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

Xijun Li 1,2, Qingyu Qu 2, Fangzhou Zhu 2, Mingxuan Yuan 2, Jia Zeng 2, and Jie Wang 1

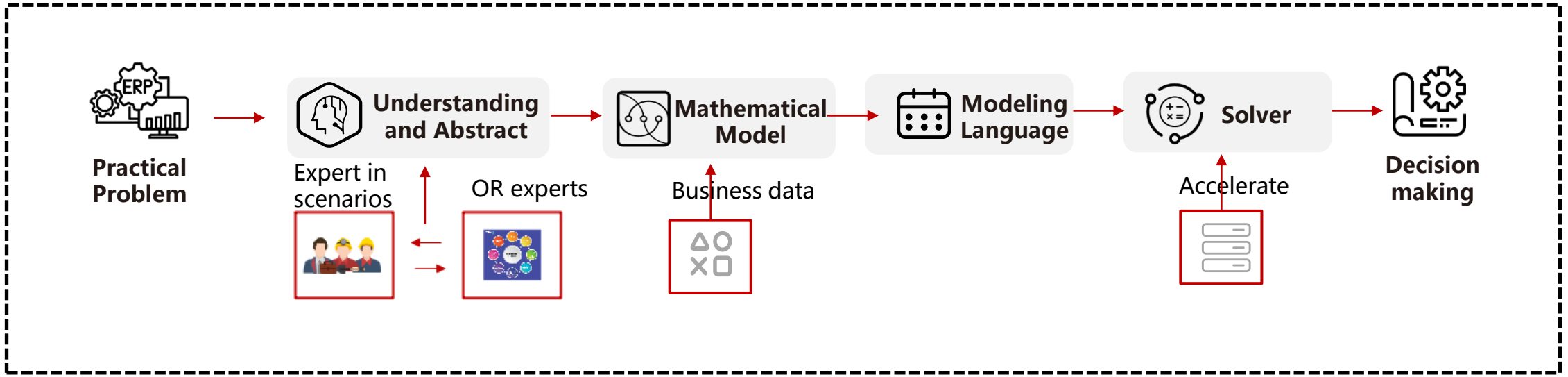
1 University of Science and Technology of China

2 Huawei Noah’ s Ark Lab

Outline

- **Preliminary**
- **Motivation: Who Cares and Why We Cares**
- **The Proposed Method**
- **Experiment and conclusion**

Preliminary



The procedure of calling solver to solve mathematical optimization problem

Preliminary

Mixed-Integer Linear Program

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

- $\mathbf{c} \in \mathbb{R}^n$ the objective coefficients
- $\mathbf{A} \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $\mathbf{b} \in \mathbb{R}^m$ the constraint right-hand-sides
- $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ the lower and upper variable bounds
- $p \leq n$ integer variables

Linear Program (LP) relaxation

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Convex problem, efficient algorithms (e.g., simplex).

- $\mathbf{x}^\star \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ (lucky) $\rightarrow$ solution to the original MILP
- $\mathbf{x}^\star \notin \mathbb{Z}^p \times \mathbb{R}^{n-p} \rightarrow$ lower bound to the original MILP

Branch and Bound

Stopping criterion:

- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

Split the LP recursively over a non-integral variable, i.e. $\exists i \leq p \mid x_i^\star \notin \mathbb{Z}$

$$x_i \leq \lfloor x_i^\star \rfloor \quad \vee \quad x_i \geq \lceil x_i^\star \rceil.$$

Lower bound (L): minimal among leaf nodes.
Upper bound (U): minimal among integral leaf nodes.

Order matters: who cares and why we care

Prof. Andrea Lodi [1]

1. "The performance of MIP solvers is subject to some unexpected variability that appears, for example, when changing from one computing platform to another, when permuting rows and/or columns of a model, etc."
2. "This phenomenon has certainly been observed for decades and has been the topic of an extensive literature in the artificial intelligence and SATisfiability communities."
3. "One source of performance variability is rooted in the so-called imperfect tie breaking. Most of the decisions taken by an MIP solver are based on ordering candidates according to scores and selecting the candidate with the best score. This is true for cut separation and cut filtering as well as for one of the most crucial decisions of the MIP solution process, namely, the variable to branch on at each node."

Dr. Ivet Galabova [2], Developer of HiGHS

For some problems the presolve performance is very sensitive to the order in which the rules are applied. Hence, an analysis of presolve would be incomplete without an investigation of this effect for particular LP instances. Alternative orderings may be more suitable for a problem, depending on the problem structure.

