

Searching large neighborhoods for integer linear programs with contrastive learning

Bistra Dilkina

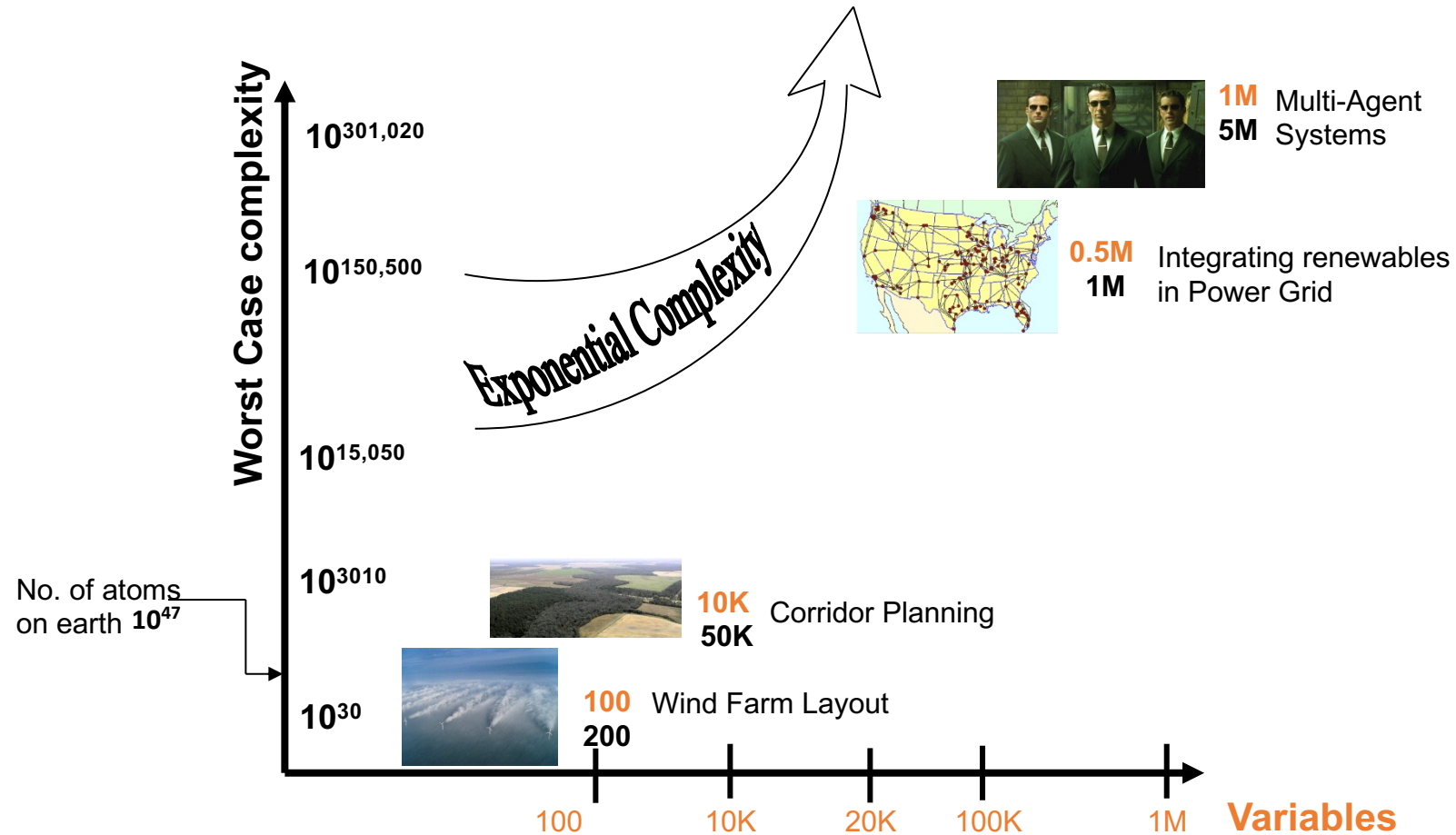
Associate Professor of Computer Science
Co-Director of USC Center on AI in Society
University of Southern California

MIP Workshop

Apr 25, 2023

Constraint Reasoning and Optimization

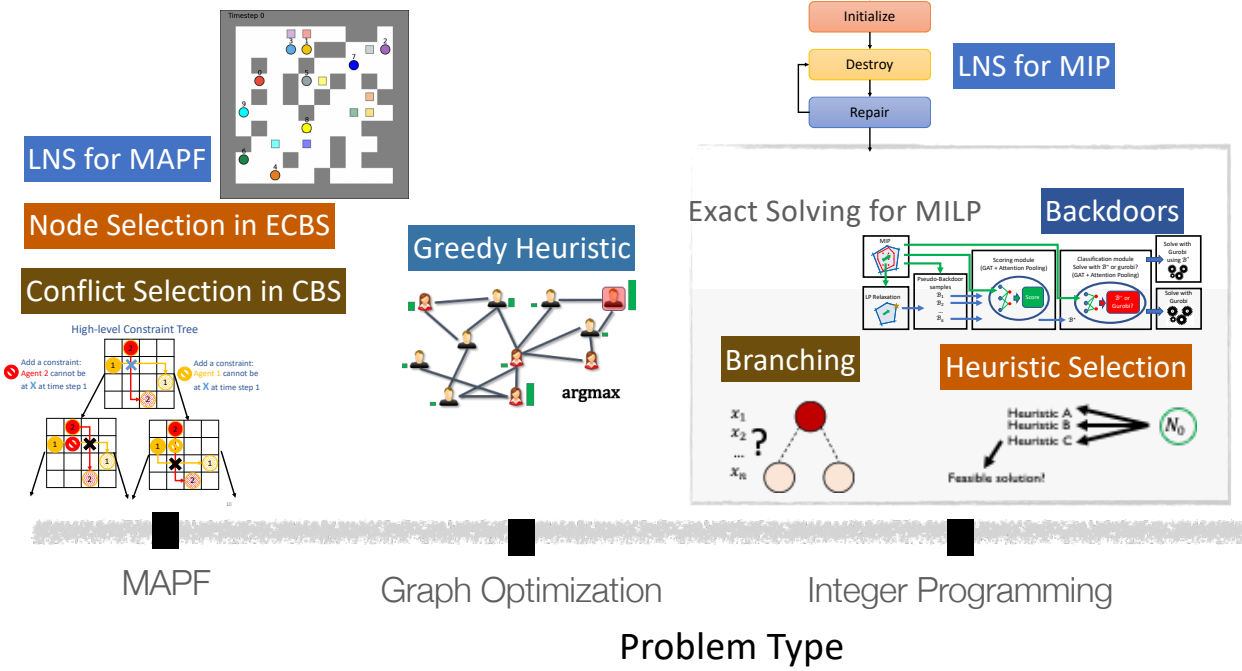
Decision making problems of **larger size** and **new problem structure** drive the continued need to **improve combinatorial solving** methods



ML ↔ Combinatorial Optimization

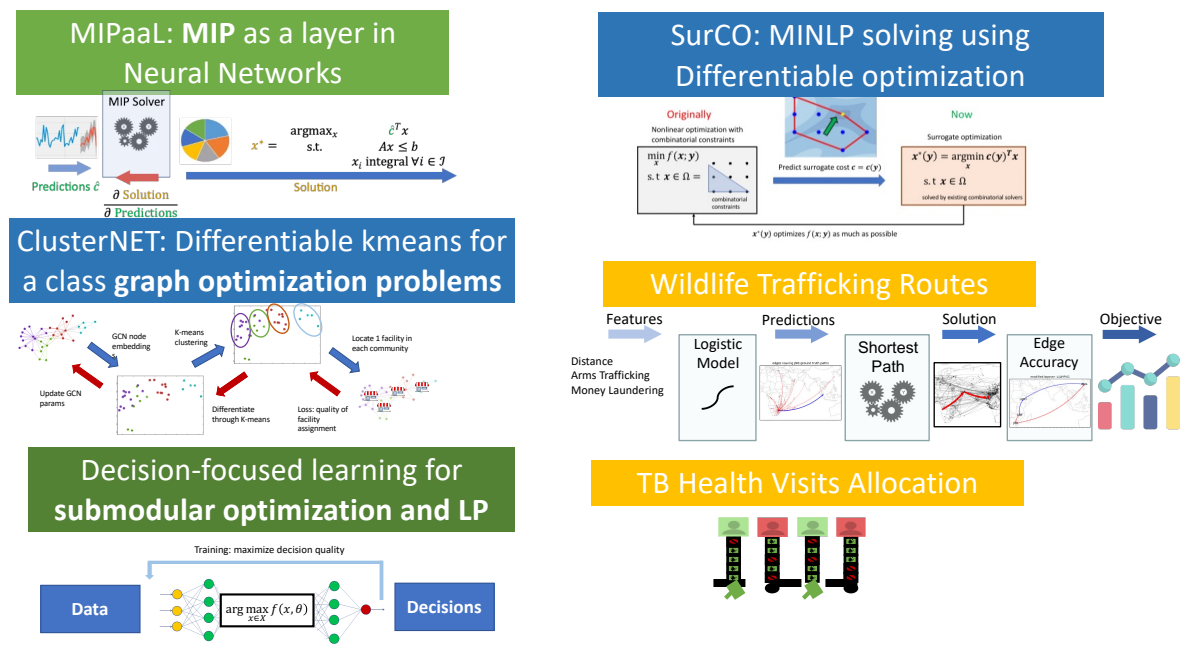
► Exciting and growing research area

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

Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning

Taoan Huang, Aaron Ferber, Yuandong Tian, Bistra Dilkina, Benoit Steiner

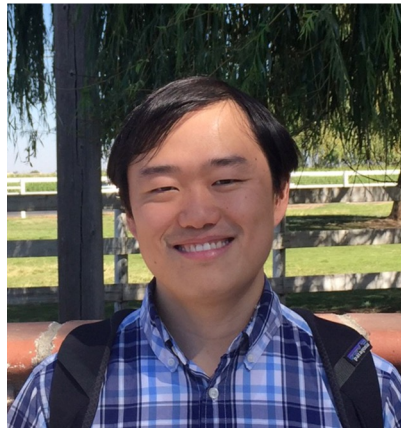
(USC)



(USC)



(Meta AI, FAIR)



(USC)



(Meta AI, FAIR)

