

ML-guided search for Mixed Integer Linear Programming

Bistra Dilkina

Associate Professor of Computer Science

Co-Director, USC Center on AI in Society (CAIS)
USC Site Lead, AI Institute for Optimization (AI4OPT)

University of Southern California

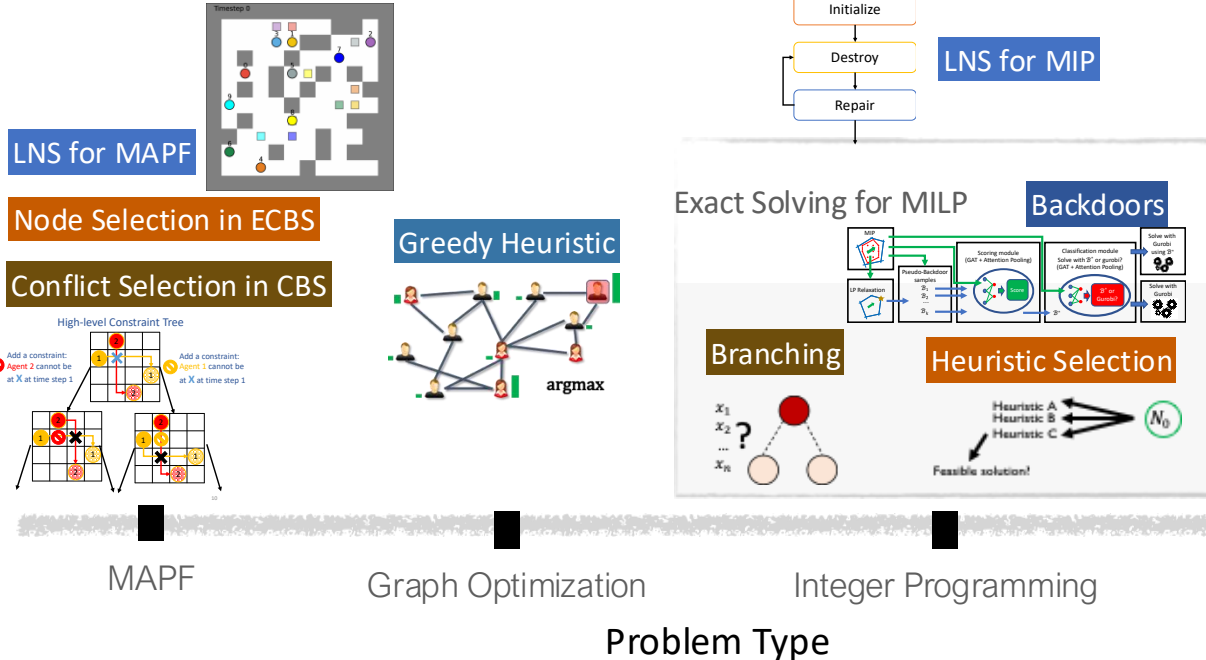
Aug 10, 2025

at SAT'25 and CP'25 workshop on “Machine Learning for Solvers and Provers (ML4SP)”

ML ↔ Combinatorial Optimization

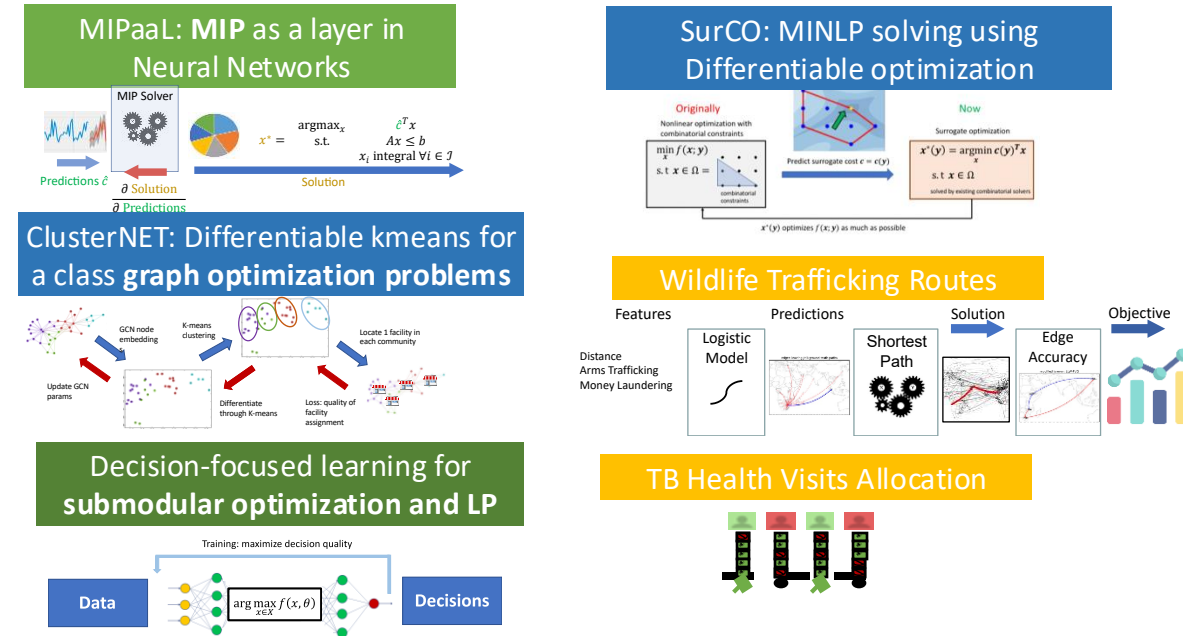
- Exciting and growing research area

Infusing Discrete Optimization with Machine Learning



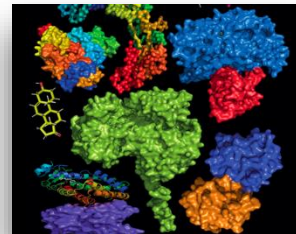
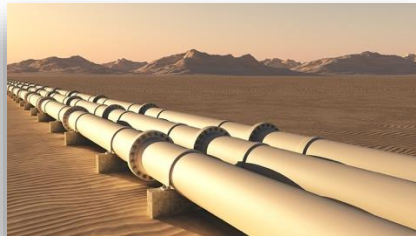
Augment discrete optimization algorithms with learning components

Infusing ML with Constrained Decision Making



Learning methods that incorporate the combinatorial decisions they inform

Mixed Integer Linear Programs (MIP)



Flexible mathematical program framework

$$\begin{array}{lll}
 \min_x & c^T x & \text{objective} \\
 \text{s.t.} & Ax \leq b & \text{constraints} \\
 & x_j \in \mathbb{Z} \forall j \in \mathcal{I} & \text{integrality}
 \end{array}$$

Scalable solving methods key to many real-world applications:

- Branch-and-bound (SCIP, Gurobi, CPLEX)
- Large Neighborhood Search
- Etc.

Distributional MIP Solving

Learn from:



Day 1



Day 2

...



Day T

Solve



Day T + Δ

Repeated solving of similar problems in many real-world settings

Learn search policies tailored to the MIP distribution

- get better performance than generic one-fits-all approaches by fitting to specific problem structure
- no need for hand-designed custom approaches to new / challenging domains

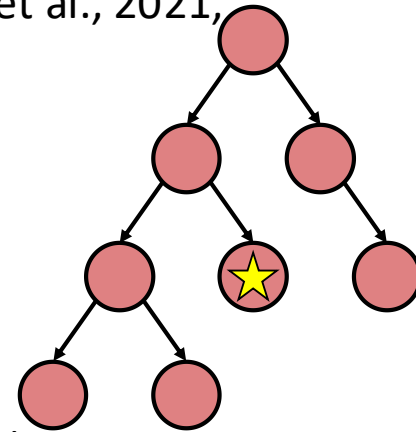
Landscape: ML for MIP solving

ML at the top-level: algorithm configuration / solver ensembling

- SMAC, HydraMIP, AutoML, many more

ML to improve tree search (BnB) for MIP:

- **select variables to branch on** (Khalil et al., 2016; Gasse et al., 2019; Gupta et al., 2020; Zarpellon et al., 2021, Ferber et al, 2022, Scavuzo etl al 2024, Lin et al 2024, Cai et al 2024)
- **select nodes to expand** (He et al., 2014; Labassi et al., 2022).
- **select primal heuristics** (Khalil et al., 2017; Chmiela et al., 2021, others)
- **select cutting planes** (Tang et al., 2020; Paulus et al., 2022; Huang et al., 2022c, Li et al, others)



ML for Incomplete / Heuristic Search for MIP

- Large Neighborhood Search / LNS (Song et al, Wu et al, Sonnerat et al, Liu et al, Huang et al 2023)
- Solution Prediction and Predict-and-Search: (Ding et al, Nair et al, Khalil et al, Han et al, Huang et al 2024)

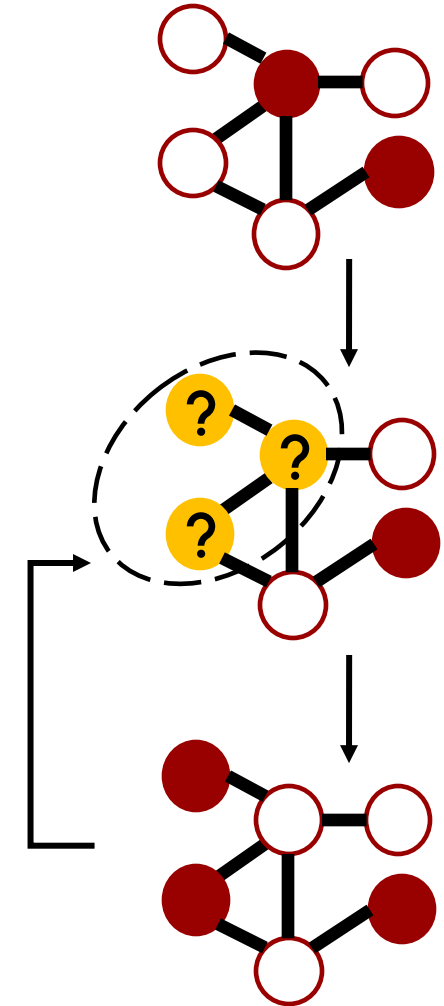
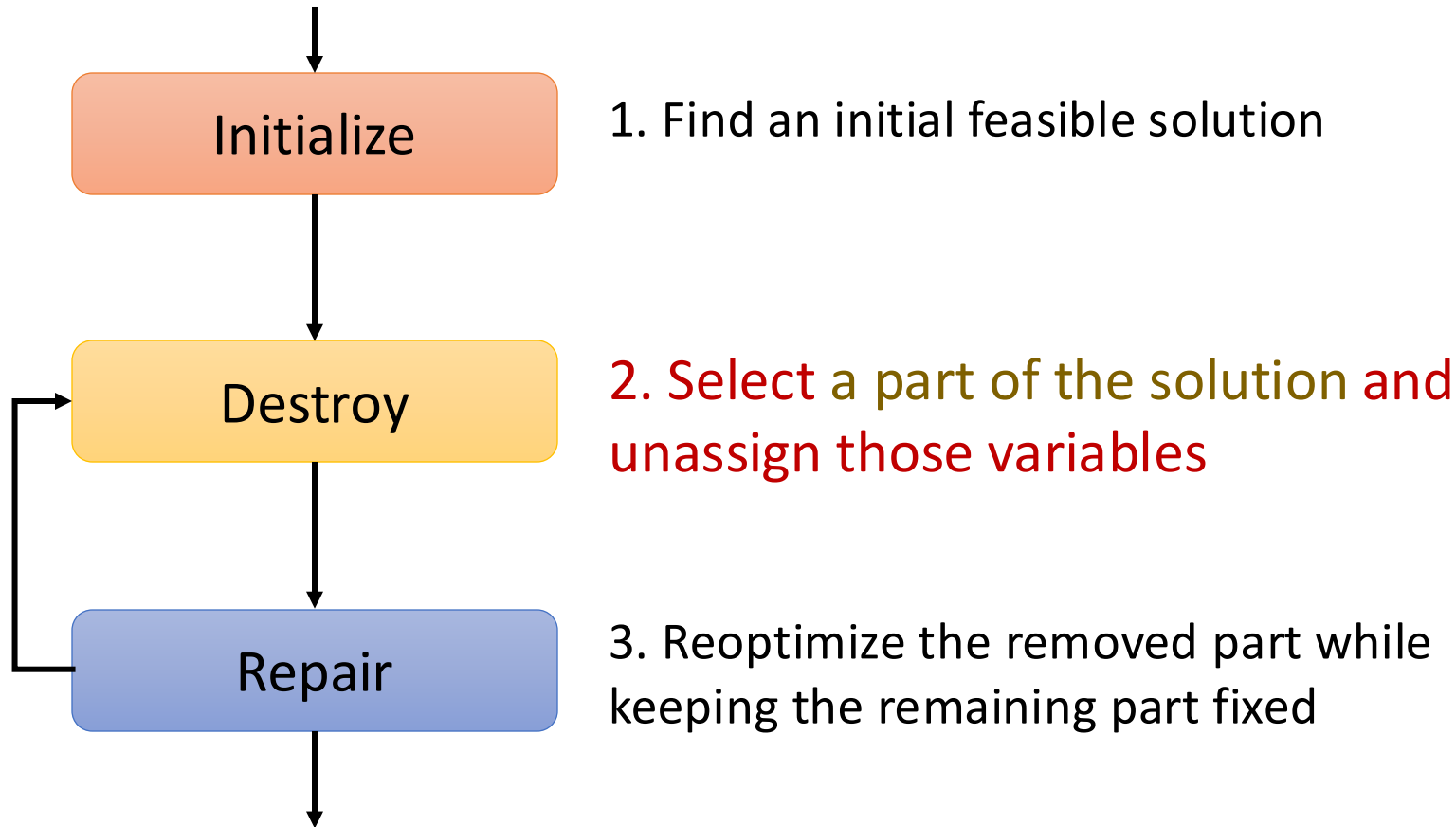
ML for other Combinatorial Optimization Problems:

