Co-training for Policy Learning

Jialin Song [1] **Ravi Lanka** [2] **Yisong Yue** [1] **Masahiro Ono** [2]

Abstract

We study the problem of learning sequential decision-making policies in settings with multiple state-action representations. Such settings naturally arise in many domains, such as planning (e.g., multiple integer programming formulations) and various combinatorial optimization problems (e.g., those with both integer programming and graph-based formulations). Inspired by the classical co-training framework for classification, we study the problem of co-training for policy learning. We present sufficient conditions under which learning from two views can improve upon learning from a single view alone. Motivated by these theoretical insights, we present an algorithm for co-training for sequential decision making. Our framework is compatible with both reinforcement learning and imitation learning. We validate the effectiveness of our approach on a challenging class of combinatorial optimization problems: minimum vertex cover.

1. Introduction

A common wisdom in problem solving is that there is more than one way to look at a problem. For sequential decision making problems, such as those in reinforcement learning and imitation learning, one can often utilize multiple different state-action representations to characterize the same problem. A canonical application example is learning solvers for hard optimization problems such as combinatorial optimization (1; 2; 3; 4; 5; 6). It is well-known in the operations research community that many combinatorial optimization problems have multiple formulations. Prominent examples include the maximum cut problem where one can describe with a quadratic integer problem as well as a linear integer program (7; 8). Another example is the travelling salesperson problem, which admits multiple integer programming formulations (9; 10). One can also formulate many problems using a graph-based representation (see Figure 1). Beyond learning combinatorial optimization solvers, other examples with multiple state-action representations include robotic applications with multiple sensing modalities such as third-person view demonstrations (11) and multilingual machine translation (12).

In the context of policy learning, one natural question is how different state-action formulations impact learning and, more importantly, how learning can take advantage of multiple formulations. The multiple formulation scenario is related to the co-training problem (13; 14), where different feature representations of the same problem enable more effective learning compared with using only a single representation (15; 16). While co-training has received much attention in classical tasks such as classification, little effort has been made on applying it to sequential decision making problems. One immediate consequence considering the sequential case is that some settings have completely separate state-action representations while others can share the action space.

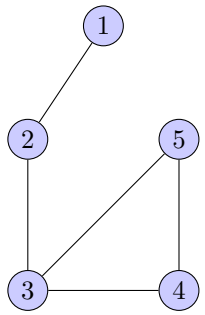
In this paper, we propose CoPiEr (co-training for policy learning), a framework for policy co-training that can incorporate both reinforcement learning and imitation learning as subroutines. Our approach is based on a novel theoretical result that integrates and extends results from general policy learning with demonstrations (17). To the best of our knowledge, we are the first to formally extend the co-training framework to policy learning.

Our contributions can be summarized as:

- We present a formal theoretical framework for policy co-training. We provide a general theoretical characterization of policy improvement. This theoretical characterization sheds light on a rigorous algorithm design for policy learning that can appropriately exploit multiple state-action representations.
- We present CoPiEr (co-training for policy learning), an algorithm for policy co-training that is based on the theoretical analysis.
- We empirically evaluate on a challenging combinatorial optimization problems: minimum vertex cover. We showcase the practicality of our approach by demonstrating superior performance compared to a

[1]California Institute of Technology [2]Jet Propulsion Laboratory. Correspondence to: Jialin Song <jssong@caltech.edu>.

Real-world Sequential Decision Making workshop at ICML 2019.



$$\max \; -\sum_{i=1}^{5} x_i,$$

subject to:

$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

Figure 1. Two ways to encode minimum vertex cover (MVC) problems. Left: policies learn to operate directly on the graph view to find the minimal cover set (18). Right: we express MVC as an integer linear program, then polices learn to traverse the resulting combinatorial search space, i.e., learn to branch-and-bound (1; 5).

wide range of strong learning-based benchmarks as well as commercial solvers such as Gurobi.

2. Related Work

Co-training Our work is inspired by the classical co-training framework for classification (13), which utilizes two different feature representations, or views, to effectively use unlabeled data to improve the classification accuracy. Subsequent extensions of co-training includes co-EM (19) and co-regularization (20). Co-training has been widely used in natural language processing (15; 21), clustering (16; 22), domain adaptation (23) and game playing (24). For policy learning, some related ideas have been explored where multiple estimators of the value or critic function are trained together (25; 26).

