3	223349.8	-	-	-
GBDT30%G	18487.6	281021.2	27700.8	281234.5	22115.9	210019.2	-	-	-
Ours-50%S	16147.2↑	166966.9↑	26956.8↑	269771.3↑	22963.6↑	230278.1↑	**36125.5↑**	357483.8↑	944166.1↑
Ours-50%G	**16108.1↑**	**160015.5↑**	**26950.7↑**	**269571.5↑**	**22966.5↑**	**230432.9↑**	36108.2↑	**362265.1↑**	**980688.0↑**
GBDT50%S	16728.8	268294.9	27107.9	271777.2	22795.7	227006.4	-	-	-
GBDT50%G	17503.4	252797.2	27329.9	274600.8	22530.1	215393.6	-	-	-
SCIP	25191.2	385708.4	31275.4	491042.9	18649.6	9104.3	29974.7	168289.9	924954.5
Gurobi	17934.5	320240.4	28151.3	283555.8	21789.0	216591.3	32960.0	329642.4	-
Time	2000s	12000s	2000s	8000s	2000s	8000s	2000s	6000s	1000s

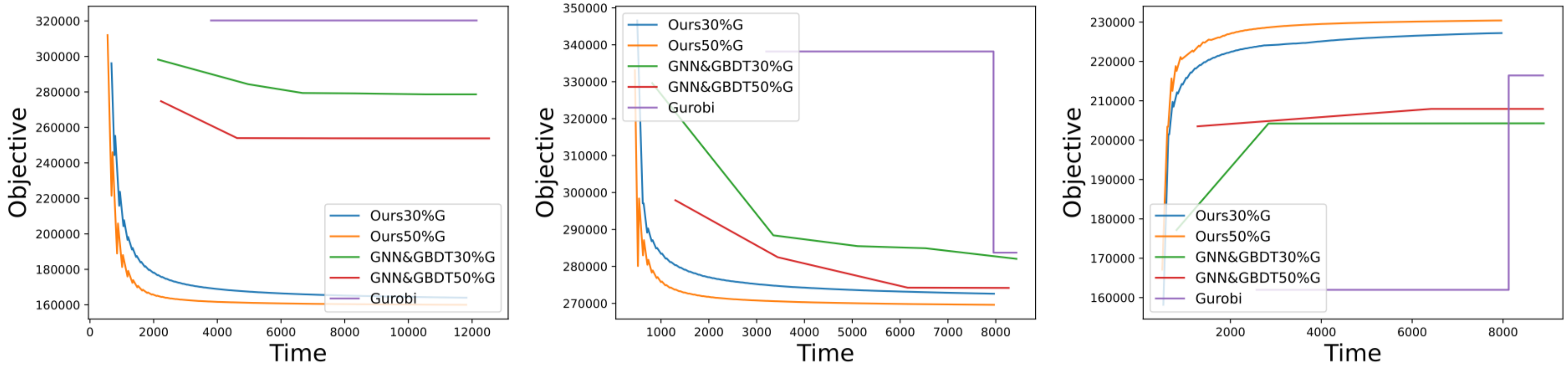
Comparison of Running Time

- Compared to the large-scale baseline solvers, the proposed framework can **achieve the same results in only 0.5% of the time** for the benchmark MILPs, including SC_1, MVC_1, MIS_1 and $MIKS_1$
- Even compared to the state-of-the-art ML-based frameworks, our Light-MILPopt **can save more than 90% of the time** on most MILPs to achieve the same results

