Contrastive Learning for ML-guided MIP search

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

Feb 26, 2024

At AAAI-24 Workshop "LEARNOPT"

NSF AI Institute for Advances in Optimization (AI4OPT)

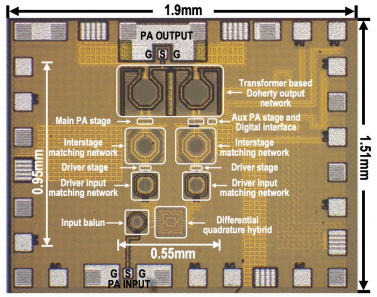
with Georgia Tech, Berkeley, Clark Atlanta University, University of Texas at Arlington; and Spelman College

Objective: aims at delivering a paradigm shift in automated decision making at massive scales by fusing AI and Mathematical Optimization (MO), to achieve breakthroughs that neither field can achieve independently.

Ubiquity of optimization problems in society, larger scales, complex settings

Challenge: *ability to move from optimization solutions to intelligent systems that predict and quantify uncertainty, reason and optimize, learn continuously, and coordinate and collaborate.*

End-Use AI Applications



Logistics and Supply Chains

e.g.: multi-agent coordination in warehouses, last mile delivery

Hardware design and control

e.g.: Optimization and ML play a key role in both the pre-silicon design and post-silicon control of circuits

Resilience and Sustainability

e.g.: disaster mitigation planning, wildfire management, energy

Relevant AI Research

***Next-gen Optimization Solvers**: incorporating data-driven learning approaches to discover new algorithmic policies customized to the problem distributions*

***End-to-End learning**: the integration of differentiable combinatorial optimization layers as parts of the deep-learning pipeline, in order to develop predictive models that best guide downstream decision tasks.*

***Multi-agent Planning, Decentralized Optimization and Reinforcement Learning**: develop methods that can help agents reason and optimize, learn continuously, and coordinate and collaborate.*

***Responsible AI**: fairness, bias, privacy,, explanation*

***Education and Outreach**: an innovative longitudinal education and workforce development program, focus on historically black and Hispanic-serving high schools and colleges, in Georgia and California.*

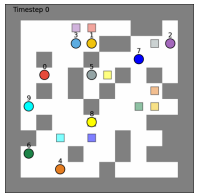
ML ⟺ Combinatorial Optimization

- Exciting and growing research area

AI4OPT USC

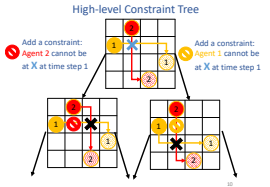
Infusing Discrete Optimization with Machine Learning

LNS for MAPF

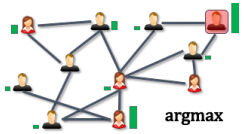


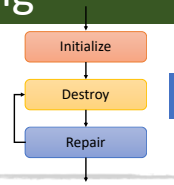
Node Selection in ECBS

Conflict Selection in CBS



Greedy Heuristic

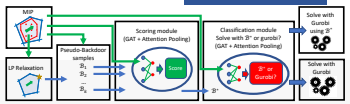




LNS for MIP

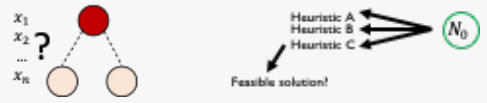
Exact Solving for MILP

Backdoors



Branching

Heuristic Selection



MAPF | Graph Optimization | Integer Programming

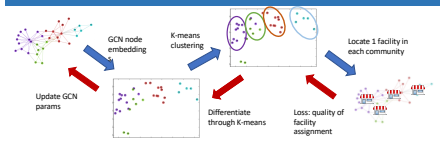
Problem Type

Augment discrete optimization algorithms with learning components

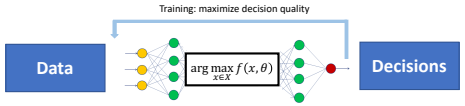
Infusing ML with Constrained Decision Making

MIPaaL: **MIP** as a layer in Neural Networks

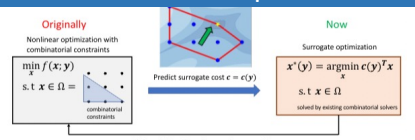
ClusterNET: Differentiable kmeans for a class **graph optimization problems**



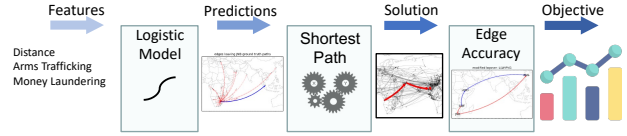
Decision-focused learning for **submodular optimization and LP**



SurCO: MINLP solving using Differentiable optimization



Wildlife Trafficking Routes

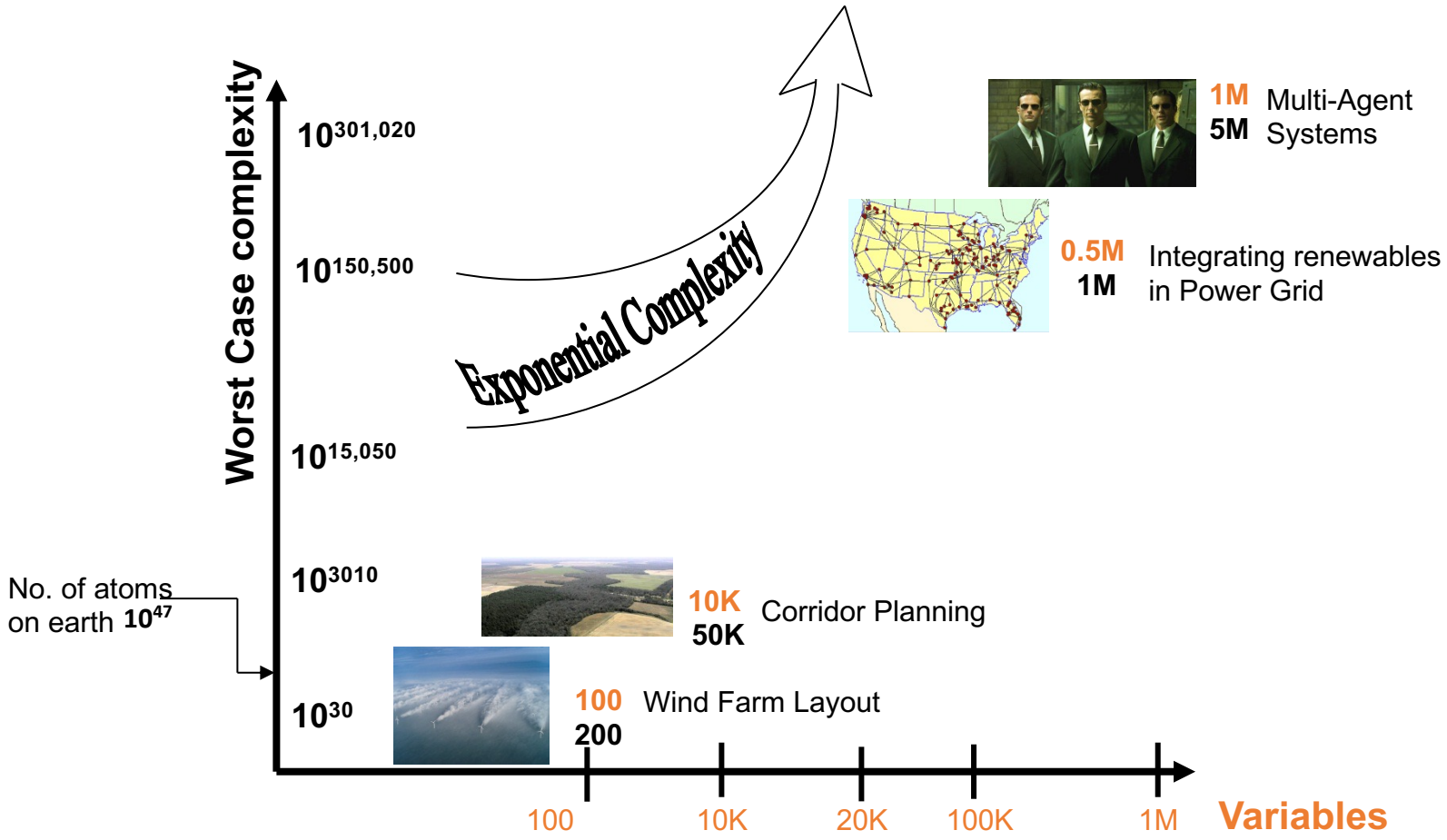


TB Health Visits Allocation

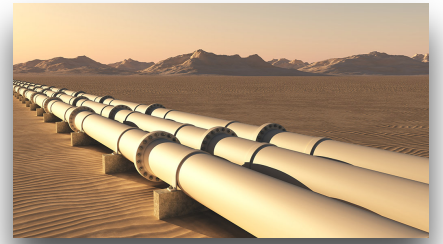
Learning methods that incorporate the combinatorial decisions they inform

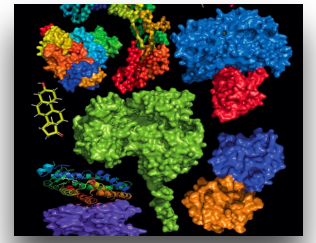
Constraint Reasoning and Optimization

Decision making problems of larger size and new problem structure drive the continued need to improve combinatorial solving methods



Mixed Integer Linear Programs





Flexible mathematical program framework

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

Scalable solving methods key to many real-world applications:

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

Image sources: https://www.r-bloggers.com/2010/11/any-r-packages-to-solve-vehicle-routing-problem/
[https://youtu.be/8gy5tYVR-28]
https://www.cnbc.com/2022/10/16/how-the-uber-lyft-gig-economy-battle-over-drivers-ends.html
https://itsvinayak.hashnode.dev/all-about-knapsack-problem-or-python

Distributional MIP Solving

Learn from:

Solve

Day T + Δ

Repeated solving of similar problems in many real-world settings

Learn search policies tailored to the MIP distribution

Outline

Contrastive learning for LNS-MIP

- Improvement over Imitation Learning and Reinforcement Learning approaches to ML-Guided Large Neighborhood Search (LNS) for MIP

Contrastive learning for backdoors + BnB in MIP

- Previous success in showing existence of small backdoors in MIP
- Improvement over Sample+Learn-to-Rank approach to predicting backdoors in MIP

Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning

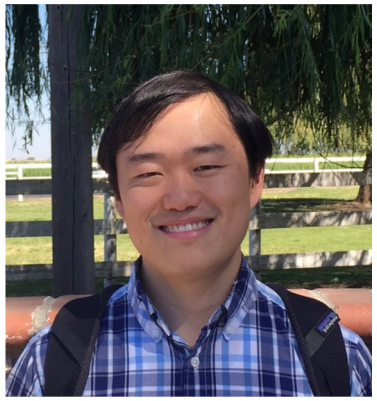
Taoan Huang[1], Aaron Ferber[1], Yuandong Tian[2], Bistra Dilkina[1], Benoit Steiner[3]

