Contrastive Predict-and-Search for Mixed Integer Linear Programs

ICML'24

Taoan Huang, Aaron Ferber, Arman Zharmagambetov,
Yuandong Tian, Bistra Dilkina

Stanford CS/MS&E 331

ML for integer programming

Mixed integer linear programs (MILP):

- Flexible modeling tool for NP-hard combinatorial optimization
- E.g., scheduling, network design, …

Solvers:

- Typically solved using Branch-and-Bound (e.g., used by Gurobi)
- Can be very computationally expensive

Motivation for ML-based heuristics:

- Learn heuristics to find high-quality primal solutions quickly
- Guide solver's search to accelerate convergence to good solutions

(Binary) integer linear program

minimize $\boldsymbol{c} \cdot \boldsymbol{x}$

subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

$\boldsymbol{x} \in \{0,1\}^n$

(Paper generalizes beyond binary)

"Predict-and-Search" framework

1. Learn model/distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid M)$
   - $\boldsymbol{\theta}$: trainable parameters
   - M: MILP
   - $\boldsymbol{x}$: solution
2. Use prediction to **reduce** MILP search space
3. Solve **reduced MILP** with standard solver

This paper focuses on Step 1:

How to train an effective prediction model

Contrastive Predict-and-Search (ConPaS)

Existing models are often trained with, e.g., BCE loss

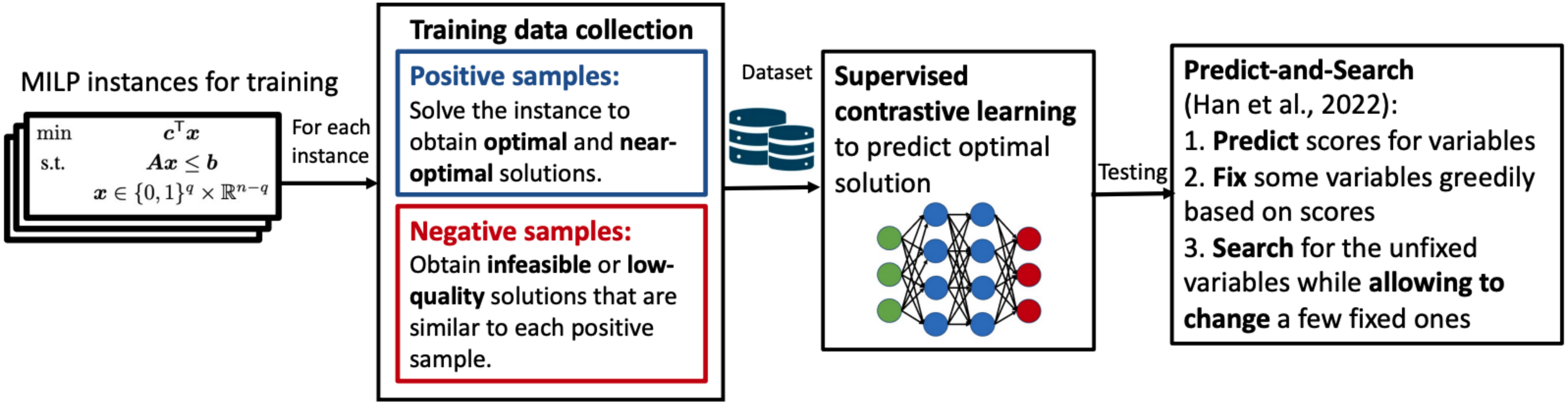
- May not provide a sufficiently discriminative signal

This paper: Train a model to *contrast:*

1. Positive/high-quality solutions
2. Negative/low-quality/infeasible solutions

Contribution: New **contrastive learning** strategy for MILPs

ConPaS framework



- **Positive examples:** Solve each instance M (e.g., with Gurobi)
  - Collect (e.g., ≤ 50) solutions with opt/near-opt objective value
  - Forms a set $\mathcal{S}_p^M$
- **Key question:** How to generate negative examples?