- SAT, TSP, VRP, SMT, Multi-agent Path Finding, other Graph Opt

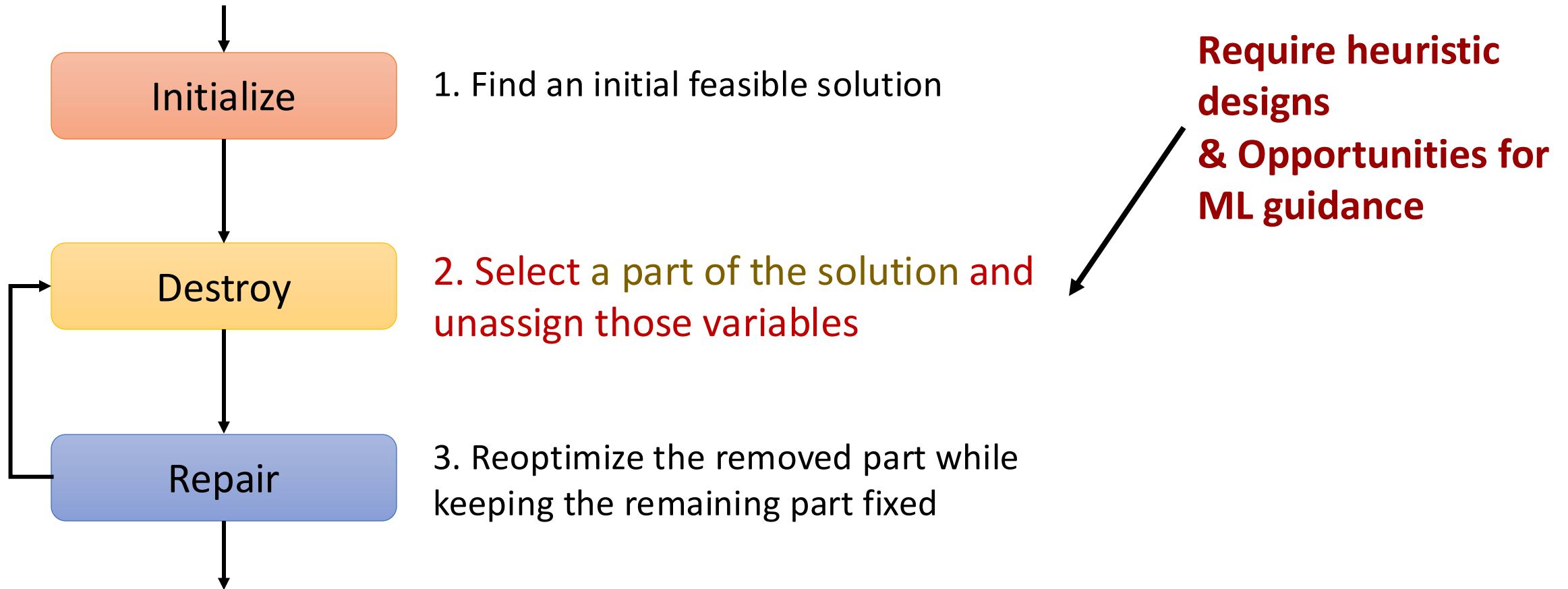
Outline

- 1) **Large Neighborhood Search (LNS-MIP)** via contrastive loss
- 2) **Contrastive loss as a unifying paradigm for ML-guided MIP**
 - Predict-and-Search with contrastive loss
 - Branching priorities in B&B (Backdoor prediction) with contrastive loss
- 3) **Multi-task Learning** for ML-guided MIP solving
- 4) Benchmarking: **Distributional MIPLIB**

Large Neighborhood Search (LNS)



Large Neighborhood Search (LNS)



Local Branching (LB) Heuristic [Fischetti & Lodi, 2003]

Destroy step: Given an ILP and the current best solutions x^* ,
choose at most k variables to reoptimize while fixing the rest

How do we find the **optimal** subset of k variables?

Local Branching: solve a ILP with n variables and $m + 1$ constraints

$$\begin{array}{ll} \min_x & c^T x \\ \text{s.t.} & Ax \leq b \\ & x_i \in \{0,1\} \forall i \end{array}$$

objective
constraints
integrality

$$\sum_{i:x_i^*=0} x_i + \sum_{i:x_i^*=1} (1 - x_i) \leq k$$

Variable i is selected
for LNS if changed
value from x^*

LB Heuristic [Fischetti & Lodi, 2003]

- Solve the Local Branching ILP
- In hindsight, variables whose values change are the optimal subset
- ☹ But very expensive to solve computationally

Allow to change $k = 3$

Current best solution

LB solution

Difference

x_1	x_2	x_3	x_4	x_5
1	1	1	0	0
1	0	0	1	0
0	1	1	1	0



IL-LNS: Imitation Learning Approach

Google DeepMind

Learning to **imitate Local Branching** [Sonnerat et al., 2021]

- **Data collection:** run Local Branching ILP exhaustively to find one great example (label) of the variable destroy set for each 'state' (problem, current complete solution)
- **Training:** Imitate Local Branching by classifying the variables (destroy or not)
- **Testing:** Sample neighborhoods based on the predicted scores of each variable


[Sonnerat et al., 2021] Learning a large neighborhood search algorithm for mixed integer programs. Arxiv.


Contrastive Loss

Instead of learning only from the best samples provided by local branching...

We also learn to distinguish between good and bad samples with ***contrastive loss***


Contrastive pairs






\equiv

 \neq





Positive Pair

Negative Pair

$$l(\mathbf{x}, \mathbf{y}) = -\log \left(\frac{e^{\text{similarity}(\mathbf{x}, \mathbf{y})}}{e^{\text{similarity}(\mathbf{x}, \mathbf{y})} + e^{\text{similarity}(\mathbf{x}, \mathbf{z})} + e^{\text{similarity}(\mathbf{x}, \mathbf{w})}} \right)$$

<https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>

Recent use of contrastive learning in other settings:

Contrastive learning of visual representations (Hjelm et al., 2019; He et al., 2020; Chen et al., 2020)
and graph representations (You et al., 2020; Tong et al., 2021)

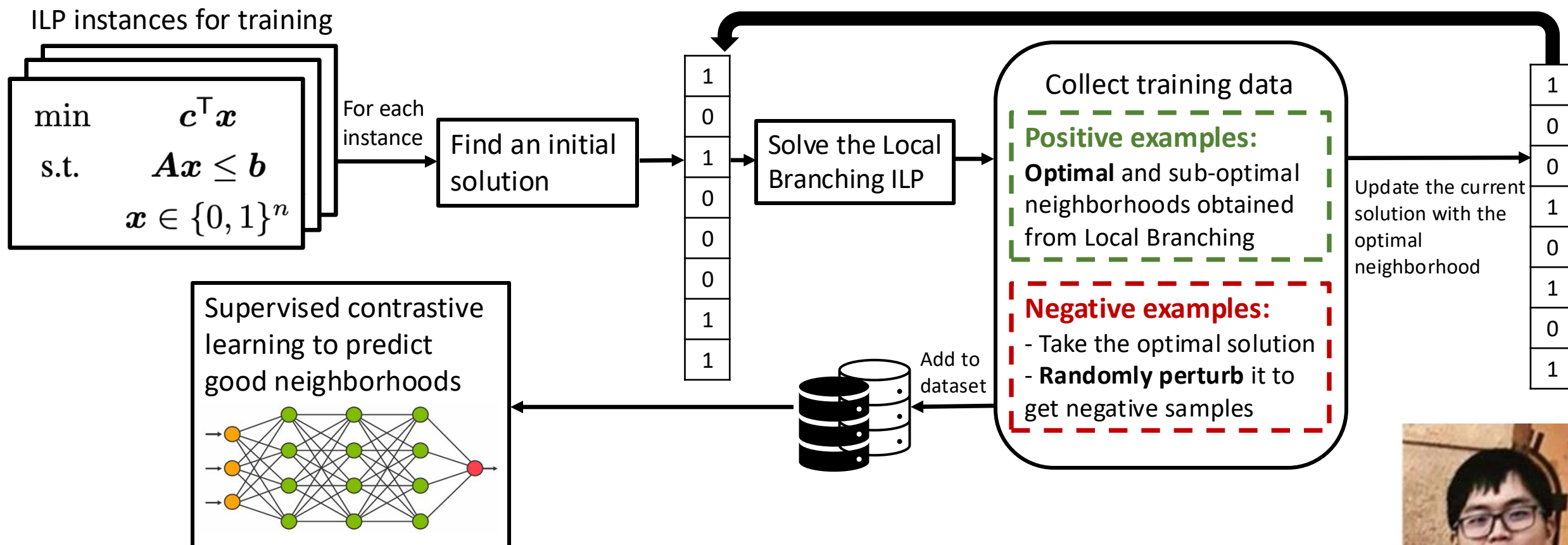
[Mulamba et al., 2021] Contrastive losses and solution caching for predict-and-optimize. IJCAI 2021.

[Duan et al., 2022] Augment with care: Contrastive learning for combinatorial problems. ICML 2022

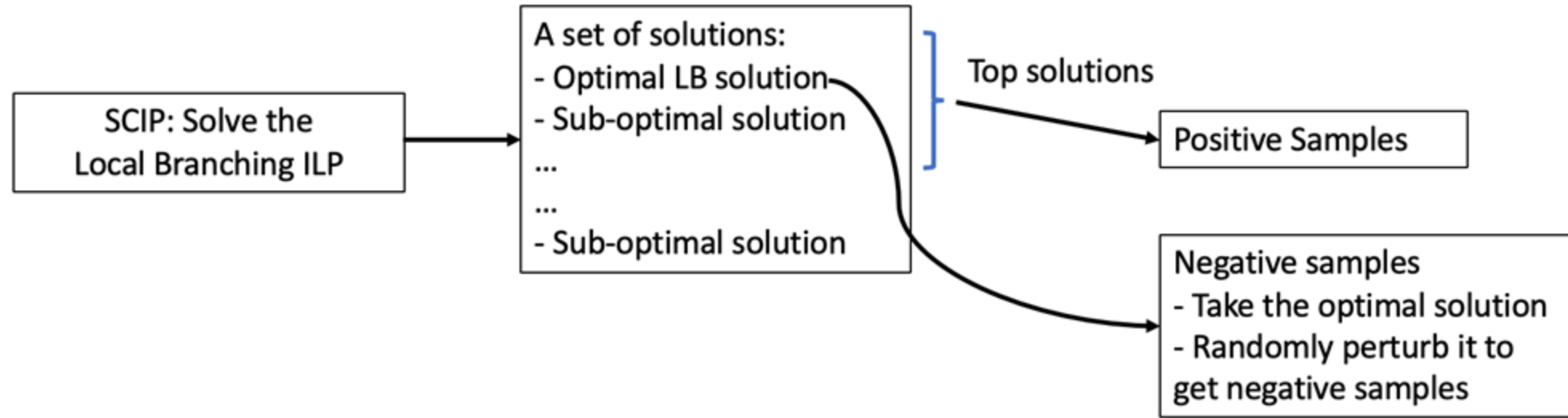
Our Approach: CL-LNS

[Taoan Huang et al., ICML 2023] Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning

Training and data collection pipeline



Contrastive loss - Data collection details

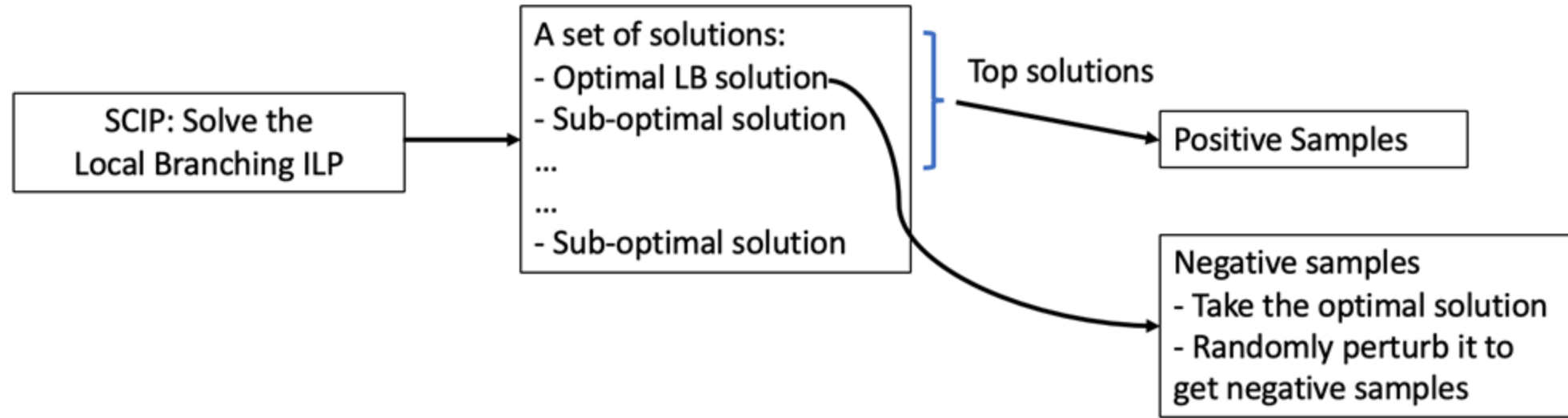


Optimal sample := variables changed in the optimal solution found by
local branching ILP solved by SCIP within 1 hours
best_improve := improvement of the optimal sample over incumbent objective value

Positive samples of destroy sets:

- All solutions [0/1 vector] found by local branching ILP with *improvement* $\geq 0.5 * \text{best_improve}$
- up to max of 10 solutions

Contrastive loss - Data collection details



Optimal sample := variables changed in the optimal solution found by
local branching solved by SCIP within 1 hours

best_improve := improvement of the optimal sample over incumbent objective value

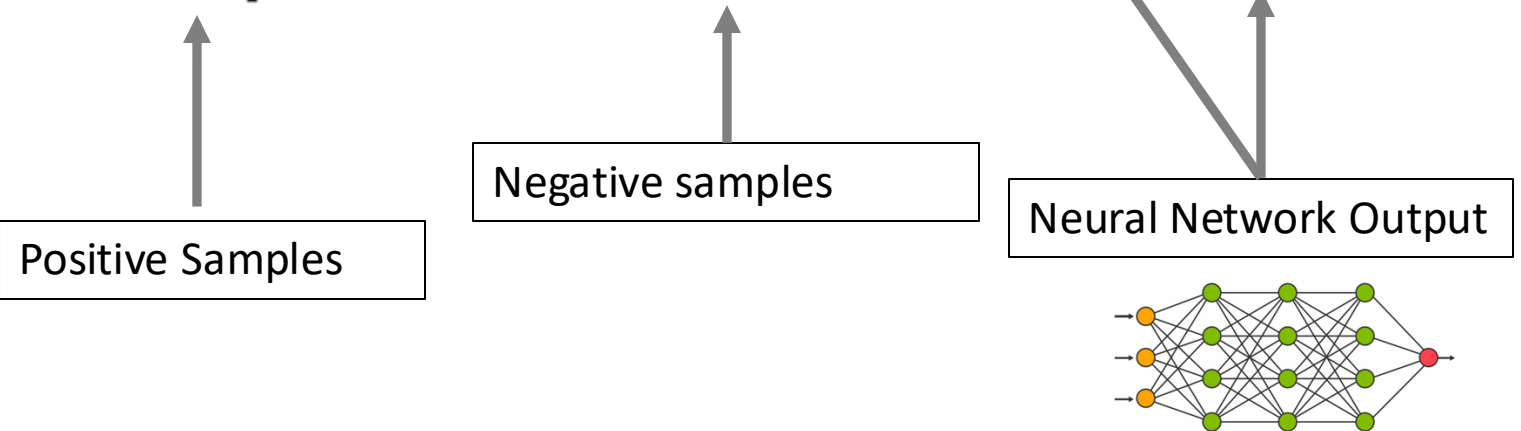
Negative samples of destroy sets ($k * \text{Num Positive Samples}$, $k=9$):

- Randomly replace 5% of variables in the optimal sample, Solve LNS MIP-subproblem with SCIP
- Record it as a negative sample if $\text{improvement} \leq 0.05 * \text{best_improve}$
- If not enough negative samples found, increase to 10% to 20%, 30%...100%