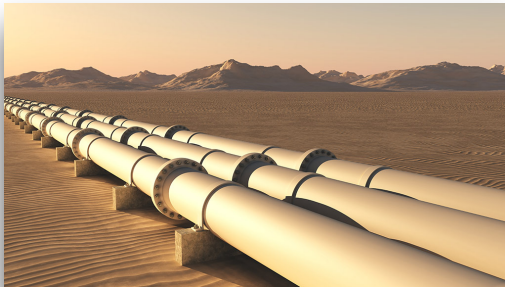


Mixed Integer Programs & their applications

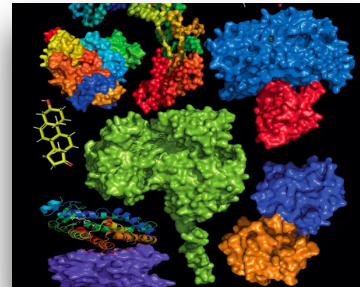
Flexible mathematical program framework

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

Energy Systems



Scientific Discovery



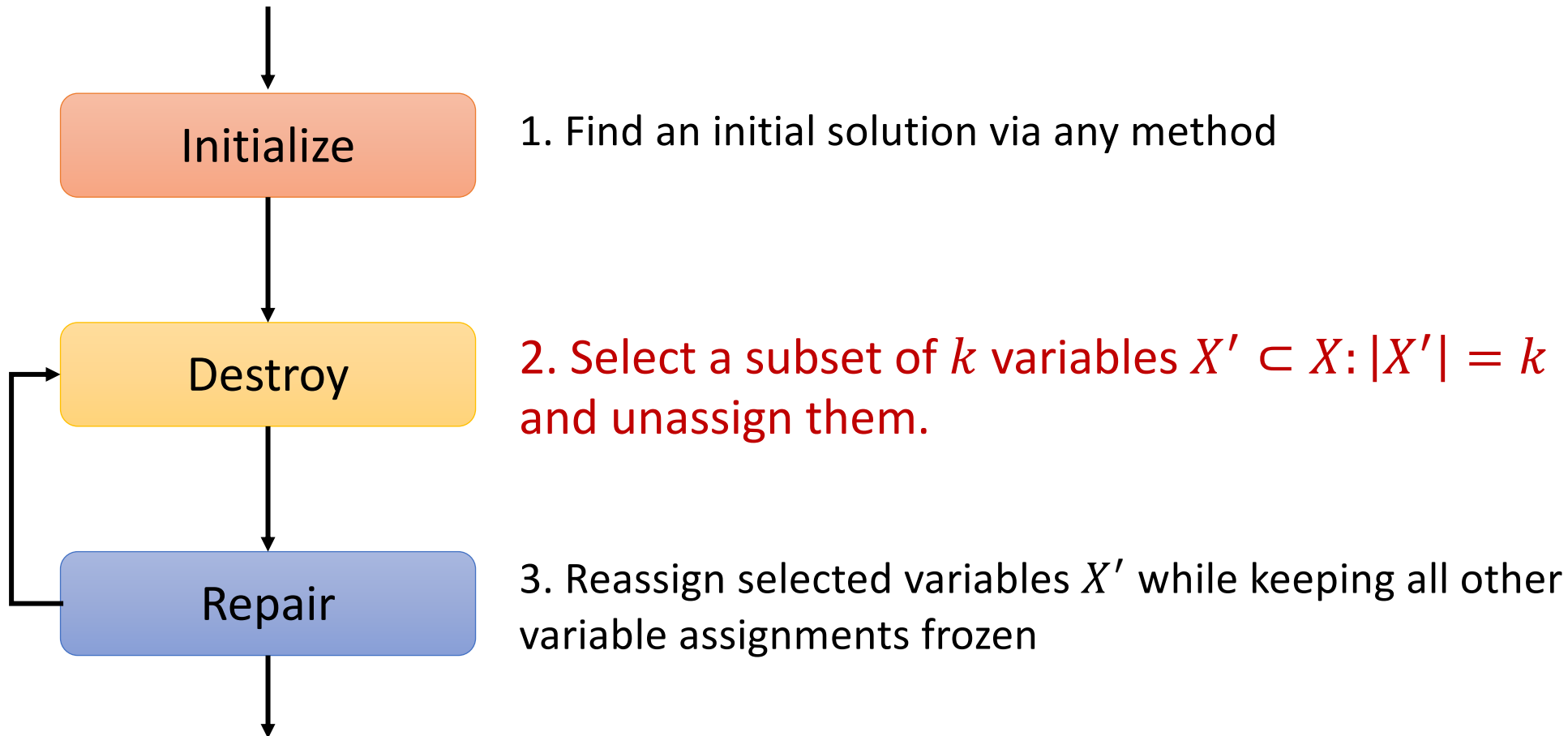
Conservation Planning



Disaster Response and Planning



Large Neighborhood Search (LNS)



Compared to other local search methods,
LNS explores a large neighborhood of possible next solutions in each step

LNS and its applications

- LNS for Constraint Programming
 - Perron et al, 2004, 2006; Berthold et al, 2012
- LNS for Vehicle Routing Problems
 - Ropke & Pisinger, 2006; Azi et al., 2014
- LNS for Traveling Salesman Problem
 - Smith & Imeson, 2017
- LNS for scheduling
 - Kovacs et al., 2012; Zulj et al., 2018
- LNS for path planning problems
 - Li et al., 2022; 2021
- LNS for Mixed Integer Problems
 - Fischetti & Lodi, 2003; Danna et al., 2005; Ghosh, 2007; Berthold, 2014; Maher et al., 2017; Hendel, 2022

Large Neighborhood Search (LNS) for MIP

Initialize

1. Find an initial solution via any method

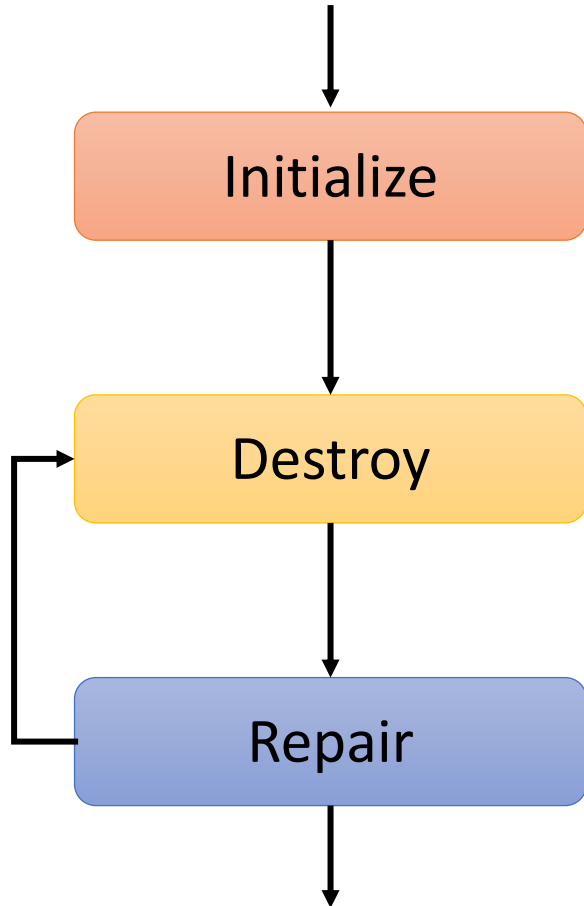
Opportunity for ML guidance

Destroy

2. Select a subset of k variables $X' \subset X: |X'| = k$ and unassign them.

Repair

3. Reassign selected variables X' while keeping all other variable assignments frozen:
solve sub-ILP over X'



Imitation Learning in Decomposition-based LNS

[Jialin Song, Ravi Lanka, Yisong Yue, Bistra Dilikina.

A General Large Neighborhood Search Framework for Solving Integer Linear Programs, NeurIPS, 2020]

The first work on applying ML-guided LNS to solve ILP

Decompose variables into **k equally sized variable subsets**, re-optimize each variable subset in turn

Algorithm 1 Decomposition-based LNS

- 1: **Input:** an optimization problem P , an initial solutions S_X , a decomposition $X = X_1 \cup X_2 \cup \dots \cup X_k$, a solver F
 - 2: **for** $i = 1, \dots, k$ **do**
 - 3: $S_X = \text{FIX_AND_OPTIMIZE}(P, S_X, X_i, F)$
 - 4: **end for**
 - 5: **return** S_X
-

Learn predict good decompositions from offline data

with imitation learning using behavior cloning (BC-LNS) and forward training (FT-LNS)

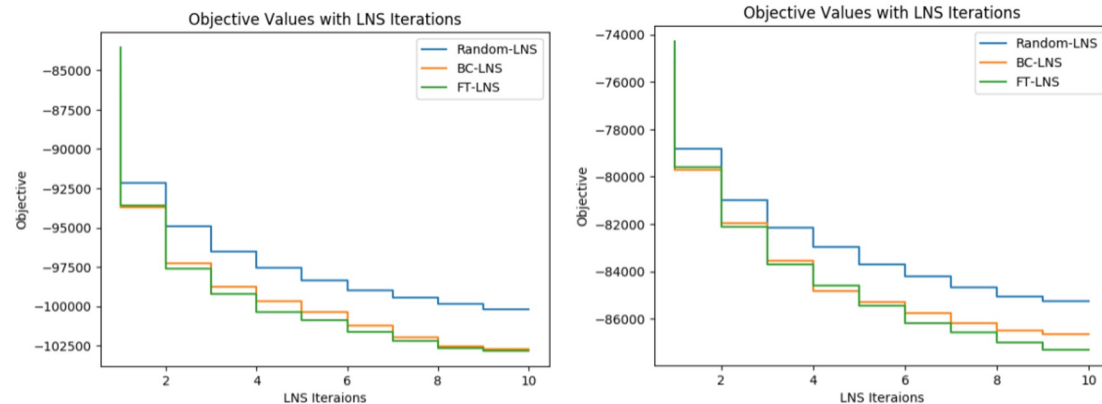
Data collection: given current solution, sample multiple random decompositions and evaluate solution improvement, add the best decomposition for training set for imitation learning

	MVC BA 1000	MVC ER 1000	MAXCUT BA 500	MAXCUT ER 500
Gurobi	440.08 \pm 1.26	482.15 \pm 0.82	-3232.53 \pm 16.61	-4918.07 \pm 12.43
Random-LNS	433.59 \pm 0.51	471.21 \pm 0.36	-3583.63 \pm 3.81	-5488.49 \pm 6.60
BC-LNS	433.09 \pm 0.53	470.20 \pm 0.34	-3584.90 \pm 4.02	-5494.76 \pm 6.51
FT-LNS	432.00 \pm 0.52	470.04 \pm 0.37	-3586.29 \pm 3.33	-5496.29 \pm 6.69
RL-LNS	434.16 \pm 0.38	471.52 \pm 0.15	-3584.70 \pm 1.49	-5481.57 \pm 2.97

Table 1: Comparison of different LNS methods and Gurobi for MVC and MAXCUT problems.

	CATS Regions 2000	CATS Regions 4000	CATS Arbitrary 2000	CATS Arbitrary 4000
Gurobi	-94559.9 \pm 2640.2	-175772.9 \pm 2247.89	-69644.8 \pm 1796.9	-142168.1 \pm 4610.0
Random-LNS	-99570.1 \pm 790.5	-201541.7 \pm 1131.1	-85276.6 \pm 680.9	-170228.3 \pm 1711.7
BC-LNS	-101957.5 \pm 752.7	-207196.2 \pm 1143.8	-86659.6 \pm 720.2	-172268.1 \pm 1594.8
FT-LNS	-102247.9 \pm 709.0	-208586.3 \pm 1211.7	-87311.8 \pm 676.0	-169846.7 \pm 5293.2

Table 2: Comparison of different LNS methods and Gurobi for CATS problems.



(a) CATS with 2000 items and 4000 bids from regions distribution.

(b) CATS with 2000 items and 4000 bids from arbitrary distribution.

Large Neighborhood Search (LNS) for MIP

Initialize

1. Find an initial solution via any method

Opportunity for ML guidance

Destroy

2. Select a subset of k variables $X' \subset X: |X'| = k$ and unassign them.

Repair

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

Algorithm 1 LNS for ILPs

1: **Input:** An ILP.

2: $\mathbf{x}^0 \leftarrow$ Find an initial solution to the input ILP

3: $t \leftarrow 0$

4: **while** time limit not exceeded **do**

5: $\mathcal{X}^t \leftarrow$ Select a subset of variables to destroy

6: $\mathbf{x}^{t+1} \leftarrow$ Solve the ILP with additional constraints $\{x_i = x_i^t : x_i \notin \mathcal{X}^t\}$

7: $t \leftarrow t + 1$

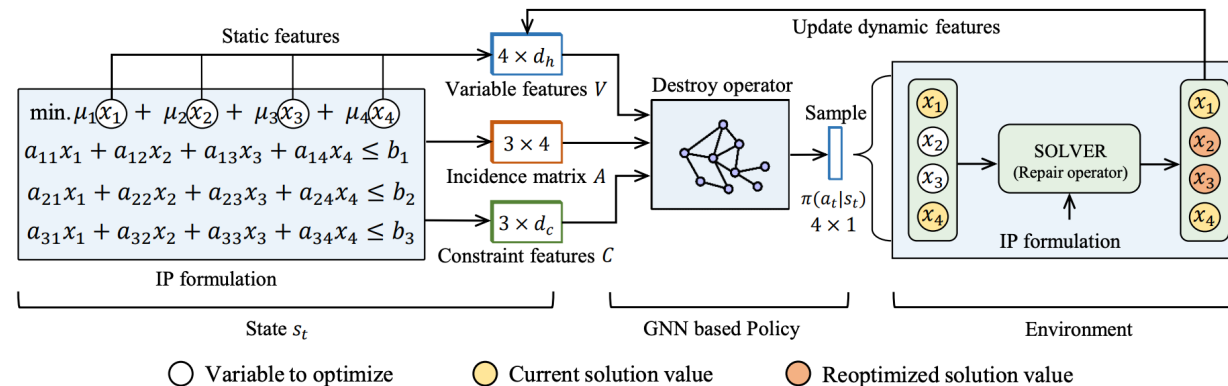
8: **return** \mathbf{x}^t

RL-LNS: Reinforcement Learning Approach in LNS for MIP

[Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang.