Negative examples: Variant **ConPaS-Inf**

Goal: Collect **infeasible** solutions similar to positive samples

Method (for each $\boldsymbol{x}_p \in \mathcal{S}_p^M$)**:**

- Randomly perturb ~10% of binary variables to get $\boldsymbol{x}'$
- Check if $\boldsymbol{x}'$ is infeasible (using solver)
- If infeasible, add to set $\mathcal{S}_n^M$

Negative examples: Variant **ConPaS-LQ**

Goal: Collect **low-quality** solutions similar to positive samples

Method (for each $\boldsymbol{x}_p \in \mathcal{S}_p^M$), solve and add to set $\mathcal{S}_n^M$ **:**

$$
\begin{array}{ll}
\textbf{maximize} & \boldsymbol{c} \cdot \boldsymbol{x}' \\
\text{subject to} & A\boldsymbol{x}' \leq \boldsymbol{b} \\
& \left\| \boldsymbol{x}_p - \boldsymbol{x}' \right\|_1 \leq k
\end{array}
$$

Contrastive loss

- GNN predicts a score vector $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid M)$
- Loss is weighted by solution quality

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{M} \frac{-1}{\left|\mathcal{S}_p^M\right|} \sum_{\boldsymbol{x}_p \in \mathcal{S}_p^M} \ell(\boldsymbol{\theta} \mid \boldsymbol{x}_p, M)$$

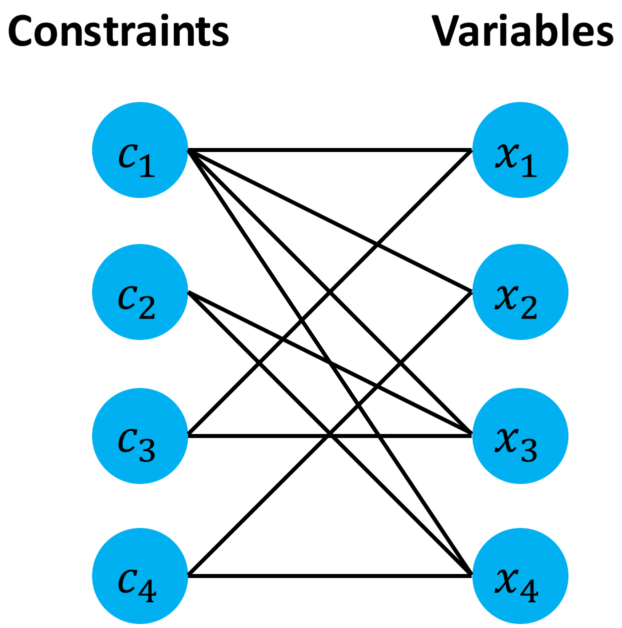
$$\ell(\boldsymbol{\theta} \mid \boldsymbol{x}_p, M) = \log \frac{\exp\left(\frac{\boldsymbol{x}_p^\top p_{\boldsymbol{\theta}}(\boldsymbol{x}_p \mid M)}{\tau(\boldsymbol{x}_p \mid M)}\right)}{\sum_{\tilde{\boldsymbol{x}} \in \mathcal{S}_n^M \cup \{\boldsymbol{x}_p\}} \exp\left(\frac{\tilde{\boldsymbol{x}}^\top p_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}} \mid M)}{\tau(\tilde{\boldsymbol{x}} \mid M)}\right)}$$

  - $\tau(\boldsymbol{x} \mid M)$ inverse proportional to obj when feasible; constant else

Graph representation

IP represented as bipartite graph

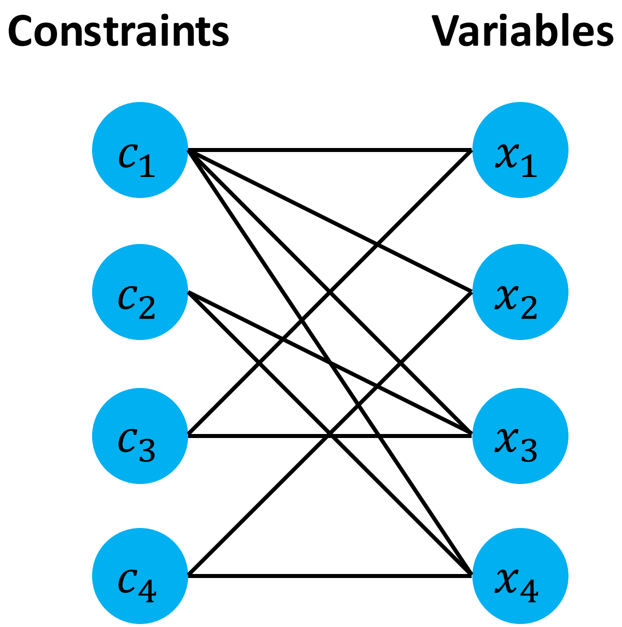
$$
\begin{array}{lll}
\max & 9x_1 + 5x_2 + 6x_3 + 4x_4 & \\
\text{s.t.} & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 & (c_1) \\
 & x_3 + x_4 \leq 10 & (c_2) \\
 & -x_1 + x_3 \leq 0 & (c_3) \\
 & -x_2 + x_4 \leq 0 & (c_4) \\
 & x_1, x_2, x_3, x_4 \in \{0,1\} &
\end{array}
$$



Graph representation

IP represented as bipartite graph

- **Edge feature:** constraint coefficient
- **Example node features:**
  - Constraints:
    - Cosine similarity with objective
    - Tight in LP solution?
  - Variables:
    - Objective coefficient
    - Solution value equals upper/lower bound?