[1] Lodi, Andrea, and Andrea Tramontani. "Performance variability in mixed-integer programming." Theory driven by influential applications. INFORMS, 2013. 1-12.
[2] Galabova, Ivet. "Presolve, crash and software engineering for HiGHS." (2023).

Motivation

An easily overlooked phenomenon: the appearing order of variables in a given LP instance indeed affects the solver's performance. The solver takes input the model constructed by human experts.

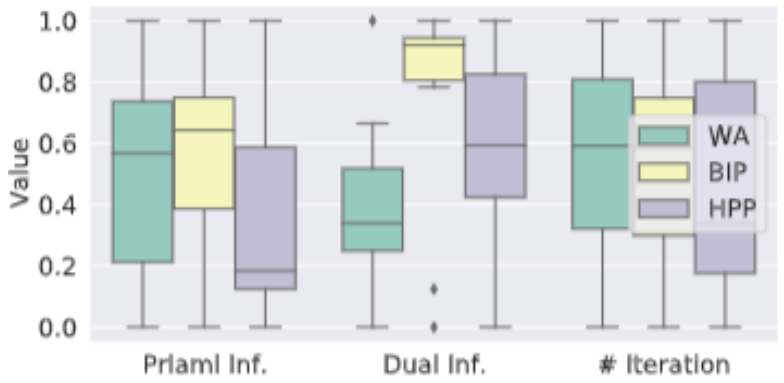


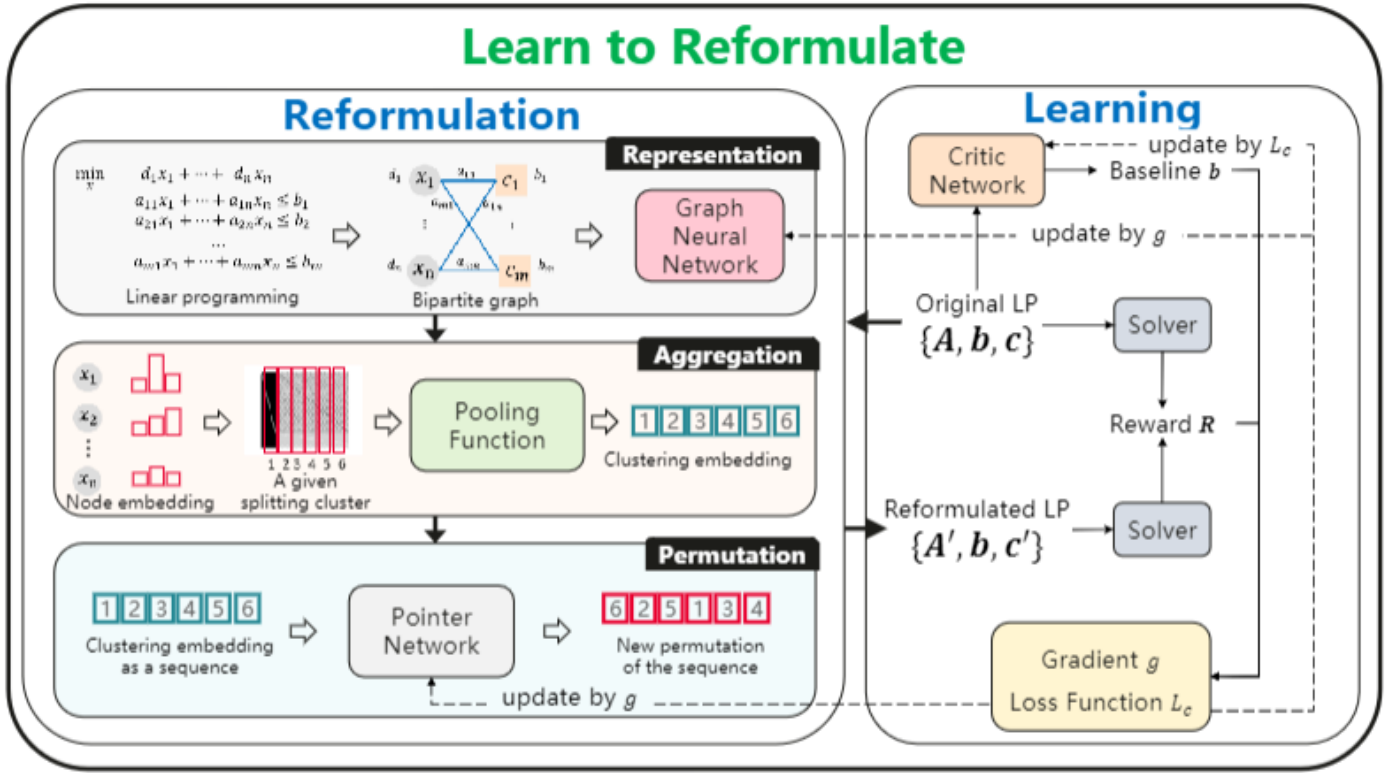
Figure: Three distinct LP instances are selected to perform the preliminary experiment. The significant variance of metrics shows that solver performance is quite sensitive to the different formulations for a given LP instance.