[1]University of Southern California

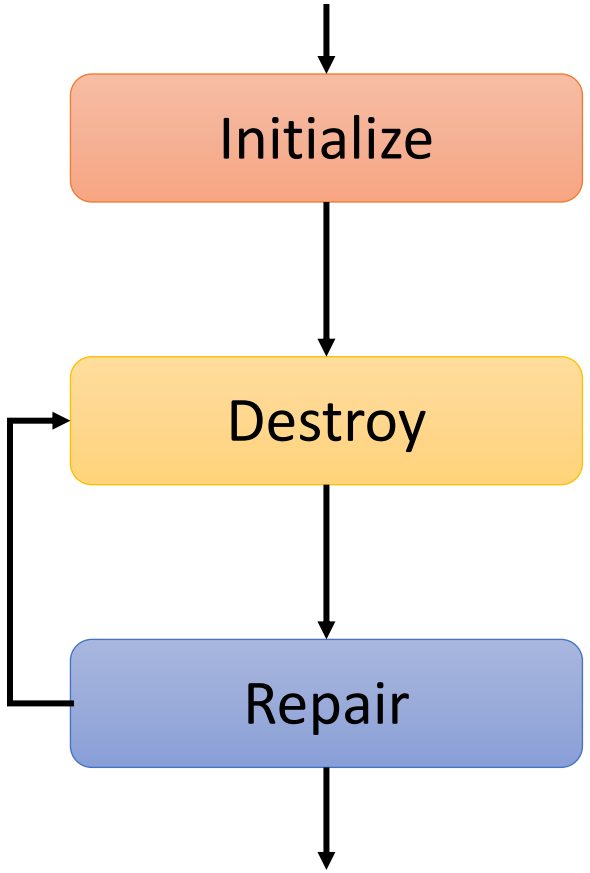
[2]Meta AI Research

[3]Anthropic





Large Neighborhood Search (LNS)



1. Find an initial solution via any method

Require heuristic designs & Opportunities for ML guidance

2. Select a neighborhood, or a subset of k variables, and unassign them.

3. Reoptimize selected variables X' while keeping all other variable assignments frozen

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

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

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

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

[Fischetti & Lodi, 2003] Local Branching. Mathematical programming.

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 $k = 3$

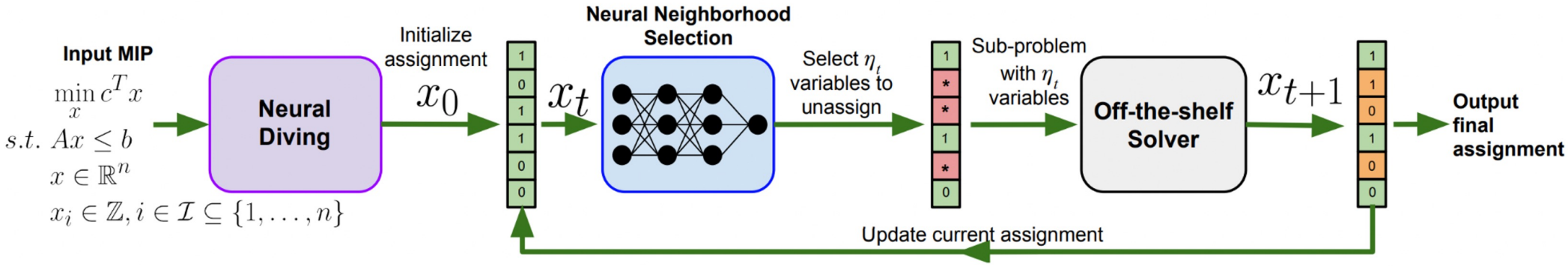
	x_1	x_2	x_3	x_4	x_5
Current best solution	1	1	1	0	0
LB solution	1	0	0	1	0
Difference	0	**1**	**1**	**1**	0

✓ ✓ ✓

IL-LNS: Imitation Learning Approach

Google DeepMind

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



Data collection: solve Local Branching exhaustively (2-3 hours each) in LNS

Training: Imitate Local Branching with a binary cross entropy loss on each variable

Testing: Sample neighborhoods based on the predicted scores of each variable

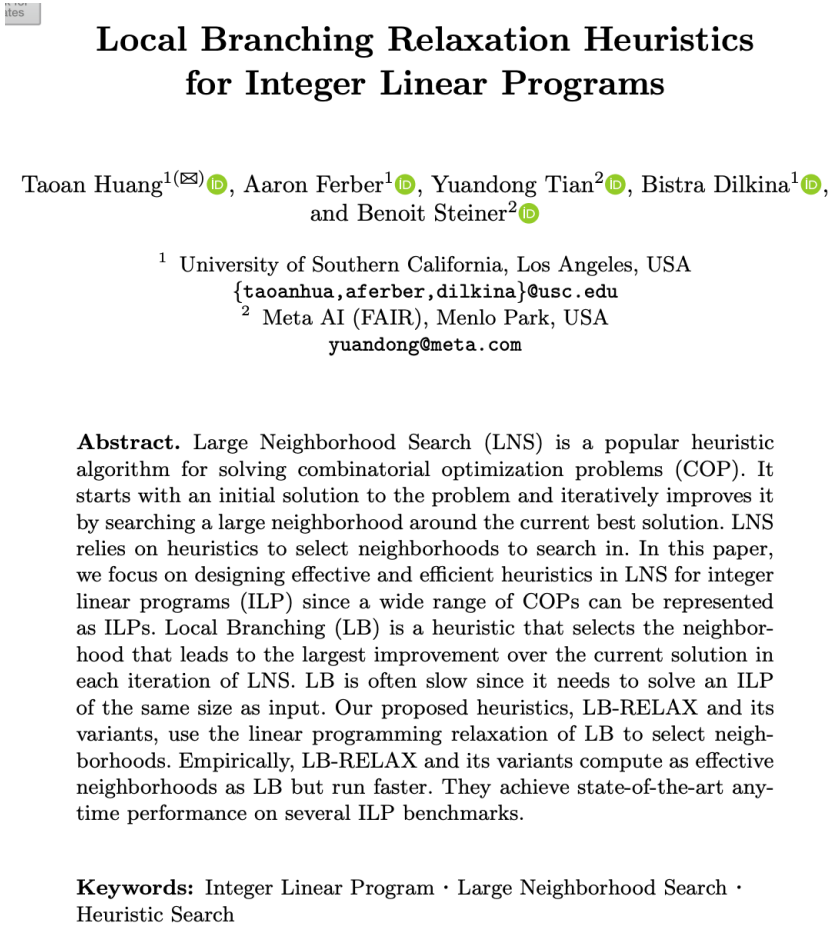
[Sonnerat et al., 2021] Learning a large neighborhood search algorithm for mixed integer programs. Arxiv.

Detour from ML Direction: LB-RELAX

Local Branching Relaxation: solve the LP relaxation of the Local Branching ILP [**Huang et al.**, CPAIOR 2023]

$$\begin{aligned} \min_x \quad & c^T x && \text{objective} \\ \text{s.t.} \quad & Ax \le b && \text{constraints} \\ & x_i \in \{0,1\}\ \forall i && \text{~~integrality~~} \rightarrow \text{continuous} \end{aligned}$$

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



Local Branching Relaxation Heuristics for Integer Linear Programs

Taoan Huang[1(✉)], Aaron Ferber[1], Yuandong Tian[2], Bistra Dilkina[1], and Benoit Steiner[2]

[1] University of Southern California, Los Angeles, USA
{taoanhua,aferber,dilkina}@usc.edu
[2] Meta AI (FAIR), Menlo Park, USA
yuandong@meta.com

Abstract. Large Neighborhood Search (LNS) is a popular heuristic algorithm for solving combinatorial optimization problems (COP). It starts with an initial solution to the problem and iteratively improves it by searching a large neighborhood around the current best solution. LNS relies on heuristics to select neighborhoods to search in. In this paper, we focus on designing effective and efficient heuristics in LNS for integer linear programs (ILP) since a wide range of COPs can be represented as ILPs. Local Branching (LB) is a heuristic that selects the neighborhood that leads to the largest improvement over the current solution in each iteration of LNS. LB is often slow since it needs to solve an ILP of the same size as input. Our proposed heuristics, LB-RELAX and its variants, use the linear programming relaxation of LB to select neighborhoods. Empirically, LB-RELAX and its variants compute as effective neighborhoods as LB but run faster. They achieve state-of-the-art anytime performance on several ILP benchmarks.

Keywords: Integer Linear Program · Large Neighborhood Search · Heuristic Search

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

Local Branching Relaxation Heuristics

LB-RELAX:

- Solve the **LP relaxation** of the Local Branching ILP
- Selects k variables **greedily** whose **relaxation values differ most from the current solution**
- → integer variables in the LP relaxation solution that have changed will always be selected first

	x_1	x_2	x_3	x_4	x_5
Current best solution	1	1	1	0	0
Relaxation solution	1	0.6	0	0.7	0.9
Difference	**0**	**0.4**	**1**	**0.7**	**0.9**

$k = 3$

Sort + Greedily select →

x_3	x_5	x_4	x_2	x_1
1	0	0	1	1
0	0.9	0.7	0.6	1
1	**0.9**	**0.7**	**0.4**	**0**
✓	✓	✓		

Achieved the best performance on 24 out of 31 hard instances from MIPLIB

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

Local Branching Relaxation Heuristics

- Achieved the best performance on 24 out of 31 hard instances from MIPLIB (a well-known ILP benchmark library)
  - Compared to 4 other commonly used LNS versions
- Even outperform some of the state-of-the-art machine learning methods in the distributional settings
- Incorporating LB-RELAX into existing solvers

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

Contrastive Learning

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

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

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

Recent use of contrastive learning in other settings:

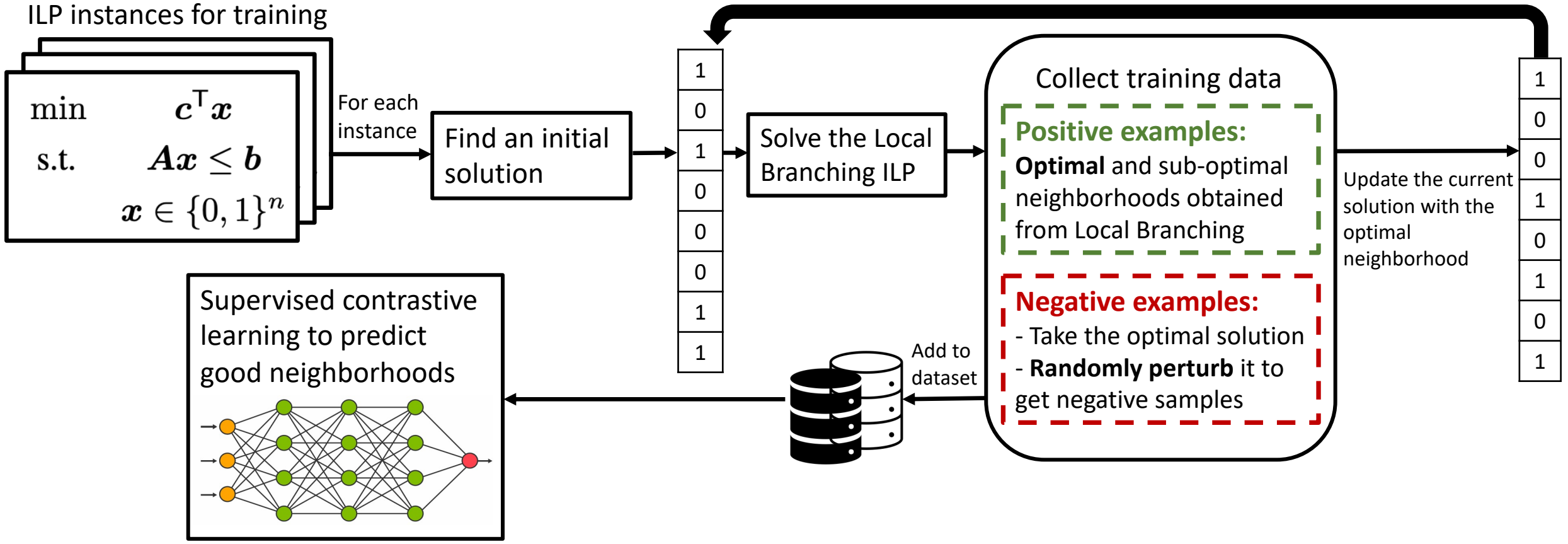
Contrastive learning of visual representations (Hjelm et al., 2019; He et al., 2020; Chen et al., 2020) and graph representations (You et al., 2020; Tong et al., 2021)

