Fast and Interpretable Mixed-Integer Linear Program Solving by Learning Model Reduction

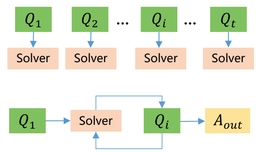
Yixuan Li[1], Can Chen[1], Jiajun Li[1], Jiahui Duan[2], Xiongwei Han[2], Tao zhong[2], Vincent Chau[1], Weiwei Wu[1], Wanyuan Wang[1]*.

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China

[2] Huawei Noah's Ark Lab, HUAWEI, Shenzhen, China

Email: {yixuanli,cchen,jiajun_li,vincentchau,weiweiwu,wywang}@seu.edu.cn, {jiahui.duan,hanxiongwei,zhongtao}@huawei.com

AI for MILP Solving: Backgrounds



- Large number of homogeneous Mixed-Integer Linear Programming (MILP) problems with similar combinatorial structureas need to be solved simultaneously.
- AI methods can exploit the correlation between the structure and the solution of MILP to accelerate large-scale MILP solving.

Previous Work

- **End-to-end Solution Prediction**
  neural diving, predict and search
- **Learning to Optimize Process**
  learn to branch/cut, neighborhood search
- **Learning to to Simplify the MILP**
  learn to presolve, model decomposition

Limitations:

Poor Interpretability: Intuition for variable predicting is unexplainable.
Large Search Space: Solution space is high-dimensional.

Limited Acceleration: Slower than end-to-end methods.
Frequent Interaction: Switching between AI algorithms and internal solver algorithms is time-consuming.

Limited Reduction: Presolved models are still complex.
Poor Reliability: Predictive reduction model's performance and reliability depend on ML fitting performance.

Our Contributions:

- We propose an end-to-end, model reduction approach to learn optimal solutions for MILP problems **efficiently** and **interpretably**.
- Integrate an attention architecture and preference-based loss function to fully exploit the preference information in terms of the performance of reduced models on instances to improve the **reliability**.
- Propose a SETCOVER technique to generate the minimum labels to **scale down the search space**.

Model Reduction: Strategy

Strategy: Encode the essential information of the solution of MILP, including the values of **integer variables** and **tight constraints** at optimality. [1]

Strategy

Instances Formulation

$$g(\bar{\theta},\bar{x}) = \begin{cases} g_1(x_I^1,...,x_I^d,x_1,...,x_n) \le b_1 \\ g_2(x_I^1,...,x_I^d,x_1,...,x_n) \le b_2 \\ g_3(x_I^1,...,x_I^d,x_1,...,x_n) \le b_3 \\ ... \\ g_q(x_I^1,...,x_I^d,x_1,...,x_n) \le b_q \end{cases} \quad \min_{\bar{x}} f(\bar{\theta},\bar{x}) \quad x_I \in \mathbb{Z}^d \quad x \in \mathbb{R}^n$$

Instance $\bar{\theta}_i$

Applying the strategy s_i to instance $\bar{\theta}_i$

Strategy s_i

Integer Variables $\bar{x}_I^*(\bar{\theta})$

Tight Constraints $\mathcal{T}(\bar{\theta}) = \{i \in \{1,\dots,m\} \mid g_i(\bar{\theta},x^*) = 0\}$

Original **MILP** problem can be reduced to an **LP** problem involving only continuous variables and fewer constraints.

$$g'(\bar{\theta},\bar{x}) = \begin{cases} g'_1(x_{p1},...,x_{p1}) \le b'_1 \\ g'_2(x_{p2},...,x_{q2}) \le b'_2 \\ ... \\ g'_k(x_{p1},...,x_{pn}) \le b'_k \end{cases} \quad \min_{\bar{x}} f'(\bar{\theta}_0,\bar{x}) \quad x \in \mathbb{R}^n$$

Reduced Formulation

End to End Framework

Through this strategy, we can transform the task of solving MILP problem instances into **learning a mapping from instances to the optimal strategies.**

① Strategy Generation and Pruning

② Preference-based Strategy Learning

The proposed framework comprises two phases:

① generate a set of useful strategies that can be applied to all MILP instances.

② predict the optimal strategy for any MILP instance.

Preference-based Strategy Learning

Motivation Existing strategy learning approaches typically only focus on the exact optimal strategy [2], or treat a set of feasible strategies equally desirable [3], overlooking the significant **preference** information available in the instance-strategy space.

To address this issue, we transform the performance (i.e., objective function value and constraint feasibility) of strategies on MILP instances as preferences, and proposes a novel preference-based learning method.