Idx	HPP (m=146722, n=260636, nnz=668270)			WA (m= 64282, n=61000, nnz=359428)			BIP (m=195, n=1083, nnz=7440)		
	Primal Inf.	Dual Inf.	# Iteration	Primal Inf.	Dual Inf.	# Iteration	Primal Inf.	Dual Inf.	# Iteration
1	7.721295E+09	1.884180E+11	19408.00	311.22	1.9980E+09	14083.00	9.43	1.922493E+07	539.00
2	7.720309E+09	1.882416E+11	19242.00	295.33	1.9790E+09	17020.00	9.58	1.937180E+07	486.00
3	7.725508E+09	1.869514E+11	19208.00	269.29	1.9530E+09	16495.00	9.38	1.916960E+07	557.00
4	7.764919E+09	1.876644E+11	19408.00	262.35	2.0620E+09	15149.00	9.47	1.932945E+07	453.00
5	7.726977E+09	1.878485E+11	19175.00	299.10	1.9080E+09	16351.00	9.37	1.915617E+07	586.00
6	7.746786E+09	1.877944E+11	19266.00	281.19	1.8650E+09	14971.00	9.43	1.922493E+07	528.00
7	7.746866E+09	1.879217E+11	19274.00	308.00	1.9630E+09	15778.00	9.22	1.706866E+07	591.00
8	7.729257E+09	1.873669E+11	19391.00	269.86	2.1070E+09	16586.00	9.00	1.880115E+07	519.00
9	7.745738E+09	1.875424E+11	19203.00	296.18	2.0280E+09	14984.00	9.22	1.902808E+07	453.00
10	7.727762E+09	1.882426E+11	19240.00	284.81	2.2290E+09	15861.00	9.05	1.674231E+07	554.00

Motivation: using machine learning technique to automatically find 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

The proposed method



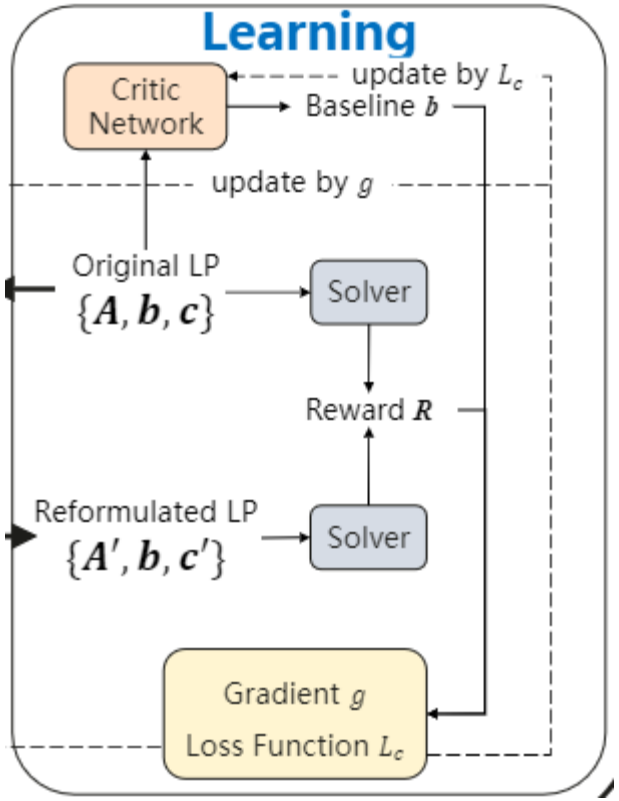
Representation: The inputting LP instance is represented by a bipartite graph, and then the embedding of variables is obtained via a GNN;

Aggregation: The embedding of variables will be further aggregated with a given group of variable clusters

Permutation: Taking as input the previous embeddings, a pointer network (PN) is used to output a new permutation of variables

Learning: The learning part interacts with the reformulation part to update the parameters of GNN and PN, trained with REINFORCE algorithm.