[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[Huang et al., ICML 2023]

Training and data collection pipeline



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

CL-LNS: Data Collection

Collect training data

Positive examples:
Optimal and sub-optimal neighborhoods obtained from Local Branching

Negative examples:
- Take the optimal solution
- **Randomly perturb** it to get negative samples

Optimal sample := variables changed in the optimal solution found by local branching solved within 1 hours (using a black-box solver SCIP)
best_improve := improvement of the optimal sample over the current objective value

Positive samples:

- All solutions found by local branching with *improvement >= 0.5 * **best_improve***
- Up to max of 10 solutions

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

CL-LNS: Data Collection

Collect training data

Positive examples:
Optimal and sub-optimal neighborhoods obtained from Local Branching

Negative examples:
- Take the optimal solution
- **Randomly perturb** it to get negative samples

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 the current objective value

Negative samples (k * Num Positive Samples, k=9):

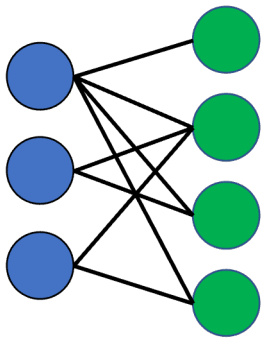
- Randomly replace 5% of variables in the ***optimal sample*** Try to repair it
- Record it as a negative sample if *improvement <= 0.05 * **best_improve***
- If not enough negative samples found, increase to 10% to 20%, 30%...100%

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

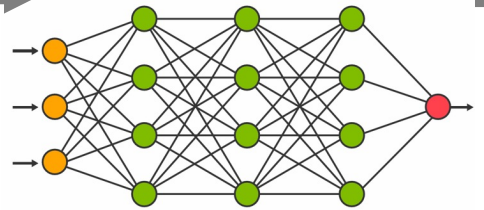
CL-LNS: Neural Network Architecture

MIP Bipartite Graph and Features

-Variable nodes -Constraint nodes



Graph Attention Network
- With message passing



Neural Network Output

$$\pi(\mathbf{x})$$

$$n \times 1$$

The final output is a score in [0,1] per variable

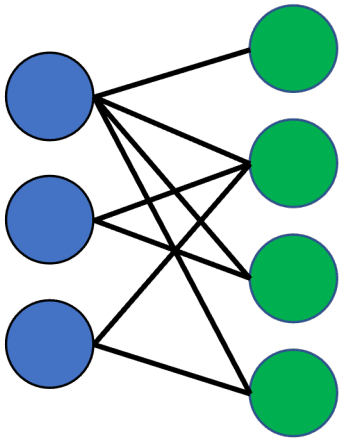
indicating how promising it is to select each variable

We use the same graph and message-passing mechanism in previous work used for BnB:
[Gasse et al., NeurIPS 2019] Exact Combinatorial Optimization with Graph Convolutional Neural Networks.
But **we replace convolution layers with attention layers**

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

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

-**Variable nodes** -**Constraint nodes**



Variable features (**V**):

- Objective coefficient and its constrain coefficient
- The number of constraints in which it appears
- …
- Features based on the most recent best-found solutions

Edge features (**E**):

- Coefficient values

Constraint features (**C**):

- The number of non-zero coefficients
- Average coefficient values
- …

[Gasse et al., 2019] Exact Combinatorial Optimization with Graph Convolutional Neural Networks. NeurIPS
[Khalil et al., 2016] learning to branch in mixed integer programming. AAAI

CL-LNS: Training with a Contrastive Loss

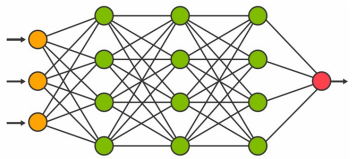
Contrastive Loss function (InfoNCE):
optimizes the negative log probability of the final embedding being similar to the positive samples

$$\frac{-1}{|\mathcal{S}_{pos}|} \sum_{\boldsymbol{a} \in \mathcal{S}_{pos}} \log \frac{\exp(\boldsymbol{a}^\top \pi(\boldsymbol{x})/\tau)}{\sum_{\boldsymbol{a}' \in \mathcal{S}_{neg} \cup \{\boldsymbol{a}\}} \exp(\boldsymbol{a}'^\top \pi(\boldsymbol{x})/\tau)}$$

Positive Samples

Negative samples

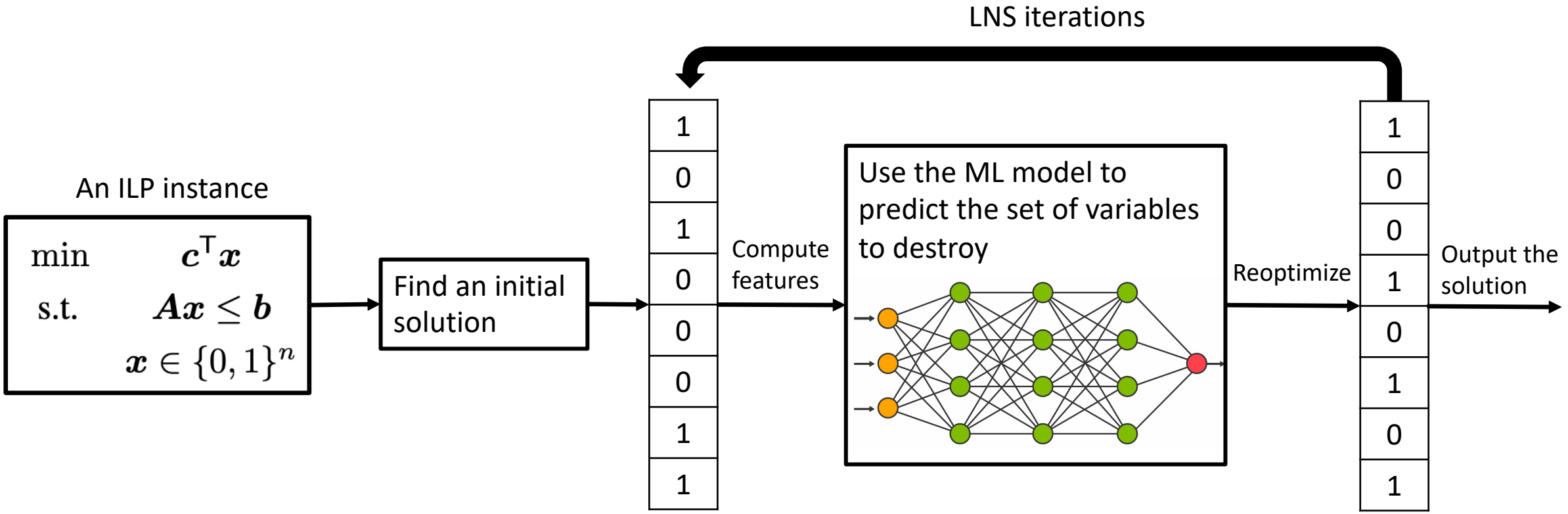
Neural Network Output



Each positive / negative sample (potential destroy set) is a binary 0/1 vector of length n variables
Model output $\pi(\boldsymbol{x})$ is a continuous [0,1] vector of length n variables

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

CL-LNS: Testing



Test Time: Greedily choose the k variables with max scores from the ML model

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

Benchmarking

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

Results on two more domains (MIS, MSC) 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]

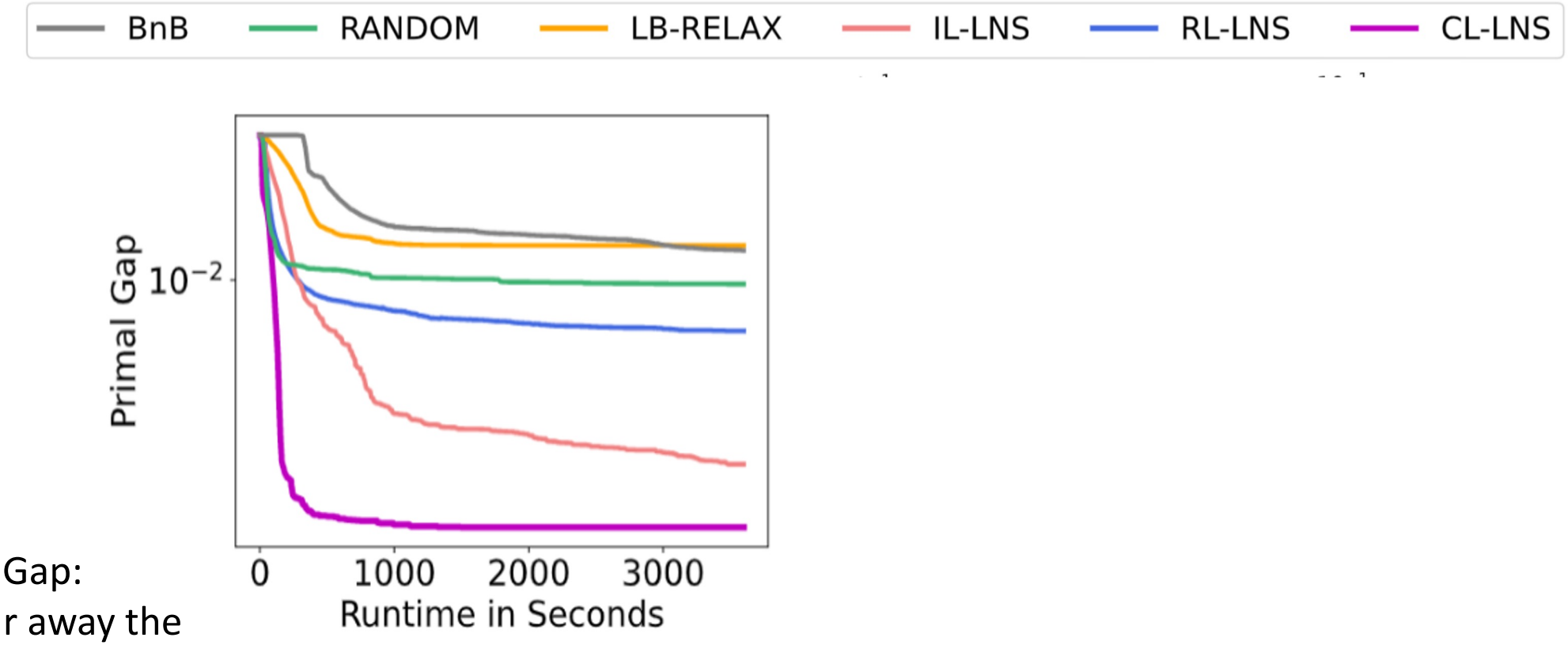
vs.