Policy Learning for Sequential Decision Making Sequential decision making pertains to tasks where the policy performs a series of actions in a stateful environment. A popular framework to characterize the interaction between the agent and the environment is a Markov Decision Process (MDP). There are two main approaches for policy learning in MDPs. The first is reinforcement learning, which uses the observed environmental rewards to perform policy optimization. Recent work include Q-Learning approaches such as deep Q-Networks (27), as well as policy gradient approaches such as DDPG (28), TRPO (29) and PPO (30). Despite its successful applications to a wide variety of tasks including playing games (27; 31), robotics (32; 33) and combinatorial optimization (2; 4; 3), high sample complexity and unstable learning pose significant challenges in practice (34).

The second approach, imitation learning, uses demonstrations (from an expert) as the primary learning signal. One popular class of algorithms is reduction-based (35; 36; 37; 38; 39), which generates cost-sensitive supervised examples from demonstrations. Other approaches include estimating the expert's cost-to go (40), inverse reinforcement learning (41; 42; 43), and behavioral cloning (44). Recent works have also explored on how to combine these two types of learning (45; 17; 46; 47). One major limitation of imitation learning is the reliance on demonstrations for reliable learning. For both imitation learning and reinforcement learning, we show that co-training on two views can provide surrogate demonstrations in the former and improved exploration in the latter, in both cases leading to superior performance.

3. Background & Preliminaries

Markov Decision Process with Two State Representations. A Markov decision process (MDP) is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mathcal{S}_T)$. Let $\mathcal{S}$ denote the state space, $\mathcal{A}$ the action space, $\mathcal{P}(s'|s, a)$ the (probabilistic) state dynamics, $r(s, a)$ the reward function, γ the discount factor and (optional) $\mathcal{S}_T$ a set of terminal states where the decision process ends. We consider both stochastic and deterministic MDPs. An MDP with two views can be written as $\mathcal{M}^A = (\mathcal{S}^A, \mathcal{A}^A, \mathcal{P}^A, r^A, \gamma^A, \mathcal{S}_T^A)$ and $\mathcal{M}^B = (\mathcal{S}^B, \mathcal{A}^B, \mathcal{P}^B, r^B, \gamma^B, \mathcal{S}_T^B)$. To connect the two views, we make the following assumption about the ability to translate trajectories between the two views.

Assumption 1. *For a (complete) trajectory in $\mathcal{M}^A$, $\tau^A = (s_0^A, a_0^A, s_1^A, a_1^A, \cdots, s_n^A)$, there is a function $f_{A\to B}$ such that $f_{A\to B}(\tau^A) = \tau^B = (s_0^B, a_0^B, s_1^B, a_1^B, \cdots, s_m^B)$ is a (complete) trajectory in $\mathcal{M}^B$. And rewards for τ^A and τ^B are the same under their respective reward functions, i.e., $\sum_{i=0}^{n-1} r^A(s_i^A, a_i^A) = \sum_{j=0}^{m-1} r^B(s_j^B, a_j^B)$. Similarly, there is also a function $f_{B\to A}$ that maps trajectories in $\mathcal{M}^B$ to $\mathcal{M}^A$ which preserves the total rewards. Moreover, $f_{A\to B}$ and $f_{B\to A}$ are the inverse maps of each other.*

Combinatorial Optimization Example. Minimum vertex cover (MVC) is a classical combinatorial optimization defined over a graph $G = (V, E)$. A cover set is defined as a subset $U \subset V$ such that every edge $e \in E$ is incident to at least one $v \in U$. The objective is to find a U with minimal cardinality. For the graph in Figure 1, a minimal cover set is $\{2, 3, 4\}$.

There are two natural ways to represent an MVC problem as an MDP. The first is graph-based (4), and sets the action space as V, and the state space as sequences of vertices in V representing partial solutions. The deterministic transition function is the obvious choice of adding vertices to

the current partial solution. The rewards are -1 for each selected vertex. A terminal state is reached if the selected vertices form a cover.

The second way is to formulate an integer linear program (ILP) that encodes MVC problem:

$$\begin{aligned}&\max -\sum_{v\in V} x_v,\\ &\text{subject to :}\\ &x_u + x_v \geq 1, \forall e = (u,v) \in E,\\ &x_v \in \{0,1\}, \forall v \in V.\end{aligned}$$

We use branch-and-bound (48) to solve this ILP. Branch-and-bound represents the optimization problem as a search tree, and explores different areas of a search tree through a sequence of branching operations. The MDP states represent current search tree, and the actions correspond to which node to explore next. The deterministic transition function is the obvious choice of adding a new node into the search tree. The rewards are zero during intermediate steps and the agent receives reward equal to the best objective value found in the end. A terminal state is a search tree which contains an optimal solution.