Learning Large Neighborhood Search Policy for Integer Programming . NeurIPS, 2021]

Learning to **select destroy sets by reinforcement learning**



Data collection: collected with training

Training: GCN over bipartite MIP graph with features from [Gasse et al, 2019],

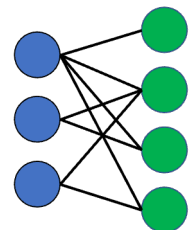
each variable is predicted separately (shared parameters)

trained with Actor-Critic

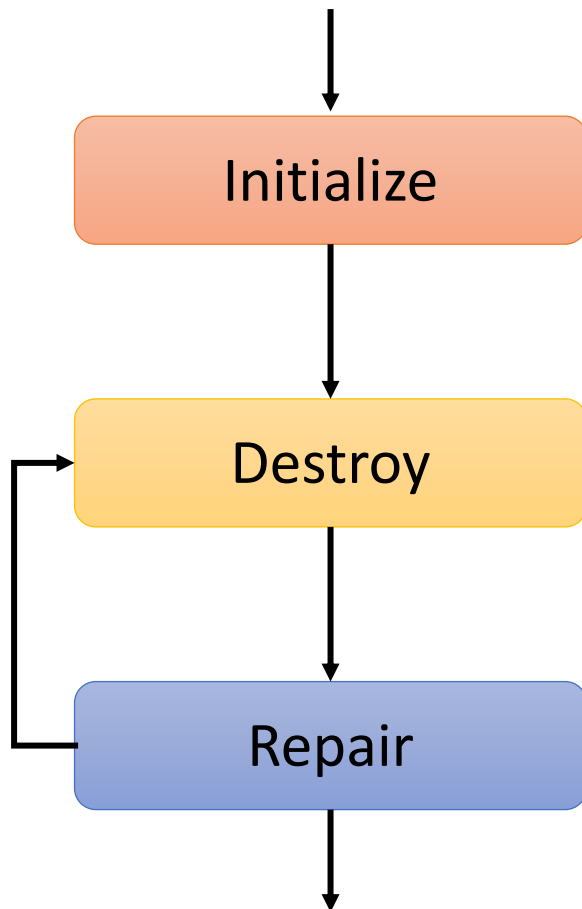
Test: compares favorably to decomposition-based R-LNS and FT-LNS,

as well as random U-LNS, SCIP and Gurobi

-Variable nodes -Constraint nodes



Large Neighborhood Search (LNS) for MIP



1. Find an initial solution via any method

Opportunity for ML guidance

Oracle: Local Branching (slow but good)

2. Select a subset of k variables $X' \subset X: |X'| = k$ and unassign them.

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

Algorithm 1 LNS for ILPs

1: **Input:** An ILP.

2: $\mathbf{x}^0 \leftarrow$ Find an initial solution to the input ILP

3: $t \leftarrow 0$

4: **while** time limit not exceeded **do**

5: $\mathcal{X}^t \leftarrow$ Select a subset of variables to destroy

6: $\mathbf{x}^{t+1} \leftarrow$ Solve the ILP with additional constraints $\{x_i = x_i^t : x_i \notin \mathcal{X}^t\}$

7: $t \leftarrow t + 1$

8: **return** \mathbf{x}^t

Local Branching (Fischetti & Lodi, 2003)

LNS-destroy: Given an ILP and a feasible solutions x^* ,
choose at most k variables to reoptimize while fixing the rest

How to find the **optimal** subset of k variables?
(the k variables to select, which maximize the objective value of the repaired solution)

Local Branching: solve a MIP 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

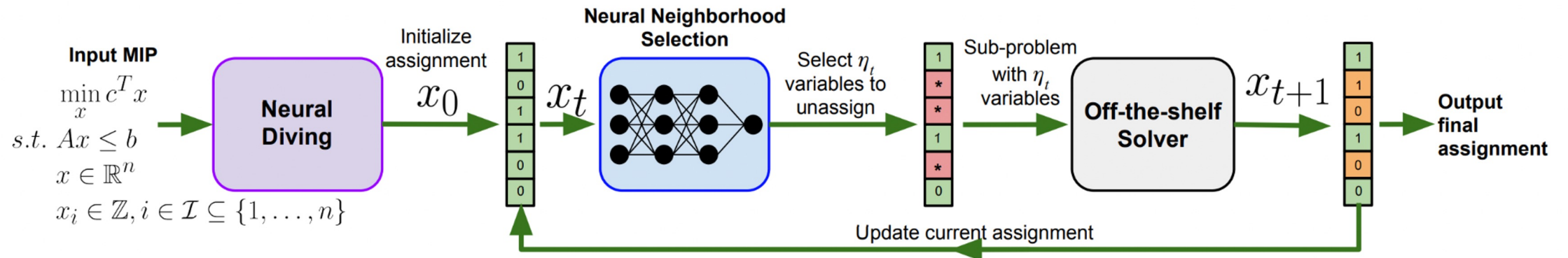
$$\sum_{i:x_i^*=0} x_i + \sum_{i:x_i^*=1} (1 - x_i) \leq k$$

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

IL-LNS: Imitation Learning Approach in LNS for MIP

[Sonnerat, N., Wang, P., Ktena, I., Bartunov, S., and Nair, V. Learning a large neighborhood search algorithm for mixed integer programs". arXiv preprint, 2021]

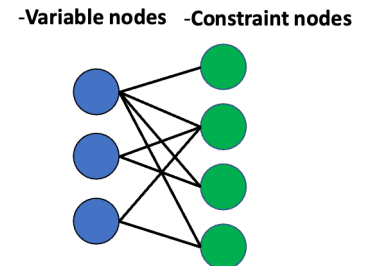
Learning to imitate Local Branching



Data collection: solve MIPs for Local Branching (2-3 hours each)

Training: GCN over bipartite MIP graph with features from [Gasse et al, 2019],
 each variable is predicted separately (shared parameters)

Test: combines Initial Solution by Neural Diving + IL-LNS, compares to SCIP



Summary: ML in LNS for MIP

Method	Representation	Neural Architecture	Action	Training method
Song et al. NeurIPS 2020	PCA of A matrix	MLP	Partition variables into p sets	Imitating best sampled partition
Sonnerat et al. 2021 IL-LNS	Variable-constraint graph	GCN	Select subset of k variables	Imitating oracle (local branching)
Wu et al. NeurIPS 2021 RL-LNS	Variable-constraint graph	GCN	Select subset of k variables	Standard RL

- Liu, Fischetti, Lodi. Learning to search in local branching. AAAI 2022 - ML to tune the runtime limit and neighborhood sizes for Local Branching.

Learn the Repair heuristic for Routing Problems

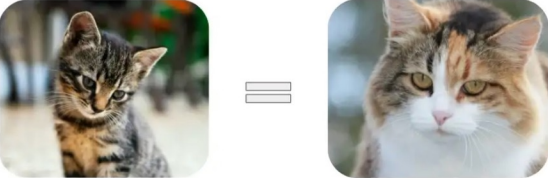
- André Hottung, Kevin Tierney. Neural large neighborhood search for routing problems. Artificial Intelligence 2022
- Falkner et al. Large Neighborhood Search based on Neural Construction Heuristics, 2022

Our approach: CL-LNS


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***

Contrastive pairs



Positive Pair



Negative Pair

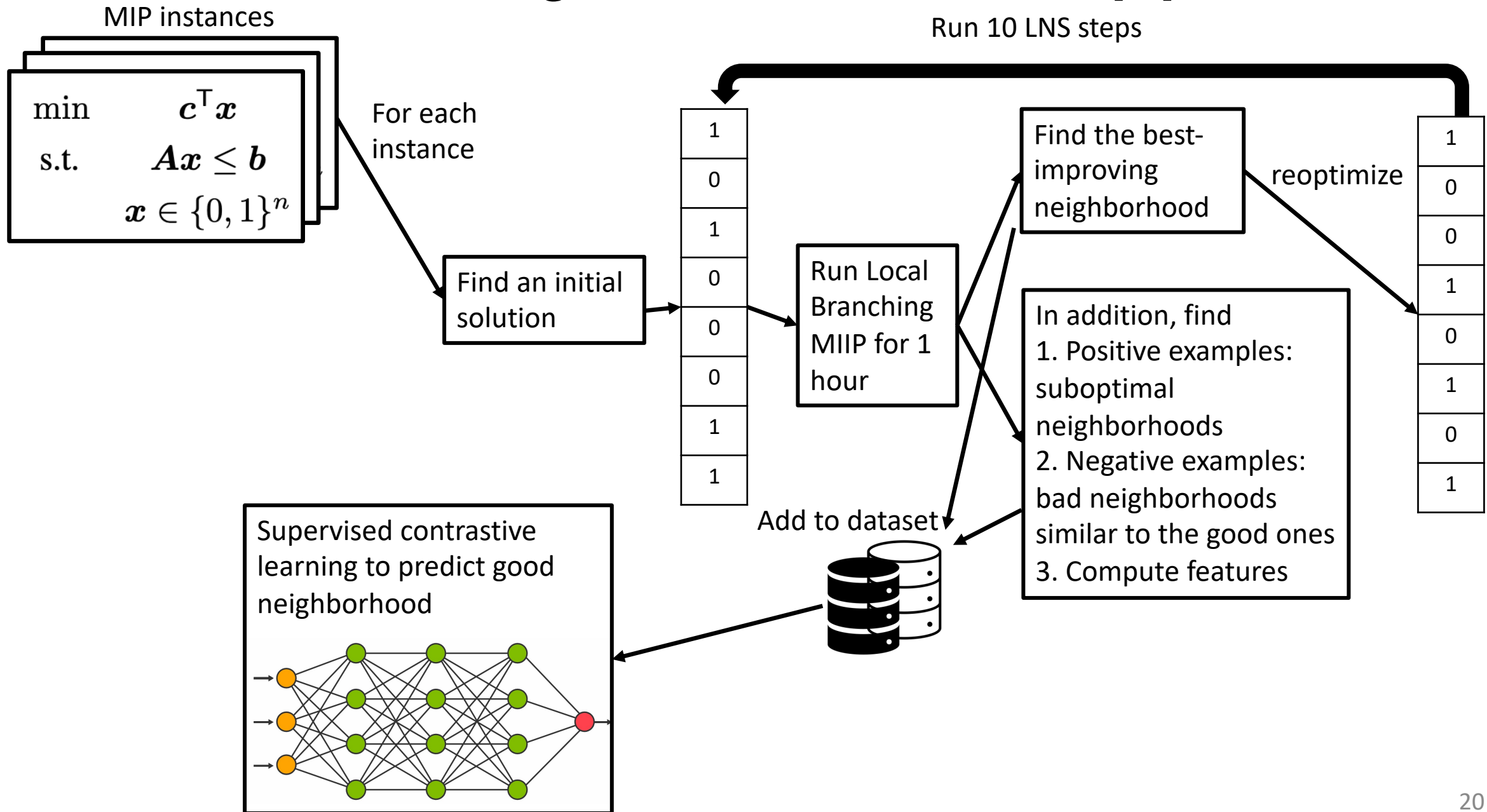
$$l(\text{img}_1, \text{img}_2) = -\log \left(\frac{e^{\text{similarity}(\text{img}_1, \text{img}_2)}}{e^{\text{similarity}(\text{img}_1, \text{img}_2)} + e^{\text{similarity}(\text{img}_1, \text{img}_3)} + e^{\text{similarity}(\text{img}_1, \text{img}_4)}} \right)$$

A contrastive loss is a function whose value is low when the predicted action is similar to the positive samples and dissimilar to the negative samples (similarity measured by dot products)

Prior Work on Contrastive Learning for COP

- 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) derive a contrastive loss for decision-focused learning to solve COPs with uncertain inputs that can be learned from historical data
- Duan et al. (2022) use contrastive pre-training to learn good representations for SAT.