CL-LNS: Training with a Contrastive Loss

Contrastive Loss function (InfoNCE):

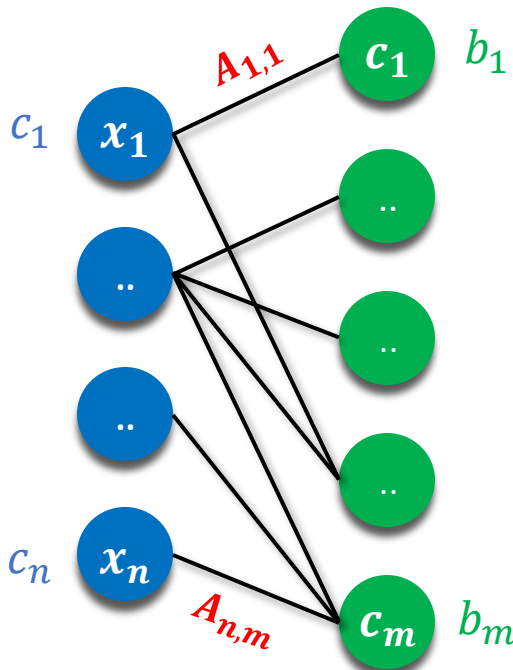
optimizes the negative log probability of the final embedding being similar to the positive samples

$$\frac{-1}{|\mathcal{S}_{pos}|} \sum_{\mathbf{a} \in \mathcal{S}_{pos}} \log \frac{\exp(\mathbf{a}^T \boldsymbol{\pi}(\mathbf{x}) / \tau)}{\sum_{\mathbf{a}' \in \mathcal{S}_{neg} \cup \{\mathbf{a}\}} \exp(\mathbf{a}'^T \boldsymbol{\pi}(\mathbf{x}) / \tau)}$$


\mathbf{a} (**destroy set**): each positive/negative sample is a binary 0/1 vector of length n variables
 \mathbf{x} is a set of **features of the MIP to be solved and the current search state**
 $\boldsymbol{\pi}(\mathbf{x})$: NN output is a continuous $[0,1]$ vector of length n variables: **suggested destroy set**

CL-LNS: Features [Khalil et al., 2016] [Gasse et al., 2019]

$$\min \mathbf{c}^T \mathbf{x} \quad \text{s.t.} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \\ x_i \text{ integers}$$



Variable

Constraints



Variable features (V):

- Objective coefficient c
- The number of constraints in which it appears
- ...
- **Features of the most recent best-found solutions**

Edge features (E):

- Coefficient values A

Constraint features (C):

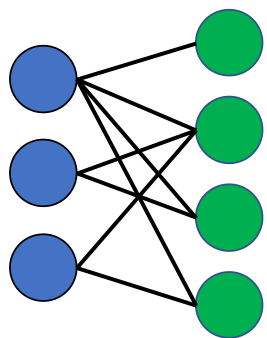
- Right-hand side b
- The number of non-zero coefficients
- Average coefficient values

...

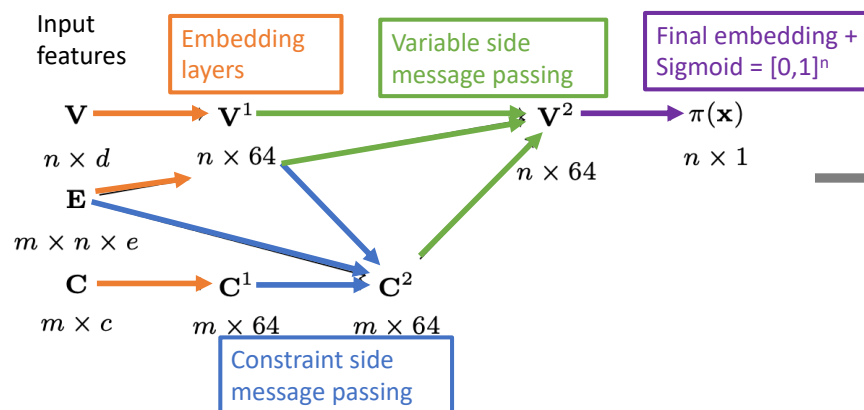
CL-LNS: Neural Network Architecture (permutation invariant model)

Bipartite
Graph and
Features
 $x=(V,E,C)$

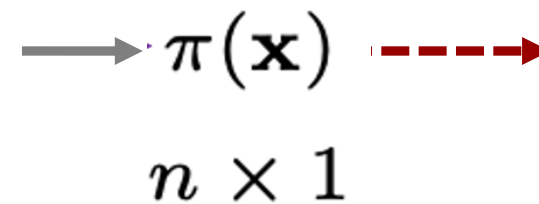
-Variable nodes -Constraint nodes



Graph Attention
Network

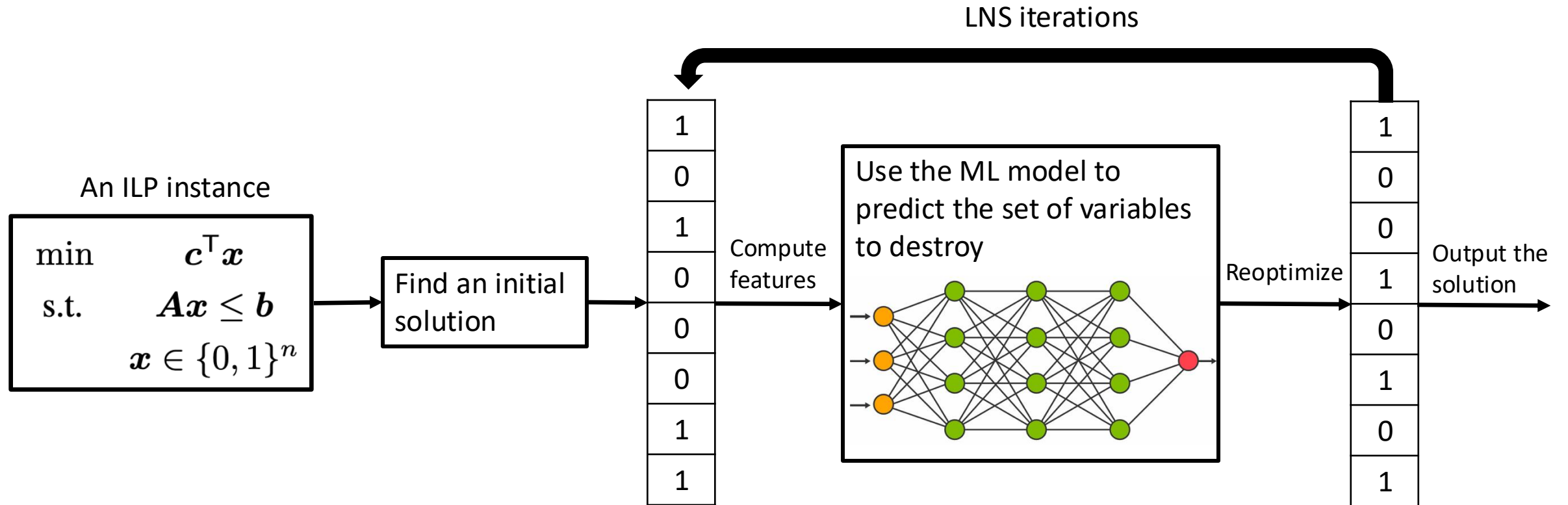


Neural
Network
Output



The final output is a score in $[0,1]$ indicating how promising it is to select each variable.

CL-LNS: Testing

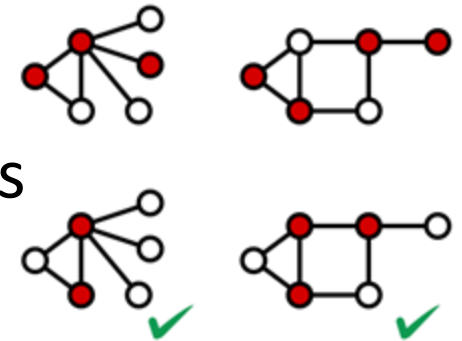


Test Time: Greedily choose the k variables with max scores from the ML model output $\pi(\mathbf{x})$ as destroy set for LNS move and reoptimize sub-MIP

Problem Domains

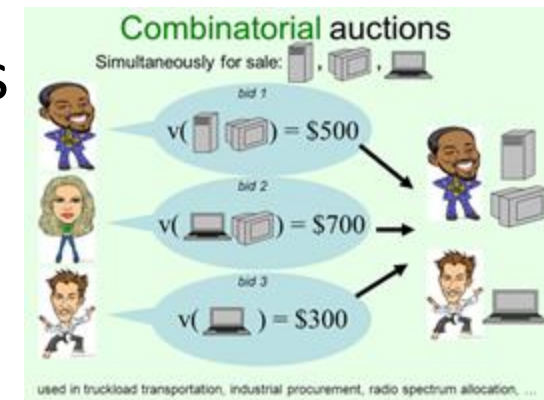
MVC: Weighted Minimum Vertex Cover

“Small” instance: 1,000 variables and 65,100 constraints



CA: Combinatorial Auctions

“Small” instance: 4,000 variables and 2,675 constraints



- **Small** instance: Training (1024) and testing (100)
- **Large** instance: Testing only (100 instances)
 - **2x** variables and **~2x** constraints

Results on two more domains (MIS, SC) in our paper

Baselines

- **BnB**: Branch and Bound using SCIP solver
- **RANDOM**: LNS with random neighborhood selection
- **LB-RELAX**: LNS with local branching relaxation heuristics [Huang et al., CPAIOR2023]
- **IL-LNS**: SOTA imitation learning approach [Sonnerat et al., 2021]
- **RL-LNS**: SOTA reinforcement learning approach [Wu et al., 2021]

vs.

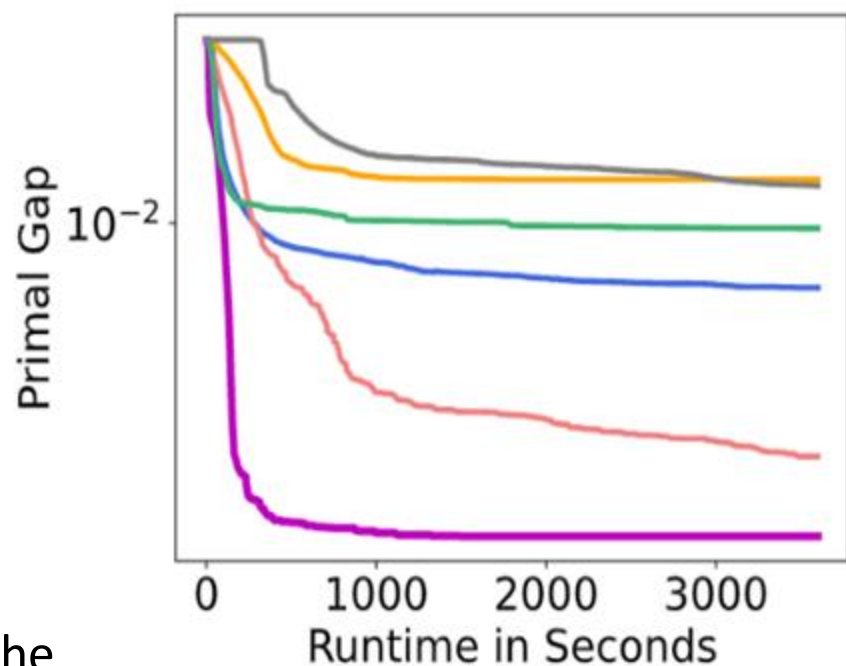
- **CL-LNS** (ours)

[Huang et al., 2023] Local Branching Relaxation Heuristics for Integer Linear Programs. CPAIOR

[Sonnerat et al., 2021] Learning a large neighborhood search algorithm for mixed integer programs. Arxiv.

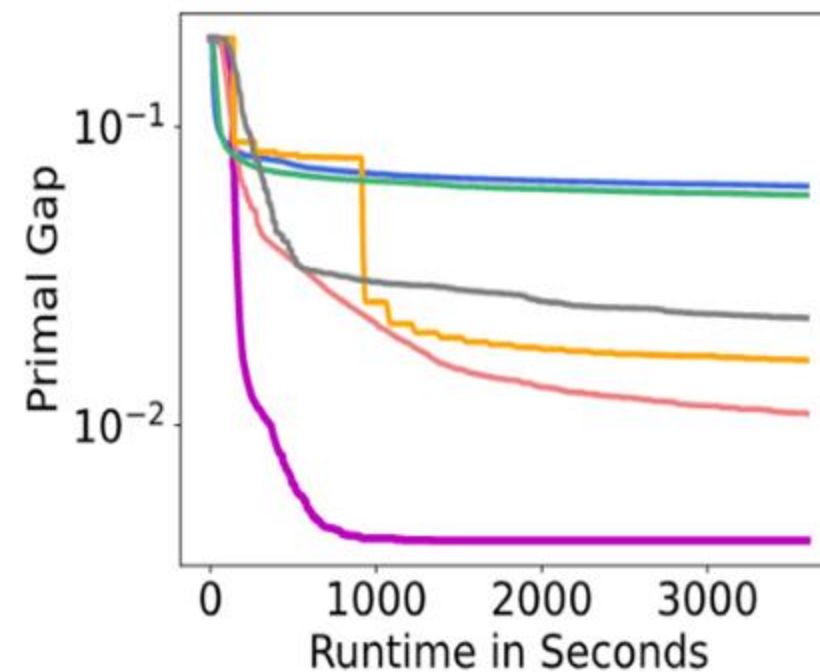
[Wu et al., 2021] Learning large neighborhood search policy for integer programming. NeurIPS