Infeasiblity: $p(\theta_i,s_j) = \|(g(A,\hat{x}^*_{i,j}) - b)_+\|_\infty / \|b\|$

Suboptimality: $d(\theta_i,s_j) = |f(c,\hat{x}^*_{i,j}) - f(c,x^*_{i,j})| / |f(c,x^*_{i,j})|$

Reward: $r(\theta_i,s_j) = -log(p(\theta_i,s_j) + d(\theta_i,s_j))$

Define the preference $\succ$ by the rewards r:

$$(\theta_i,s_j) \succ (\theta_i,s_k) \Leftrightarrow r(\theta_i,s_j) > r(\theta_i,s_k)$$

Previous preference-based learning (e.g., RLHF [4]) requires selecting **all possible pairwise** comparisons as samples (e.g., $\binom{M}{2}$ sample pairs when there are M strategies).

Fortunately, the instance preferences on strategies have a **transitivity** structure: if $s_i \succ s_j, s_j \succ s_k$, then $s_i \succ s_k$.

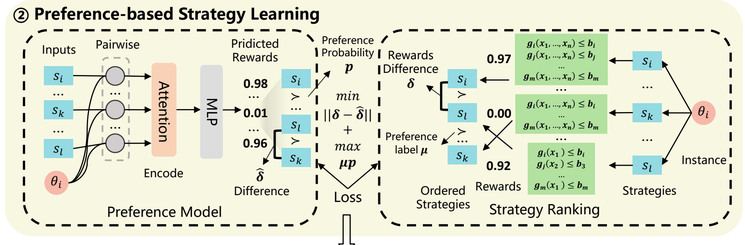
Therefore, when the candidate strategies are **ranked** by their preferences:

$$s_{\sigma(j)} \in S^P_\sigma = \{s_{\sigma(1)},...,s_{\sigma(M^P)}\}$$

only the M ordered preference samples are necessary as the preference set:

$$\{\langle(\theta_i,s_{\sigma(1)}) \succ (\theta_i,s_{\sigma(2)})\rangle, \cdots, \succ(\theta_i,s_{\sigma(M^P)})\rangle\}$$

② Preference-based Strategy Learning



Preference loss $L_d(\phi)$: maximize the probability for every ordered adjacent sample pair: $p_{i,j} = \frac{\exp(\hat{r}_{i,\sigma(j)})}{\exp(\hat{r}_{i,\sigma(j)}) + \exp(\hat{r}_{i,\sigma(j+1)})}$.

$$L_d(\phi) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M^P-1}(\hat{r}_{i,\sigma(j)} - \hat{r}_{i,\sigma(j+1)} - \delta_{i,j})^2$$

If the model outputs an higher $\hat{r}_{\sigma(2)}$, leading $\hat{r}_{\sigma(1)} < \hat{r}_{\sigma(2)}, \hat{r}_{\sigma(2)} > \hat{r}_{\sigma(3)}$, although $\hat{r}_{\sigma(1)} < \hat{r}_{\sigma(2)}$ would be penalized by $L_d(\phi)$, but $\hat{r}_{\sigma(2)} > \hat{r}_{\sigma(3)}$ might even reduce the value of $L_d(\phi)$. So we introduce **difference loss $L_p(\phi)$** to reinforce the correct order:

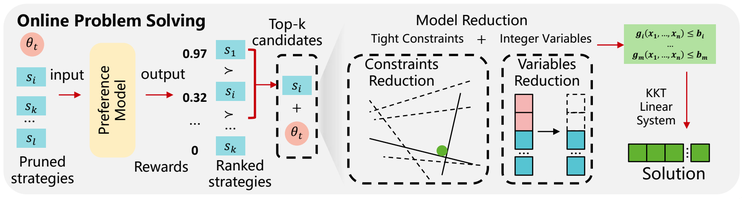
$$L_p(\phi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M^P-1}|\mu_{i,j}\log(p_{i,j}) + (1-\mu_{i,j})\log(1-p_{i,j})|$$

Total loss $L_{total}(\phi)$: $L_{total}(\phi) = \lambda_1 L_p(\phi) + \lambda_2 L_d(\phi)$

After training, reliability can be increased by taking the **Top-k** model output strategies as candidates.

To solve the reduced model, for special types of problems such as MIQP and MILP, the linear system can be simplified based on the **KKT** optimality conditions [1].

