A Network Architecture

We give full details of the GAT architecture described in Section 3.2. The policy takes the bipartite graph $P' = (G, V, C, E)$ as input and output a score vector with one score per decision variable. We use 2-layer MLPs with 64 hidden units per layer and ReLU as the activation function to map each node feature and edge feature to $\mathbb{R}^L$ where $L = 64$.

Let $\boldsymbol{v}_j, \boldsymbol{c}_i, \boldsymbol{e}_{i,j} \in \mathbb{R}^L$ be the embeddings of the i-th variable, j-th constraint and the edge connecting them output by the embedding layers V^1, C^1, E^1. We perform two rounds of message passing through the GAT. In the first round, each constraint node $\boldsymbol{c}_i$ attends to its neighbors $\mathcal{N}_i$ using an attention structure with $H = 8$ attention heads:

$$\boldsymbol{c}_i' = \frac{1}{H}\sum_{i=1}^{H}\left(\alpha_{ii,1}^{(h)}\boldsymbol{\theta}_{c,1}^{(h)}\boldsymbol{c}_i + \sum_{j\in\mathcal{N}_i}\alpha_{ij,1}^{(h)}\boldsymbol{\theta}_{v,1}^{(h)}\boldsymbol{v}_j\right)$$

where $\boldsymbol{\theta}_{c,1}^{(h)} \in \mathbb{R}^{L\times L}$ and $\boldsymbol{\theta}_{v,1}^{(h)} \in \mathbb{R}^{L\times L}$ are learnable weights. The updated constraints embeddings $\boldsymbol{c}_i'$ in updated embedding layer C^2 are averaged across H attention heads using attention weights

$$\alpha_{ij,1}^{(h)} = \frac{\exp(\boldsymbol{w}_1^T\rho([\boldsymbol{\theta}_{c,1}^{(h)}\boldsymbol{c}_i, \boldsymbol{\theta}_{v,1}^{(h)}\boldsymbol{v}_j, \boldsymbol{\theta}_{e,1}^{(h)}\boldsymbol{e}_{i,j}]))}{\sum_{k\in\mathcal{N}_i}\exp(\boldsymbol{w}_1^T\rho([\boldsymbol{\theta}_{c,1}^{(h)}\boldsymbol{c}_i, \boldsymbol{\theta}_{v,1}^{(h)}\boldsymbol{v}_k, \boldsymbol{\theta}_{e,1}^{(h)}\boldsymbol{e}_{i,k}]))}$$

where the attention coefficients $\boldsymbol{w}_1 \in \mathbb{R}^{3L}$ and $\boldsymbol{\theta}_{e,1}^{(h)} \in \mathbb{R}^{L\times L}$ are both learnable weights and $\rho(\cdot)$ refers to the LeakyReLU activation function with negative slope 0.2. In the second round, similarly, each variable node attends to its neighbors to get updated variable node embeddings

$$\boldsymbol{v}_j' = \frac{1}{H}\sum_{i=1}^{H}\left(\alpha_{jj,2}^{(h)}\boldsymbol{\theta}_{c,2}^{(h)}\boldsymbol{c}_i' + \sum_{j\in\mathcal{N}_i}\alpha_{ji,2}^{(h)}\boldsymbol{\theta}_{v,2}^{(h)}\boldsymbol{v}_j\right)$$

with attention weights

$$\alpha_{ji,2}^{(h)} = \frac{\exp(\boldsymbol{w}_2^T\rho([\boldsymbol{\theta}_{c,2}^{(h)}\boldsymbol{c}_i', \boldsymbol{\theta}_{v,2}^{(h)}\boldsymbol{v}_j, \boldsymbol{\theta}_{e,2}^{(h)}\boldsymbol{e}_{i,j}]))}{\sum_{k\in\mathcal{N}_i}\exp(\boldsymbol{w}_2^T\rho([\boldsymbol{\theta}_{c,2}^{(h)}\boldsymbol{c}_i', \boldsymbol{\theta}_{v,2}^{(h)}\boldsymbol{v}_j, \boldsymbol{\theta}_{e,2}^{(h)}\boldsymbol{e}_{i,k}]))}$$

where $\boldsymbol{w}_2 \in \mathbb{R}^{3L}$ and $\boldsymbol{\theta}_{c,2}^{(h)}, \boldsymbol{\theta}_{v,2}^{(h)}, \boldsymbol{\theta}_{e,2}^{(h)} \in \mathbb{R}^{L\times L}$ are learnable weights. After the two rounds of message passing, the final representations of variables $\boldsymbol{v}'$ in updated embedding layer V^2 are passed through a 2-layer MLP with 64 hidden units per layer to obtain a scalar value for each variable. Finally, we apply the sigmoid function to get a score between 0 and 1.