Results: Primal Gap over Time



Min Vertex Cover-Small

n=1,000 m=65,100



Combinatorial Auctions-Small

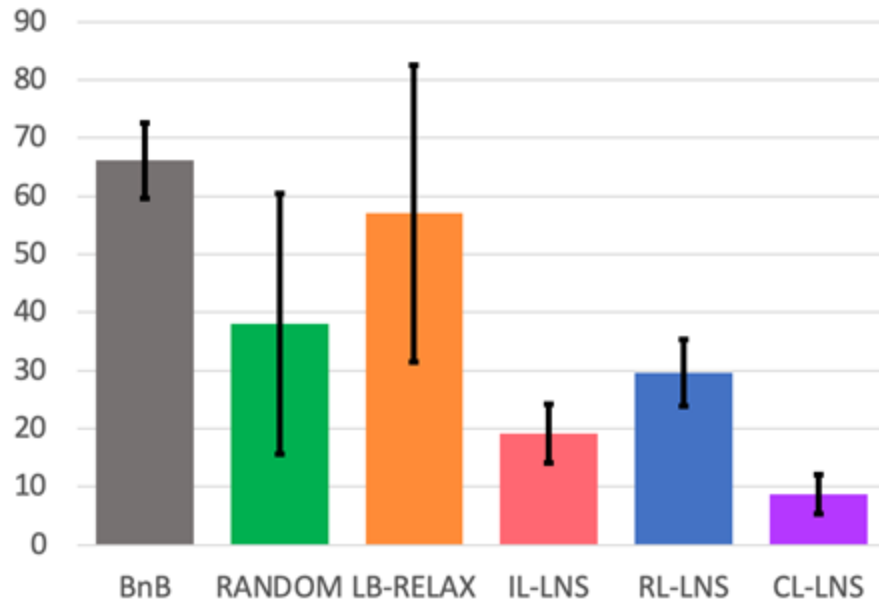
n=4,000 m= 2,675

Primal Gap:
How far away the
solution is from the
best known one

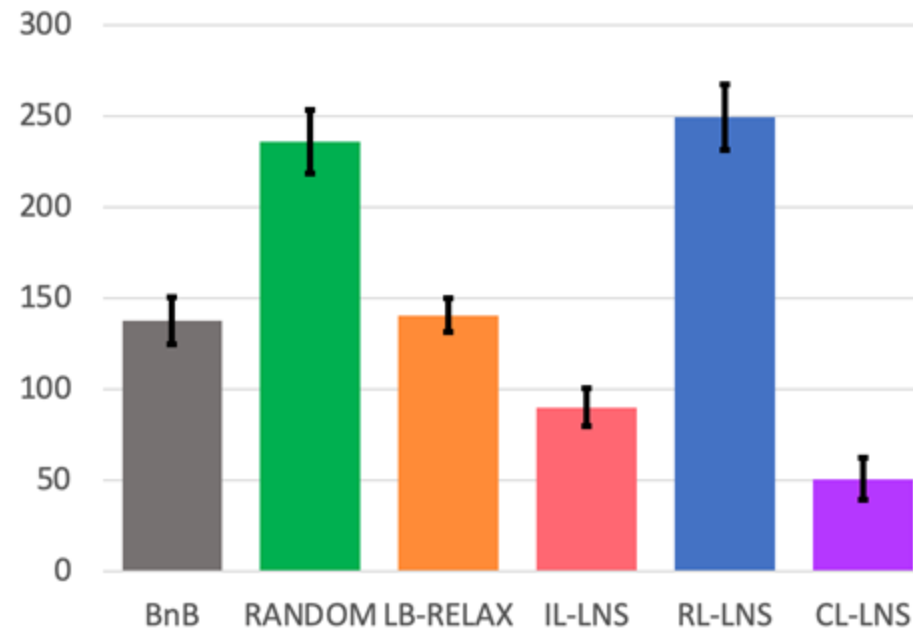
Results: Primal Integral

60 minutes - **Small** instances

Min Vertex Cover-**Small**



Combinatorial Auctions-**Small**



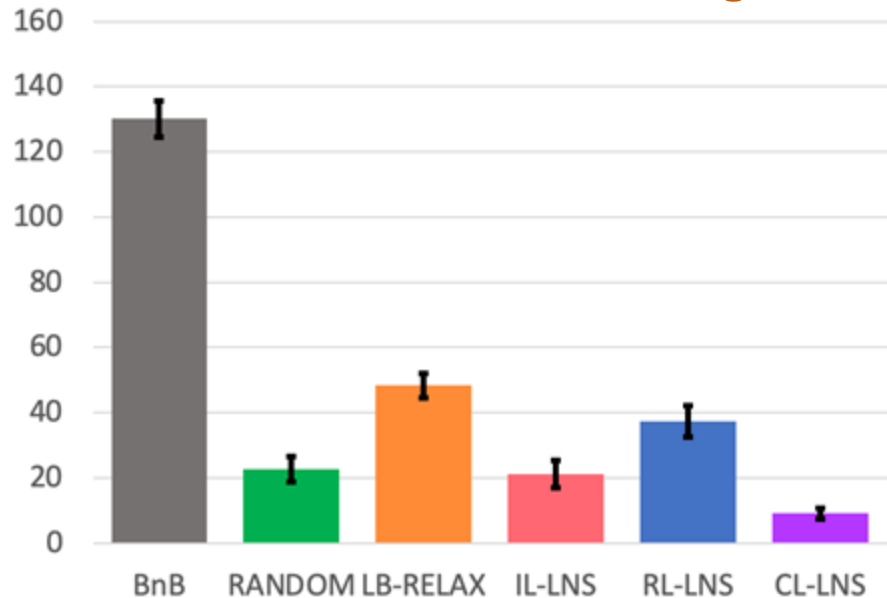
The **primal integral** at time q is the integral on $[0, q]$ of the primal gap as a function of runtime.

Captures the quality of and the speed at which solutions are found.

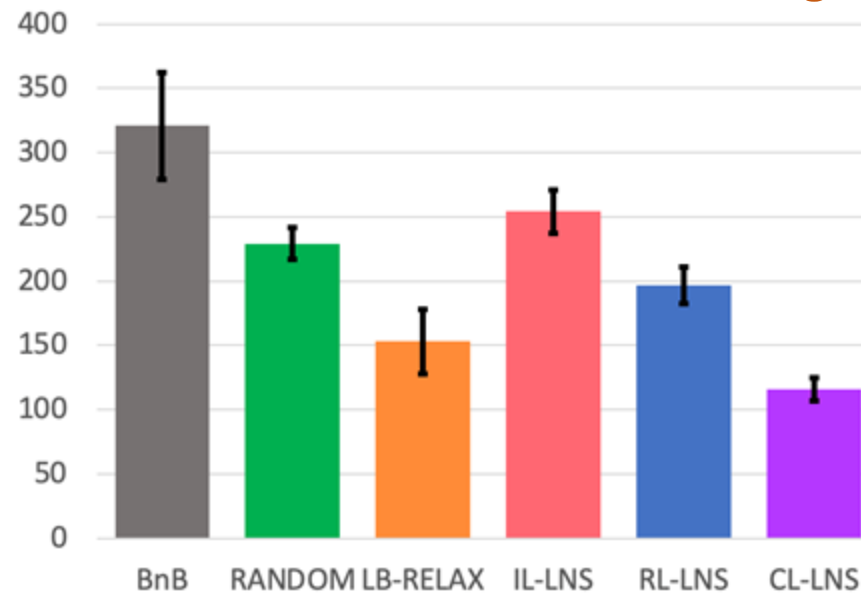
Results: Primal Integral (Generalization)

60 minutes - **Large** instances

Min Vertex Cover-Large****



Combinatorial Auctions-Large****



The **primal integral** at time q is the integral on $[0, q]$ of the primal gap as a function of runtime.

Captures the quality of and the speed at which solutions are found.

Outline

Contrastive learning for **LNS-MIP**

- gives **26% - 69% better rate of improvement (primal integral)** than the second-best approach (primal integral at 60 minutes) among non-ML, Imitation Learning and RL SOTA approaches
- strong **generalization to larger instances: up to 57% better** than second-best

Contrastive loss framework for MILP solving

- Predict-and-Search [ICML 2024]
- Backdoors for branching priority in BnB [ECAI 2024]

A contrastive-loss framework for ML-guided MILP solving

Identify a decision to improve in the search algorithm

Define positive and negative samples of the decisions: design algorithms to compute them

Data collection:
Run the search.
Compute positive and negative samples and record corresponding features

Contrastive loss:
Learn a model to make discriminative predictions

Deploy: ML-guided Search



- Positive = Good decisions (similar to imitation learning)
- Negative = Bad decisions



Imitate positive samples (similar to imitation learning)
But also pull the prediction away from negative samples

Predict-and-Search

Given a binary MILP

1. Select a subset of variables and fix them to 0
2. Select a subset of variables and fix them to 1
3. Solve for the unfixed variables (a reduced-size MILP)
 - When solving, allow changing $\Delta \geq 0$ previously-fixed variables (aka trust region)

Predict-and-Search with Imitation Learning

[Han et al., ICLR 2023]

Imitation Learning:

- Use optimal and near-optimal solutions as the expert demonstration
- Predict each variable as 0 or 1
- Use a binary classification loss

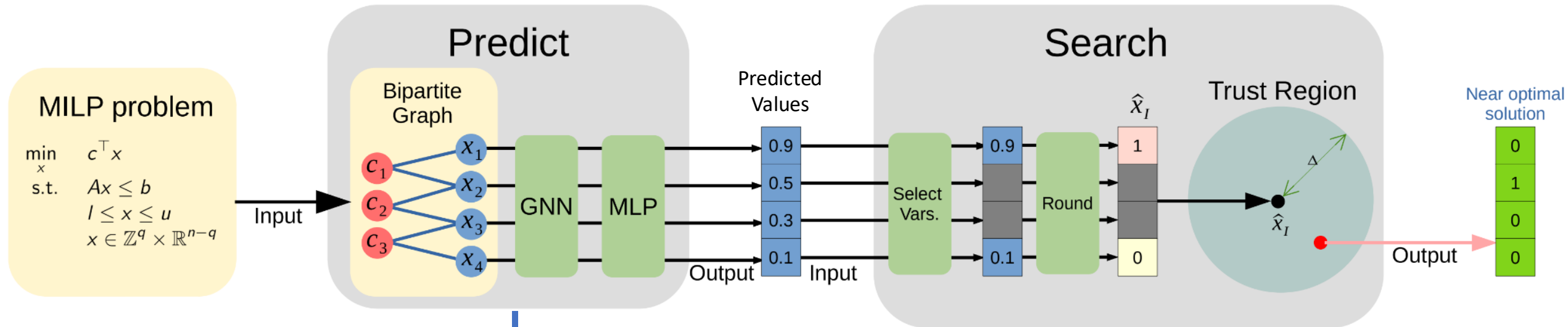
The smaller the objective of solution j , the higher the weight in the loss (assuming minimization problems)

$$L(\theta) = - \sum_{i=1}^N \sum_{j=1}^{N_i} w_{ij} \log p_{\theta}(\underline{x^{i,j}} | \underline{M_i})$$

Assignment j of instance M_i Problem Instance M_i

Learn from the data of all feasible solutions discovered during BnB (not necessary the optimal ones)

Predict-and-Search with Contrastive Learning

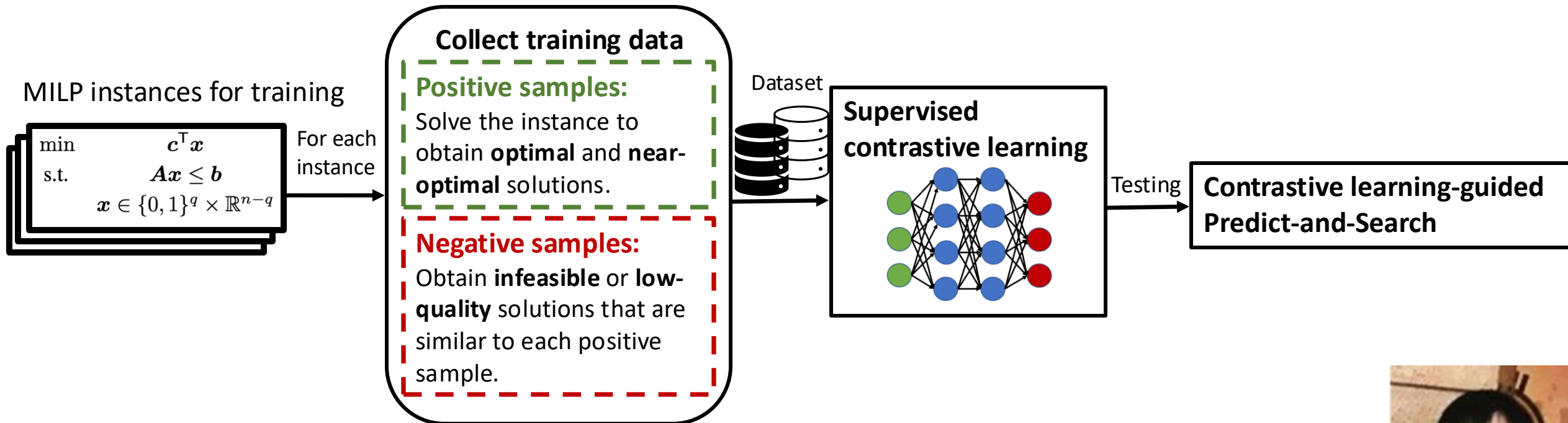


Our work

Use **contrastive loss** to make better variable assignment predictions

ConPaS: Contrastive Predict-and-Search

[Taoan Huang et al. ICML 2024] Contrastive Predict-and-Search for Mixed Integer Linear Programs



Positive and Negative Samples

Positive Samples

- Optimal solutions
- Near-optimal solutions

Negative Samples

- (Inf) Infeasible solutions that are similar to optimal ones
 - Obtained by random perturbation
- (LQ) Low-quality solutions that are similar to optimal ones
 - Obtained by solving a **novel minimax optimization problem**

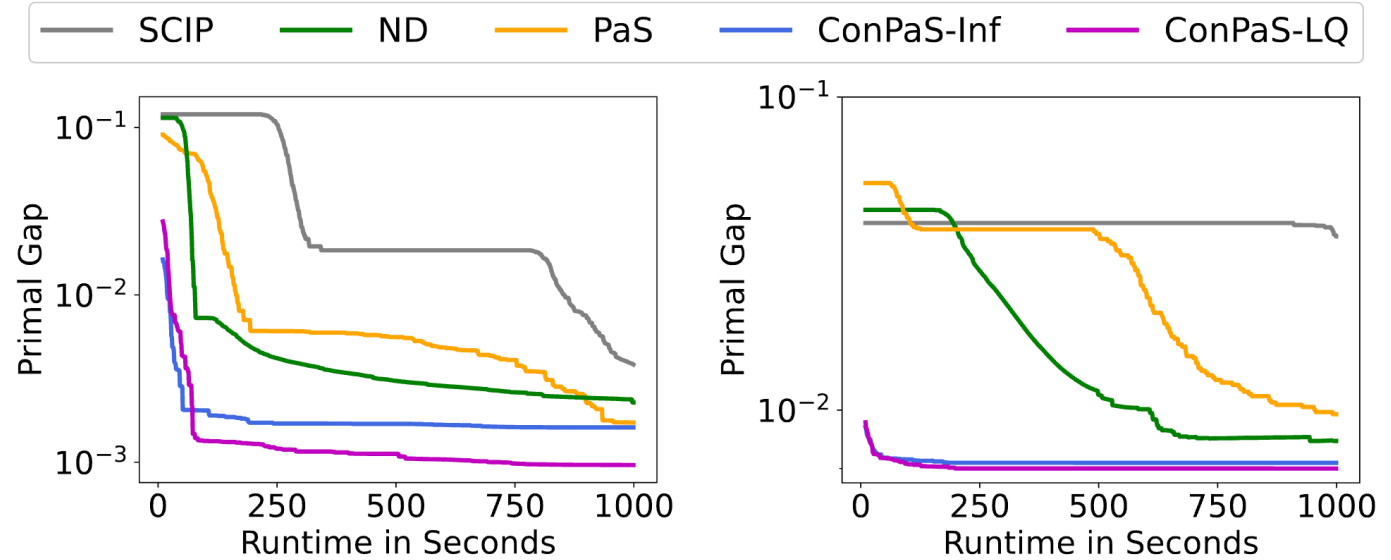
Results

Baselines

- **SCIP**: Branch and Bound using SCIP solver
- **ND**: Neural Diving [Nair et al., 2021]
- **PaS**: GNN-based Predict-and-Search [Han et al., 2023]

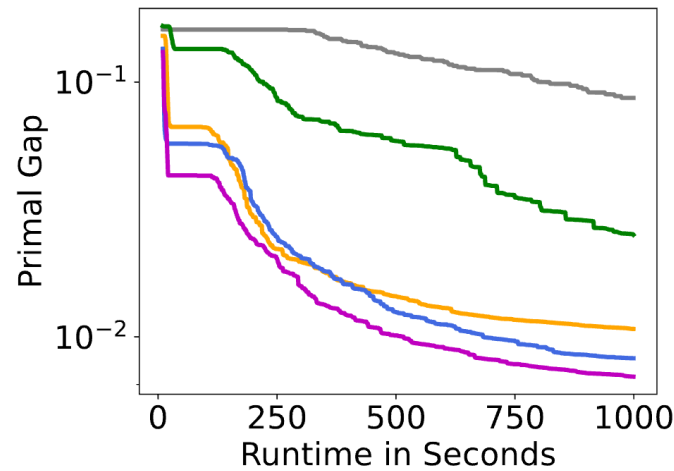
vs.