- **CL-LNS** (ours)

Comparison with two more baselines in our paper

[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

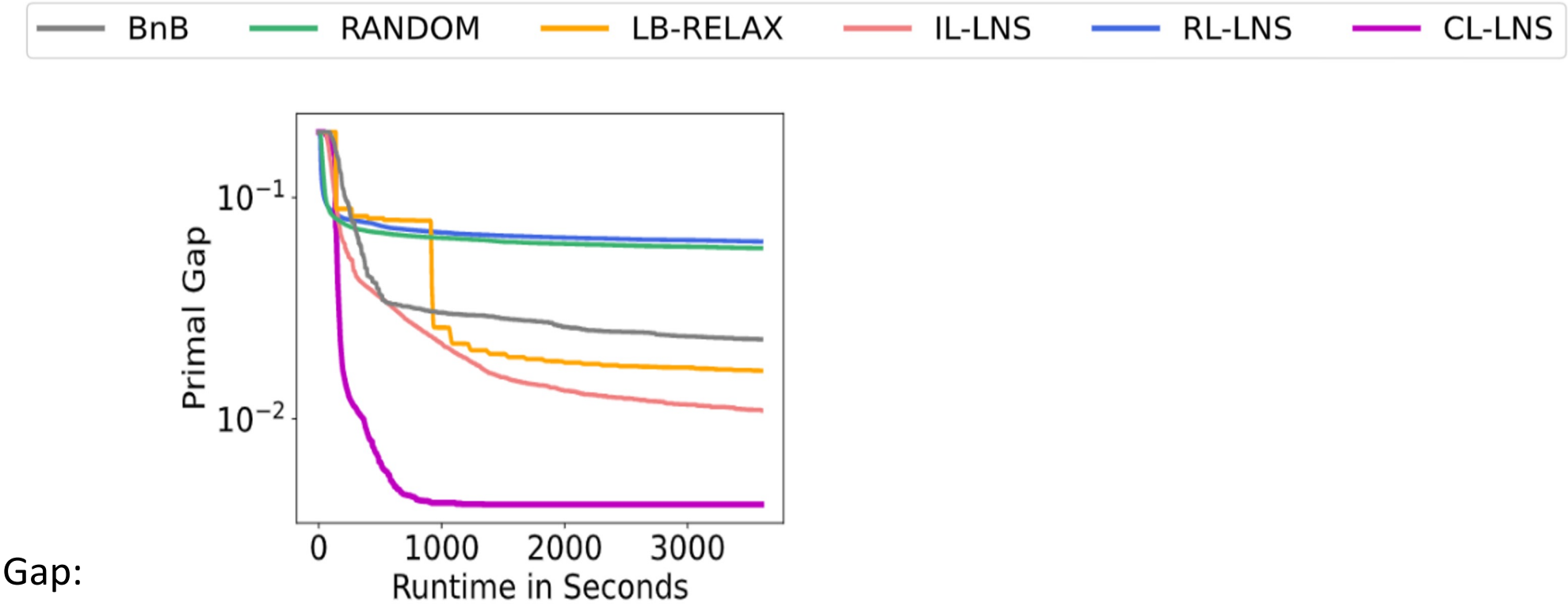
Results: Primal Gap over Time
Minimum Vertex Cover



Min Vertex Cover-Small

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

Results: Primal Gap over Time
Combinatorial Auctions

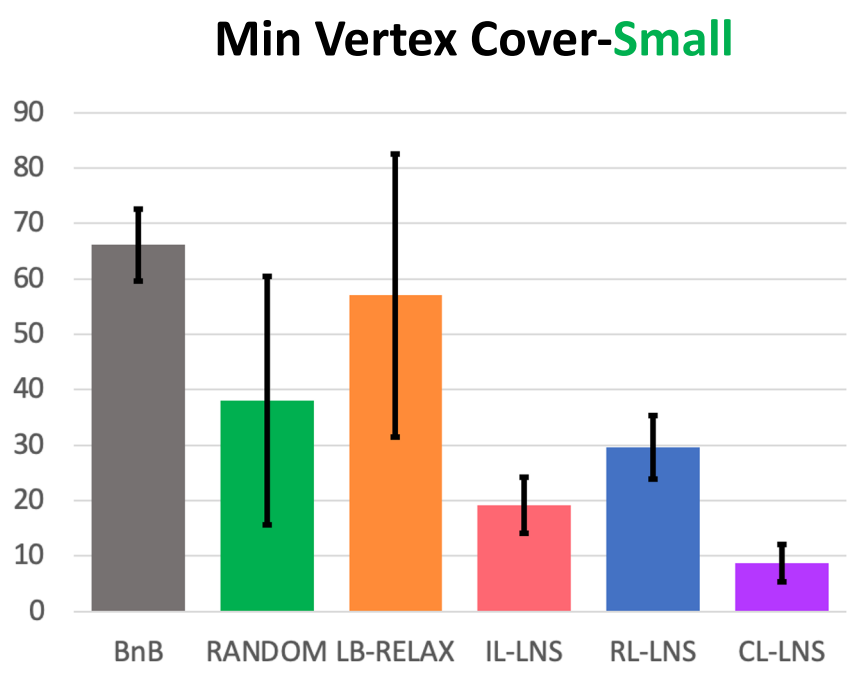


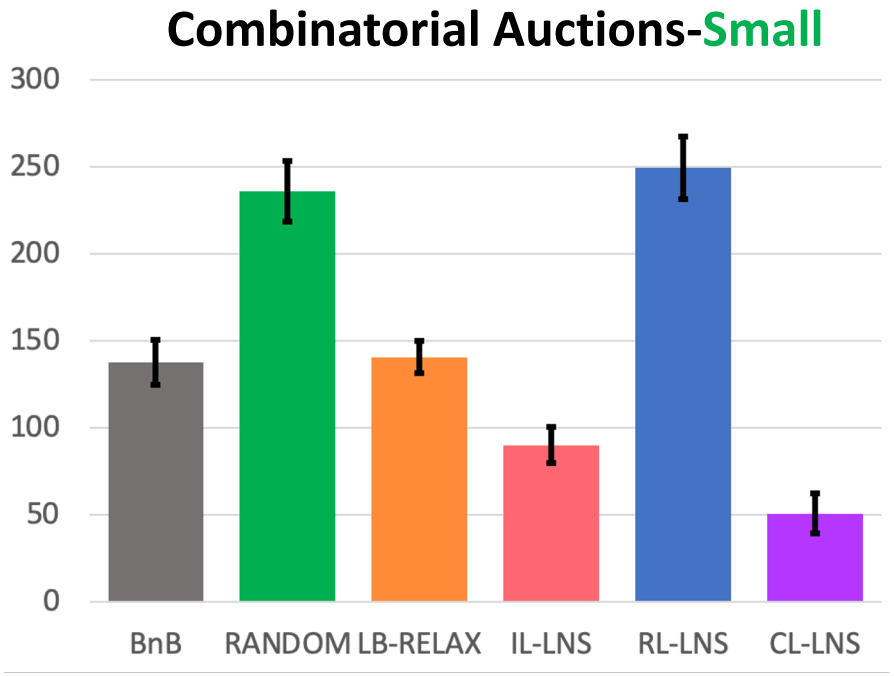
Combinatorial Auctions-Small

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

Results: Primal Integral

60 minutes - Small instances



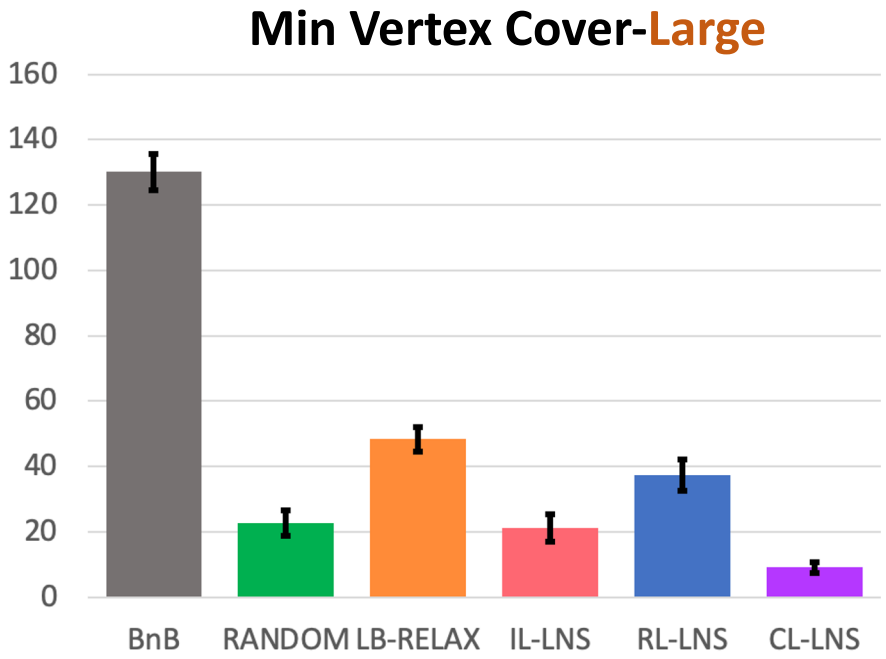


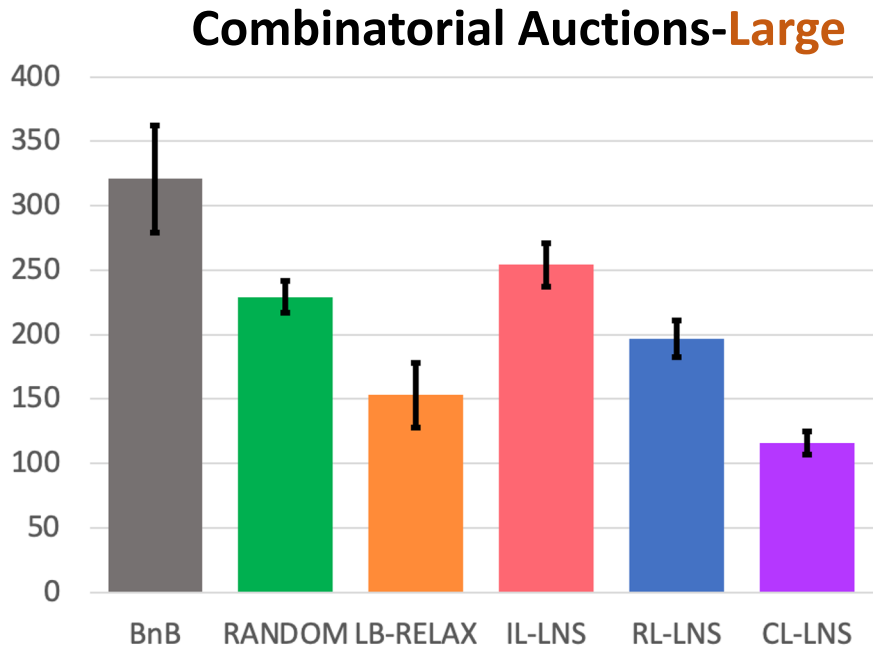
The **primal integral** at time q is the integral on [0, q] of the primal gap as a function of runtime.