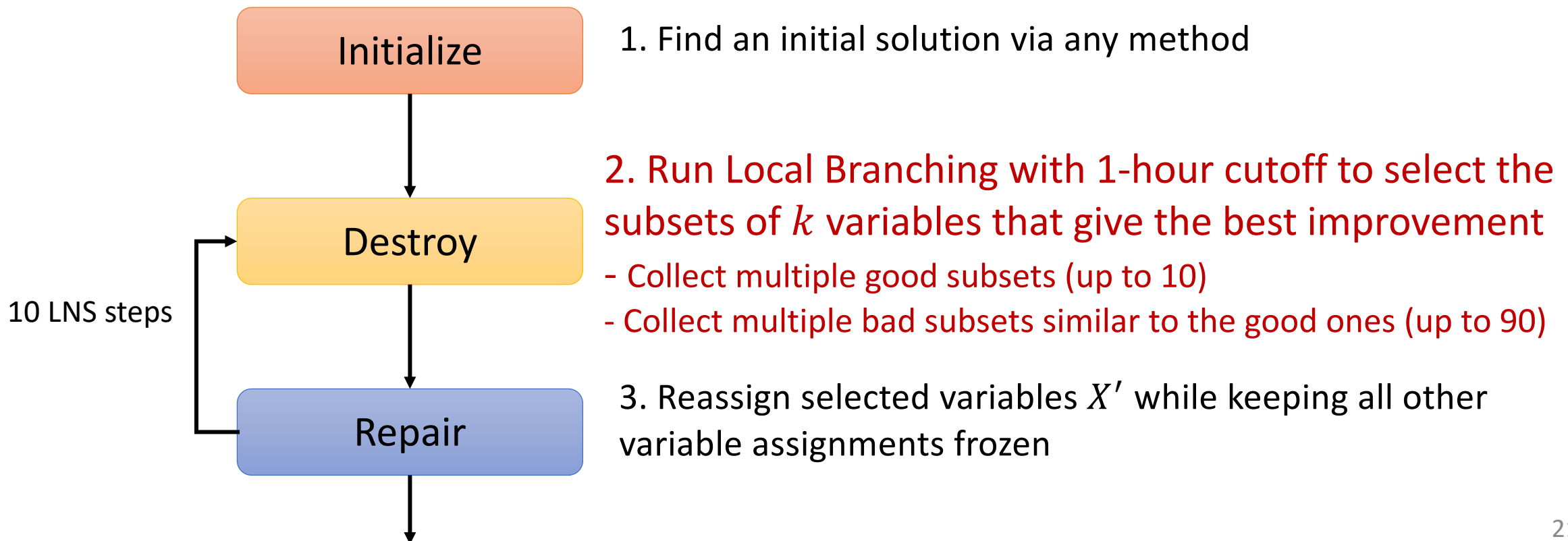
Training and data collection pipeline



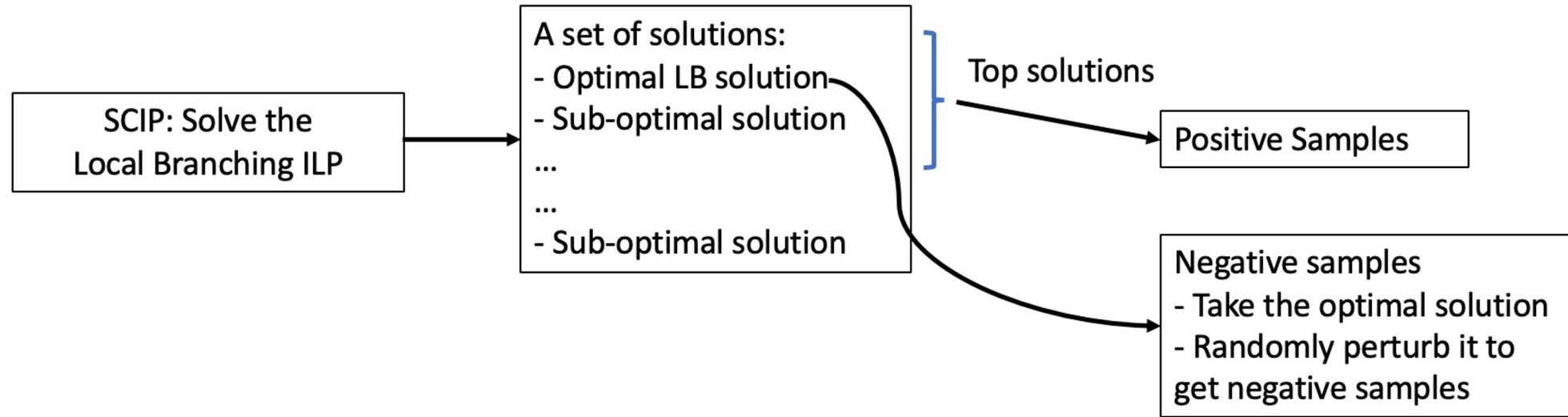
Contrastive learning - Data collection

For each training instance, we use the following procedure to collect data

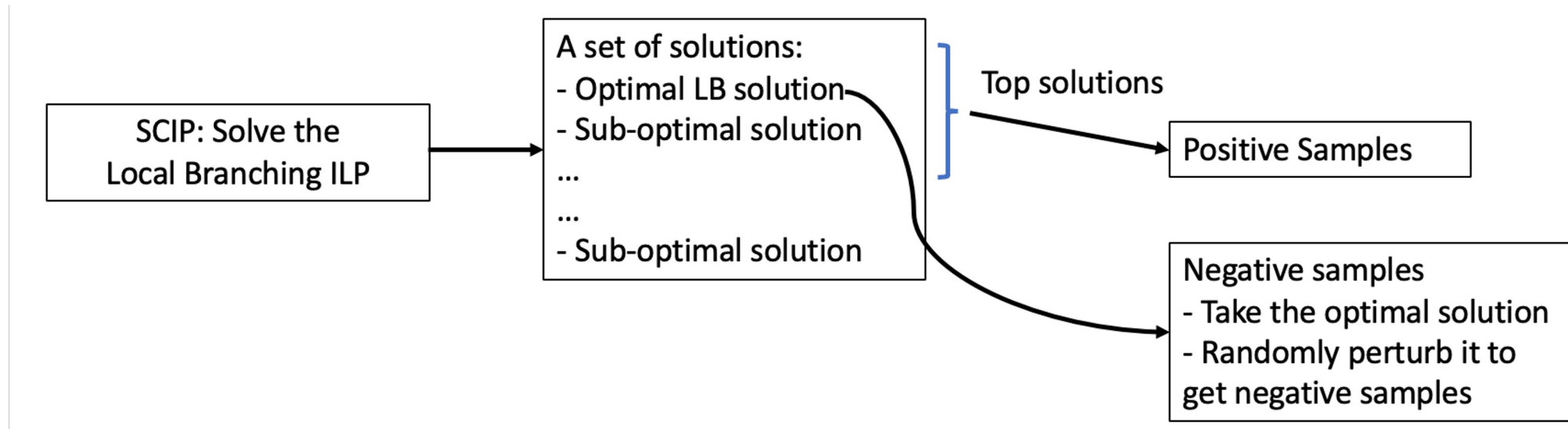
- LNS + an exhaustive Local Branching search in the destroy step



Contrastive learning - Data collection details



Contrastive learning - Data collection details



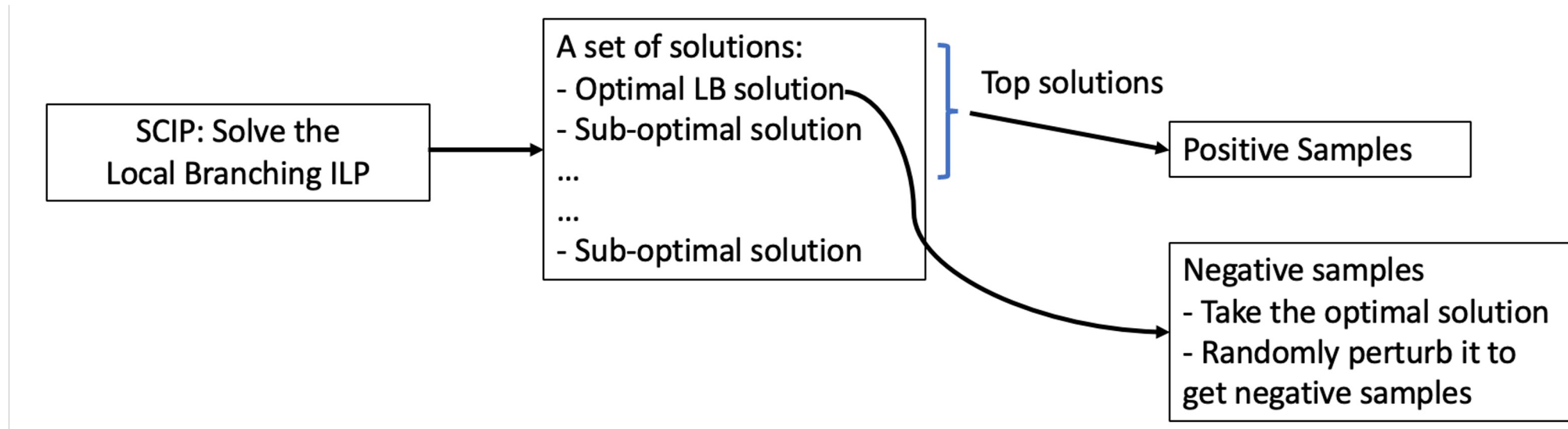
Optimal sample := variables changed in the optimal solution found by
local branching solved by SCIP within 1 hours

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

Positive samples:

- All solutions found by local branching with *improvement* $\geq 0.5 * best_improve$
- up to max of 10 solutions

Contrastive learning - 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 ($P * \text{Num Positive Samples}$, $P=9$):

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

Features + ML Architecture

Variable features:

Static features from MIP:

[Gasse et al 2019] features (23)

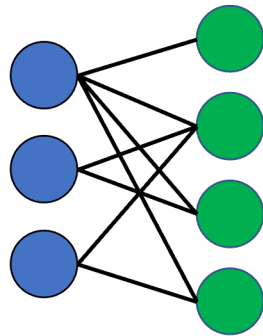
+ [Khalil et al 2017] features (72) – not used in [Sonnerat et al, 2021]

Dynamic features: Include the features of past W incumbent solutions, Feature size = $W * 3$

Features: incumb exists, incumbent value, LB-relax value

Edge features and **constraint** features: Static from MIP, the same as used in previous work

-Variable nodes -Constraint nodes



Features

Gasse et al 2019

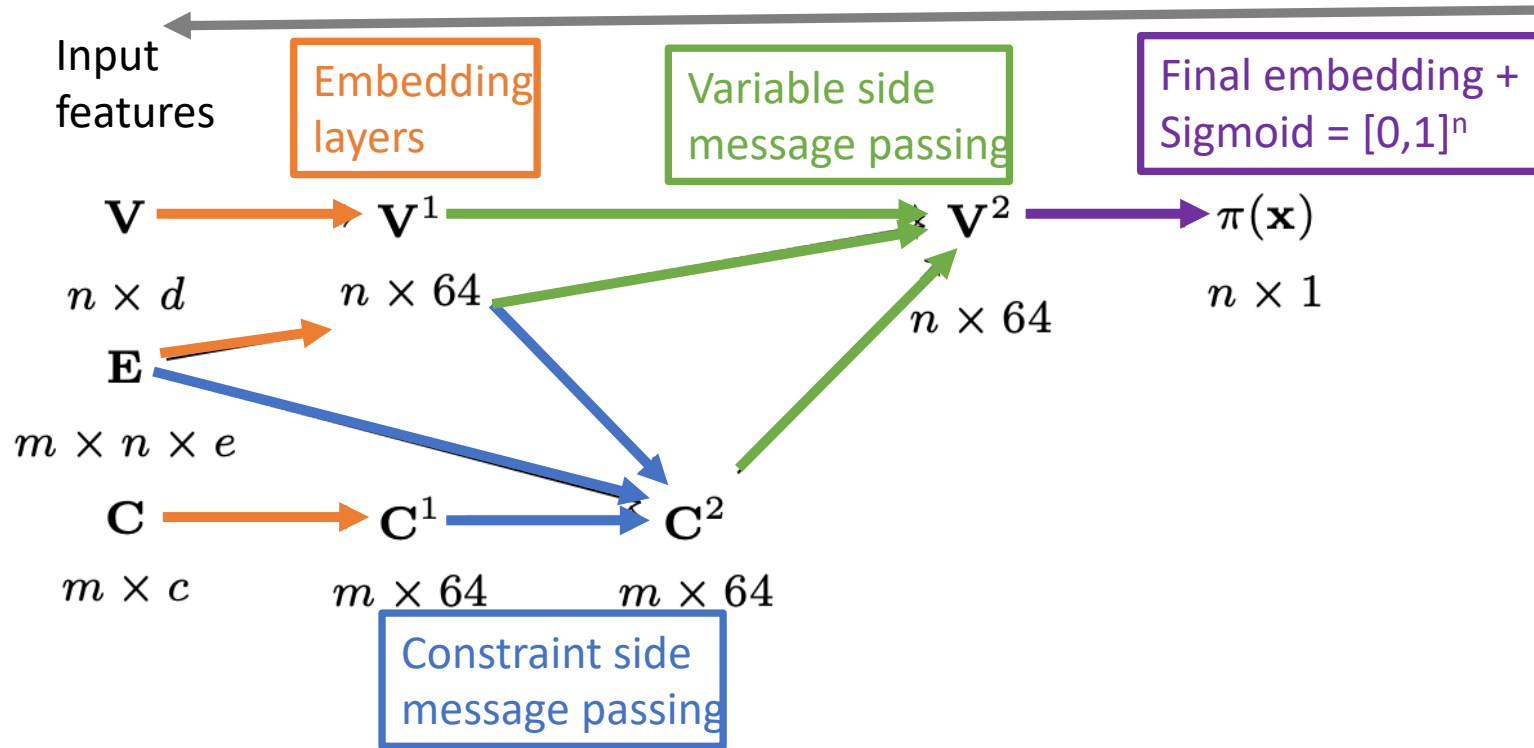
Category	Name (count)	Description
V Vars	type (3)	Variable type (continuous, binary, integer)
	obj_coef	The decision variable's objective coefficient
	has_lb	Does the variable have a lower bound?
	has_ub	Does the variable have an upper bound?
	root_lp_at_lb	Variable at it's lower bound in the root LP?
	root_lp_at_ub	Variable at it's upper bound in the root LP?
	root_lp_frac	Is the variable fractional in the root LP?
	lp_basis (4)	Variable root LP status (basic, lower, upper, superbasic)
E Edges	coef	Constraint coefficient
C Constrs	obj_cos_sim	Cosine similarity of objective and constraint
	bias	Constant bias
	root_lp_tight	Constraint tightness in the root LP
	root_lp_dual	Dual value In the root LP
	sense (3)	Constraint direction ($\leq, \geq, =$)