The relationship between solutions in the two formulations are clear. For a graph $G = (V, E)$, a feasible solution to the ILP corresponds to a vertex cover by selecting all the vertices $v \in V$ with $x_v = 1$ in the solution.

Note that, despite the deterministic dynamics, solving MVC other combinatorial optimization problems can be extremely challenging due to the very large state space. Indeed, policy learning for combinatorial optimization is a topic of active research (18; 1; 5; 3; 6).

Policy Learning. We consider policy learning over a distribution of MDPs. For instance, there can be a distribution of MVC problems. Formally, we have a distribution $\mathcal{D}$ of MDPs that we can sample from (i.e., $\mathcal{M} \sim \mathcal{D}$). For a policy π, we define the following terms: $J(\pi) = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}}[\eta(\pi, \mathcal{M})]$, $\eta(\pi, \mathcal{M}) = \mathbb{E}_{\tau\sim\pi}[\sum_{i=0}^{n-1} \gamma^i r(s_i, a_i)]$, $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$, $Q_\pi(s, a) = \mathbb{E}_{\tau\sim\pi}[\sum_{i=0}^{n-1} \gamma^i r(s_i, a_i)|s_0 = s, a_0 = a]$, $V_\pi(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{i=0}^{n-1} \gamma^i r(s_i, a_i)|s_0 = s]$ with J being the overall objective, η the expected cumulative reward of an individual MDP $\mathcal{M}$, A the advantage function, Q the Q function, and V the value function. The performance of two policies can be related via the advantage function (29; 49):

$$\eta(\pi', \mathcal{M}) = \eta(\pi, \mathcal{M}) + \mathbb{E}_{\tau\sim\pi'}[\sum_{i=0}^{n-1} \gamma^i A_\pi(s_i, a_i)] \quad (1)$$

Based on the equivalence between a policy and its occupancy measure (50), we can rewrite the final term in (1) with the occupancy measure, $\rho_\pi(s, a) = \mathbb{P}(\pi(s) = a) \sum_{i=0}^{\infty} \gamma^i \mathbb{P}(s_i = s|\pi)$.

With slight notation abuse, define $\rho_\pi(s) = \sum_{i=0}^{\infty} \gamma^i \mathbb{P}(s_i = s|\pi)$ to be the state visitation distribution. In policy iteration, we aim to maximize:

$$\begin{aligned}&\mathbb{E}_{\tau\sim\pi'}[\sum_{i=0}^{n-1} \gamma^i A_\pi(s_i, a_i)]\\ &= \sum_{i=0}^{n-1} \mathbb{E}_{s_i\sim\rho_{\pi'}(s)}[\mathbb{E}_{a_i\sim\pi'(s_i)}[\gamma^i A_\pi(s_i, a_i)]],\\ &\approx \sum_{i=0}^{n-1} \mathbb{E}_{s_i\sim\rho_{\pi}(s)}[\mathbb{E}_{a_i\sim\pi'(s_i)}[\gamma^i A_\pi(s_i, a_i)]].\end{aligned}$$

This is done instead of taking an expectation over $\rho_{\pi'}(s)$ which has a complicated dependency on a yet unknown policy π'. Policy gradient methods tend to use the approximation by using ρ_π which depends on the current policy. We define the approximate objective as $\eta_\pi(\pi', \mathcal{M}) = \eta(\pi, \mathcal{M}) + \sum_{i=0}^{n-1} \mathbb{E}_{s_i\sim\rho_\pi(s)}[\mathbb{E}_{a_i\sim\pi'(s_i)}[\gamma^i A_\pi(s_i, a_i)]]$, and its associated expectation over $\mathcal{D}$ as $J_\pi(\pi') = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}}[\eta_\pi(\pi', \mathcal{M}]$.

4. A Theory of Policy Co-training

In this section, we provide a theoretical characterization of policy co-training, which motivates the design of our CoPiEr algorithm presented in Section 5. Our theoretical analysis quantifies the **policy improvement** in terms of policy advantages and differences, and caters to policy gradient approaches. Due to space constraint, we defer all proofs to the appendix.