B Additional Details of Instance Generation

We present the MILP formulations for the Generalized Independent Set Problem (GISP), Set Cover Problem (SC), Combinatorial Auction (CA), Maximal Independent Set (MIS), Facility Location (FC), and Neural Network Verification (NN).

B.1 GISP

In a GISP instance, we are given a graph G with vertex set V and edge set where E_1 is a set of non-removable edges and E_2 is a set of removable edges. A revenue $w_i > 0$ is associated with each vertex $i \in V$, and a cost $c_{ij} > 0$ is associated with each removable edge $(i, j) \in E_2$. The goal is to find an independent set, i.e., a set of vertices such that no two vertices in the set are adjacent, that maximizes the difference between the total revenue associated with the vertices in the set and the total cost associated with the removal of edges with both endpoints in the set:

$$\begin{aligned}\max \sum_{i\in V} w_i x_i - \sum_{(i,j)\in E_2} c_{ij}y_{ij}\\ \text{s.t. } x_i + x_j \le 1, \forall (i,j) \in E_1,\\ x_i + x_j - y_{ij} \le 1, \forall (i,j) \in E_2,\\ x_i \in \{0,1\}, \forall i \in V,\\ y_{ij} \in \{0,1\}, \forall (i,j) \in E_2.\end{aligned}$$

B.2 SC

In a SC instance, we are given m elements and a collection S of n sets whose union is the set of all elements. The goal is to select a minimum number of sets from S such that the union of the selected set is still the set of all elements:

$$\begin{aligned}\max -\sum_{s\in S} x_s\\ \text{s.t.} \sum_{s\in S: i\in s} x_s \ge 1, \forall i \in [m],\\ x_s \in \{0,1\}, \forall s \in S.\end{aligned}$$

B.3 CA

In a CA instance, we are given n bids $\{(B_i, p_i) : i \in [n]\}$ for m items, where B_i is a subset of items and p_i is its associated bidding price. The objective is to allocate items to bids such that the total revenue is maximized:

$$\begin{aligned}\max \sum_{i\in[n]} p_i x_i\\ \text{s.t.} \sum_{i: j\in B_i} x_i \le 1, \forall j \in [m],\\ x_i \in \{0,1\}, \forall i \in [n].\end{aligned}$$

B.4 MIS

In a MIS instance, we are given an undirected graph $G = (V, E)$. The goal is to select the largest subset of nodes such that no two nodes in the subsets are connected by an edge in G:

$$\begin{aligned}\max \sum_{v\in V} x_v\\ \text{s.t.} x_u + x_v \le 1, \forall (u,v) \in E,\\ x_v \in \{0,1\}, \forall v \in V.\end{aligned}$$

B.5 FC

In a FC instance, we are given a set of potential facility locations I and a set of customers J. Each facility $i \in I$ can serve multiple customers and has a fixed cost f_i associated with opening it. Each customer $j \in J$ must be served by exactly one open facility, and the cost of serving customer j from facility i is denoted as c_{ij}. The goal is to minimize the total cost of opening facilities and serving all customers:

$$\begin{aligned}
\min \sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}& \\
\text{s.t.} \sum_{i\in I} x_{ij} = 1, \quad \forall j \in J,& \\
x_{ij} \le y_i, \quad \forall i \in I, j \in J,& \\
\sum_{j\in J} D_j x_{ij} \le S_i y_i, \quad \forall i \in I,& \\
y_i \in \{0,1\}, \quad \forall i \in I,& \\
x_{ij} \in \{0,1\}, \quad \forall i \in I, j \in J.&
\end{aligned}$$