Online Problem Solving



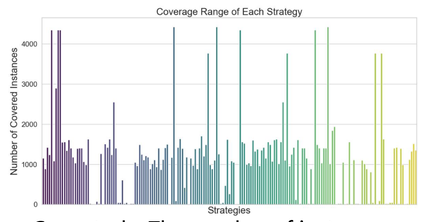
Strategy Generation and Pruning

Strategy Generation: To identify a set of useful strategies that will be used as labels for model reduction training and learning.

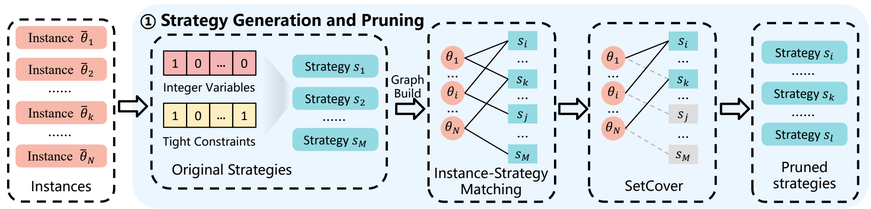
We randomly generate instances as well as its optimal strategy until the Good-Turning estimator falls below a tiny value. [5]

However, in large-scale scenario, the number of strategies (i.e., labels) can grow quickly, making the learning task very difficult.

Strategy Pruning: A SETCOVER based method while ensuring that all generated instances can be covered, in which the cover means that there is at least one feasible strategy for an instance.



Case study: The number of instances covered by the strategies.



Experiments

Experimental Settings

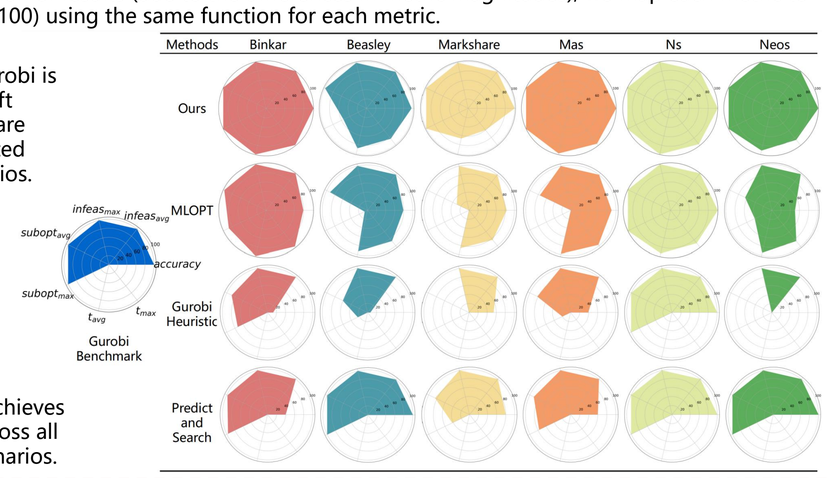
Datasets: MIPLIB (six real-world scenarios), Fuel Cell Energy Management (scale can be increased by increasing T) [6], Inventory Management (five large-scale with average number of 100,000 constraints).
Baselines: Gurobi, Gurobi Heuristic, MLOPT [5], Predict and Search [7].
metrics: infeasiblity $p(\theta_i,s_j)$, suboptimality $d(\theta_i,s_j)$, accuracy, $\epsilon_1 = \epsilon_2 = 1e-4$

$$accuracy = \frac{1}{N}|\{\theta_i \mid p(\theta_i,\hat{s_j}) \le \epsilon_1 \wedge d(\theta_i,\hat{s_j}) \le \epsilon_2\}|$$

Experiments on MIPLIB

The performance on six scenarios from MIPLIB and each vertex in the subplot represents a metric. For better presentation of results (due to the differences in metric magnitudes), we map each metric to the range of (0, 100) using the same function for each metric.

The result of Gurobi is shown on the left since its results are similarly presented across all scenarios.

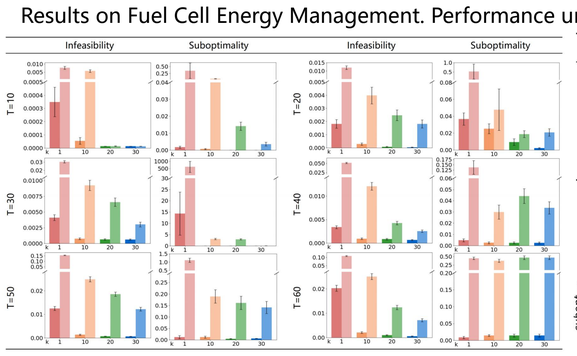


Our approach achieves good results across all metrics and scenarios.