Khalil et al 2017 (at root)

Feature	Description	Count	Reference
<i>Static Features (18)</i>			
Objective function coeffs.	Value of the coefficient (raw, positive only, negative only)	3	
Num. constraints	Number of constraints that the variable participates in (with a non-zero coefficient)	1	
Stats. for constraint degrees	The <i>degree of a constraint</i> is the number of variables that participate in it. A variable may participate in multiple constraints, and statistics over those constraints' degrees are used. The constraint degree is computed on the root LP (mean, stdev., min, max)	4	
Stats. for constraint coeffs.	A variable's positive (negative) coefficients in the constraints it participates in (count, mean, stdev., min, max)	10	
<i>Dynamic Features (54)</i>			
Slack and ceil distances	$\min\{\bar{x}_j^i - \lfloor \bar{x}_j^i \rfloor, \lceil \bar{x}_j^i \rceil - \bar{x}_j^i\}$ and $\lceil \bar{x}_j^i \rceil - \bar{x}_j^i$	2	
Pseudocosts	Upwards and downwards values, and their corresponding ratio, sum and product, weighted by the fractionality of x_j	5	(Achterberg 2009)
Infeasibility statistics	Number and fraction of nodes for which applying SB to variable x_j led to one (two) infeasible children (during data collection)	4	
Stats. for constraint degrees	A dynamic variant of the static version above. Here, the constraint degrees are on the current node's LP. The ratios of the static mean, maximum and minimum to their dynamic counterparts are also features	7	
Min/max for ratios of constraint coeffs. to RHS	Minimum and maximum ratios across positive and negative right-hand-sides (RHS)	4	(Alvarez, Louveaux, and Wehenkel 2014)
Min/max for one-to-all coefficient ratios	The statistics are over the ratios of a variable's coefficient, to the sum over all other variables' coefficients, for a given constraint. Four versions of these ratios are considered: positive (negative) coefficient to sum of positive (negative) coefficients	8	(Alvarez, Louveaux, and Wehenkel 2014)
Stats. for active constraint coefficients	An active constraint at a node LP is one which is binding with equality at the optimum. We consider 4 weighting schemes for an active constraint: unit weight, inverse of the sum of the coefficients of all variables in constraint, inverse of the sum of the coefficients of only candidate variables in constraint, dual cost of the constraint. Given the absolute value of the coefficients of x_j in the active constraints, we compute the sum, mean, stdev., max. and min. of those values, for each of the weighting schemes. We also compute the weighted number of active constraints that x_j is in, with the same 4 weightings	24	(Patel and Chinneck 2007)

Table 1: Description of the atomic features.

Contrastive learning – ML Architecture



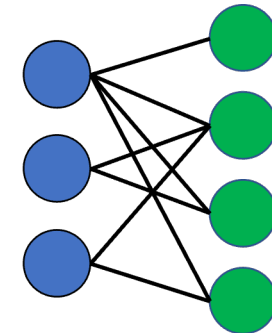
Training data for one LNS step

V, E, C Features

S_{pos} Positive Samples $a = \{0,1\}^n$

S_{neg} Negative samples $a' = \{0,1\}^n$

-Variable nodes -Constraint nodes

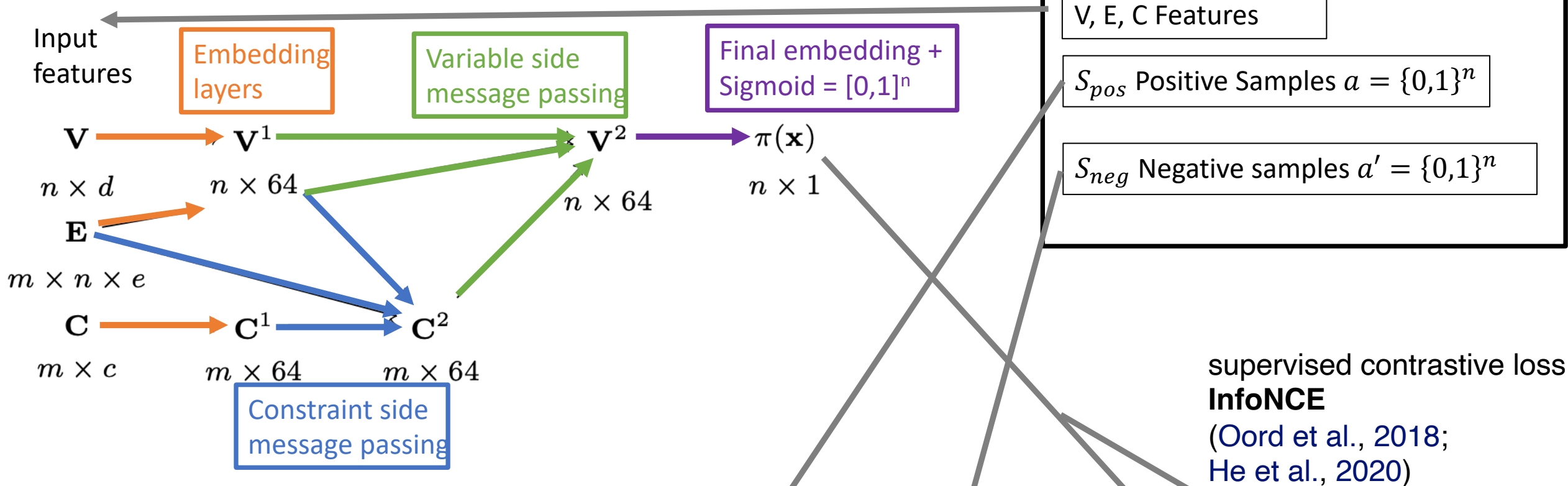


ML Architecture: ILP Graph + Embedding layers to $d=64$ + graph attention network (GAT) (Brody et al., 2022) with $H=8$ attention heads + two rounds of message passing + MLP + sigmoid $\rightarrow [0,1]$ score per variable

We use the same message-passing mechanism in previous work (Gasse et al., 2019)

We replace convolution layers with attention layers

Contrastive learning – Loss Function



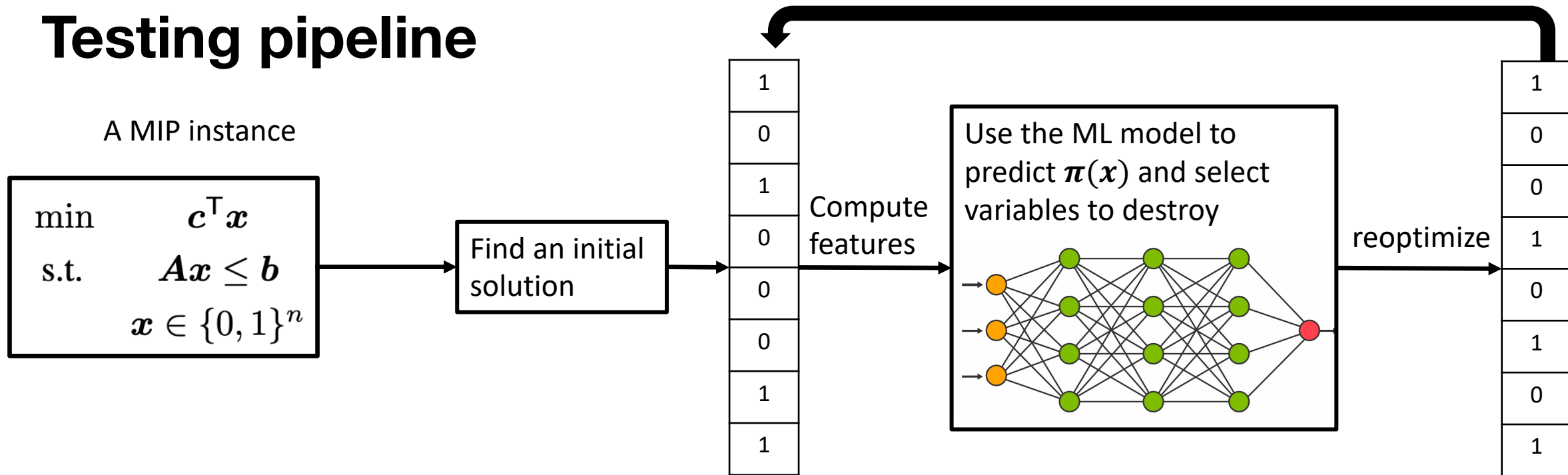
Loss computation:

The final loss is summed over multiple LNS steps in the batch

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

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

Testing pipeline



Neighborhood selection during testing:

- Given a neighborhood size k
- **Greedy** choose k variables with the **largest embedding values $\pi(x)$**

Adaptive Neighborhood Size (Sonnerat et al.)

- When no improving solution is found, $k \leftarrow k \times \alpha$ where $\alpha > 1$
- We upper bound k by half of the number of variables

Experimental Setup

MVC: Minimum Vertex Cover

MIS: Maximum Independent Set

CA: Combinatorial Auction

SC: Minimum Set Cover

	Small Instances				Large Instances			
Name	MVC-S	MIS-S	CA-S	SC-S	MVC-L	MIS-L	CA-L	SC-L
#Variables	1,000	6,000	4,000	4,000	2,000	12,000	8,000	8,000
#Constraints	65,100	23,977	2,675	5,000	135,100	48,027	5,353	5,000

Training and Testing

Testing only

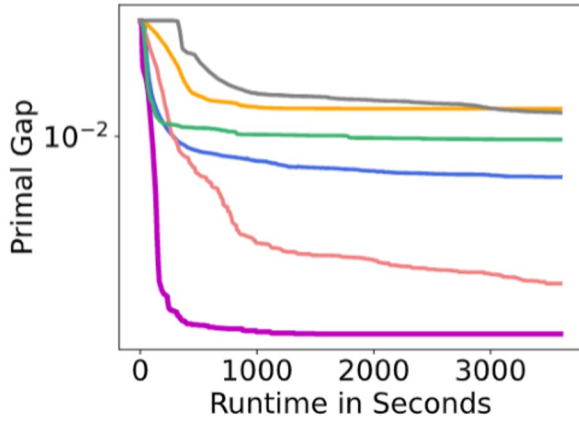
Baselines

- **BnB**: SCIP Branch and Bound solver (with the aggressive mode -improving the objective value)
- **RANDOM**: LNS with random neighborhood selection
- **LB-RELAX**: LNS with local branching relaxation heuristics [Huang et al, CPAIOR 2023]
- **IL-LNS**: SOTA imitation learning approach [Sonnerat et al, 2021]
- **RL-LNS**: SOTA reinforcement learning approach [Wu et al, 2021]

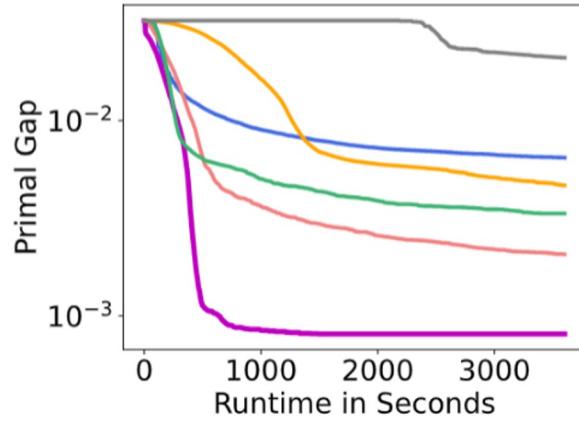
vs.

