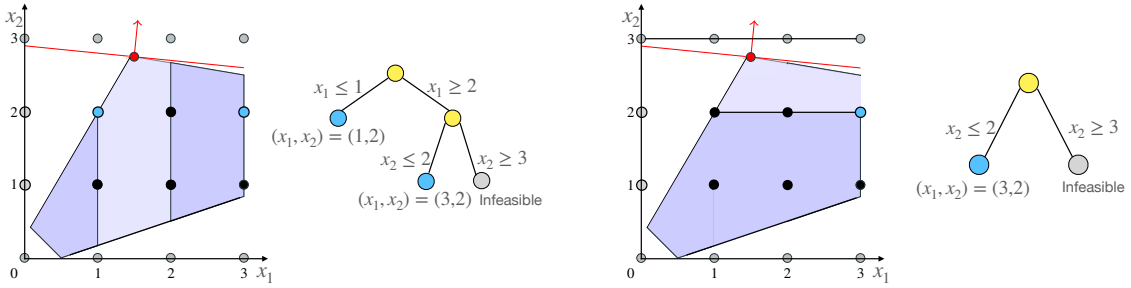
A APPENDIX

A.1 BRANCH AND BOUND

Here we gives a simple illustration of B&B algorithm in Figure 5. Given the LP relaxation, the polytope represents the feasible region of the LP relaxation and the red arrow represents the objective vector. We first solve the LP relaxation and obtain the solution $\hat{x}$ as the red point. Noticing it is not feasible for MIP, we branch the LP relaxation into two subproblems. In (a) we select to split variable x_1 and in (b) we select to split variable x_2. The subproblems obtained after branching are displayed by the shaded purple regions. After finishing solve these two MIPs, we obtain the search trees t_1 and t_2. We can see that a wise selection of variable x_2 can solve the problem faster.



(a) split on x_1 and search tree t_1

(b) split on x_2 and search tree t_2

Figure 5: Illustration of splitting in B&B and the corresponding search tree

A.2 IMPLEMENTATION

A.2.1 HARDWARE

All the experiments were run at a Ubuntu 18.04 machine with Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz, 256 GB Memory and Nvidia RTX 2080Ti graphic cards.

A.2.2 PD POLICY

Comparison. PD policy is similar to the GCN in Gasse et al. (2019) but has two major differences. First, we use a dynamic reduced graph where fixed variables and trivial constraints are removed due to the variable bounds changing during the solving process while Gasse et al. (2019) do not consider it. The reduced graph can not only save computation, but also give a more accurate description of the solving state by ignoring the redundant information. The ablation in Section 5.5 shows it is indispensable in the success of RL. Second, we use a simple matrix multiplication in our PD policy while Gasse et al. (2019) use a complicated edge embedding in GCN. In some sense, GCN can be seen as an overparameterized version of our method. And our success reveals that message passing on the LP relaxation is the true helpful structure.

detail. We implement our primal dual policy net using dgl (Wang et al., 2019), with hidden dimension $h = 64$ and ReLU activation. The feature $\mathbf{X}$ for variable is a 17 dimension vector and feature $\mathbf{Y}$ for constraint is a 5 dimension vector. We list the detail of feature in Table. 3

Tensor	Name	Description
$\mathbf{X}$	type	a one-hot encoding for (binary, integer, implicit, continuous)
	coef	objective coefficient
	lb	variable lower bound
	ub	variable upper bound
	at-lb	indicator whether solution value equals lower bound
	at-ub	indicator whether solution value equals upper bound
	sol-frac	solution value fractionality
	basis-status	a one-hot encoding for simplex basis status (lower, basic, upper, zero)
	red	reduced cost
	age	normalized LP age
	sol-val	solution value
$\mathbf{Y}$	obj-sim	cosine similarity with objective
	bias	bias value
	is-tight	tightness indicator in LP solution
	dualsol-val	dual solution value
	age	normalized LP age

Table 3: Feature $\mathbf{X}$ for variable and feature $\mathbf{Y}$ for constraint

A.2.3 BASELINE

FSB. We use the implementation in SCIP Gamrath et al. (2020)

VFS. We use the implementation in SCIP Gamrath et al. (2020)

RPB. We use the implementation in SCIP Gamrath et al. (2020)

GCN. We tried to implement GCN in dgl (Wang et al., 2019), however, it is significantly slower than the original implementation in Gasse et al. (2019). Hence, we still use the implementation in Gasse et al. (2019).

SVM. We use the implementation in Gasse et al. (2019).

A.3 TRAINING

We have two settings *clean*, *default*. In experiments, we always train and test under the same setting.

Imitation Learning. We initialize our PD policy using imitation learning similar to Gasse et al. (2019). The difference is we only use 10000 training samples, 2000 validation samples and 10 training epochs as a warm start. In our setting, a policy from scratch can hardly solve an instance in a reasonable time, hence a warm start is necessary.

Novelty Search Evolution Strategy. We improve our RL agent using Algorithm 2. The parameters are set as $\alpha = 1e-4$, $\sigma = 1e-2$, $n = 40$, $V = 200$, $w = 0.25$, $\beta = 0.99$, $T = 1000$, $k = 10$.

Algorithm 2: Evolutionary Strategy with Novelty Score.