- **ConPaS-Inf** (ours)
- **ConPaS-LQ** (ours)

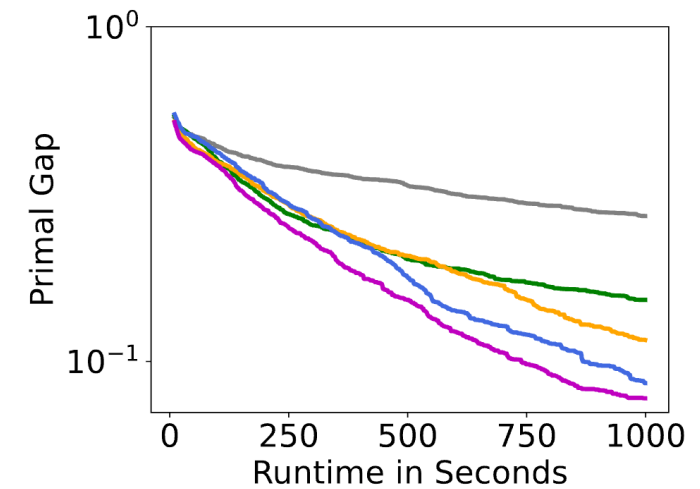


(a) MVC.

(b) MIS.



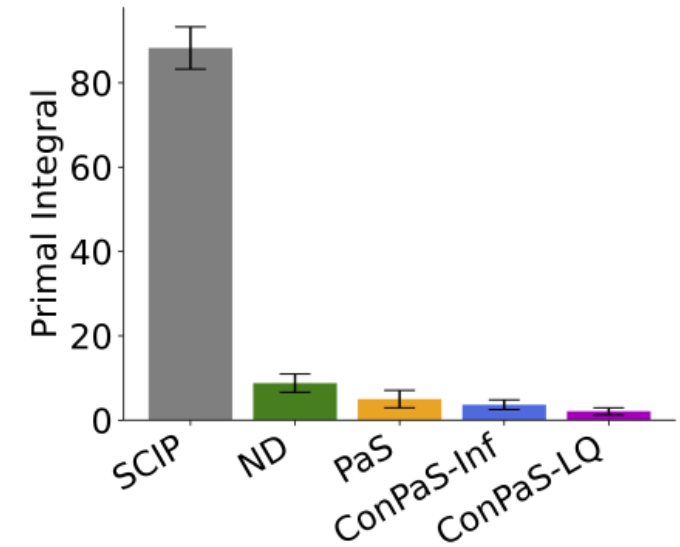
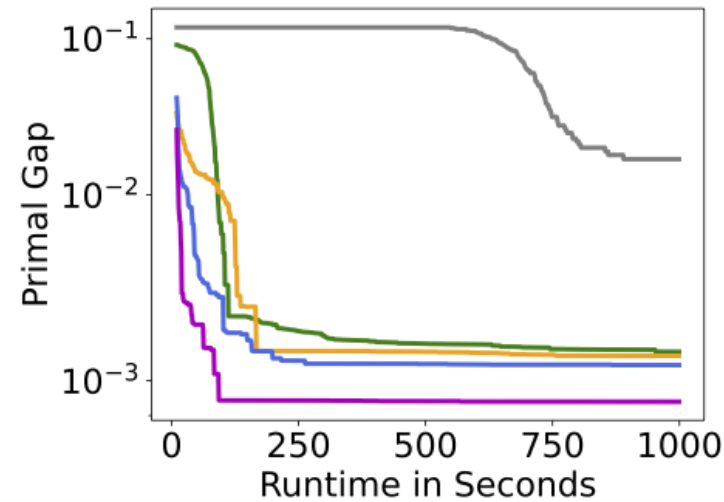
(c) CA.



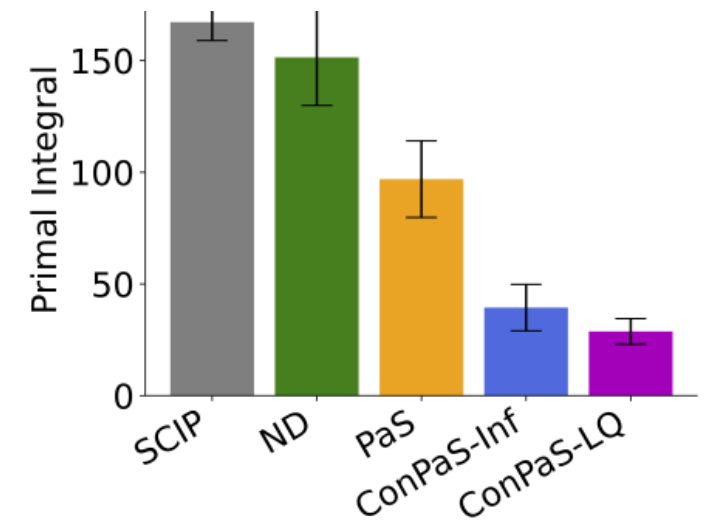
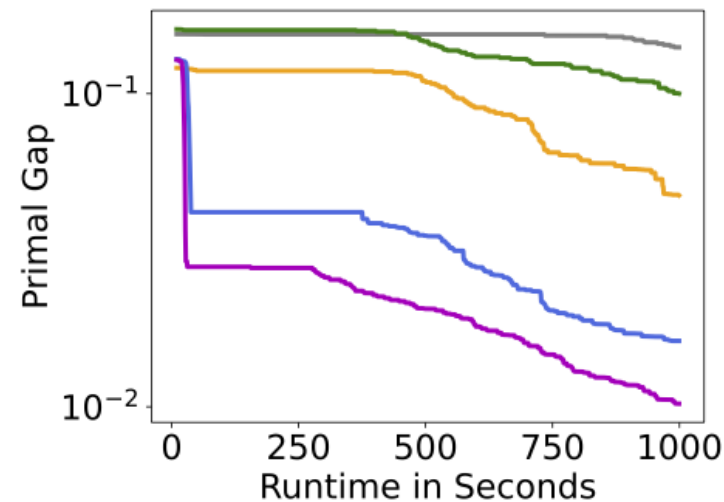
(d) IP.

Results: Generalization

Min Vertex Cover-**Large**



Combinatorial Auctions-**Large**



Primal Integral is **58% - 70% lower** than the best baseline PaS

Outline

Contrastive loss for **LNS-MIP** [ICML 2023]

Contrastive loss framework for MILP solving

- **Predict-and-Search** [ICML 2024]
 - Novel strategy for effective negative samples
 - **19%-76%** lower primal integral than second best in distribution
 - **58-70%** lower primal integral than second best when generalizing to larger
- **Backdoors for branching priority in BnB [ECAI 2024]**

Backdoors

Backdoor = a set of key decision variables such that searching over them is enough to solve the problem

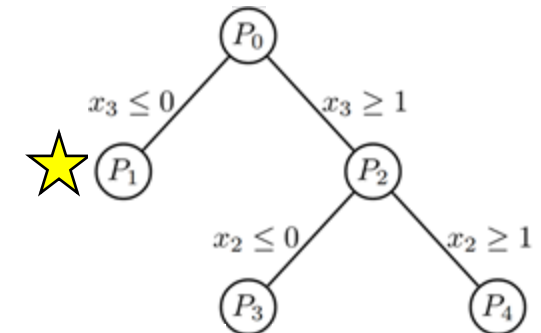
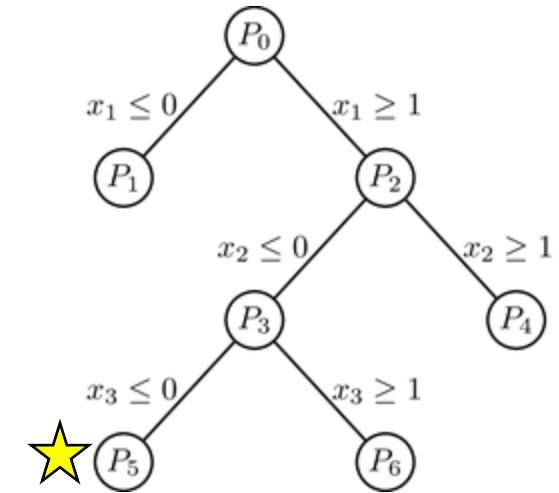
- first introduced in the context of SAT [Williams et al, 2003]

Strong backdoors for MIP

- Subset of decision variables \mathcal{B}
- Branching on only this subset yields optimal solution
- generalized to MIP [Dilkina et al, 2009]

Goal: Predict pseudo-backdoors

- Learn to predict a set of key decision variables
- such that prioritizing branching on these variables will lead to improving solving time



Previous Work: Backdoors

SAT/CSP: first introduced, with empirical limitations

- Williams et al. IJCAI 2003
- Paris et al. ICTAI 2006
- Kottler et al. SAT 2008

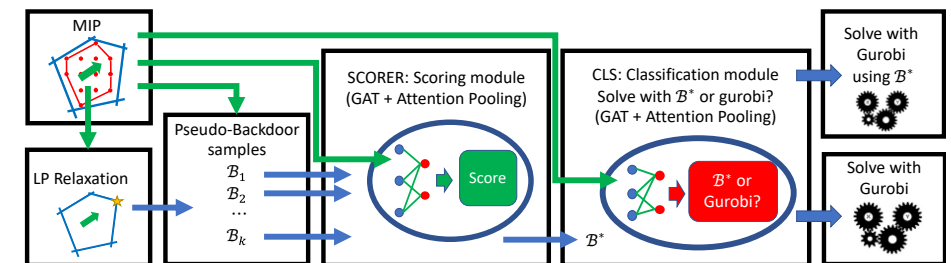
MIP Sampling backdoors

- Dilkina et al. CPAIOR 2009
- Fiscetti et al. IPCO 2011
- Dvořák et al. IJCAI 2017
- Khalil et al. AAAI 2022

Overall: core sets of variables potentially exist
+ they can be used for fast MIP solving

Learning Pseudo-Backdoors for Mixed Integer Programs. Aaron Ferber, Jialin Song, Bistra Dilkina, Yisong Yue. CPAIOR 2022

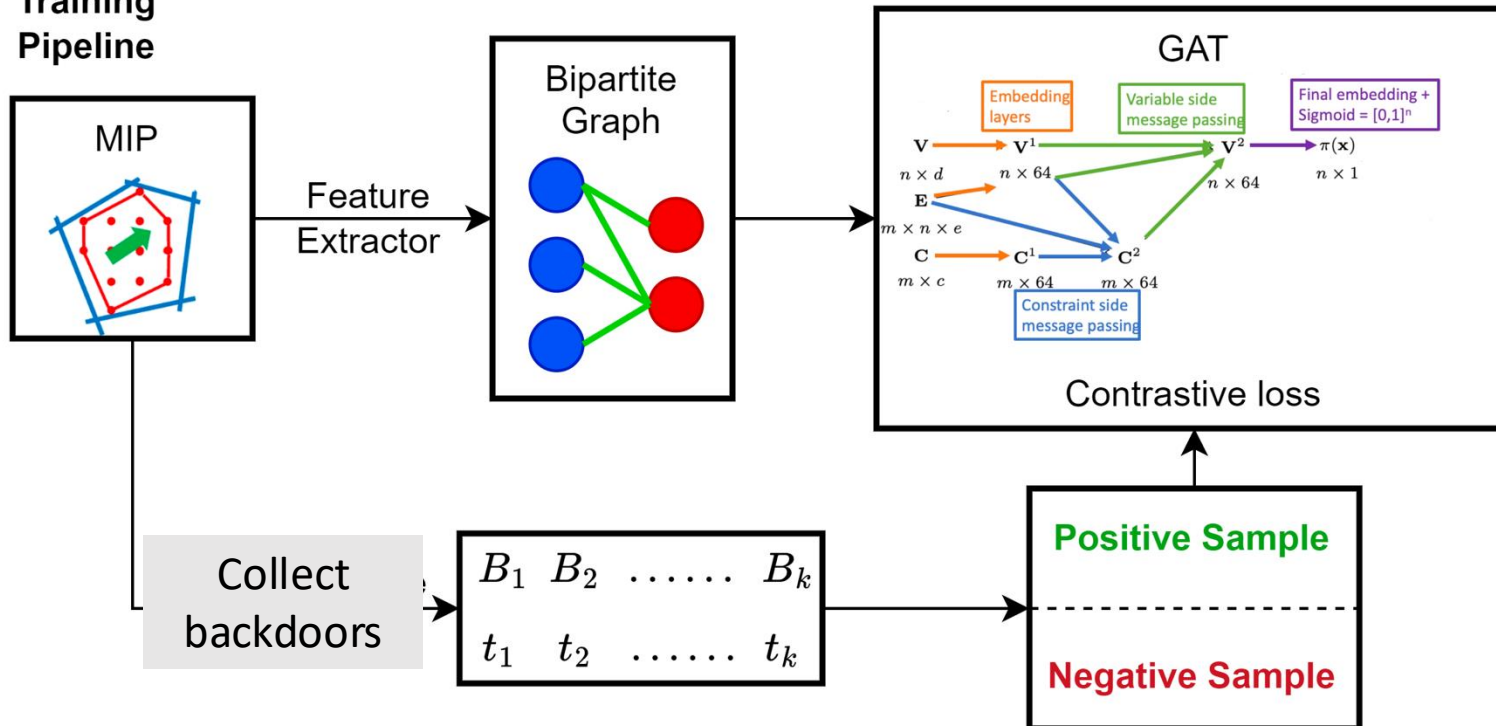
- First ML approach to Backdoor prediction
- Sample backdoor candidates and Rank
- Set branching priority based on highest- ranked backdoor candidate [with reject option]