The proposed method



Policy

$$p(\pi|\{C_j\}_{j=1}^M) = \prod_{j=1}^{M} p(\pi(j)|\pi(<j), \{C_j\}_{j=1}^M)$$

- $\{C_j\}_{j=1}^M$: a splitting clustering
- π: the permutation of given splitting clustering

Reward

$$R(\pi|l) = 1 - \frac{S_{\mathcal{M}}(l|\pi)}{S_{\mathcal{M}}(l)}$$

- l: a given LP instance
- $S_{\mathcal{M}}(l|\pi)$: the solving performance of the reformulated LP instance
- $S_{\mathcal{M}}(l)$: the solving performance of the input LP instance

Optimization

$$J(\theta_G, \theta_P|l) = \mathbb{E}_{\pi \sim p_{\theta_G, \theta_P}(.|l)}[R(\pi|l)]$$

- θ_G: the parameterized representation (i.e., GCN)
- θ_P: the parameterized policy (i.e., pointer network)

Experimental evaluation

Table 1: Statistical description of used dataset

Dataset	m	n	NNZ
BIP	195.00 ± 00.00	1083.00 ± 00.00	7440.00 ± 00.00
WA	$6.43e04 \pm 54.51$	$6.1e04 \pm 00.00$	$3.62e05 \pm 6007.41$
HPP	$1.25e06 \pm 6.93e04$	$2.66e06 \pm 1.83e05$	$6.64e06 \pm 4.29e05$

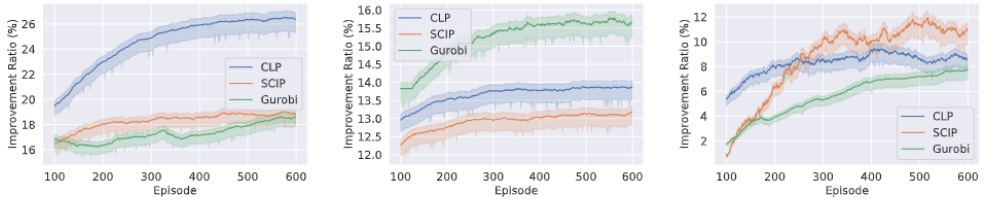
Table 2: Improvement of the iteration number (the lower, the better) over testing instances of three datasets.

Dataset	Solver	Avg. # Iteration		Improv. (%)
		Original	Reformulated	
BIP	CLP	956.32	705.10	26.27
	SCIP	673.58	545.33	19.04
	Gurobi	525.46	428.30	18.49
WA	CLP	6943.37	5985.18	13.8
	SCIP	11354.34	9874.87	13.03
	Gurobi	17962.85	15110.35	15.88
HPP	CLP	98352.27	90277.55	8.21
	SCIP	94124.48	82933.08	11.89
	Gurobi	23526.64	21656.27	7.95

Table 3: Improvement of solving time (the lower, the better) by our proposed method over all instances of three datasets

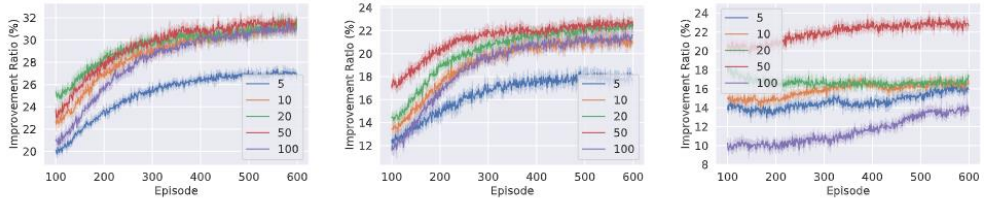
Dataset	Seen/Unseen	Solver	Avg. Solving Time (s)		Avg. Reordering Time (s)	Improv. (%)
			Original	Reformulated		
BIP	Seen	CLP	0.03862	0.02851	0.00011	26.18
		SCIP	0.04625	0.03453	0.00012	25.35
		Gurobi	0.01086	0.00849	0.00012	21.78
	Unseen	CLP	0.01792	0.01355	0.00019	24.39
		SCIP	0.04636	0.03472	0.00017	25.11
		Gurobi	0.01144	0.00939	0.00018	17.86
WA	Seen	CLP	6.74099	5.76085	0.00695	14.54
		SCIP	3.96057	3.39302	0.00811	14.33
		Gurobi	5.21913	4.23533	0.00723	18.85
	Unseen	CLP	7.17389	6.17385	0.00689	13.94
		SCIP	4.04960	3.29840	0.00709	18.55
		Gurobi	4.31686	3.59897	0.00685	16.63
HPP	Seen	CLP	71.0665	63.7396	0.01441	10.31
		SCIP	40.4433	34.2676	0.01719	15.27
		Gurobi	43.2238	37.7647	0.01996	12.63
	Unseen	CLP	70.9742	65.9208	0.01334	7.12
		SCIP	40.7626	34.9947	0.01755	14.15
		Gurobi	43.9299	40.0245	0.01454	8.89