- **CL-LNS** (ours)

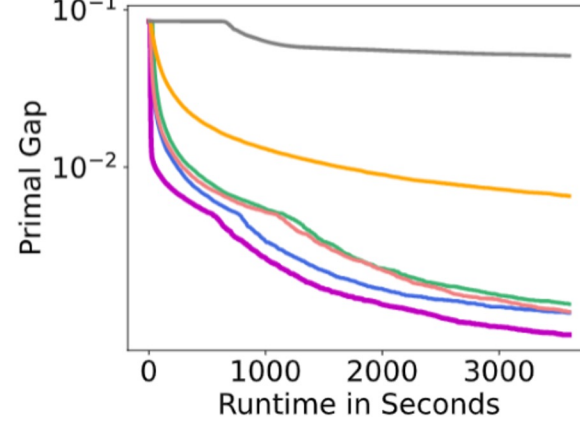
Results: Primal Gap over Time (secs)



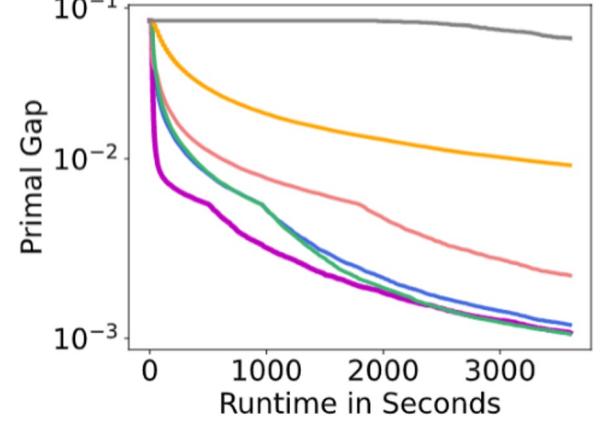
Min Vertex Set-S



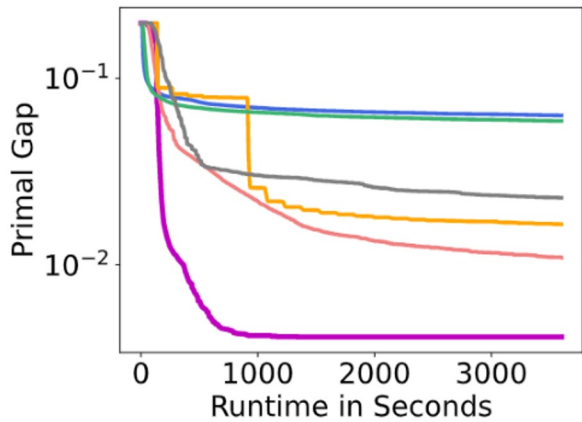
Min Vertex Set-L



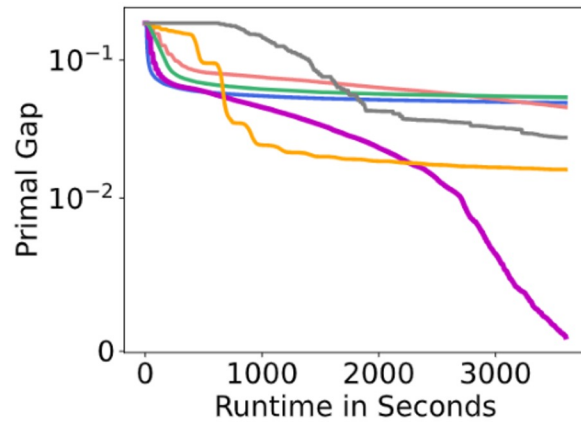
Max Independent Set-S



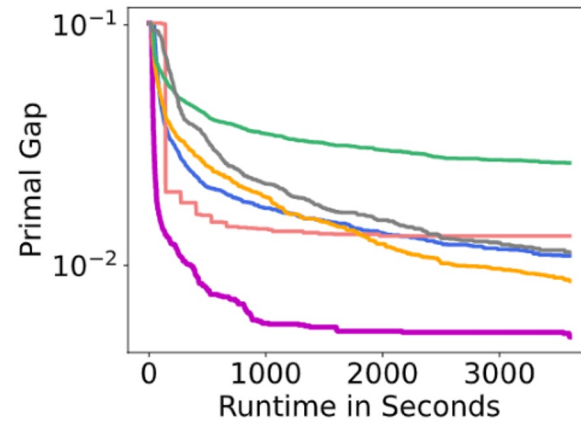
Max Independent Set-L



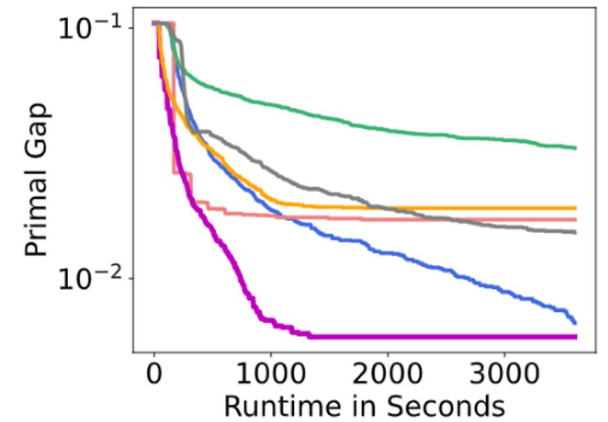
Combinatorial Auctions-S



Combinatorial Auctions-L

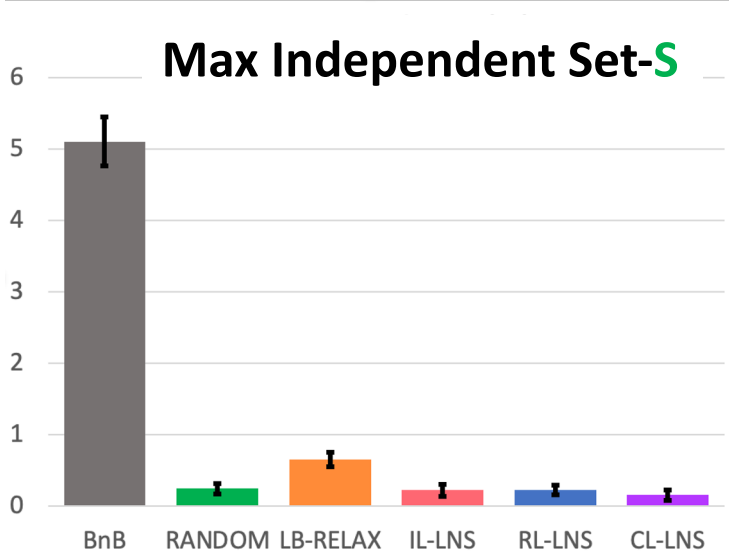
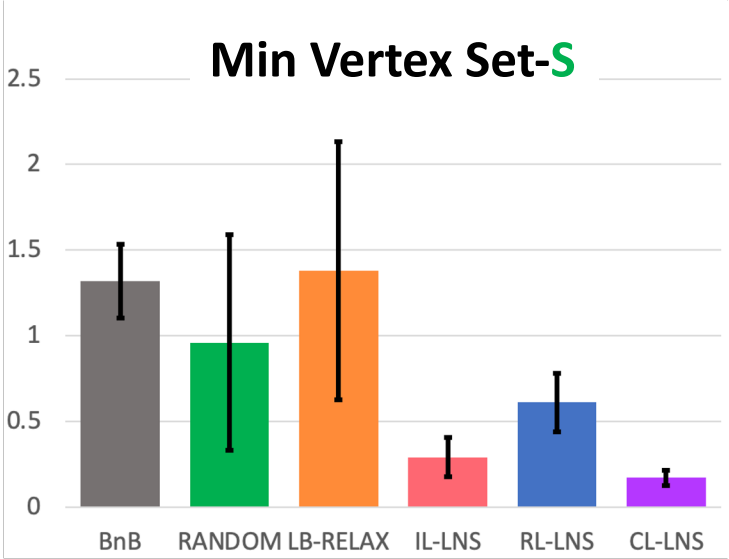


Min Set Cover-S

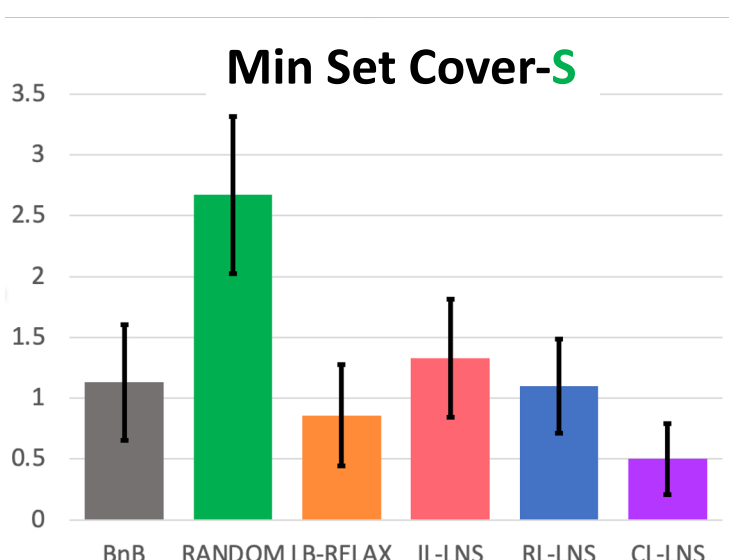
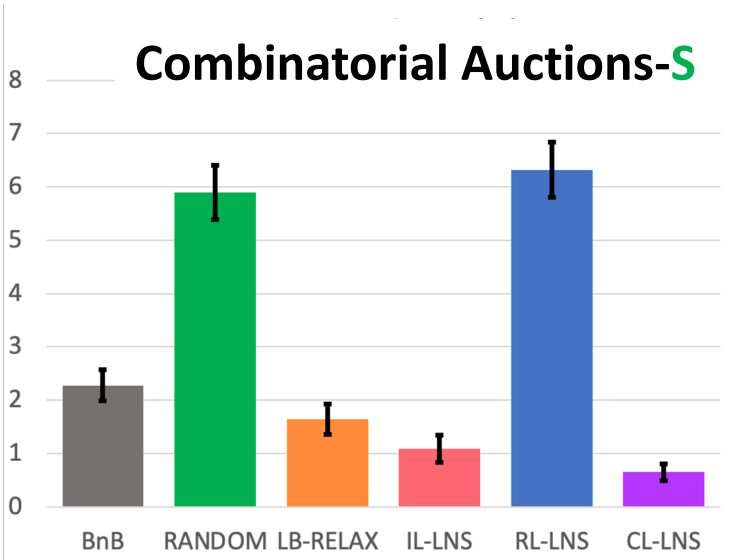


Min Set Cover-L

Primal gap % at 60 minutes - Small instances

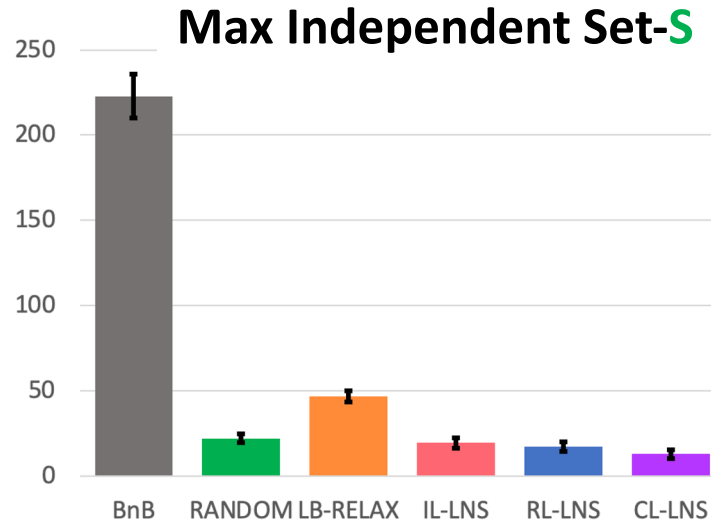
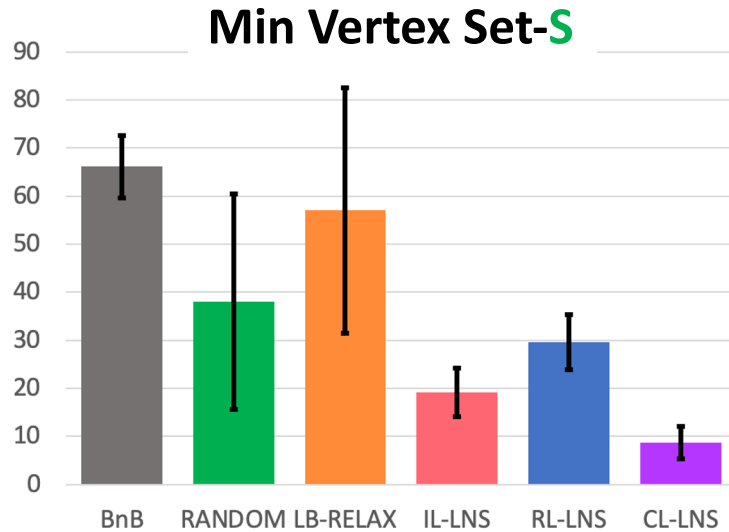