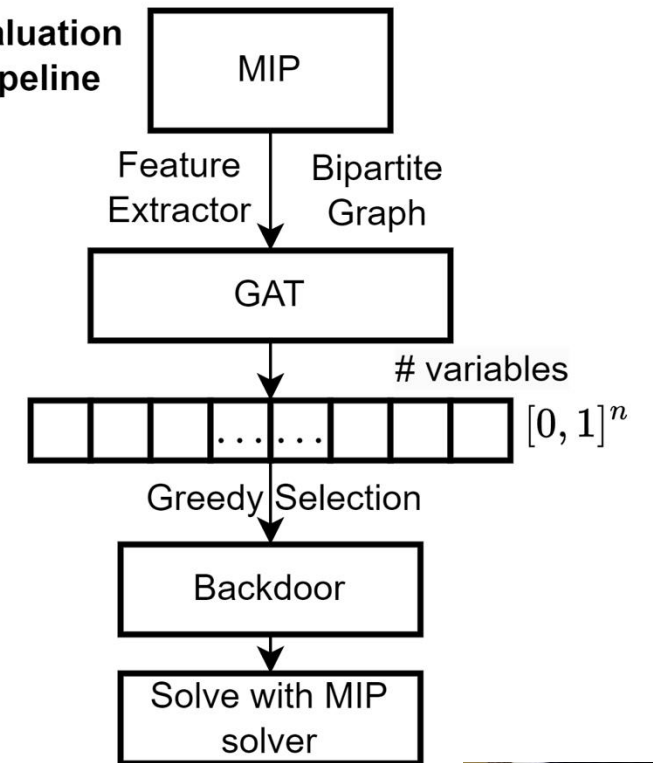
CL-MIP-Backdoor Pipeline

[Junyang Cai et al. ECAI 2024] Learning Backdoors for Mixed Integer Programs with Contrastive Learning

Training Pipeline



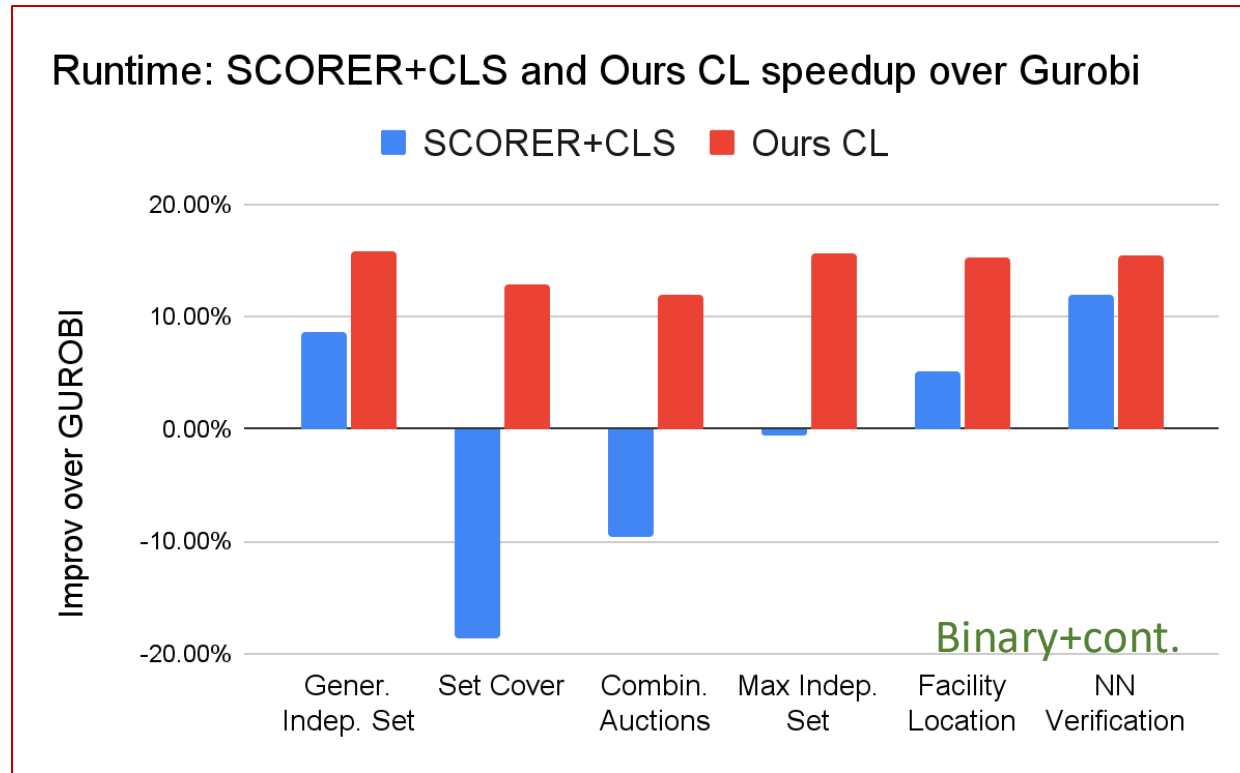
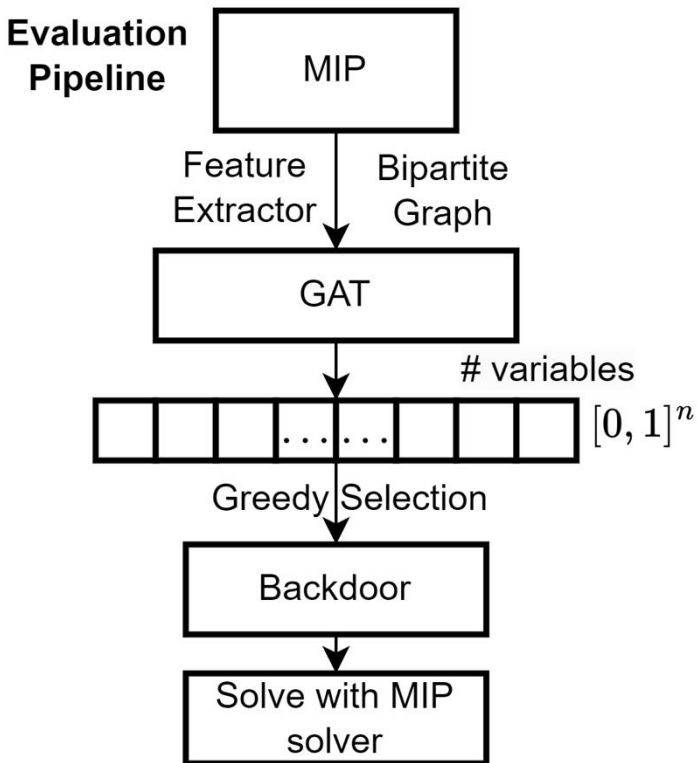
Evaluation Pipeline



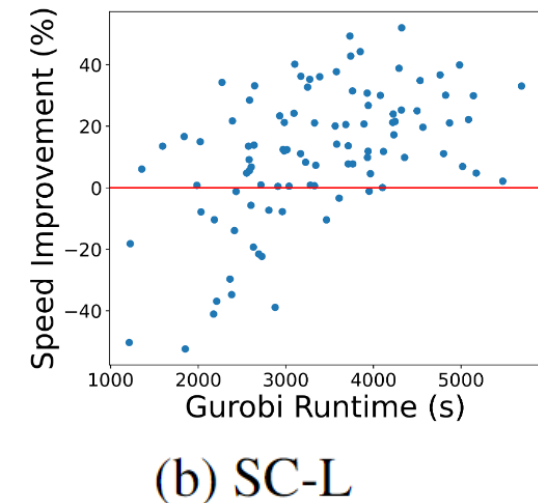
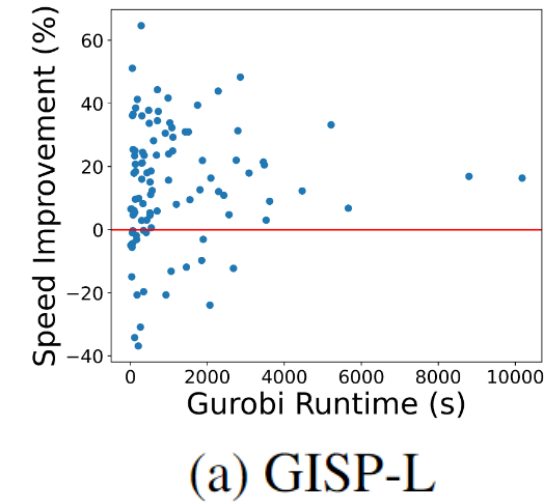
Training data collection (for each MIP)

1. Collect candidate backdoors using MCTS [Khalil et al., AAAI2022]
 2. Select the top k candidate backdoors S with the highest tree weight
 3. Solve MIP with each backdoor to obtain BnB runtime
- Positive samples $S_p = p$ shortest-runtime backdoors in S
 - Negative samples $S_n = q$ longest-runtime backdoors in S

Deploy: use ML model to guide solver on unseen MIPs

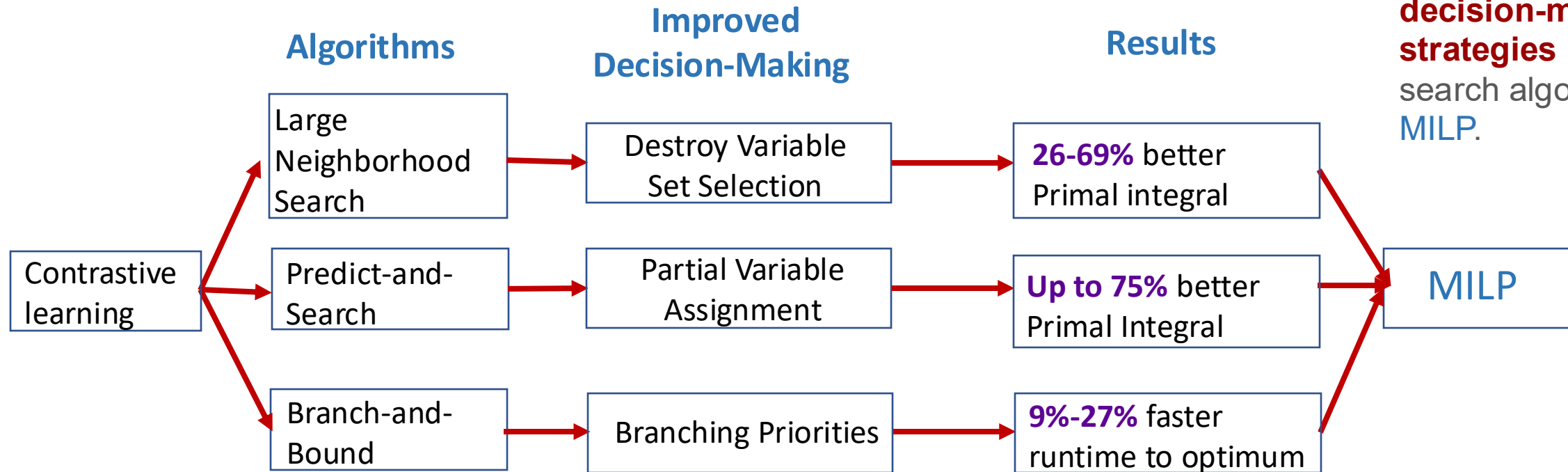


Performance: ~15% improvement over Gurobi on 6 problem domains
CL model gives speed-ups on **hardest** instances in each distribution



Contrastive Loss for ML-guided MIP

Contrastive loss machine learning framework to improve **decision-making strategies** in different search algorithms for **MILP**.



Can we do something more synergistic?

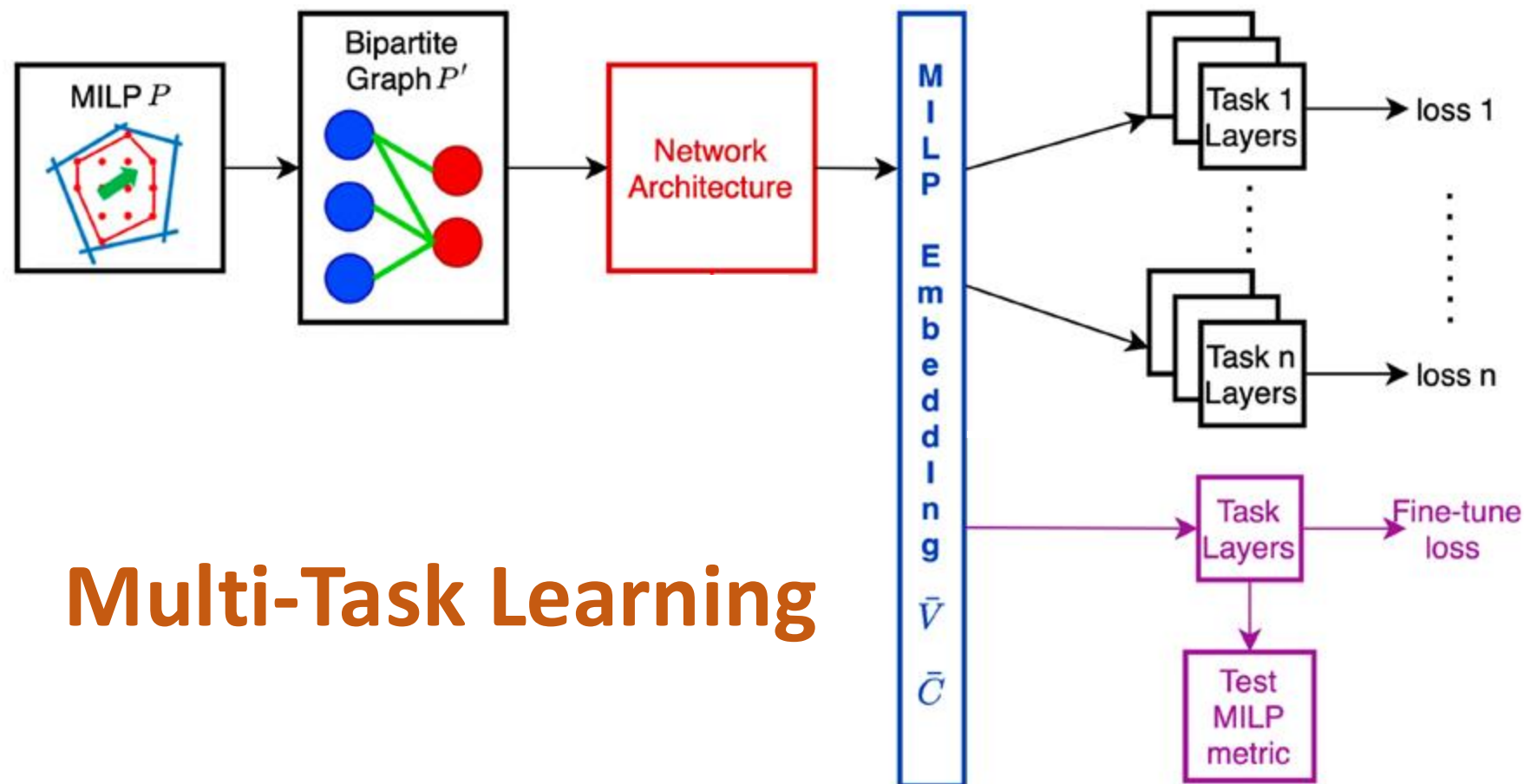
Mixing algorithmic tasks

[Junyang Cai et al, CPAIOR 2025] Multi-task Representation Learning for Mixed Integer Linear Programming

