Benchmarking of Multi-agent Reinforcement Learning Strategies for Optimizing Cutting Plane Selection ⋆

Arjun M.[1] **Hariprasad Kodamana**[1,2,3,†]
Manojkumar Ramteke[1,2,†]

[1] *Department of Chemical Engineering, Indian Institute of Technology Delhi*
[2] *Yardi School of Artificial Intelligence, Indian Institute of Technology Delhi*
[3] *Indian Institute of Technology Delhi - Abu Dhabi, Khalifa City B, UAE*

Abstract:
Cutting planes refine the feasible region of relaxed Integer Programming (IP) problems, but traditional methods relying on static heuristics often fail to generalize effectively. This study investigates Multi-Agent Reinforcement Learning (MARL) frameworks – TD3-TD3, PPO-PPO, and TD3-PPO – for dynamic, adaptive cut selection. MARL improves scalability and exploration by distributing decision-making across specialized agents, outperforming conventional techniques. The hybrid TD3-PPO configuration balances TD3's sample-efficient learning with PPO's robust exploration. PPO-PPO demonstrates superior exploration and success rates in a sensor network design problem relevant to process control, while TD3-TD3 offers greater stability. The results highlight MARL's potential for enhancing IP solvers and solving complex optimization problems.

Keywords: Multi-Agent Reinforcement Learning, Integer Programming, Cutting Planes, Optimization Algorithms, Sensor Network Design

1. INTRODUCTION

In mathematical programming, Linear Programming (LP) is foundational for solving problems involving finding the extremum of a linear objective function subject to linear constraints. Its wide applicability has solidified its role in fields such as supply chain logistics, financial planning, and industrial process control. However, many real-world problems will involve integer decision variables, leading to Integer Programming (IP) problems, where the solution space is discrete, significantly increasing computational complexity. This contrasts with LP problems, where the solution space is continuous and thus more straightforward to explore. The structure of an IP problem is given as:

$$\begin{aligned} z = \min \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x \geq 0 \\ & x \in Z^n \end{aligned} \tag{1}$$

Here, $x \in \mathbb{R}^n$ represents the vector of decision variables, $c \in \mathbb{R}^n$ denotes the cost vector, and n is the dimension of the problem (Boyd and Vandenberghe (2004)). The last constraint enforces the integrality of the decision variables, thereby introducing combinatorial complexity, as the number of feasible solutions grows exponentially with problem size. Consequently, solving IP problems requires techniques like branch-and-bound, branch-and-cut, and cutting plane methods to effectively search the complex solution space for optimal solutions.

Cutting planes, an essential component of modern IP solvers, iteratively refine the LP relaxation's feasible region by introducing additional constraints, known as cuts, to exclude non-integer solutions while preserving feasible integer solutions. This process begins by solving a relaxed version of the IP problem, in which x can take fractional values, forming the feasible region $\mathcal{C}^{(0)} = \{x \mid Ax \leq b, x \geq 0\}$. The optimal solution of this relaxation, $x_{\text{LP}}^{(0)}$, has an objective value $z_{\text{LP}}^{(0)}$. Since $\mathcal{C}^{(0)}$ includes the original IP problem's feasible region, $z_{\text{LP}}^{(0)} \leq z_{\text{IP}}$. If $x_{\text{LP}}^{(0)}$ is integer-valued, it is optimal for the original IP problem. Otherwise, if $x_{\text{LP}}^{(0)} \notin \mathbb{Z}^n$, the cutting plane method adds a new constraint, $\alpha^T x \leq \beta$, that is satisfied by all integer solutions but violated by $x_{\text{LP}}^{(0)}$, thereby excluding it from the feasible region. This new constraint updates the feasible region to $\mathcal{C}^{(1)} = \mathcal{C}^{(0)} \cap \{x \mid \alpha^T x \leq \beta\}$, and the LP relaxation is solved again to obtain a new solution $x_{\text{LP}}^{(1)}$ and its objective value $z_{\text{LP}}^{(1)}$. At each iteration k, the feasible region is refined by adding cuts, $\mathcal{C}^{(k+1)} = \mathcal{C}^{(k)} \cap \{x \mid \alpha_k^T x \leq \beta_k\}$, leading to a new solution $x_{\text{LP}}^{(k+1)}$ and objective value $z_{\text{LP}}^{(k+1)}$. This itera-

⋆ The authors sincerely acknowledge the funding received from DST-SERB India with file number CRG/2022/003722.
† Corresponding Authors. Email: kodamana@iitd.ac.in, mcramteke@chemical.iitd.ac.in

tive refinement continues until an optimal integer solution $x_{\text{LP}}^{(opt)} \in \mathbb{Z}^n$ is found, satisfying all original IP constraints and representing the optimal solution for the problem.