B.6 NN

In a NN instance, we are given a neural network represented by a series of layers with nodes interconnected by weights. The goal is to verify whether, for all possible inputs within a specified range, the network's output adheres to certain safety or correctness criteria. The problem is typically formulated as determining if there exists any input that leads to an undesired or unsafe output, which can be expressed as:

$$\begin{aligned}
&\max\{z_k\} - c_k \\
\text{s.t.} \quad & x_{i+1} = f(W_i x_i + b_i), \quad \forall i, \\
& a \le x_0 \le b, \\
& z_k \text{ must satisfy safety criteria}, \\
& x_i, z_k \in \mathbb{R}^n.
\end{aligned}$$

C Additional Details on Hyperparameter Tuning

For SCORER and SCORER+CLS, we use all the hyperparameters provided in Ferber et.al.'s code in our experiments.

We limit the number of backdoor samples k to 50 for each instance during data collection due to the inefficiency of MCTS, thus giving us a restricted range of p and q. We ran several experiments with minor, reasonable variations of p and q $(p = 5, q = 15)(p = 10, q = 10)$ during training, and we did not observe much difference in the performance of the resulting models. For L and H, we are following previous work to avoid complicate ablation studies.

In our experiments, we greedily taking the q worst candidate backdoors as negative samples. Other ways of determining negative samples have been experimented with, such as choosing the candidate backdoors with highest number of common variables but have worse performance or randomly sampling 5 backdoors from the 15 worst samples. The current strategy outperforms the others.

In Table 1, we summarize all the hyperparameters with their notations and values used in our experiments.

D Additional Figures

Figure 1 presents a runtime comparison between the collected backdoors and the standard Gurobi runtime for GISP-S, SC-S, CA-S, MIS-S, FC, and NN. The histograms provide a visual representation, highlighting the instances where the best backdoor among the 50 collected outperforms the standard Gurobi runtime. Notably, GISP-S and FC exhibits the highest backdoor quality, followed by SC-S and NN, where the backdoor quality is considered good. However, for CA-S and MIS-S, the backdoor quality is less favorable, with a majority of backdoors unable to surpass the standard Gurobi runtime.

Table 1: Hyperparameters with their notations and values used.

Hyperparameter	**Notation**	**Value**
Number of candidate backdoors for each instance	k	50
Number of positive samples	p	5
Number of negative samples	q	5
Feature embedding dimension	L	64
Number of attention heads in the GAT	H	8
Temperature parameter in the contrastive loss	τ	0.07
Learning rate		5×10^{-4}
Weight decay		0.01
Batch size		32
Number of training epochs		100

Figure 2 compares the runtime of GRB and CL in terms of speed improvement and finish rate on small instances. On the left part, the dots under the red line are mostly easy instances, and our model almost gains most win for the hard instances. On the right side, the majority of the orange line is above the blue line. The results clearly show CL is better than GRB in terms of running time.