Experimental evaluation and conclusion



(a) BIP (b) WA (c) HPP

Figure 3: Learning convergence and improvement of the iteration number over three datasets' training (seen) instances. Several findings can be pointed out: 1) the networks' parameters can converge over the three datasets; 2) with each testing solver, our method is effective in reducing the solving iteration number; 3) on the LP instances from WA and HPP, our method performs slightly worse than those from BIP, which demonstrates that it is relatively harder to learn neural network parameter over the large-scale and complex LP instances.



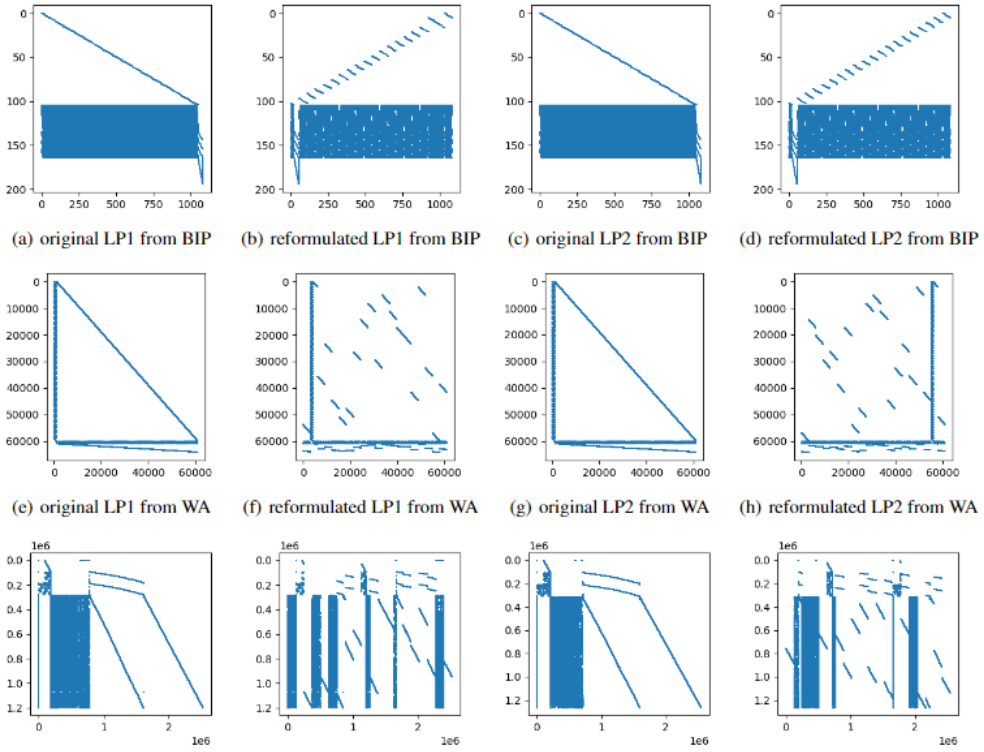
(a) CLP over BIP (b) SCIP over BIP (c) Gurobi over BIP

Figure 4: How different number of clustering block impacts our method's performance on BIP. Several findings can be pointed out: 1) it can be noted that increasing the number of clustering block within a specific range will lead to an improvement in the performance of our learning-based method; 2) it can also suffer a dramatic performance degradation if the number of clustering blocks is too large. Similar conclusions are drawn on the other two datasets (see Figure 6 in Appendix).

Take-away messages:

1) Various solvers (including commercial and open-source) can benefits from reformulation.
2) Gurobi is the most robust to the varying formulation.

Hindsight: what better formulation the agent found



(a) original LP1 from BIP (b) reformulated LP1 from BIP (c) original LP2 from BIP (d) reformulated LP2 from BIP

(e) original LP1 from WA (f) reformulated LP1 from WA (g) original LP2 from WA (h) reformulated LP2 from WA

(i) original LP1 from HPP (j) reformulated LP1 from HPP (k) original LP2 from HPP (l) reformulated LP2 from HPP

Figure 5: Visualization of reformulation process over BIP, WA and HPP datasets