Experiments on The Other Datasets

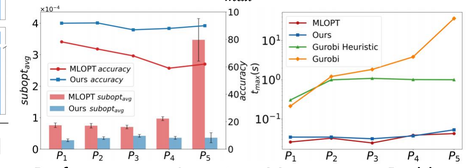
Results on Fuel Cell Energy Management. Performance under varying problem scale (T).

T	$t^H_{\max}(s)$	$t^G_{\max}(s)$	M	method	M^P	$t_{\max}(s)$	accuracy	gain
20	0.366	0.379	578	Ours	41	0.061	97.66%	**10.98%**
				MLOPT	110	0.047	86.68%	
40	1.120	4.840	2041	Ours	75	0.069	94.20%	**16.35%**
				MLOPT	2026	0.054	77.85%	
60	1.184	33.733	2924	Ours	129	0.086	84.90%	**38.93%**
				MLOPT	2863	0.069	45.97%	



Average infeasibility and suboptimality for our method (bar on the left) and MLOPT under varying k and T.

t^H_{max} means the maximum computation time from Gurobi Heuristic and t^G_{max} is from Gurobi.



Performance on Inventory Management Problems.

Experiment on Directly Reward Fitting

Performance of our method versus Reward Fitting (RF) on Fuel Cell Energy Management when k = 1.

	$T=10$				$T=20$				$T=30$			
method	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain
Ours	95.65%	0.00035	0.00176	**4.58%**	83.04%	0.00182	0.03676	**10.60%**	73.88%	0.00409	14.33399	**9.38%**
RF	91.07%	0.00680	0.00836		72.43%	0.00685	0.03883		64.51%	0.00705	0.04398	
	$T=40$				$T=50$				$T=60$			
method	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain
Ours	73.21%	0.00338	0.00477	**9.60%**	56.14%	0.01250	0.01210	**14.29%**	39.71%	0.02029	0.00865	**6.82%**
RF	63.62%	0.00740	0.01307		41.85%	0.02385	0.01668		32.90%	0.01387	0.04394	

Directly fitting the reward is difficult and can lead to errors. Moreover, in practical policy deployment, only the relative preference between policies is required, not the absolute reward values.

Experiment on Ranking Sampling

Performance of our method versus No Ranking (NR) on Fuel Cell Energy Management when k = 1.

	$T=10$				$T=20$				$T=30$			
method	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain
Ours	95.65%	0.00035	0.00176	**5.08%**	83.04%	0.00182	0.03676	**8.48%**	73.88%	0.00409	14.33399	**6.70%**
NR	89.84%	0.00119	0.00069		74.55%	0.00315	0.01899		67.19%	0.00639	0.02675	
	$T=40$				$T=50$				$T=60$			
method	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain	accuracy	$infeas_{avg}$	$subopt_{avg}$	gain
Ours	73.21%	0.00338	0.00477	**9.49%**	56.14%	0.01250	0.01210	**5.25%**	39.71%	0.02029	0.00865	**13.80%**
NR	63.73%	0.00623	0.01223		50.89%	0.01184	0.00366		25.91%	0.01411	0.00581	

Ranking-based sampling and loss functions can enhance performance and reduce training overhead.

[1] Boyd, S. P.; and Vandenberghe, L. 2014. Convex Optimization. Cambridge University Press.
[2] Misra, S.; Roald, L.; and Ng, Y. 2022. Learning for Constrained Optimization: Identifying Optimal Active Constraint Sets. INFORMS J. Comput., 34(1): 463–480.
[3] Bertsimas, D.; and Kim, C. W. 2023. A Prescriptive Machine Learning Approach to Mixed-Integer Convex Optimization. INFORMS J. Comput., 35(6): 1225–1241.
[4] Christiano, P. F.; Leike, J.; Brown, T. B.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. In NIPS'17, 4302–4310.
[5] Bertsimas, D.; and Stellato, B. 2022. Online mixed-integer optimization in milliseconds. INFORMS Journal on Computing, 34(4): 2229–2248.
[6] Frick D, Domahidi A, Morari M.2015. Embedded Optimization for Mixed Logical Dynamical Systems, Comput. Chemical Engrg. 72:21–33.
[7] Han, Q.; Yang, L.; Chen, Q. et.al. 2023. A GNN-Guided Predict-and Search Framework for Mixed-Integer Linear Programming. In ICLR'23.