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

Results: Primal Integral

60 minutes - Large instances

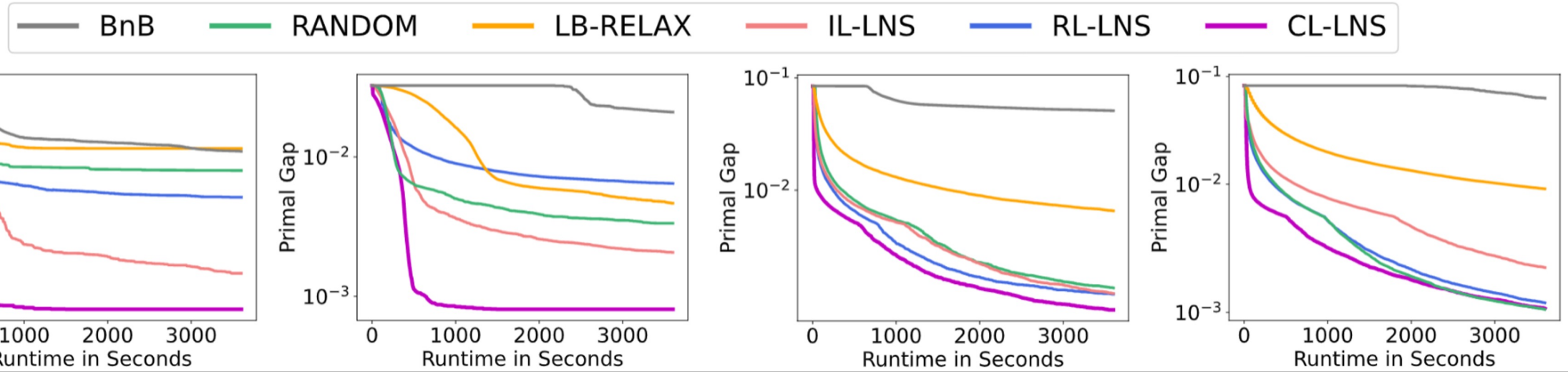




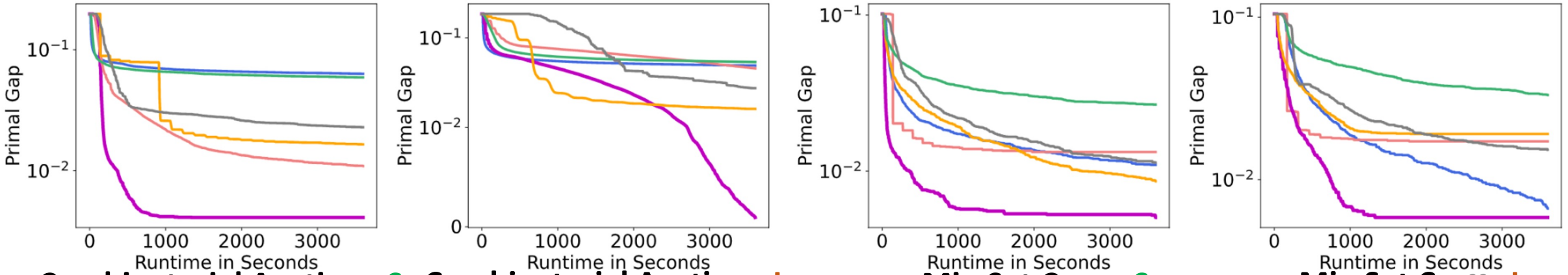
The **primal integral** at time q is the integral on [0, q] of the primal gap as a function of runtime.

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

Results: Primal Gap over Time



Min Vertex Cover-S **Min Vertex Cover-L** **Max Independent Set-S** **Max Independent Set-L**



Combinatorial Auctions-S **Combinatorial Auctions-L** **Min Set Cover-S** **Min Set Cover-L**

Outline

Contrastive learning for LNS-MIP

- ML-Guided Large Neighborhood Search for Mixed Integer Program
  - **Up to 1.69x** speedup for improving solutions

Contrastive learning for backdoors in MIP

- Previous success in finding small backdoors in MIP
- Previous success with learning to predict backdoors in MIP
- Improvement with Contrastive learning

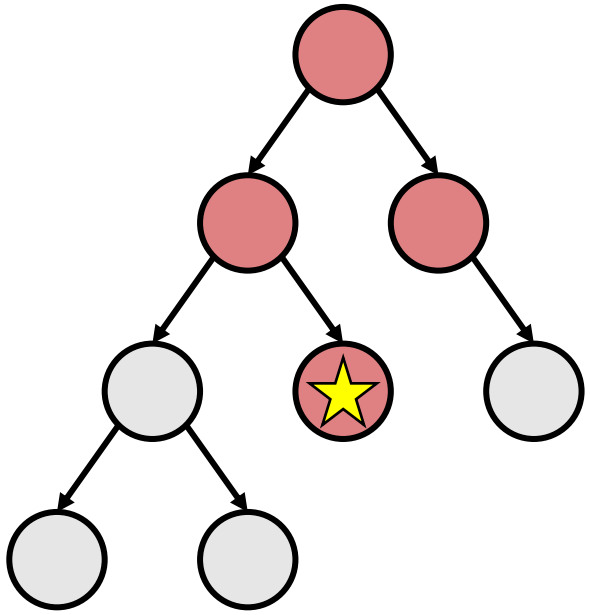
Backdoors

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]



Previous Work: Backdoors

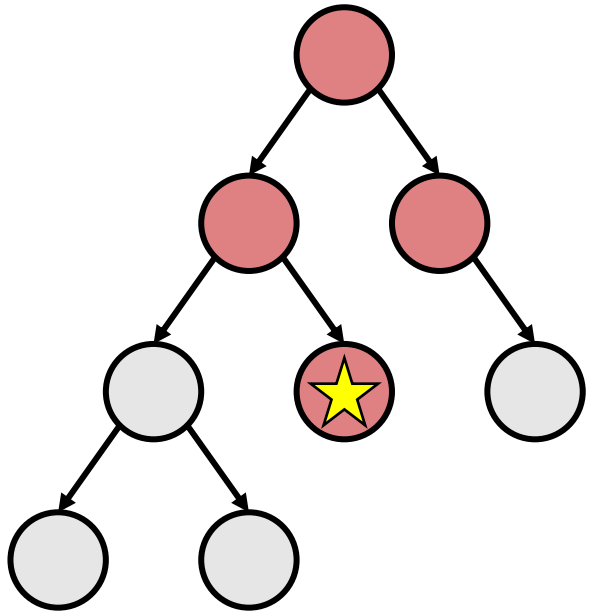
SAT/CSP: first introduced, with empirical limitations

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

MIP Sampling backdoors

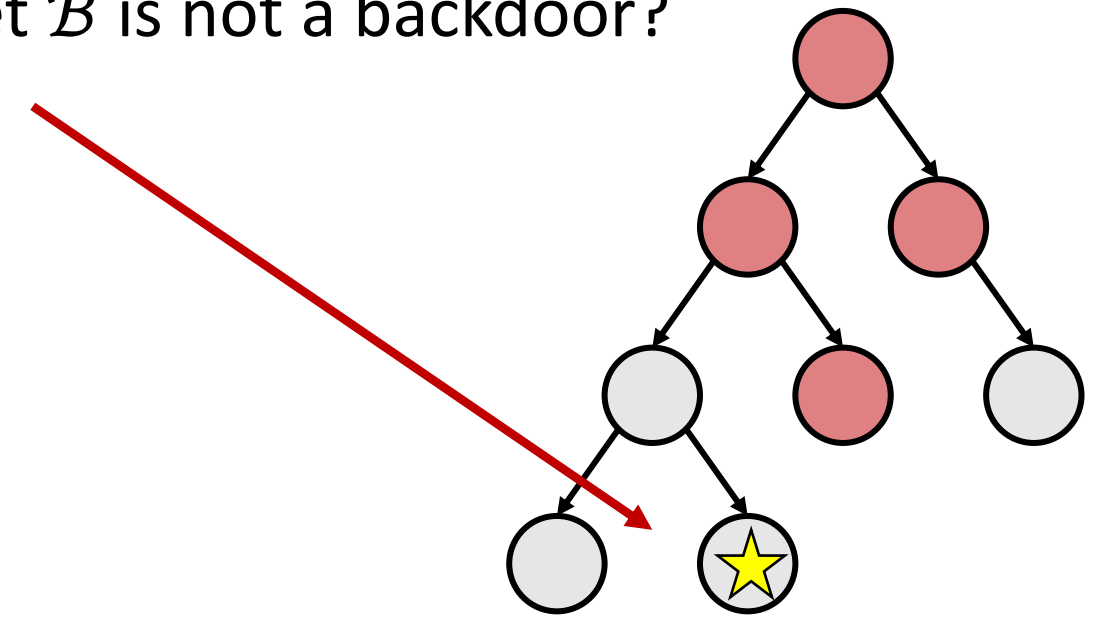
- Dilkina et al. CPAIOR 2009
- Fiscetti et al. IPCO 2011
- Dvořák et al. IJCAI 2017
- Khalil et al. AAAI 2022

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



Learning MIP Backdoors

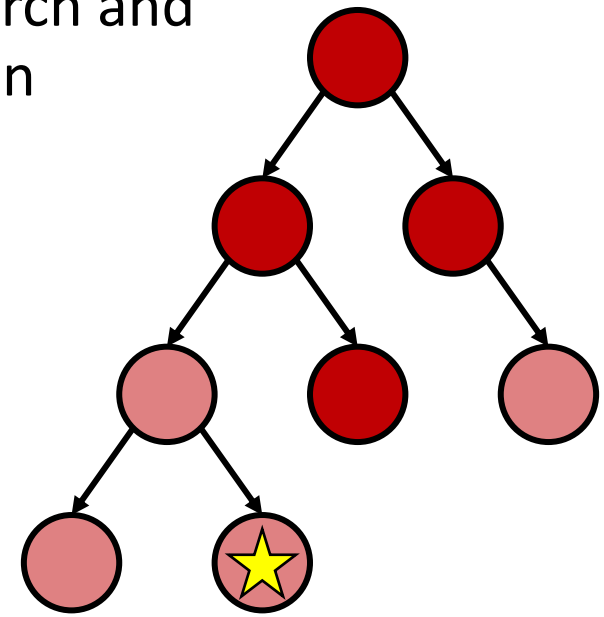
- [CPAIOR 2022] “Learning Pseudo-Backdoors for Mixed Integer Programs”. Aaron Ferber, Jialin Song, Bistra Dilkina, Yisong Yue.
- **Idea:** Learn to choose a backdoors subset B and provide to BnB solver
- But what happens if the subset $\mathcal{B}$ is not a backdoor?
  - Can’t find the optimal solution
  - -> pseudo-backdoors



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

Pseudo-Backdoors

- Subset of decision variables $\mathcal{B}$
- Prioritizing branching on this subset $\mathcal{B}$ yields **efficient solving**
- What happens if we are wrong?
  - Solver continues the search and finds the optimal solution