For an MDP $\mathcal{M} \sim \mathcal{D}$, consider the rewards of two policies with different views $\eta^A(\pi^A, \mathcal{M}^A)$ and $\eta^B(\pi^B, \mathcal{M}^B)$. If $\eta^A(\pi^A, \mathcal{M}^A) > \eta^B(\pi^B, \mathcal{M}^B)$, then on this instance π^A performs better than π^B, and thus we could use the translated trajectory of π^A as a demonstration for π^B. Even when $J(\pi^A) > J(\pi^B)$, because J is computed in expectation over $\mathcal{D}$, π^B can still perform better than π^A on some MDPs. Thus it is possible for the exchange of demonstrations to go in both directions.

Formally, we can decompose the distribution $\mathcal{D}$ into two parts $\mathcal{D}_1$ and $\mathcal{D}_2$ such that the support of $\mathcal{D}$, $\text{supp}(\mathcal{D}) = \text{supp}(\mathcal{D}_1) \cup \text{supp}(\mathcal{D}_2)$ and $\text{supp}(\mathcal{D}_1) \cap \text{supp}(\mathcal{D}_2) = \emptyset$, where for an MDP $\mathcal{M} \in \text{supp}(\mathcal{D}_1), \eta(\pi^A, \mathcal{M}^A) \geq \eta(\pi^B, \mathcal{M}^B)$ and for an MDP $\mathcal{M} \in \text{supp}(\mathcal{D}_2), \eta(\pi^B, \mathcal{M}^B) > \eta(\pi^A, \mathcal{M}^A)$. By construction, we can quantify the perfor-

mance gap as:

Definition 1.

$$\delta_1 = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}_1}[\eta(\pi^A, \mathcal{M}^A) - \eta(\pi^B, \mathcal{M}^B)] \geq 0$$
$$\delta_2 = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}_2}[\eta(\pi^B, \mathcal{M}^B) - \eta(\pi^A, \mathcal{M}^A)] > 0$$

We can now state our first result on policy improvement.

Theorem 1. *(Extension of Theorem 1 in (17)) Define:*

$$\alpha^A_{\mathcal{D}} = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}}[\max_s D_{KL}(\pi^A(s)\|\pi'^A(s))],$$
$$\beta^B_{\mathcal{D}_2} = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}_2}[\max_s D_{JS}(\pi^B(s)\|\pi^A(s))],$$
$$\alpha^B_{\mathcal{D}} = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}}[\max_s D_{KL}(\pi^B(s)\|\pi'^B(s))],$$
$$\beta^A_{\mathcal{D}_1} = \mathbb{E}_{\mathcal{M}\sim\mathcal{D}_1}[\max_s D_{JS}(\pi^A(s)\|\pi^B(s))],$$
$$\epsilon^B_{\mathcal{D}_2} = \max_{\mathcal{M}\in supp(\mathcal{D}_2)} \max_{s,a} |A_{\pi^B}(s,a)|,$$
$$\epsilon^A_{\mathcal{D}} = \max_{\mathcal{M}\in supp(\mathcal{D})} \max_{s,a} |A_{\pi^A}(s,a)|,$$
$$\epsilon^A_{\mathcal{D}_1} = \max_{\mathcal{M}\in supp(\mathcal{D}_1)} \max_{s,a} |A_{\pi^A}(s,a)|,$$
$$\epsilon^B_{\mathcal{D}} = \max_{\mathcal{M}\in supp(\mathcal{D})} \max_{s,a} |A_{\pi^B}(s,a)|.$$

Here D_{KL} & D_{JS} denote the Kullback-Leibler and Jensen-Shannon divergence respectively. Then we have:

$$J(\pi'^A) \geq J_{\pi^A}(\pi'^A) - \frac{2\gamma^A(4\beta^B_{\mathcal{D}_2}\epsilon^B_{\mathcal{D}_2} + \alpha^A_{\mathcal{D}}\epsilon^A_{\mathcal{D}})}{(1-\gamma^A)^2} + \delta_2$$
$$J(\pi'^B) \geq J_{\pi^B}(\pi'^B) - \frac{2\gamma^B(4\beta^A_{\mathcal{D}_1}\epsilon^A_{\mathcal{D}_1} + \alpha^B_{\mathcal{D}}\epsilon^B_{\mathcal{D}})}{(1-\gamma^B)^2} + \delta_1$$

Compared to conventional analyses on policy improvement, the new key terms that determine how much the policy improves are the β's and δ's. The β's, which quantify the maximal divergence between π^A and π^B, hinders improvement, while the δ's contribute positively. If the net contribution is positive, then the policy improvement bound is larger than that of conventional single view policy gradient. This insight motivates co-training algorithms that explicitly aim to minimize the β's.