Primal Gap:
 normalized difference
 between the primal bound v
 (at time cutoff) and a
 precomputed best known
 objective value v^*
 as %



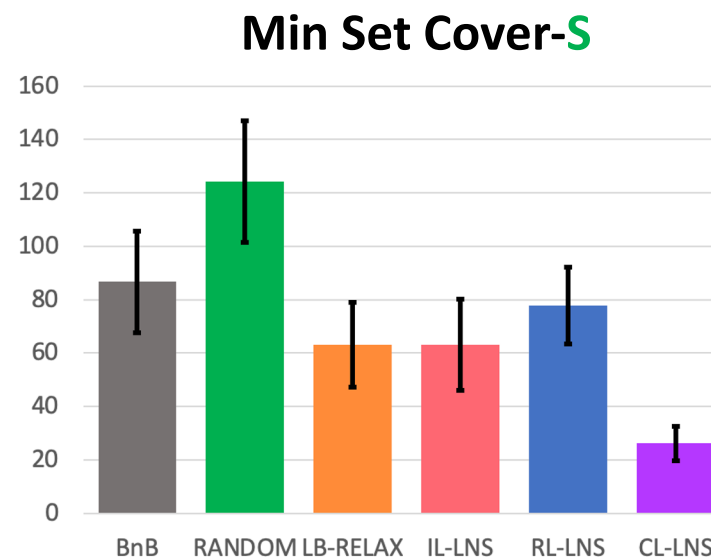
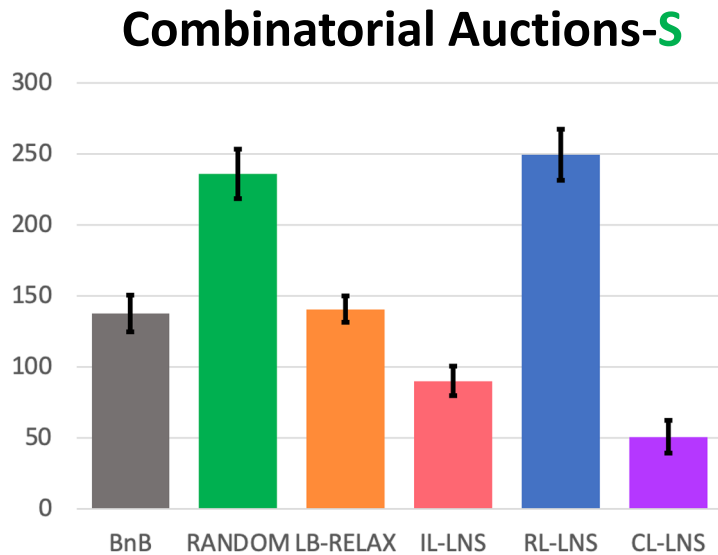
$$\frac{|v - v^*|}{\max(v, v^*, \epsilon)} * 100\%$$

Primal Integral at 60 minutes - Small instances

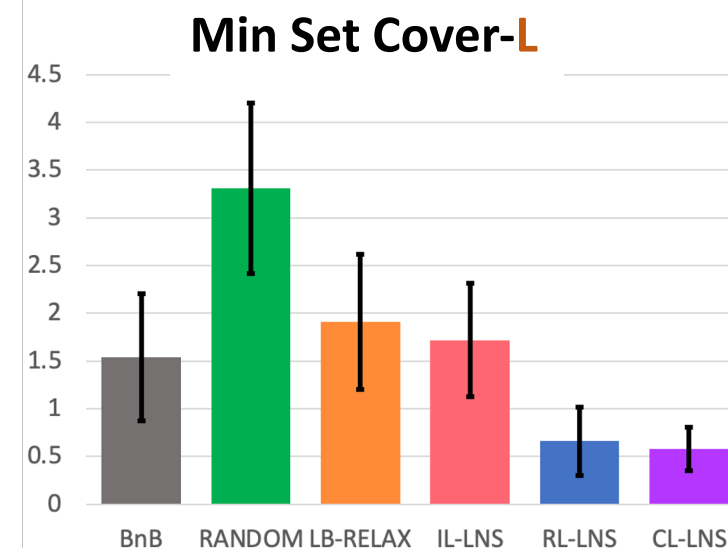
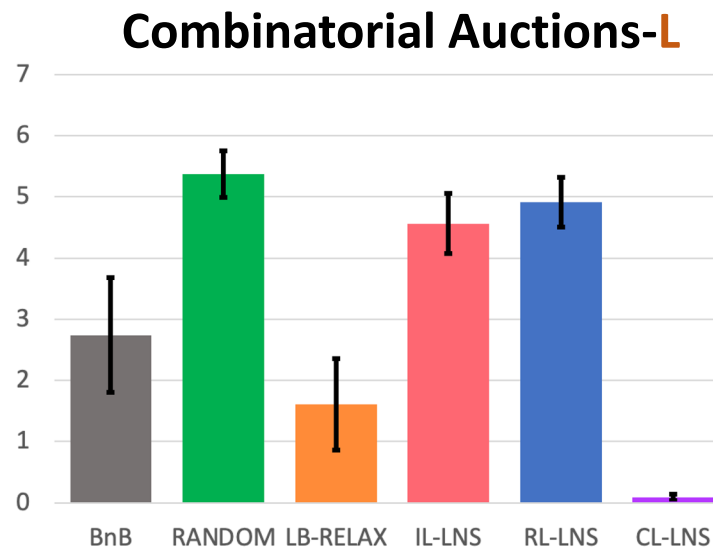
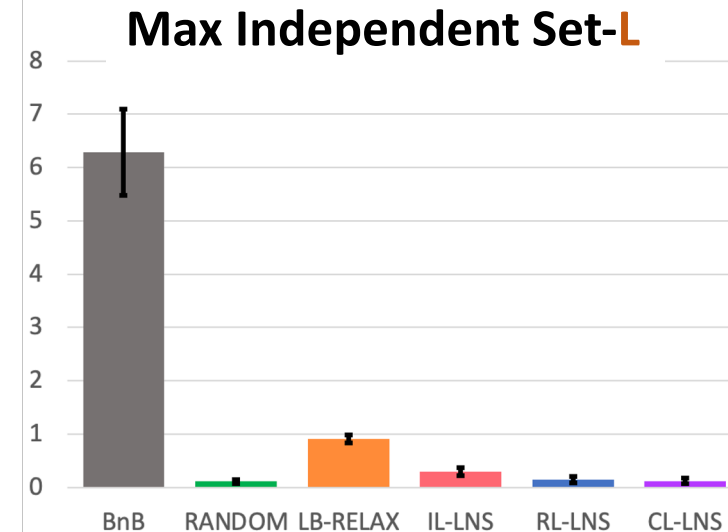
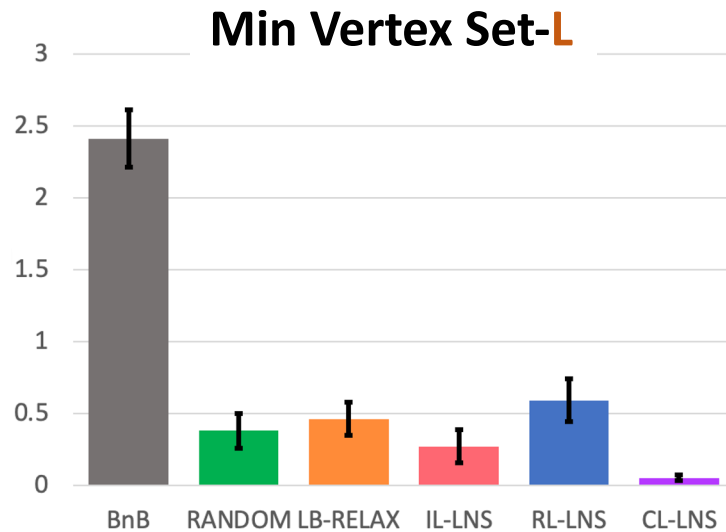


The **primal integral** at time q is the integral on $[0, q]$ of the primal gap as a function of runtime.

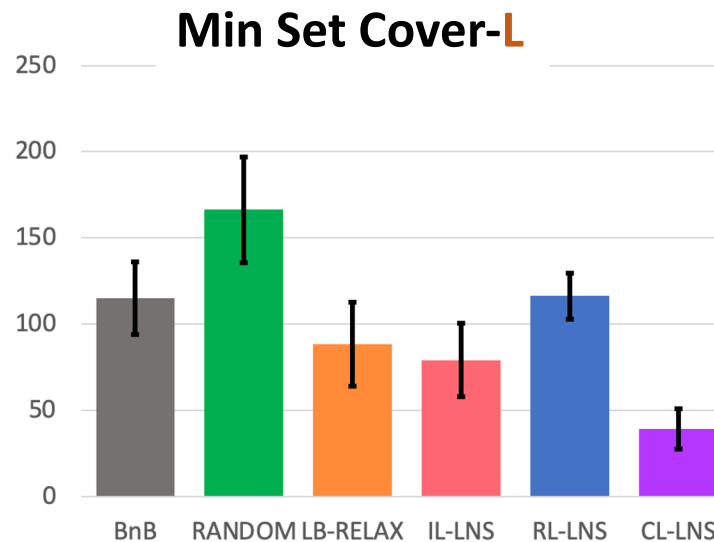
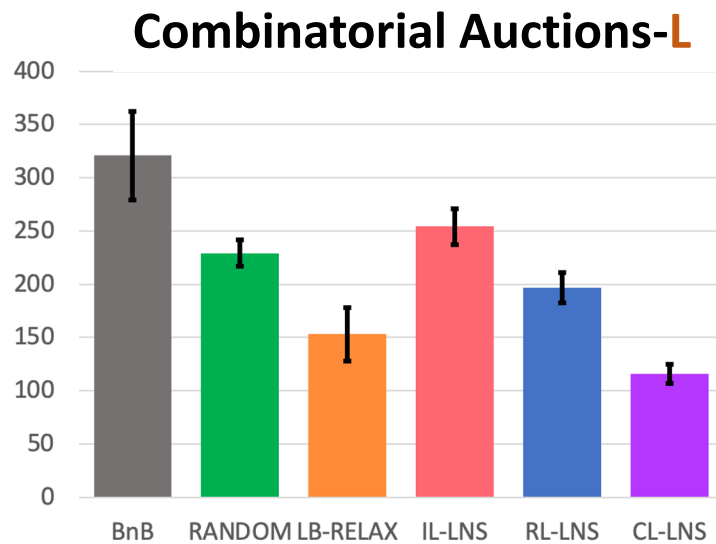
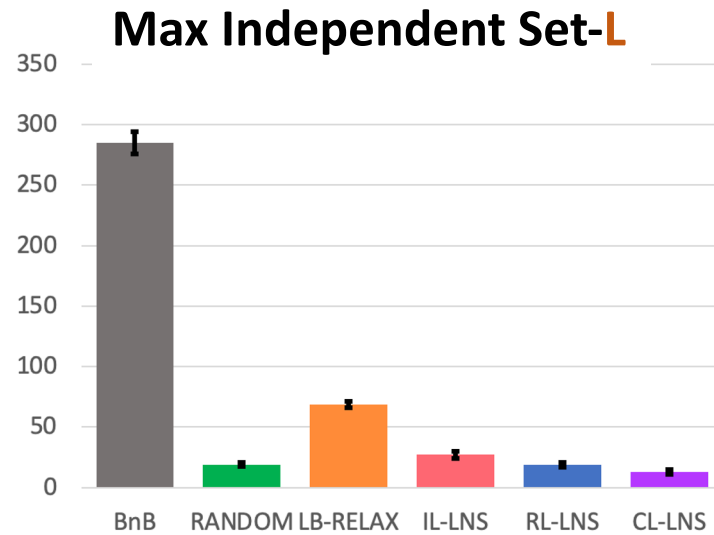
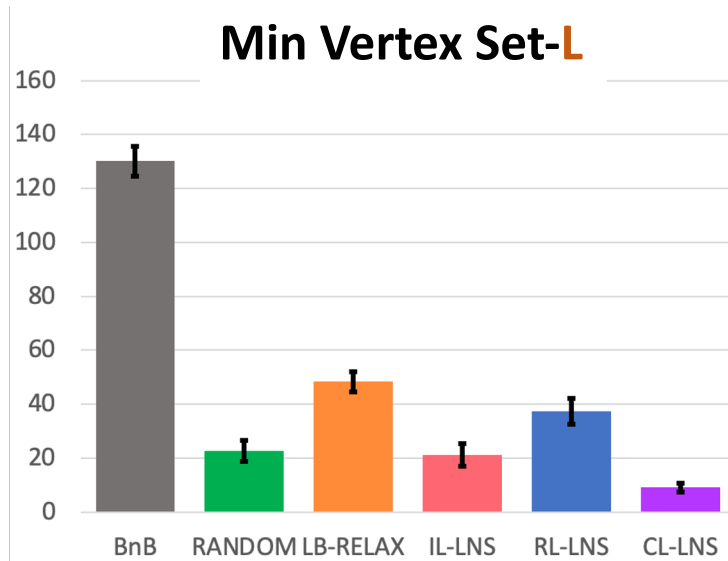
Captures the quality of and the speed at which solutions are found.