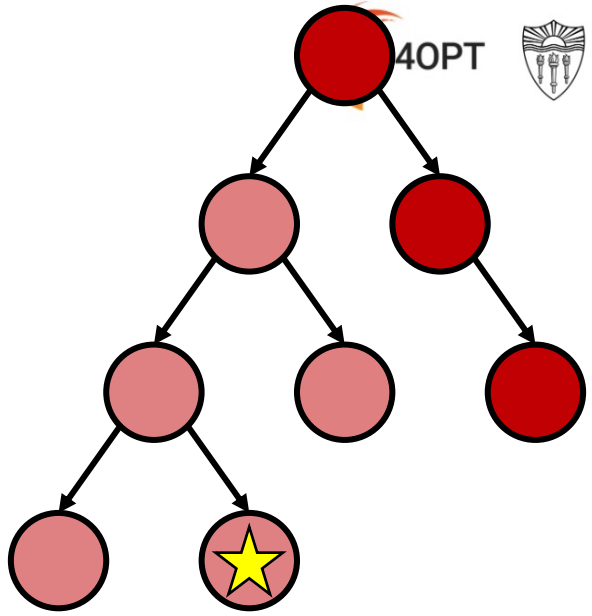
- **Faster solve time**
- **Efficient gap convergence**
- **High quality solutions found at cutoff**

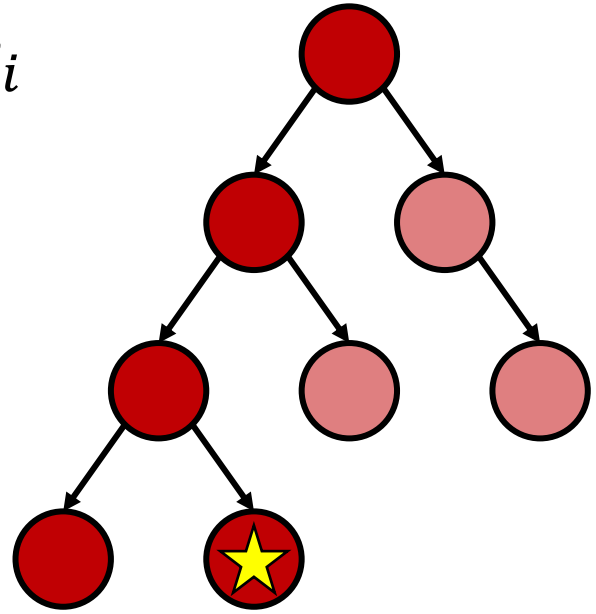
[CPAIOR 2022] “Learning Pseudo-Backdoors for Mixed Integer Programs”. Aaron Ferber, Jialin Song, Bistra Dilkina, Yisong Yue.

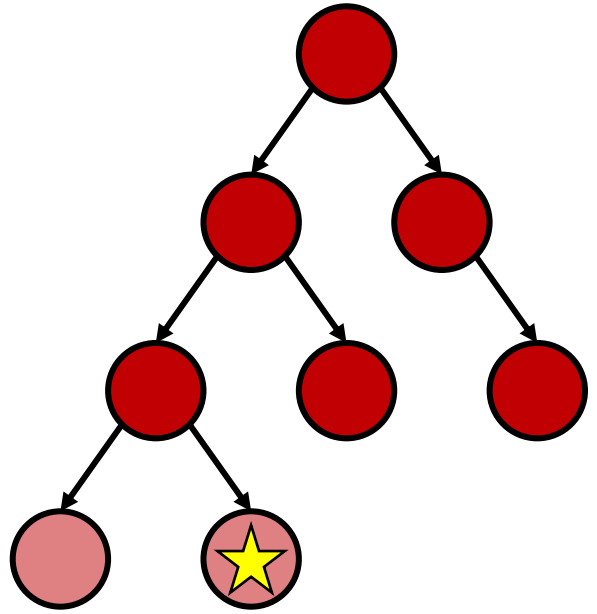
Training: data collection

- Given training MIPs
- Sample k pseudo-backdoors $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_i, \ldots \mathcal{B}_k$
- Solve MIPs by **setting branching priority**
- Collect solve performance labels v_i
  - Runtime

- Additionally get Gurobi runtime (without backdoors)

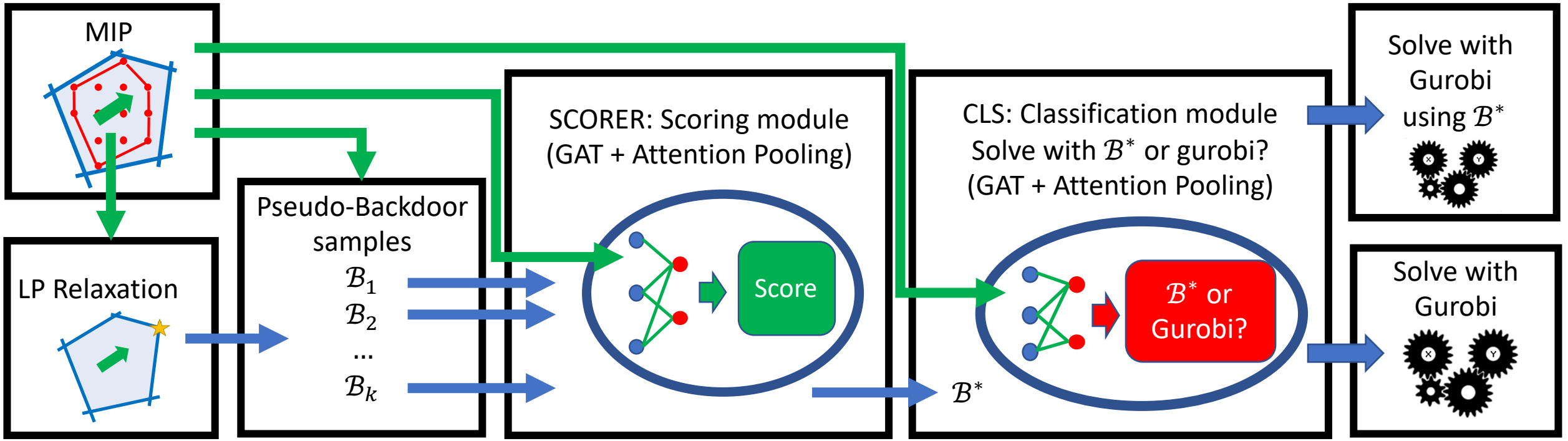






[CPAIOR 2022] “Learning Pseudo-Backdoors for Mixed Integer Programs”. Aaron Ferber, Jialin Song, Bistra Dilkina, Yisong Yue.

ML-MIP-Backdoor Pipeline [CPAIOR 2022]



Approach: two-part model

- **Scorer**: Train value estimator to **rank** backdoors among a set of backdoor candidates (LP-based sampling)
- **Classifier**: Train classifier model to predict 0/1 whether highest ranked backdoor solves faster than Gurobi

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

Results: win / tie / loss

dataset	solver	mean	stdev	W/T/L vs grb
nnverify	grb	6.5	7.9	0 / 100 / 0
nnverify	scorer	7.0	12.8	68 / 0 / 31
nnverify	scorer+cls	**5.6**	**7.4**	62 / 18 / 20
facilities easy	grb	27.4	21.6	0 / 100 / 0
facilities easy	scorer	24.9	**18.2**	65 / 0 / 35
facilities easy	scorer+cls	**22.8**	18.8	55 / 33 / 12
facilities hard	grb	46.9	31.6	0 / 100 / 0
facilities hard	scorer	56.0	42.5	28 / 0 / 72
facilities hard	scorer+cls	**44.5**	**30.9**	25 / 74 / 1
gisp easy	grb	611	**182**	0 / 100 / 0
gisp easy	scorer	960	755	41 / 0 / 59
gisp easy	scorer+cls	**601**	247	24 / 70 / 6
gisp hard	grb	2533	939	0 / 100 / 0
gisp hard	scorer	2373	**855**	47 / 0 / 53
gisp hard	scorer+cls	**2326**	**855**	41 / 39 / 20

- Scorer only has many wins but also many losses, decreasing performance
- Adding classifier (cls) reduces losses while keeping many wins improving overall pipeline performance

Learning Backdoors for Mixed Integer Programs with Contrastive Learning

Junyang Cai, Taoan Huang, Bistra Dilkina

University of Southern California

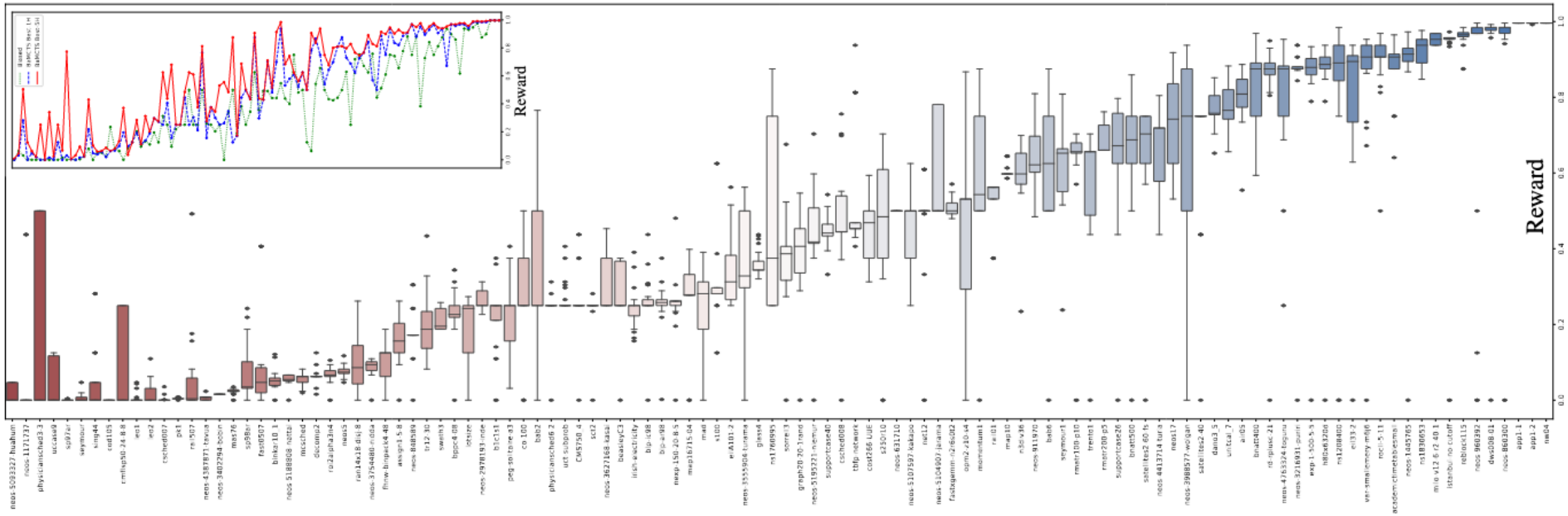
Better Backdoor Discovery [Khalil et al, AAAI 2022]

- Sampling backdoors for MIP via Monte Carlo Tree Search, based on tree weight
- For binary MIPs

[Khalil, Vaezipoor, Dilkina, AAAI 2022]
Finding backdoors to integer programs: A Monte Carlo Tree Search framework.