5. The CoPiEr Algorithm

We now present practical algorithms motivated by the theoretical insights from Section 4. We start with an algorithm named CoPiEr (Algorithm 1), whose important subroutines are EXCHANGE and UPDATE.

Algorithm 2 covers exchanging trajectories generated by the two policies. First we estimate the relative quality of the two policies with their sampled trajectories (Line 2-3 in Algorithm 2). Then we use the trajectories from the better policy as demonstrations for the worse policy on this MDP.

Algorithm 1 CoPiEr (Co-training for Policy Learning)

1: **Input:** A distribution $\mathcal{D}$ of MDPs, two policies π^A, π^B, mapping functions $f_{A\to B}, f_{B\to A}$
2: **repeat**
3: Sample $\mathcal{M} \sim \mathcal{D}$, form $\mathcal{M}^A, \mathcal{M}^B$
4: Run π^A on $\mathcal{M}^A$ to generate trajectories $\{\tau^A_i\}^m_{i=1}$
5: Run π^B on $\mathcal{M}^B$ to generate trajectories $\{\tau^B_j\}^n_{j=1}$
6: $\{\tau'^A_i\}, \{\tau'^B_j\} \leftarrow$ EXCHANGE$(\{\tau^A_i\}, \{\tau^B_j\})$
7: $\pi^A \leftarrow$ UPDATE$(\pi^A, \{\tau^A_i\}, \{\tau'^A_j\})$
8: $\pi^B \leftarrow$ UPDATE$(\pi^B, \{\tau^B_i\}, \{\tau'^B_j\})$
9: **until** Convergence

Algorithm 2 EXCHANGE

1: **Input:** Trajectories $\{\tau^A_i\}^m_{i=1}$ and $\{\tau^B_j\}^n_{j=1}$
2: Compute estimate $\hat{\eta}(\pi^A, \mathcal{M}^A) = \frac{1}{m}\sum_{i=1}^m r(\tau^A_i)$
3: Compute estimate $\hat{\eta}(\pi^B, \mathcal{M}^B) = \frac{1}{n}\sum_{j=1}^n r(\tau^B_j)$
4: **if** $\hat{\eta}(\pi^A, \mathcal{M}^A) > \hat{\eta}(\pi^B, \mathcal{M}^B)$ **then**
5: $\{\tau^{A\to B}_i\} \leftarrow \{f_{A\to B}(\tau^A_i)\}^m_{i=1}$
6: $\{\tau^{B\to A}_j\} \leftarrow \emptyset$
7: **else**
8: $\{\tau^{A\to B}_i\} \leftarrow \emptyset$
9: $\{\tau^{B\to A}_j\} \leftarrow \{f_{B\to A}(\tau^B_j)\}^n_{j=1}$
10: **end if**
11: **return** $\{\tau^{A\to B}_i\}, \{\tau^{B\to A}_j\}$

This mirrors the theoretical insight presented in Section 4, where based on which sub-distribution an MDP is sampled from, the relative quality of the two policies is different.

For UPDATE, we can form a loss function that is derived from either imitation learning or reinformcent learning. Recall that we aim to optimize the β terms in Theorem 1, however it is not feasible to directly optimize that. So we consider a surrogate loss C (line 2 of Algorithm 3) that measures the policy difference. In practice, we use typically behavior cloning loss as the surrogate.

6. Experiments on Minimum Vertex Cover

We now present empirical results on minimum vertex cover by applying a combination of policy co-training: reinforcement learning on one view and imitation learning on the other.

Setup. We consider the challenging combinatorial optimization problem of minimum vertex cover (MVC). We use 150 randomly generated Erdős-Rényi (51) graph instances for each scale, with scales ranging {100-200, 200-300, 300-500, 400-500} vertices. For training, we use 75 instances, which we partition into 15 labeled and 60 unla-

Algorithm 3 UPDATE

1: **Input:** Current policy π, sampled trajectories from π, $\{\tau_i\}_{i=1}^m$ and demonstrations $\{\tau'_j\}_{j=1}^n$
2: Form a loss function $\mathcal{L}(\pi) = \begin{cases} -\sum_{i=1}^m r(\tau_i) + \lambda C(\pi, \{\tau'_j\}_{j=1}^n), & \text{RL with IL loss} \\ \lambda C(\pi, \{\tau'_j\}_{j=1}^n), & \text{IL loss only} \end{cases}$
3: Update $\pi \leftarrow \pi - \alpha \nabla \mathcal{L}(\pi)$