Predict-and-search

1. Select k_0 variables with **smallest** $p_{\boldsymbol{\theta}}(x_i \mid M)$; call them $\mathcal{X}_0$
2. Select k_1 variables with **largest** $p_{\boldsymbol{\theta}}(x_i \mid M)$; call them $\mathcal{X}_1$
3. Fix all variables in $\mathcal{X}_0$ to 0, $\mathcal{X}_1$ to 1
4. Define:

$$B(\mathcal{X}_0, \mathcal{X}_1, \Delta) = \left\{ \boldsymbol{x} \in \{0,1\}^n : \sum_{x_i \in \mathcal{X}_0} x_i + \sum_{x_i \in \mathcal{X}_1} (1 - x_i) \leq \Delta \right\}$$

5. Solve:

$$\boxed{\begin{array}{ll} \text{minimize} & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in B(\mathcal{X}_0, \mathcal{X}_1, \Delta) \end{array}}$$

Performance measure: Primal gap

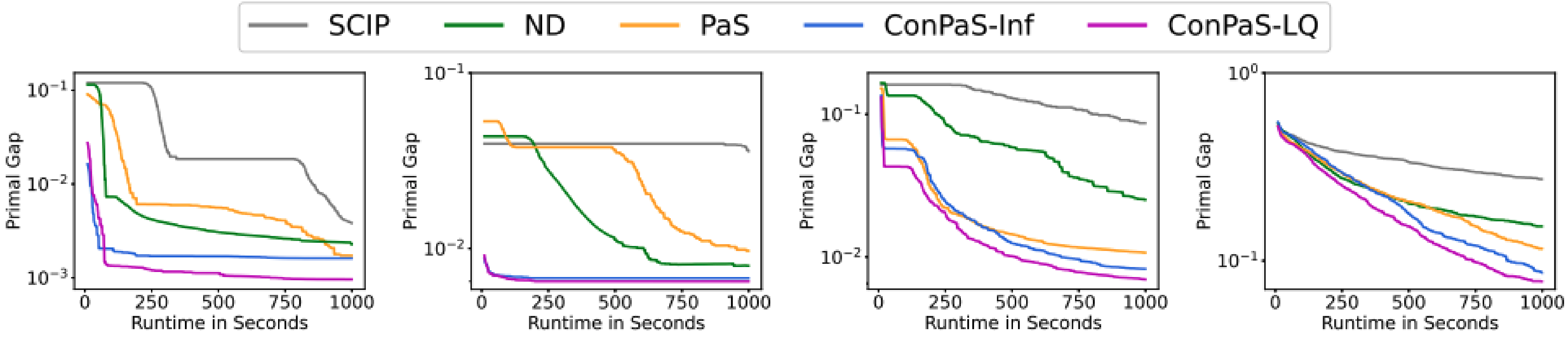
Minimize $\boldsymbol{c} \cdot \boldsymbol{x}$ subject to $A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \{0,1\}^n$

- $\boldsymbol{x}^*$: optimal solution
- IP solvers iteratively find better and better feasible solutions
- Primal bound: Objective value of best **feasible solution** so far
  - Often called the "incumbent" solution $\widehat{\boldsymbol{x}}$; $\boldsymbol{c} \cdot \boldsymbol{x}^* \leq \boldsymbol{c} \cdot \widehat{\boldsymbol{x}}$
- Dual bound: Objective value of the LP relaxation solution $\boldsymbol{x}_{\mathrm{LP}}$
  - $\boldsymbol{c} \cdot \boldsymbol{x}_{\mathrm{LP}} \leq \boldsymbol{c} \cdot \boldsymbol{x}^*$
- **Primal gap:** $\frac{\boldsymbol{c} \cdot \widehat{\boldsymbol{x}} - \boldsymbol{c} \cdot \boldsymbol{x}_{\mathrm{LP}}}{|\boldsymbol{c} \cdot \widehat{\boldsymbol{x}}|}$

Sample of results

Best open-source solver (see comparison with Gurobi in appendix)

Prior work [Nair et al., '20; Han et al., '22]



Min vertex cover

Max independent set

Combinatorial auctions

Item placement

Summary

ConPaS: New ML-based framework for MILP heuristics

Contribution: A novel contrastive learning strategy

- **Key Idea:** Use "hard negatives" (infeasible or low-quality)
- Learn a more discriminative models