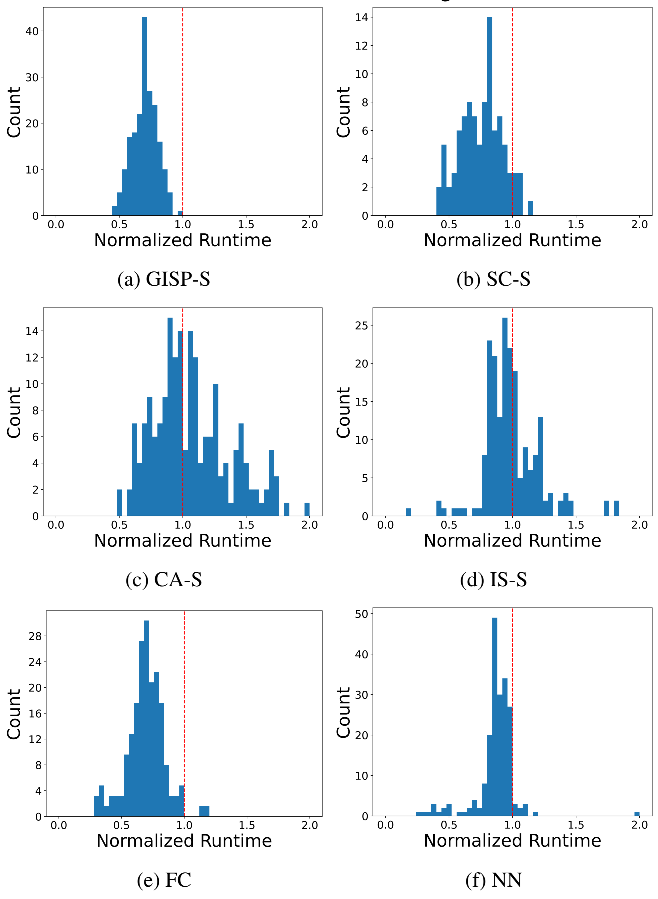
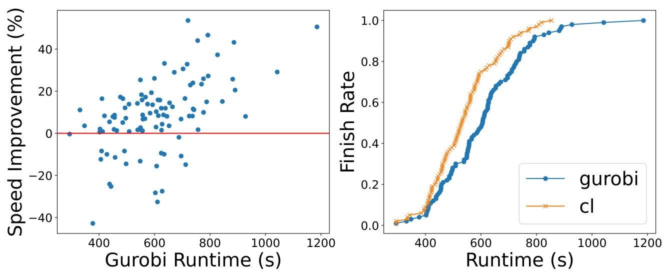
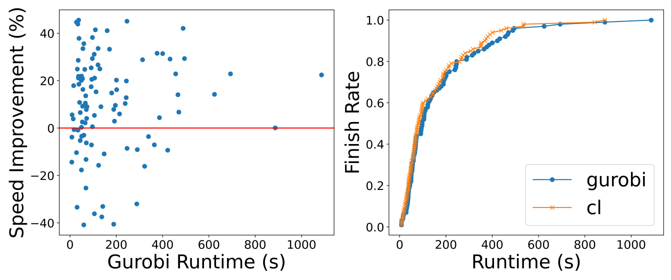


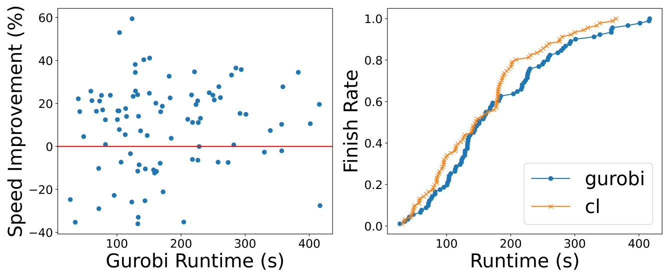
Figure 1: Histograms illustrating the normalized runtime distributions of best candidate backdoors among each instance for six distinct problem domains: (a) GISP-S, (b) SC-S, (c) CA-S, (d) IS-S, (e) FC, and (f) NN. The normalized runtime is calculated as the ratio of the solve time with the candidate backdoor to the original solve time without it. The red vertical line at 1.0 marks the threshold where the candidate backdoor's performance equals the original solve time. Values to the left of this line indicate instances where the candidate backdoor resulted in a faster solve time, while values to the right indicate a slower solve time than the original.



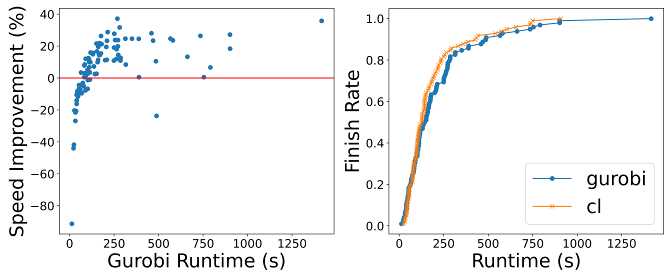
(a) GISP-S



(b) SC-S



(c) CA-S



(d) MIS-S

Figure 2: This figure shows the runtime of Gurobi (GRB) and contrastive learning model (CL) on GISP-S, SC-S, CA-S, and MIS-S through two different types of plots. The left part is a scatter plot with Gurobi runtime as the x-axis and the speed improvement as the percentage of CL over GRB as the y-axis. The points above the red line are ones where CL is better than GRB and vice versa. The right part shows the finish rate as a function of runtime. The finish rate for a given runtime is the fraction of instances solved to optimality within the runtime. The blue line is GRB and the orange line is CL, with every dot indicating one finished instance. The figures show that CL outperforms GRB on average and specifically provides speedups on the harder instances in each distribution.