Input: Learning rate α, Noise std σ, number of workers n, Validation size V, Batch size M, Initial weight λ, Weight decay rate β, Iterations T, Parameter θ_0, Policy memory M, Instance distribution $\mathcal{D}$

, Neighborhood size k. **Output:** Best parameter θ_{best}

Sample validation instances $Q_1, \cdots, Q_V \sim \mathcal{D}$

Set $R_{\text{best}} = \frac{1}{V}\sum_{j=1}^{V} R(\theta_0, Q_j), \theta_{\text{best}} = \theta_0$

for *t=0* **to** *T* **do**

  Sample instances $P_1, \cdots, P_M \sim \mathcal{D}$

  Sample $\epsilon_1, \cdots, \epsilon_n \sim \mathcal{N}(0, I)$ and compute $\theta_t^i = \theta_t + \sigma\epsilon_i$

  Set $M = \{\theta_t^1, \cdots, \theta_t^n\}$

  for *i=1* **to** *n* **do**

    Compute $R_i = \frac{1}{m}\sum_{m=1}^{M} R(\theta_t^i, P_m)$

    Compute $N_i = \frac{1}{m}\sum_{m=1}^{M} N(\theta_t^i, P_m, M)$

  end

  Set $\theta_{t+1} = \theta_t + \alpha\frac{1}{n\sigma}\sum_{i=1}^{n} \lambda \cdot N_i\epsilon_i + (1-\lambda)\cdot R_i\epsilon_i$

  Compute $R^{(t+1)} = \frac{1}{V}\sum_{j=1}^{V} R(\theta_{t+1}, Q_j)$

  if $R^{(t+1)} > R_{best}$ **then**

    Set $R_{\text{best}} = R^{(t+1)}, \theta_{\text{best}} = \theta_{t+1}, \lambda = \beta * \lambda$

  end

end

A.4 DATA SET

Set Covering. We generate a weighted set covering problem following Balas & Ho (1980). The problem is formulated as the following ILP.

$$\begin{aligned} \min \sum_{S\in\mathcal{S}} w_S x_S \\ \text{subject to } \sum_{S:e\in S} X_S \geq 1,\ \forall e \in \mathcal{U} \\ x_S \in \{0,1\},\ \forall S \in \mathcal{S} \end{aligned}$$

where $\mathcal{U}$ is the universe of elements, $\mathcal{S}$ is the universe of the sets, w is a weight vector. For any $e \in \mathcal{U}$ and $S \in \mathcal{S}$, $e \in S$ with probability 0.05. And we guarantee that for any e, it is contained by at least two sets in $\mathcal{S}$. Each w_S is uniformly sampled from integer from 1 to 100.

We first generate a set covering problem with $\mathcal{U}_0 = \{e_1, \cdots, e_{400}\}$ and $\mathcal{S}_0 = \{S_1, \cdots, S_{1000}\}$ and set it as our backbone. Then, every time we want to generate a new problem with m elements, we let $\mathcal{U} = \mathcal{U}_0 \cup \{e_{401}, e_{402}, \cdots, e_m\}$ add new e_i into $S \in \mathcal{S}$ following the pipeline mentioned above.

Maximum Independent Set. We generate maximum independent set problem using Barabasi-Albert (Albert & Barabási, 2002) graphs. The problem is formulated as the following ILP.

$$\begin{aligned} \max \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \leq 1, \ \forall e_{uv} \in E \\ & x_v \in \{0, 1\}, \ \forall v \in V \end{aligned}$$

where V is the set of vertices and E is the set of edges. We generate the BA graph using a preferential attachment with affinity coefficient 4.

We first generate a BA graph G_0 with 350 nodes. Then, every time we want to generate a new problem with n variables, we expand G_0 using preferential attachment.

Capacitated Facility Location. We generate the capacitated facility location problem following Cornuéjols et al. (1991). The problem with m customers and n facilities is formulated as the following MIP.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} d_j y_{ij} + \sum_{i=1}^{n} f_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} y_{ij} = 1, \ \forall j = 1, \cdots, m \\ & \sum_{j=1}^{m} d_j y_{ij} \leq u_i x_i, \ \forall i = 1 \cdots, n \\ & y_{ij} \geq 0, \ \forall i = 1, \cdots, n \text{ and } j = 1, \cdots, m \\ & x_i \in \{0, 1\}, \ \forall i = 1, \cdots, n \end{aligned}$$

where $x_i = 1$ indicates facility i is open, and $x_i = 0$ otherwise; f_i is the fixed cost if facility i is open; d_j is the demand for customer j; c_{ij} is the transportation cost between facility j and customer i; y_{ij} is the fraction of the demand of customer j filled by facility i. Following Cornuéjols et al. (1991), where we first sample the location of facility and customers on a 2 dimension map. Then c_ij is determined by the Euclidean distance between facility i and customer j and other parameters are sampled from the distribtuion given in Cornuéjols et al. (1991).

We first generate the location of 100 facilities and 40 customers as our backbone. Then, every time we want to generate a new problem with m customers, we generate new $m - 40$ locations for customers and follow the pipeline mentioned above.

A.5 Discussion

An interesting phenomenon is that GCN can easily beat VFS after imitation learning (Or our PD policy can obtain similar result). One possible explanation is that the primal-dual message passing is a principle structure that naturally learns the good decisions and ignores the noise brought by strong branching. Another possible reason is the biased sampling. To keep the diversity of the sample, Gasse et al. (2019) employs a mixed policy of RPB and VFS to sample the training data. It is possible that VFS is a powerful expert on the state distribution determined by the mixed policy. More studies are needed before we can answer this question.

Another point is our set representation is compatible with general B&B algorithm. Once the weight function w and distance function d are defined, we can compute the distance between two B&B solving processes.