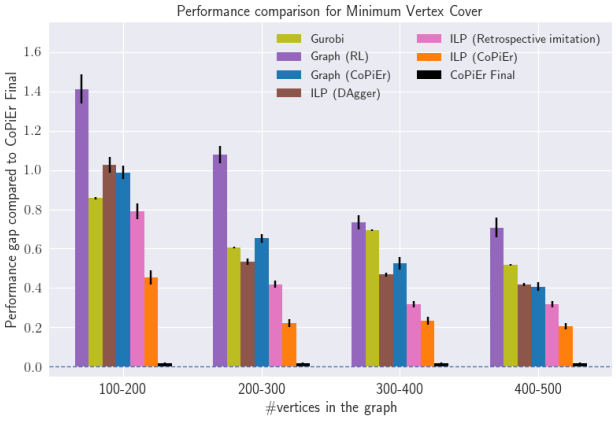


Figure 2. Comparison of CoPiEr with other learning-based baselines and a commercial solver, Gurobi. The y-axis measure relative gaps of various methods compared with CoPiEr Final. CoPiEr Final outperforms all the baselines. Notably, the gaps are significant because getting optimizing over large graphs is very challenging.

beled instances. We use the best solution found by Gurobi within 1 hour as the expert solution for the labeled set to bootstrap imitation learning. For each scale, we use 30 held-out graph instances for validation, and we report the performance on 45 test graph instances.

Views and Features. The two views are the graphs themselves and integer linear programs constructed from the graphs. For the graph view, we use DQN-based reinforcement learning (4) to learn a sequential vertex selection policy. We use `structure2vec` (52) to compute graph embeddings to use as state representations. For the ILP, we use imitation learning (1) to learn node selection policy for branch-and-bound search. A node selection policy determines which node to explore next in the current branch-and-bound search tree. We use node-specific features (e.g., LP relaxation lower bound and objective value) and tree-specific faetures (e.g., integrality gap, and global lower and upper bounds) as our state representations.

Policy Class. For the graph view, our policy class is similar to (4). In order to perform end-to-end learning of the parameters with labeled data exchanged between the two views, we use DQN (27) with supervised losses (53) to learn to imitate the better demonstrations from ILP view. For all our experiments, we determined the regularizer for the supervised losses and other parameters through cross-validation on the smallest scale (100-200 vertices). The graph view models are pre-trained with the labeled set using behavior cloning. We use the same number of training iterations for all the methods.

For the ILP view, our policy class consists of a node ranking model that prioritizes which node to consider next. We use RankNet (54) as the ranking model, instantiated using a 2-layer neural network with ReLU as activation functions. We implement our approach for the ILP view within the SCIP (55) integer programming framework.

Methods Compared. At test time, when a new graph is given, we can run both policies and return the better solution. We term this practical version "CoPiEr Final" and measure other policies' performance against it. We compare with single view learning baselines. For the graph view, we compare with RL-based policy learning over graphs (4), labelled as "Graph (RL)". And for the ILP view, we compare with imitation learning method (1) "ILP (DAgger)", retrospective imitation method (5) "ILP (Retrospective Imitation)" and a commercial solver Gurobi (56). We also show the performance of the two policies in CoPiEr as standalone policies instead of combining them, labelled "Graph (CoPiEr)" and "ILP (CoPiEr)". ILP methods are limited by the same node budget in branch-and-bound trees.

Results. Figure 2 shows the results. We see that CoPiEr Final outperforms all baselines as well as Gurobi. Interestingly, it also performs much better than either standalone CoPiEr policies, which suggests that Graph (CoPiEr) is better for some instances while ILP (CoPiEr) is better on others. This finding validates combining the two views to maximize the benefits from both.

7. Conclusion & Future Work

We have presented CoPiEr (Co-training for Policy Learning), a general framework for policy learning for sequential decision making tasks with two representations. Our approach is compatible with both reinforcement learning and imitation learning as subroutines. We evaluated on a challenging combinatorial optimization problem which shows significant improvements over numerous baselines.

There are many interesting directions for future work. On the theory front, directions include extending to more than two views. On the application front, algorithms such as CoPiEr can potentially improve performance in a wide range of robotic and other autonomous systems that utilize different sensors and image data.