Success of ML-guided MIP solving for:

- LNS
- Predict-and-Search
- BnB Branching
- Parameter configuration
- BnB Primal Heuristics
- BnB Cut selection

BUT each time we start from scratch with data collection and training

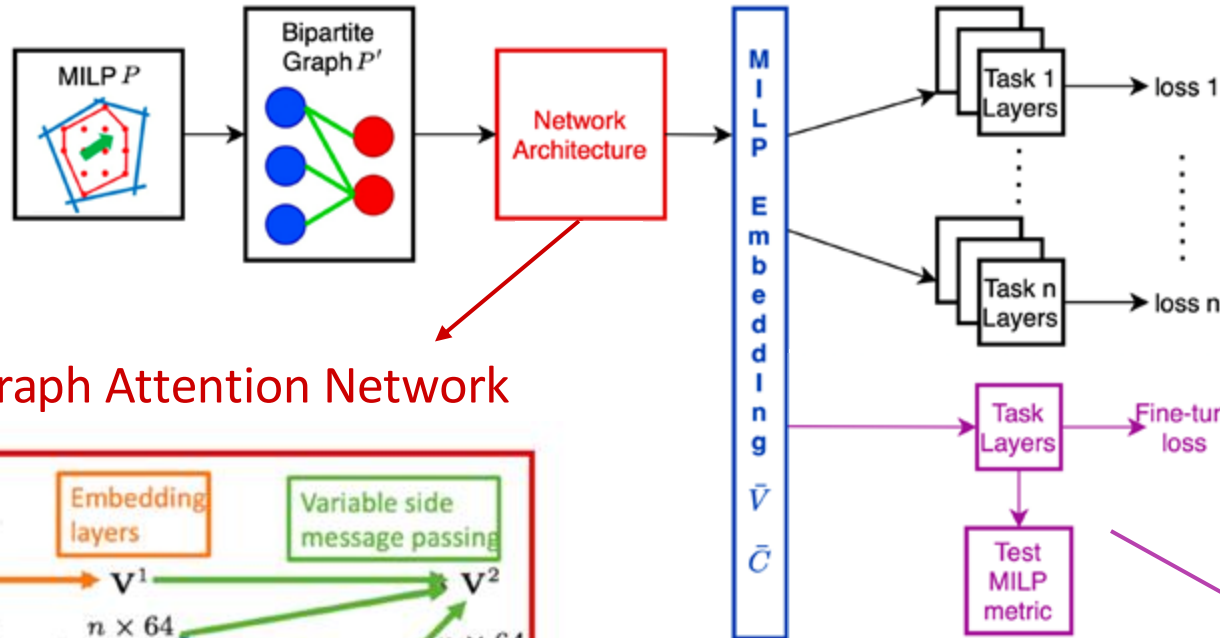


Multi-Task Learning

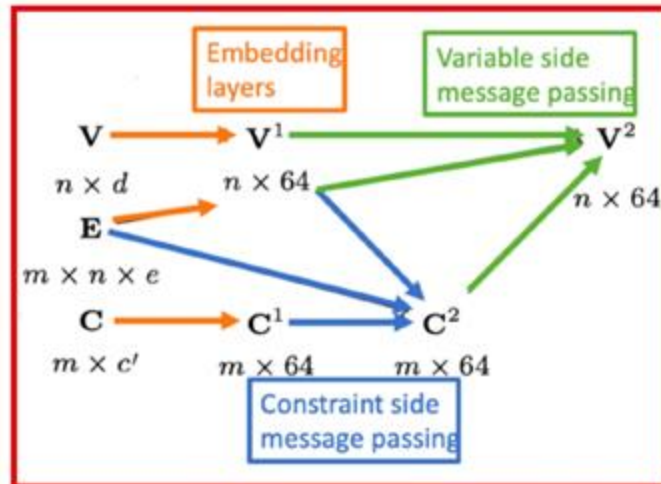
Step 1: Training network architecture with multiple fixed output layers for each task

Step 2: Fine-tune only the task-specific layers

Multi-task Representation Learning

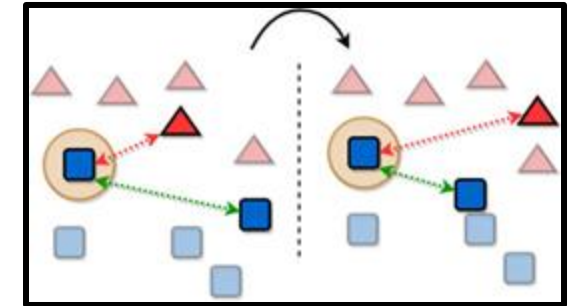


Graph Attention Network

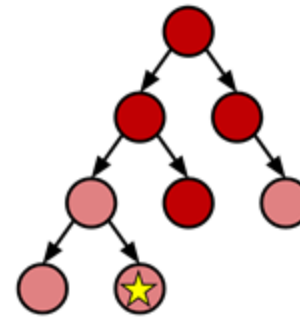


Step 1: Training network architecture with multiple fixed output layers for each task

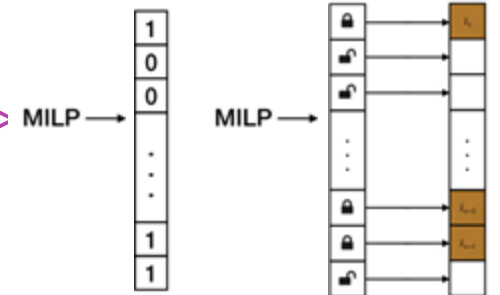
Contrastive loss



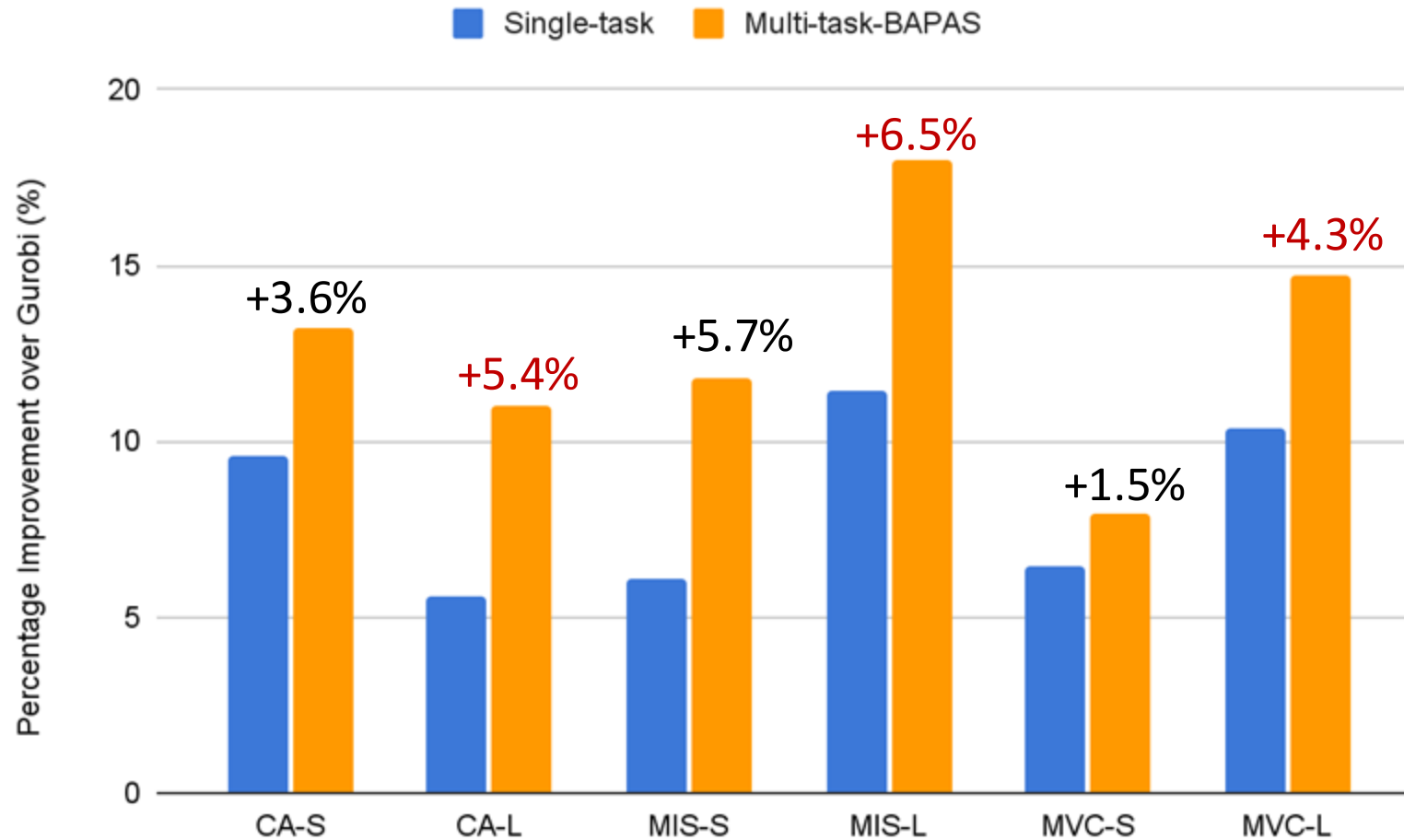
<-Backdoor
Predict-and-Search->
<-Configuration



Step 2: Fine-tune only the task-specific layers



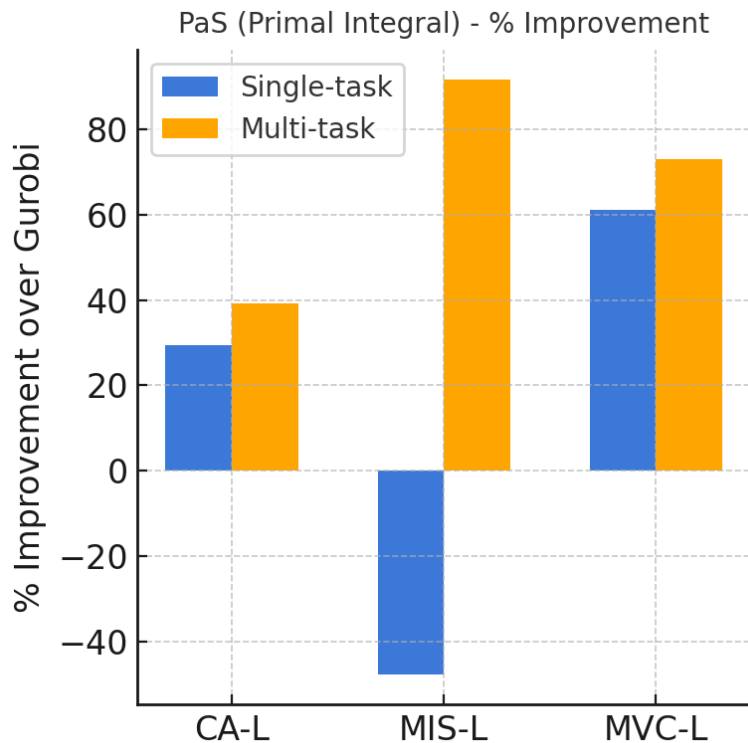
Train (Backdoors, PaS) – Test (Backdoors)



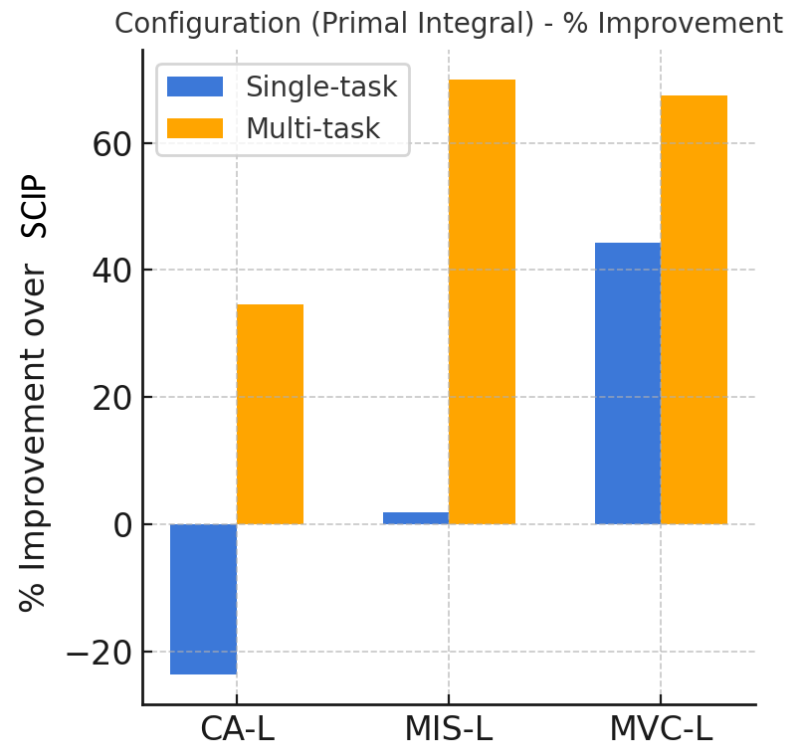
- **Better** performance In distribution
- **Even better** performance when generalization on size

Generalization to new tasks

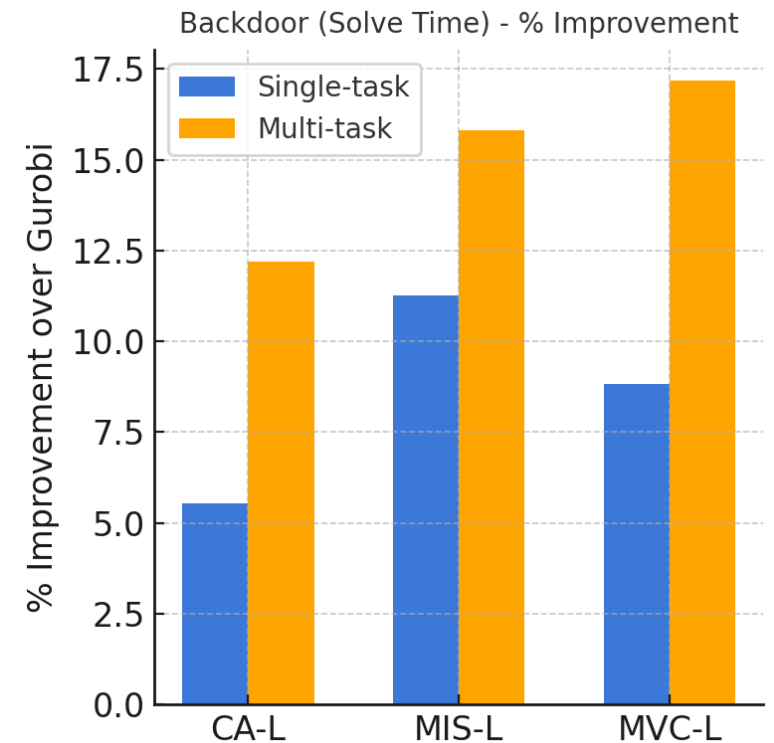
- **Single task**: train and test on same task
- **Multi-task**: Train on 2 tasks, fine-tune and test on a **new task**
- Performance on **larger instance size** then trained/finetuned