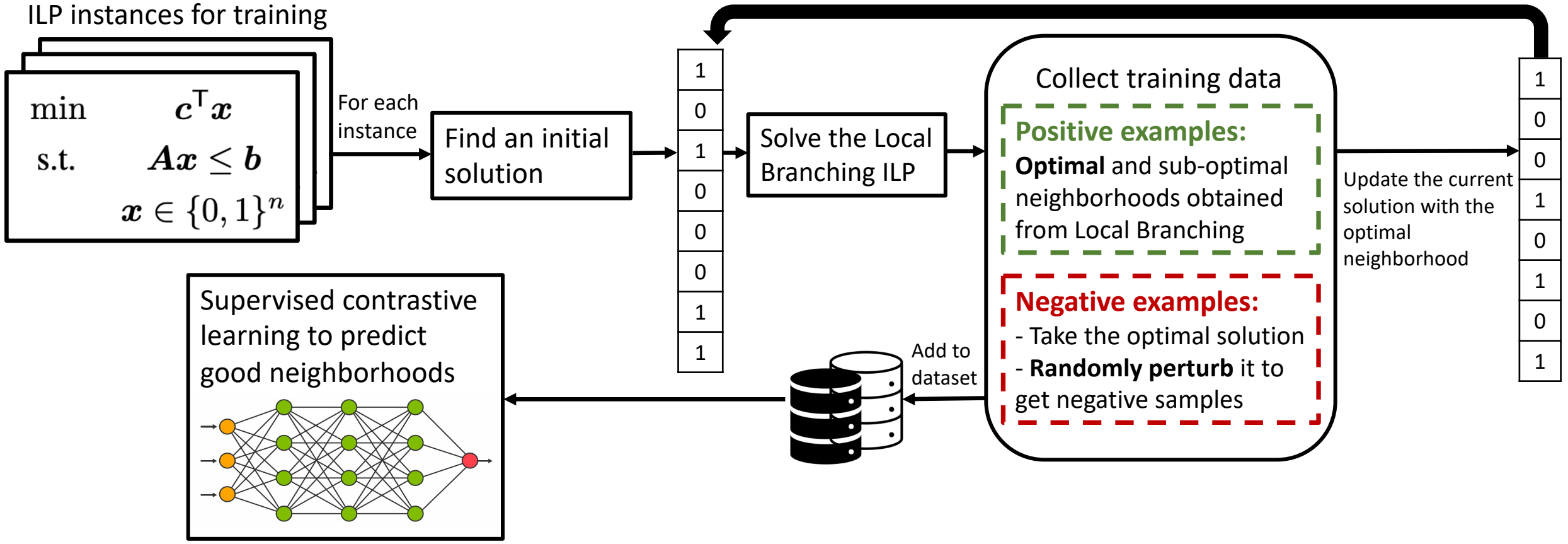
5+ hours for MCTS sampling per MIP,

but proof that good small (k=8) backdoors with BnB speedups exist for many instances



Contrastive Learning for LNS[Huang et al., ICML 2023]

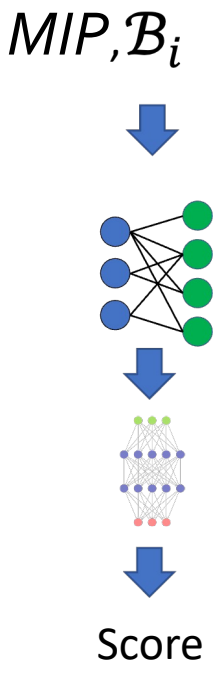
Training and data collection pipeline



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

Key Improvements

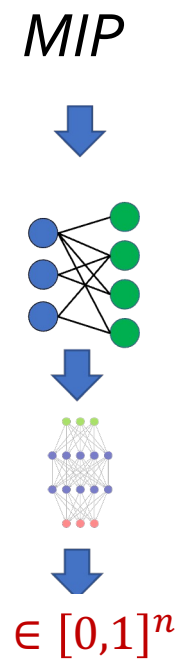
- [CPAIOR2022]
  - [CPAIOR2022] sample backdoors + rank (train with ranking loss)
  - Weakness:
    - Weak learning – only scores whole backdoor set but no feedback on individual variables
    - LP-based backdoor sampling: none of the sampled backdoors may be good



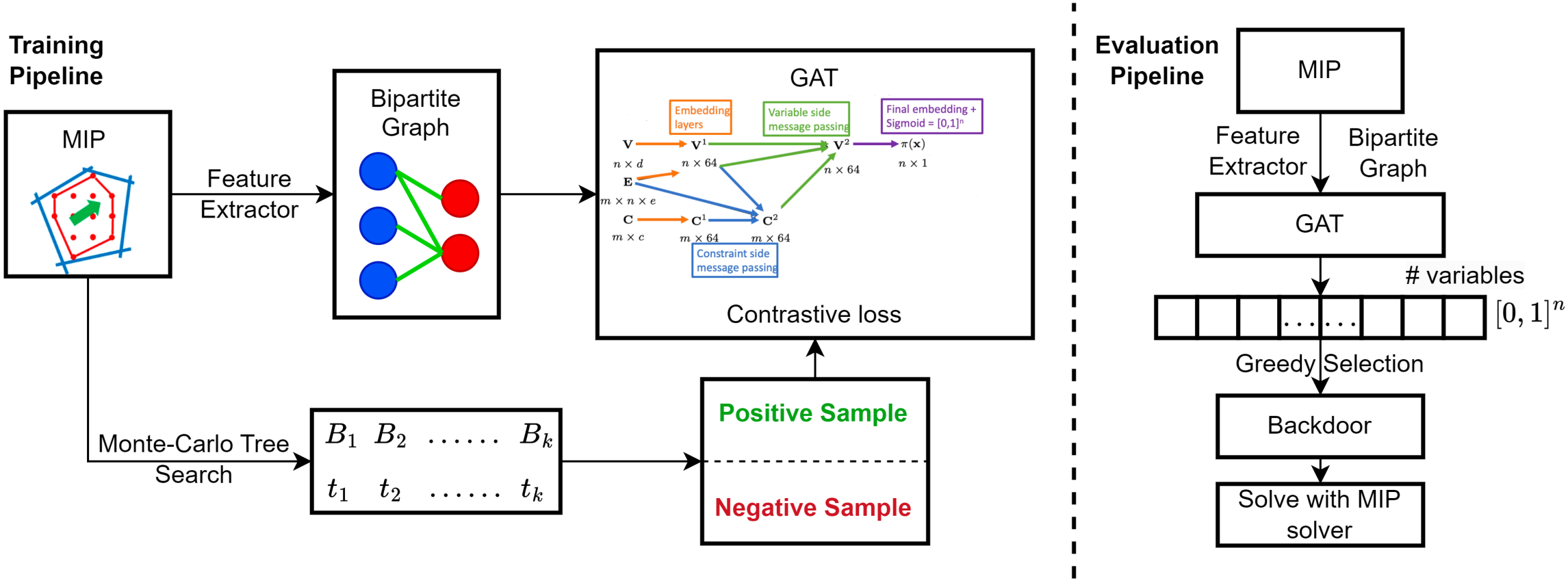
- New learning
  - Idea: train a model to <u>generate</u> a candidate backdoor

  + use **contrastive loss**

- New sampling of train data
  - **Monte Carlo Tree Search** for backdoor sampling [AAAI2022]



CL-MIP-Backdoor Pipeline



Training data collection (for each MIP)

- Run MCTS [Khalil et al., AAAI2022] to collect candidate backdoors
- Select the top **k** candidate backdoors S with the highest tree weight
- Solve MIP with these k backdoors to obtain BnB runtimes

- Positive samples S_p = **p** shortest-runtime backdoors in S
- Negative samples S_n
  - for each positive sample B in S_p, choose **q** backdoors with the most common variables with B among $S \backslash S_p$, remaining backdoors that are not positive samples
  - identified without additional data collection

- In experiments, we set k = 50, p = q = 5

Results: GISP, SC, CA, MIS

Benchmarks: Generalized Independent Set Problem (GISP), Set Cover (SC), Combinatorial Auction (CA), Maximum Independent (MIS)

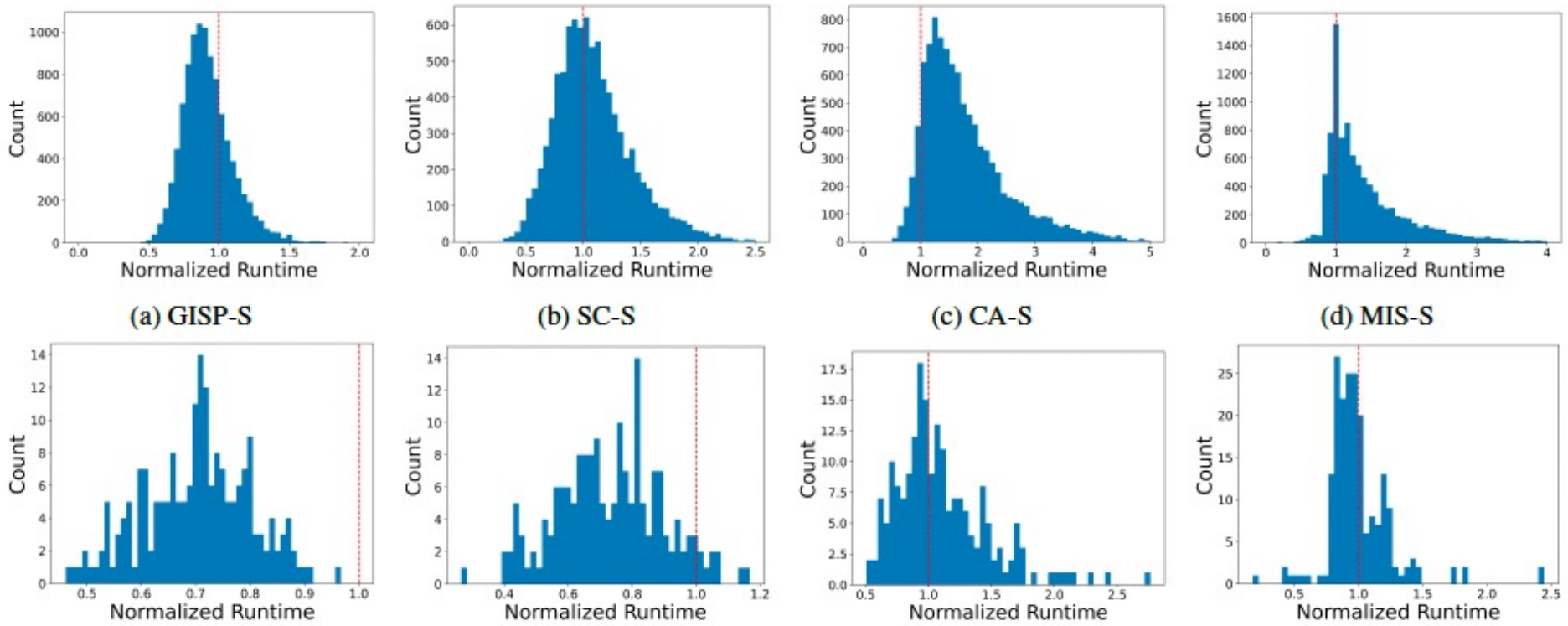
Dataset(#vars, #cons)	Solver	Mean	Std Dev	25 pct	Median	75 pct	W/T/L vs GRB
GISP-S (988, 3253)	GRB	633	154	513	615	731	-
	SCORER	544 (14.1%)	**117**	453	543 (11.7%)	**611**	74/0/26
	SCORER + CLS	578 (8.7%)	128	502	586 (4.7%)	637	43/32/25
	CL	**533 (15.8%)**	120	**441**	**524 (14.8%)**	612	**84/0/16**
	CL + CLS	560 (11.5%)	135	462	555 (9.8%)	653	59/27/14
SC-S (1000, 1200)	GRB	171	189	50.8	101	216	-
	SCORER	244 (-42.69%)	219	67.2	148 (-46.5%)	295	27/0/73
	SCORER + CLS	203 (-18.7%)	195	62.1	142 (-40.6%)	256	15/57/28
	CL	**149 (12.9%)**	**163**	**42.5**	79.8 (20.1%)	**195**	**77/0/23**
	CL + CLS	153 (10.5%)	165	49.4	**79.4 (21.4%)**	197	52/27/21
CA-S (750,282)	GRB	177	94.6	110	151	229	-
	SCORER	224 (-26.6%)	120	125	217 (-43.7%)	293	17/0/83
	SCORER + CLS	194 (-9.6%)	112	119	191 (-26.5%)	243	14/52/34
	CL	**156 (11.9%)**	**83.0**	**88.5**	146 (3.3%)	**195**	**68/0/32**
	CL + CLS	170 (4.0%)	96.2	103	**139 (7.94%)**	227	46/28/26
MIS-S (1250, 3946)	GRB	218	231	79.7	147	267	-
	SCORER	236 (-8.3%)	258	89.9	156 (-6.1%)	269	36/0/64
	SCORER + CLS	219 (-0.5%)	237	82.3	154 (-4.8%)	264	24/46/30
	CL	**184 (15.6%)**	**179**	**76.8**	**127 (13.6%)**	**214**	**69/0/31**
	CL + CLS	190 (12.8%)	197	80.2	136 (7.5%)	227	44/35/21