References

[1] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In *Advances in neural information processing systems*, pages 3293–3301, 2014.

[2] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[3] Azalia Mirhoseini, Hieu Pham, Quoc V Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

[4] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

[5] Jialin Song, Ravi Lanka, Albert Zhao, Yisong Yue, and Masahiro Ono. Learning to search via retrospective imitation. *arXiv*, 2018.

[6] Mislav Balunovic, Pavol Bielik, and Martin Vechev. Learning to solve smt formulas. In *Advances in Neural Information Processing Systems*, pages 10338–10349, 2018.

[7] Endre Boros and Peter L Hammer. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180, 1991.

[8] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of maxcut. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 53–61. Society for Industrial and Applied Mathematics, 2007.

[9] AJ Orman and HP Williams. A survey of different integer programming formulations of the travelling salesman problem. In *Optimisation, econometric and financial analysis*, pages 91–104. Springer, 2007.

[10] Temel Öncan, İ Kuban Altınel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

[11] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[12] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Googles multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[13] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Learning Theory (COLT)*, 1998.

[14] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[15] Xiaojun Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*, pages 235–243. Association for Computational Linguistics, 2009.

[16] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *International Conference on Machine Learning (ICML)*, 2011.

[17] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2474–2483, 2018.

[18] Elias Boutros Khalil, Pierre Le Bodic, Le Song, George L Nemhauser, and Bistra N Dilkina. Learning to branch in mixed integer programming. In *AAAI*, pages 724–731, 2016.

[19] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *ACM Conference on Information and knowledge Management (CIKM)*, 2000.

[20] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*, volume 2005, pages 74–79. Citeseer, 2005.

[21] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, pages 301–312. IBM Corp., 2011.

[22] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260. SIAM, 2013.

[23] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in neural information processing systems*, pages 2456–2464, 2011.

[24] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[25] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.

[26] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[29] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[32] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[33] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[34] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2018.

[35] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[36] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[37] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[38] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[39] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066, 2015.

[40] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning*, 2017.

[41] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

[42] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[43] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

[44] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2008.

[45] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé. Hierarchical imitation and reinforcement learning. In *International Conference on Machine Learning*, pages 2923–2932, 2018.

[46] Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast policy learning through imitation and reinforcement. *arXiv preprint arXiv:1805.10413*, 2018.

[47] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[48] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[49] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann Publishers Inc., 2002.

[50] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *International Conference on Machine Learning*, 2008.

[51] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[52] Hanjun Dai, Bo Dai, and Le Song. Discriminative Embeddings of Latent Variable Models for Structured Data. In *International Conference on Machine Learning (ICML)*, pages 1–23, 2016.

[53] Todd Hester, Olivier Pietquin, Marc Lanctot, Tom Schaul, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-arnold, John Agapiou, and Joel Z Leibo. Deep Q-Learning from Demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

[54] Chris Burges, Erin Renshaw, and Matt Deeds. Learning to rank using gradient descent. In *International conference on Machine learning*, 1998.

[55] Tobias Achterberg. SCIP : solving constraint integer programs. *Mathematical Programming Computation*, 2009.

[56] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

8. Appendix

8.1. Proofs

To prove Theorem 1, we need to use a policy improvement result for a single MDP (a modified version of Theorem 1 in (17)).

Theorem 2. *Assume for an MDP $\mathcal{M}$, an expert policy π_E have a higer advantage of over a policy π with a margin, i.e., $\eta(\pi_E, \mathcal{M}) - \eta(\pi, \mathcal{M}) \geq \delta$ Define*

$$\alpha = \max_s D_{KL}(\pi'(s) \| \pi(s))$$
$$\beta = \max_s D_{JS}(\pi'(s) \| \pi_E(s))$$
$$\epsilon_{\pi_E} = \max_{s,a} |A_{\pi_E}(s,a)|$$
$$\epsilon_{\pi} = \max_{s,a} |A_{\pi}(s,a)|$$

then $\eta(\pi', \mathcal{M}) \geq \eta_\pi(\pi', \mathcal{M}) - \frac{2\gamma(4\beta\epsilon_{\pi_E} + \alpha\epsilon_\pi)}{(1-\gamma)^2} + \delta$

Proof. The only difference from the original theorem is that the original assumes $\mathbb{E}_{a_E \sim \pi_E(s), a \sim \pi(s)}[A_\pi(s, a_E) - A_\pi(s,a)] \geq \delta' > 0$ for every state s. It is a stronger assumption which is not needed in their analysis. Notice that the advantage of a policy over itself is zero, i.e., $\mathbb{E}_{a \sim \pi(s)}[A_\pi(s,a)] = 0$ for every s, so the margin assumption simplifies to $\mathbb{E}_{a_E \sim \pi_E(s)}[A_\pi(s, a_E)] \geq \delta'$.