Predict-and-Search + Gurobi



Configuration + SCIP

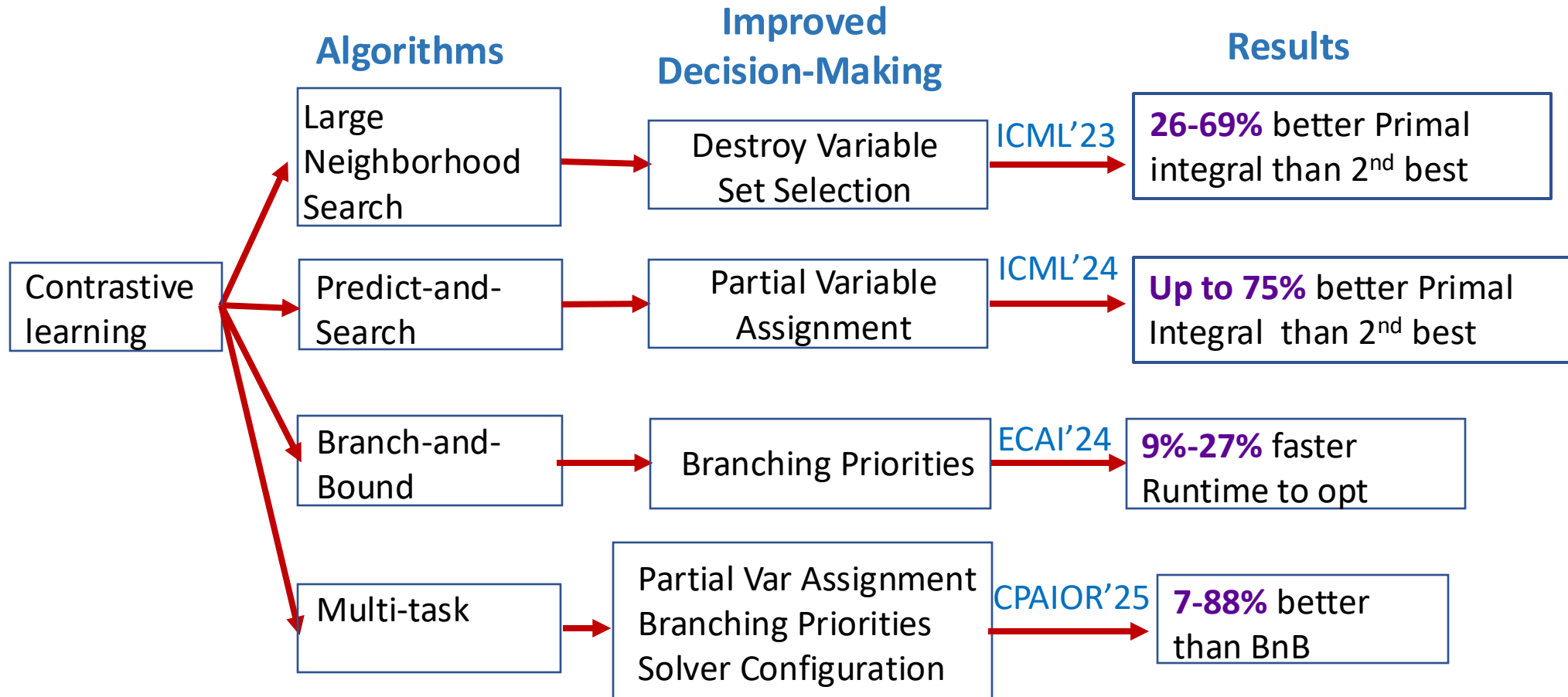


Backdoor + Gurobi

ML-guided linear MIP via contrastive loss learning

improve **decision-making strategies** in different search algorithms for MILP.

$$\begin{array}{lll} \min_x & c^T x & \text{objective} \\ \text{s.t.} & Ax \leq b & \text{constraints} \\ & x_j \in \mathbb{Z} \forall j \in \mathcal{J} & \text{integrality} \end{array}$$



Key challenge:
benchmarks and datasets

Datasets for ML-guided MILP solving

Weimin Huang et al, ArXiv 2025]

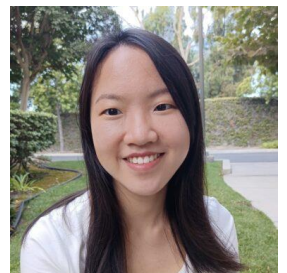
Distributional MIPLIB: a Multi-Domain Library for Advancing ML-Guided MILP Methods

Dataset Limitations in Existing Work:

- **Heterogenous (e.g, MIPLIB)**
 - Not ideal for ML methods
- MILPs distributions but independently generated with **different parameters for the same domains**
 - Hard to benchmark
- A small set of **synthetic problem domains** repeatedly used
 - Lack of evaluation on real-world problems

Need for standardized train and test sets across hardness levels
Need for a richer set of problem domains

Distributional MIPLIB:
a Multi-Domain Library
for Advancing
ML-Guided MILP Methods



8 Synthetic domains curated from existing work

1. **CA**: Combinatorial Auctions
2. **SC**: Set Covering
3. **MIS**: Maximum Independent Set
4. **MVC**: Minimum Vertex Cover
5. **GISP**: Generalized Independent Set Problem, a graph problem in forestry management
6. **CFLP**: Capacitated Facility Location Problem
7. **LB**: Load Balancing deals with apportioning workloads across workers
8. **IP**: Item Placement involves spreading items across containers to utilize them evenly

Domain	Hardness Level	Dist. Source: ML4MILPs	Instance Statistics				Performance Metrics			
			# Var B	# Var I	# Var C	# Constr	# Opt	Opt Time(s)	NonOpt Gap	Primal-dual Integral
		Synthetic								
CA [†]	Easy	Gasse et al. (2019)	1000	0	0	385.04	100	47.14	N/A	2.30
	Medium	Gasse et al. (2019)	1500	0	0	578.07	100	358.14	N/A	7.29
	Very hard	Huang et al. (2023)	4000	0	0	2676.32	0	N/A	0.10	400.28
SC [†]	Easy	Gasse et al. (2019)	1000	0	0	500	100	18.05	N/A	0.99
	Medium	Gasse et al. (2019)	1000	0	0	1000	100	214.11	N/A	15.78
	Hard	Gasse et al. (2019)	1000	0	0	2000	56	1603.66	0.04	180.25
	Very hard	Huang et al. (2023)	4000	0	0	5000	0	N/A	0.20	847.11
MIS [†]	Easy	Gasse et al. (2019)	1000	0	0	3946.25	100	50.52	N/A	0.86
	Medium	Gasse et al. (2019)	1500	0	0	5941.14	88	470.44	0.01	11.28
	Very hard	Huang et al. (2023)	6000	0	0	23994.82	0	N/A	0.30	1132.69
MVC [†]	Easy	New	1200	0	0	5975	100	27.26	N/A	0.27
	Medium	New	2000	0	0	9975	97	244.11	0.01	2.28
	Hard	New	500	0	0	30100	55	1821.04	0.02	102.74
	Very hard	Huang et al. (2023)	1000	0	0	65100	0	N/A	0.12	454.02
GISP [†]	Easy	New	605.81	0	0	1967.05	100	43.09	N/A	15.59
	Medium	Ferber et al. (2022)	988.81	0	0	3353.03	100	671.89	N/A	204.83
	Hard	Ferber et al. (2022)	1317.03	0	0	4567.83	85	2623.16	0.08	866.16
	Very hard	Cai et al. (2024)	6017	0	0	7821.87	0	N/A	0.44	2104.04
	Ext hard	Khalil et al. (2017)	12675.83	0	0	16515.44	0	N/A	2.01	8139.33
CFLP [†]	Easy	Gasse et al. (2019)	100	0	10000	10201	100	44.44	N/A	0.57
	Medium	Gasse et al. (2019)	200	0	20000	20301	100	103.51	N/A	0.88
LB [†]	Hard	Gasse et al. (2022)	1000	0	60000	64307.17	9	2665.11	0.00	33.48
IP [†]	Very hard	Gasse et al. (2022)	1050	0	33	195	0	N/A	0.44	1770.42

100 test instances, Hardness: performance of Gurobi @ 1 hour time cutoff
+ 900 train/val + generators

5 Real-world domains from critical applications

Curated from existing work in ML-guided MILP solving:

Maritime Inventory Routing Problem (MIRP)

Neural Network Verification (NNV)

New in ML-guided MILP solving:

Energy Network Optimal Transmission Switching (OTS) [Pollack et al. (2024)]

Middle-Mile Consolidation Network (MMCN) [Greening et al. (2023)]

Seismic-Resilient Pipe Network Planning (SRPN) [Huang et al. (2020)]

Domain	Hardness Level	Dist. Source: ML4MILPs	Instance Statistics				Performance Metrics			
			# Var B	# Var I	# Var C	# Constr	# Opt	Opt Time(s)	NonOpt Gap	Primal-dual Integral
Real-world										
MIRP	Medium	Gasse et al. (2022)	0	15080.57	19576.15	44429.70	10‡	697.24	0.23	728.75
NNV	Easy	Nair et al. (2020)	171.49	0	6972.60	6533.70	588‡	37.98	N/A	21.81
OTS†	Easy	New	4181	0	17137	48582	100	45.86	N/A	3.72
	Medium	New	7525	0	33202	92992	100	419.55	N/A	25.80
	Hard	New	6546	0	46423	111804	52	2564.00	0.20	1926.19
MMCN	Medium ^{BI}	New	1156.94	263.23	0	437.81	100	114.93	N/A	3.01
	Medium ^{BC}	New	4271.59	0	324.04	3171.23	100	468.17	N/A	37.30
	Hard ^{BI}	New	2074.76	346.39	0	642.57	34	1998.57	0.01	79.79
	Very hard ^{BI}	New	21596.72	1127.29	0.00	3944.01	0	N/A	0.10	369.15
	Very hard ^{BC}	New	68345.21	0	2425.87	96272.60	0	N/A	0.61	2761.52
SRPN	Easy	New	3016.42	0	3016.42	5917.27	21‡	77.91	0.02	10.00
	Hard	New	11485.33	0	11485.33	22430.84	9‡	1321.43	0.03	134.12

Sample efficiency: ML-policies with mixed distributions

- Collecting expert samples from a large number of instances is expensive
- Alternative training strategy: **pool data and train ML models on mixed distributions**

80 training instances per domain

ML-single: trained on a single domain (homogeneous)

ML-mix5: trained on a mix of instances from 5 domains

Domain	Primal-dual integral			Improv of ML-mix5 over ML-single
	SCIP	ML-single	ML-mix5	
MIS (Easy)	4.412 \pm 0.118	5.408 \pm 5.309	2.781 \pm 0.197	48.58%
GISP (Easy)	12.509 \pm 0.242	11.299 \pm 0.885	10.823 \pm 0.383	4.21%
CFLP (Easy)	0.644 \pm 0.021	0.642 \pm 0.036	0.638 \pm 0.020	0.62%
CA (Med)	2.347 \pm 0.034	1.927 \pm 0.063	1.815 \pm 0.015	5.81%
SC (Med)	6.465 \pm 0.023	5.602 \pm 0.156	5.362 \pm 0.131	4.28%

Models trained with pooled data exhibit better performance when limited training data is available

Mixed distributions & generalization

More data (320 training instances per domain)

-> Single domain training is better in-distribution

Domain	Primal-dual integral			Improv. ML-mix5 over ML-single
	SCIP	ML-single	ML-mix5	
MIS (Easy)	4.412 \pm 0.118	2.434 \pm 0.074	2.545 \pm 0.107	-4.56%
GISP (Easy)	12.509 \pm 0.242	10.700 \pm 0.442	10.420 \pm 0.279	2.62%
CFLP (Easy)	0.644 \pm 0.021	0.606 \pm 0.028	0.610 \pm 0.021	-0.66%
CA (Med)	2.347 \pm 0.034	1.775 \pm 0.056	1.795 \pm 0.199	-1.13%
SC (Med)	6.465 \pm 0.023	4.965 \pm 0.095	4.796 \pm 0.104	3.40%

Transfer ML-single and ML-mix 5 to HARDER distributions

	Domain	Primal-dual integral			Improv. ML-mix5 over ML-single
		SCIP	ML-single	ML-mix5	
Trained on MIS (Easy) →	MIS (Medium)	23.4 \pm 0.1	21.9 \pm 2.4	16.5 \pm 0.2	24.66%
	MIS (Very hard))	1479.3 \pm 2.3	1461.5 \pm 4.8	1459.0 \pm 2.4	0.17%
Trained on SC (Medium) →	SC (Hard)	53.3 \pm 0.2	49.6 \pm 0.4	48.2 \pm 0.3	2.82%
	SC (Very hard))	767.0 \pm 1.0	870.3 \pm 13.6	830.3 \pm 19.5	4.60%

Models trained with pooled data exhibits better generalization to harder instances

Conclusion

Large Neighborhood Search (LNS-MIP) via contrastive loss

- The first use of contrastive loss in ML-guided MIP solving

Contrastive loss as a unifying paradigm for ML-guided MIP

- Significant gains over other approaches in PaS and Backdoors

Multi-task Learning for ML-guided MIP solving

- **superior generalization performance** on problem **size** and unseen **tasks**
- First step to **Foundational model** for ML-guided MIP solving

Benchmarking: Distributional MIPLIB

- First standardized **benchmarks + real-world** domains
- Highlights outstanding **research challenges** such as GNN inference overhead
- Demonstrate value of **cross-domain ML training** for sample efficiency and generalization

Papers

1. [ICML 2023] Huang, Taoan; Ferber, Aaron; Tian, Yuandong; Dilkina, Bistra; Steiner, Benoit; "Searching **Large Neighborhoods** for Integer Linear Programs with Contrastive Learning"
2. [ICML 2024] Taoan Huang, Aaron M Ferber, Arman Zharmagambetov, Yuandong Tian, Bistra Dilkina. "Contrastive **Predict-and-Search** for Mixed Integer Linear Programs"
3. [ECAI 2024] J Cai, T Huang, B Dilkina. "Learning **Backdoors** for Mixed Integer Linear Programs with Contrastive Learning"
4. [CPAIOR 2025] J Cai, T Huang, B Dilkina. "**Multi-task** Representation Learning for Mixed Integer Linear Programming"
5. [Arxiv] Weimin Huang, T. Huang, A. Ferber, A. and B Dilkina. **Distributional MIPLIB**: a Multi-Domain Library for Advancing ML-Guided MILP Methods. AAAI-2025 Combining AI and OR/MS for Better Trustworthy Decision Making – Bridge Program. arXiv preprint arXiv:2406.06954.

ML ↔ Combinatorial Optimization

- ▶ Exciting and growing research area
- ▶ Design discrete optimization algorithms with learning components
- ▶ Learning methods that incorporate the combinatorial decision making they inform

Thank you!



AI4OPT