	SC_1	SC_2	MVC_1	MVC_2	MIS_1	MIS_2	$MIKS_1$	$MIKS_2$	Case Study
Ours-30%S	1998.1s↑	11823.0s↑	1951.6s↑	7967.2s↑	1951.6s↑	7967.2s↑	1982.0s↑	11980.4s↑	996.4s↑
Ours-30%G	**1166.8s↑**	**5645.0s↑**	**1475.3s↑**	**6453.3s↑**	**1487.3s↑**	**7250.5s↑**	**593.9s↑**	**7941.9s↑**	**511.5s↑**
GBDT-30%S	>48369.2s	>60000s	>60000s	>60000s	>60000s	>60000s	-	-	-
GBDT30%G	>30347.8s	>60000s	>60000s	>60000s	>60000s	>60000s	-	-	-
Ours-50%S	352.2s↑	11441.3s↑	203.1s↑	1815.3s↑	225.9s↑	**1945.7s↑**	194.9s↑	9576.1s↑	776.2s↑
Ours-50%G	**177.8s↑**	**1795.4s↑**	**193.8s↑**	**1503.3s↑**	**223.5s↑**	2062.7s↑	**160.5s↑**	**2137.8s↑**	**506.9s↑**
GBDT50%S	587.6s	>60000s	297.6s	7570.5s	348.6s	5920.7s	-	-	-
GBDT50%G	5041.6s	>60000s	29320.5s	21397.3s	4227.1s	27952.9s	-	-	-
SCIP	>60000s	>60000s	>60000s	>60000s	>60000s	>60000s	>60000s	>60000s	3097.0s
Gurobi	>60000s	>60000s	>60000s	>60000s	>60000s	>60000s	45599.4s	>60000s	2584.7s
Target	17121.5	166756.0	27337.8	273014.6	22621.7	227074.5	35067.8	355887.6	944086.4

Convergence Performance Analysis

- Convergence is an essential metric for evaluating the performance of optimization frameworks. It can be seen that the proposed framework can obtain high-quality solutions for large-scale MILPs with **only small-scale training data and a small-scale optimizer**, and the convergence performance of Light-MILPopt is **not weaker than that of the state-of-the-art solver Gurobi as well as the state-of-the-art ML-based optimization framework**



(a) The minimized SC problem. (b) The minimized MVC problem. (c) The minimized MIS problem.

Contents

Conclusion

- Future work
  - Ultra-large-scale
  - Multi-objective
  - Nonlinear constraint

References

1. RV Kulkarni and Pramod R Bhave. “Integer Programming Formulations of Vehicle Routing Problems”. In: European Journal of Operational Research 20.1 (1985), pp. 58–67.
2. Armin Fьgenschuh. “Solving a School Bus Scheduling Problem with Integer Programming”. In: European Journal of Operational Research 193.3 (2009), pp. 867–884.
3. Armann Ingolfsson et al. “Combining Integer Programming and the Randomization Method to Schedule Employees”. In: European Journal of Operational Research 202.1 (2010), pp. 153–163.
4. Jiayi Zhang et al. “A Survey for Solving Mixed Integer Programming via Machine Learning”. In: Neurocomputing 519 (2023), pp. 205–217.
5. Michel Bйnichou et al. “Experiments in Mixed-integer Linear Programming”. In: Mathematical Programming 1 (1971), pp. 76–94.
6. David Applegate et al. Finding Cuts in the TSP (A Preliminary Report). Citeseer, 1995.
7. Tobias Achterberg. “SCIP: Solving Constraint Integer Programs”. In: Mathematical Programming Computation 1.1 (2009), pp. 1–41.
8. Matteo Fischetti, Fred Glover and Andrea Lodi. “The Feasibility Pump”. In: Mathematical Programming 104 (2005), pp. 91–104.
9. Edward Rothberg. “An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions”. In: INFORMS Journal on Computing 19.4 (2007), pp. 534–541.
10. David Pisinger and Stefan Ropke. “Large Neighborhood Search”. In: Proceedings of Handbook of metaheuristics. 2010.
11. Nicolas Sonnerat et al. “Learning a Large Neighborhood Search Algorithm for Mixed Integer Programs”. In: arXiv preprint arXiv:2107.10201 (2021).
12. Jerome H Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: Annals of Statistics (2001), pp. 1189–1232.

Thanks!