By the policy advantage formula,

$$\begin{aligned}
\eta(\pi_E, \mathcal{M}) - \eta(\pi, \mathcal{M}) &= \mathbb{E}_{\tau \sim \pi_E}[\sum_{i=0}^{\infty} \gamma^i A_\pi(s_i, a_i)] \\
&= \mathbb{E}_{s_i \sim \rho_{\pi_E}} \mathbb{E}_{a_i \sim \pi_E(s_i)}[\sum_{i=0}^{\infty} \gamma^i A_\pi(s_i, a_i)] \\
&\geq \mathbb{E}_{s_i \sim \rho_{\pi_E}}[\delta' \sum_{i=0}^{\infty} \gamma^i] \\
&= \frac{\delta'}{1-\gamma}
\end{aligned}$$

So an assumption on per-state advantage translates to a overall advantage. Thus we can make this weaker assumption which is also more intuitive and the original statement still holds with a different δ term. □

Proof of Theorem 1:

Proof. Theorem 1 is a distributional extension to the theorem above. For $\mathcal{M} \sim \mathcal{D}_2$, let $\delta_{\mathcal{M}} = \eta(\pi^B, \mathcal{M}^B) - \eta(\pi^A, \mathcal{M}^A)$.

$$\begin{aligned}
&J(\pi'^A) \\
&= \mathbb{E}_{\mathcal{M} \sim \mathcal{D}}[\eta(\pi'^A, \mathcal{M}^A)] \\
&= \mathbb{E}_{\mathcal{M} \sim \mathcal{D}_1}[\eta(\pi'^A, \mathcal{M}^A)] + \mathbb{E}_{\mathcal{M} \sim \mathcal{D}_2}[\eta(\pi'^A, \mathcal{M}^A)] \\
&\geq \mathbb{E}_{\mathcal{M} \sim \mathcal{D}_1}[\eta(\pi'^A, \mathcal{M}^A)] + \\
&\mathbb{E}_{\mathcal{M} \sim \mathcal{D}_2}[\eta_{\pi^A}(\pi'^A, \mathcal{M}^A) - \frac{2\gamma^A(4\beta\epsilon_{\pi^B} + \alpha\epsilon_{\pi^A})}{(1-\gamma^A)^2} + \delta_{\mathcal{M}}] \\
&\geq \mathbb{E}_{\mathcal{M} \sim \mathcal{D}_1}[\eta_{\pi^A(}(\pi'^A, \mathcal{M}^A) - \frac{2\gamma^A\alpha\epsilon_{\pi^A}}{(1-\gamma^A)^2}] + \\
&\mathbb{E}_{\mathcal{M} \sim \mathcal{D}_2}[\eta_{\pi^A}(\pi'^A, \mathcal{M}^A) - \frac{2\gamma^A(4\beta\epsilon_{\pi^B} + \alpha\epsilon_{\pi^A})}{(1-\gamma^A)^2} + \delta_{\mathcal{M}}] \\
&= \mathbb{E}_{\mathcal{M} \sim \mathcal{D}}[\eta_{\pi^A}(\pi'^A, \mathcal{M}^A)] - \mathbb{E}_{\mathcal{M} \sim \mathcal{D}}[\frac{2\gamma^A\alpha\epsilon_{\pi^A}}{(1-\gamma^A)^2}] - \\
&\mathbb{E}_{\mathcal{M} \sim \mathcal{D}_2}[\frac{2\gamma^A \cdot 4\beta\epsilon_{\pi^B}}{(1-\gamma^A)^2}] + \mathbb{E}_{\mathcal{M} \sim \mathcal{D}_2}[\delta_{\mathcal{M}}] \\
&\geq J_{\pi^A}(\pi'^A) - \frac{2\gamma^A(4\beta^B_{\mathcal{D}_2}\epsilon^B_{\mathcal{D}_2} + \alpha^A_{\mathcal{D}}\epsilon^A_{\mathcal{D}})}{(1-\gamma^A)^2} + \delta_2
\end{aligned}$$

The derivation for $J(\pi'^B)$ is the same. □