Collecting backdoor data with MCTS

Runtime relative to Gurobi

50 backdoors per MIP

Best of 50 backdoors per MIP



- [CPAIOR2022] collected data on SC, CA, and MIS but found that using LP-based approach the best-sampled backdoors **under-**perform Gurobi on **all** instances.
- With MCTS data collection, on SC, CA, and MIS we have at least 47% of instances with at least one backdoor that can outperform Gurobi, and for SC we even reach around 85%.
- SC, CA, MIS – most variables are fractional in LP solution, confusing LP-based backdoor sampling

Ablation Study (MCTS and CL)

Dataset	Solver	Mean	Std Dev	25 pct	Median	75 pct	W/L vs GRB
GISP-S	SCORER + Sampling	601	152	518	580	681	49/51
	SCORER + MCTS	544	**117**	453	543	**611**	74/26
	CL + Sampling	565	132	455	556	634	67/33
	CL + MCTS	**533**	120	**441**	**524**	612	**84/16**
SC-S	SCORER + Sampling	287	358	85.4	167	359	3/97
	SCORER + MCTS	244	219	67.2	148	295	27/73
	CL + Sampling	186	196	48.5	107	254	43/57
	CL + MCTS	**149**	**163**	**42.5**	**79.8**	**195**	**77/23**

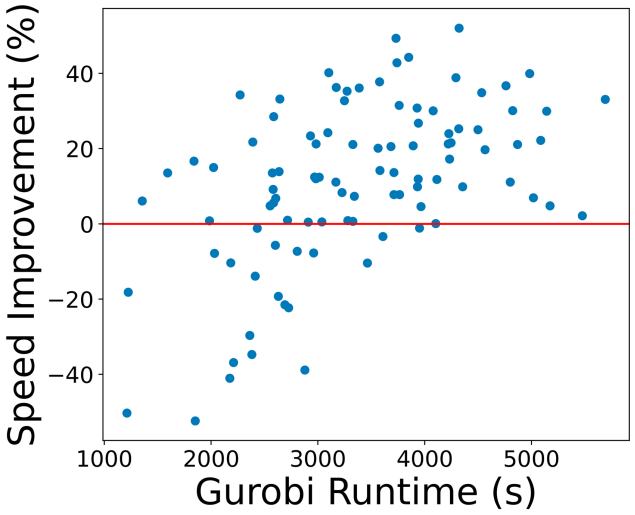
- Using **MCTS-collected training data** benefits both learning approaches over LP-based sampling
- Changing to **Contrastive Loss** gives even more substantial improvements than changing the training data collection strategy

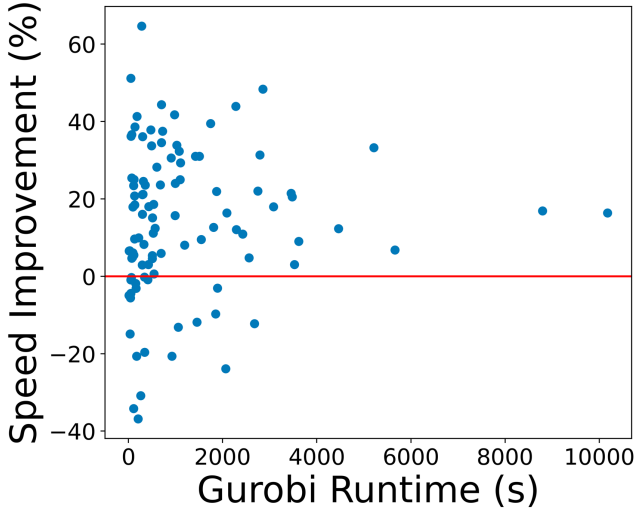
Generalization to larger sizes

Dataset(#vars, #cons)	Solver	Mean	Std Dev	25 pct	Median	75 pct	W/L vs GRB
GISP-L (1316, 4571)	GRB	3,363	989	2,611	3,303	4,049	-
	CL	**2,879 (14.4%)**	**785**	**2,288**	**2,910 (11.9%)**	**3,331**	**77/23**
SC-L (1000, 1500)	GRB	1,195	1,692	161	519	1,597	-
	CL	**1,003 (16.1%)**	**1,433**	**143**	**435 (16.0%)**	**1,309**	**78/22**
CA-L (1000, 377)	GRB	1,213	625	881	1,093	1,353	-
	CL	**1,096 (9.6%)**	**487**	**767**	**994 (9.1%)**	**1,159**	**69/31**
MIS-L (1500, 5941)	GRB	759	1,356	128	263	1,023	-
	CL	**550 (27.5%)**	**987**	**127**	**256 (2.7%)**	**501**	**75/25**

CL model gives speed-ups on **hardest** instances in each distribution

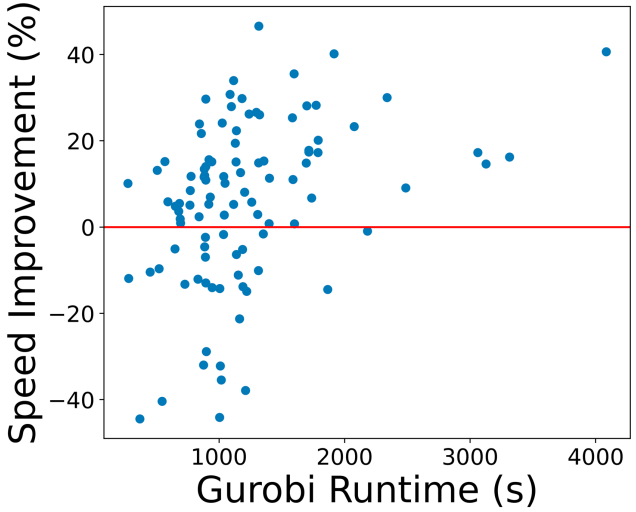
GISP-L

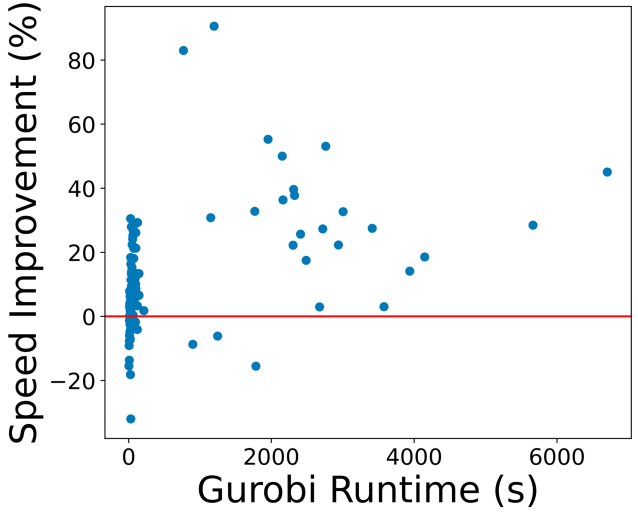




SC-L

CA-L



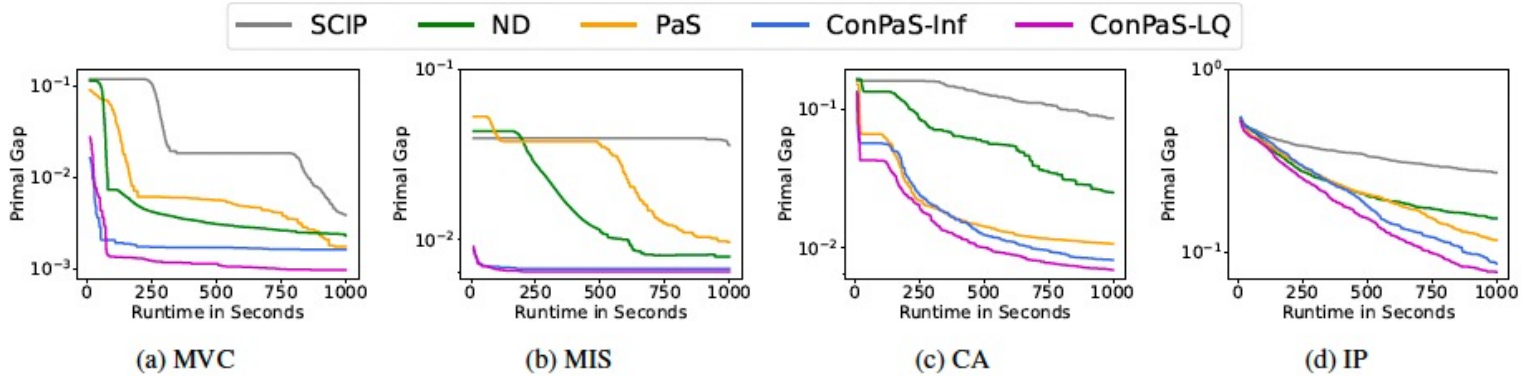


MIS-L

Conclusion: Contrastive learning to the rescue

- Developed CL-guided **Large Neighborhood Search** for MIP (CL-LNS)
  - outperforms all other (heuristic and learning-based LNS-MIP SOTAs)
- Demonstrated that CL loss can significantly improve other **set-prediction tasks in ML-guided MIP** solving

  - Contrastive learning for **Backdoor** prediction [covered here + see poster 1]
  - + Contrastive **Predict-and-Search** for MIP [see poster 2]



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!