Primal gap at 60 minutes - Large instances

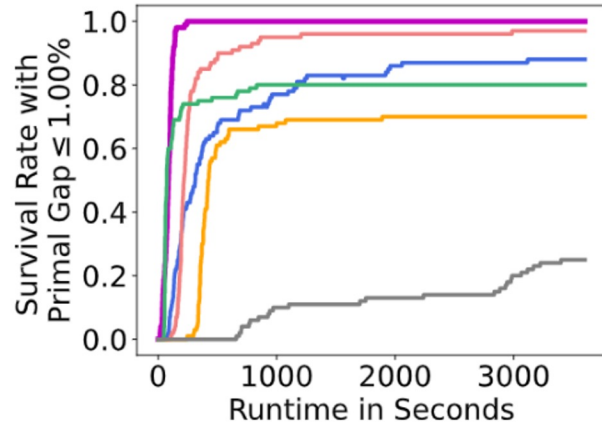


Primal **Integral** at 60 minutes - **Large** instances

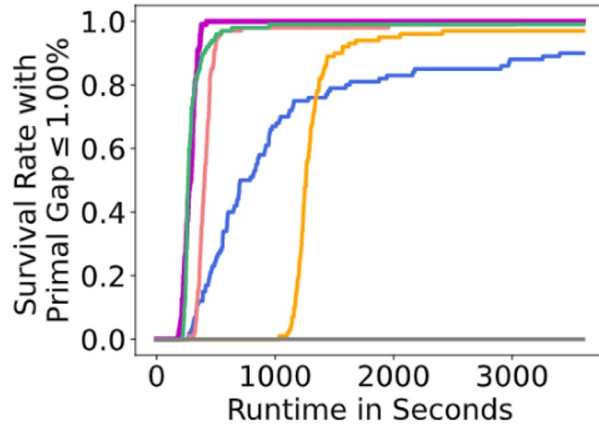


Results: Survival Rate

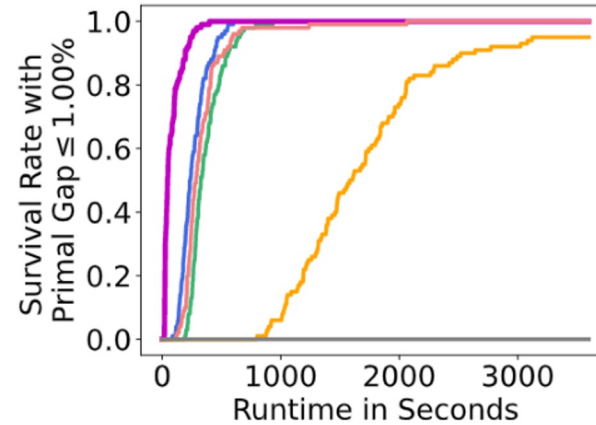
at a given primal gap threshold is the fraction of instances with primal gaps below the threshold under the method of choice



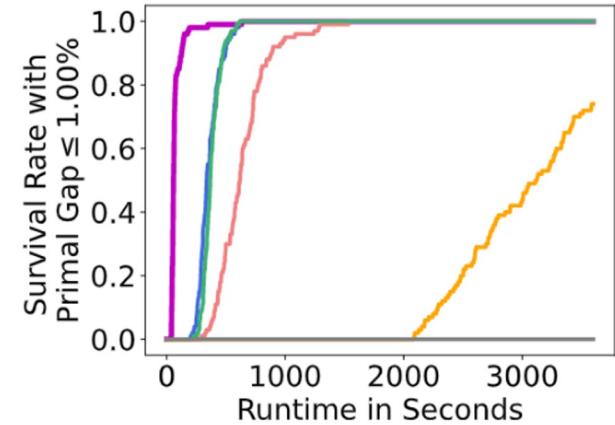
Min Vertex Set-S



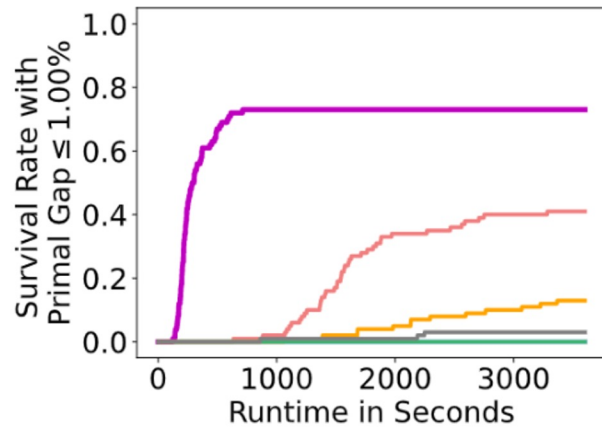
Min Vertex Set-L



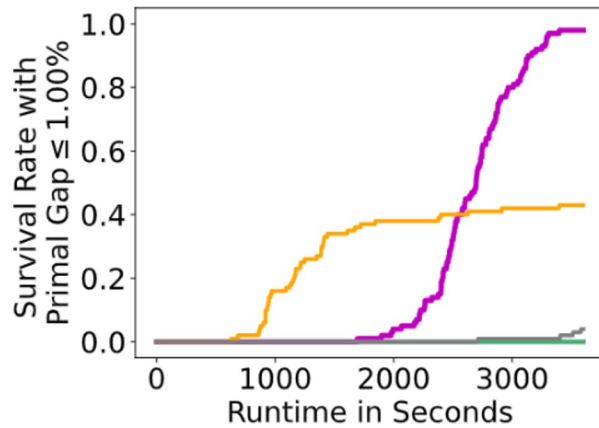
Max Independent Set-S



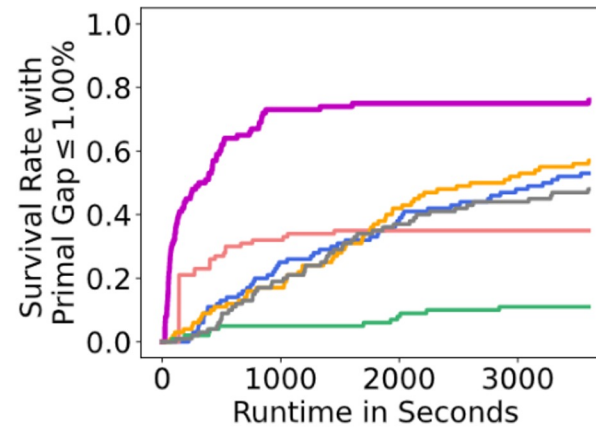
Max Independent Set-L



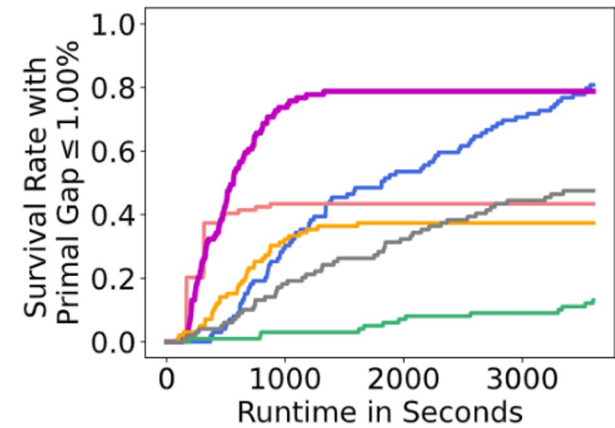
Combinatorial Auctions-S



Combinatorial Auctions-L



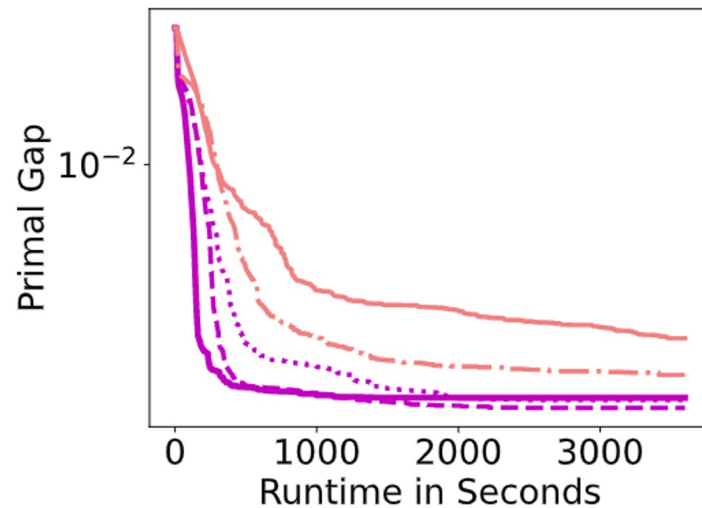
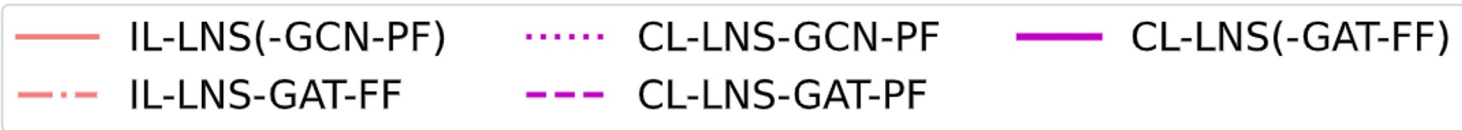
Min Set Cover-S



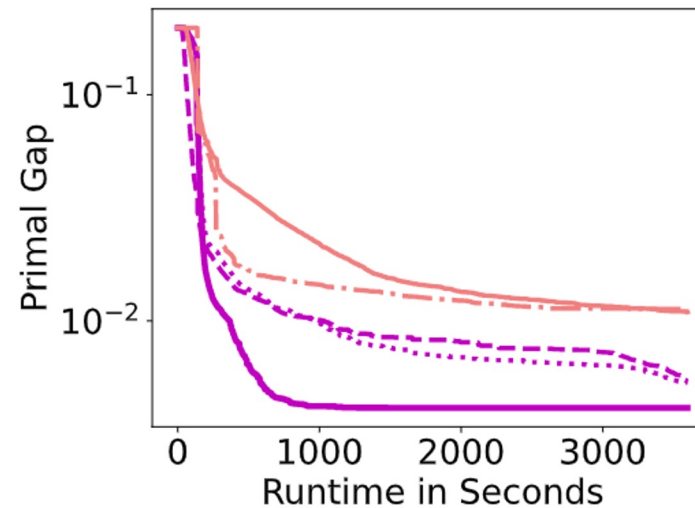
Min Set Cover-L

Ablation Study

- Imitation Learning vs. Contrastive Learning
- GCN vs GAT
- Partial features (PF) vs Full features (FF)
 - PF: Features from IL-LNS [Sonnerat et al, 2021] approach (Gasse et al. 2019)
 - FF: PF + additional variable features computed at the root node of BnB (Khalil et al. 2016)



(a) MVC-S



(b) CA-S

IL-LNS benefits from GAT+FF but still underperforms all CL-LNS variants

On MVC-S, CL-LNS-GAT-PF has better primal integral than CL-LNS-GCN-PF = benefit of replacing GCN with GAT.

On CA-S, CL-LNS-GAT-FF has better primal integral than CL-LNS-GAT-PF = benefit of replacing PF with FF.

Adding the two enhancements to the overall best performance of CL-LNS

Conclusion

- proposed CL-LNS to learn efficient and effective destroy heuristics in LNS for ILPs.
- Based on the novel idea of using Contrastive Loss
- Presented a novel data collection process tailored for CL-LNS
- Used GAT with a richer set of features to further improve its performance

- CL-LNS significantly outperformed state-of-the-art approaches on four benchmarks, according to multiple metrics
- CL-LNS achieved good generalization performance to larger instances

[Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning.](#)

Taoan Huang, Aaron Ferber, Yuandong Tian, Bistra Dilikina and Benoit Steiner. ICML 2023

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!



USC