Classical cutting plane methods, such as Gomory cuts, cover cuts, and lift-and-project cuts, have shown effectiveness in solving various IP problems (Conforti et al., 2014). However, these traditional methods often rely on static heuristics that may not generalize well across diverse problem instances. As problem complexity and variability increase, heuristic-based methods face significant limitations, prompting the exploration of dynamic, adaptive approaches based on machine learning (ML) (Bengio et al., 2021), and particularly, reinforcement learning (RL) (Sutton and Barto, 2018).

RL, in particular, has gained prominence for its ability to learn optimal policies through interaction with its environment, making it effective in navigating complex solution spaces as demonstrated in various fields, including process systems engineering (Joshi et al., 2021, 2023; Gupta et al., 2023, 2024). RL has also demonstrated remarkable potential in solving combinatorial optimization problems, as evidenced by its success in addressing tasks such as the Traveling Salesman Problem (Bello et al., 2016) and resource allocation problems (Bengio et al., 2021).

Recent RL applications in cut generation and selection, such as adaptive Gomory cuts (Turner et al., 2022) and neural cut selection (Paulus et al., 2022), have demonstrated superior performance over traditional heuristic-based methods in large-scale IP problems. However, these studies primarily focus on single-agent frameworks, which suffer from scalability limitations and restricted exploration capabilities. Single-agent RL often struggles to navigate the exponential growth of the action and state spaces in large problems, limiting its ability to discover diverse and effective cuts (Gupta et al., 2024). Multi-Agent Reinforcement Learning (MARL) has emerged as a promising alternative to address these challenges. MARL extends RL by incorporating multiple agents, each of which operates autonomously while collaborating within a shared environment. Studies comparing MARL to single-agent RL (Busoniu et al., 2008) show that distributing decision-making across specialized agents enhances exploration and improves solution diversity. This multi-agent paradigm is particularly advantageous for cut selection in IP problems, as each agent can specialize in discovering and applying specific types of cuts, such as Gomory or cover cuts. By allowing agents to collaboratively explore different regions of the solution space, MARL generates higher-quality cuts and accelerates convergence, outperforming single-agent RL in complex optimization tasks.

As IP problems grow in complexity, the transition from single-agent RL to MARL becomes crucial. In this context, this work focuses on Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) and Proximal Policy Optimization (PPO) (Schulman et al., 2017) due to their complementary characteristics. TD3, an off-policy algorithm, excels in continuous action spaces and mitigates overestimation bias using double Q-learning and delayed policy updates. Its off-policy nature allows it to utilize past experiences for efficient learning, making it particularly effective in iterative optimization tasks. PPO, in contrast, is an on-policy algorithm that constrains policy updates to ensure stability, providing robust exploration in dynamic environments. By combining TD3 and PPO within a MARL framework, this study aims to balance exploration and exploitation, leveraging PPO's adaptability alongside TD3's sample efficiency to improve cut selection.

This paper conducts a benchmarking study on cut selection using different MARL frameworks, focusing on architecture with two RL agents, namely, TD3-TD3, PPO-PPO, and TD3-PPO configurations. By applying these RL algorithms in a multi-agent context, the study evaluates their effectiveness in selecting cutting planes for IP problems. We evaluate the frameworks on a sensor network design problem—a quintessential process control task where IP models discrete sensor placement decisions under observability and cost constraints (M and Magbool Jan, 2023). This comprehensively evaluates how each method performs across different optimization scenarios.

The remainder of the article is structured as follows: Section 2 outlines the problem statement and describes the environmental setup. Section 3 details the proposed MARL framework and its components. The experimental results are discussed in Section 4, and the conclusions are presented in Section 5.

2. PROBLEM STATEMENT AND ENVIRONMENT SETUP

In this work, the primary goal is to refine the feasible region of an IP iteratively by introducing cutting planes that progressively tighten the relaxation, ultimately steering the solution toward an optimal integer outcome by using MARL frameworks.

Environment and Problem Definition: The environment is designed to model the IP problem, characterized by a cost vector $c \in \mathbb{R}^n$, a constraint matrix $A \in \mathbb{R}^{m \times n}$, and a constraint bound vector $b \in \mathbb{R}^m$. Initially, the relaxed LP version of the IP is solved, producing an optimal solution. This solution is then examined for integrality. If the solution is not integral, the environment generates cutting planes based on the current solution, which is subsequently incorporated into the LP formulation to tighten the feasible region.

State Space: The state space $\mathcal{S}$ in this environment is characterized by the current solution vector alongside the objective value of the LP problem. At any iteration t, the state is composed of the current solution values $x_{\text{LP}}^{(t)}$ and the corresponding objective value $z_{\text{LP}}^{(t)}$. Mathematically, the state at iteration t is defined as:

$$s_t = \{x_{\text{LP}}^{(t)}, z_{\text{LP}}^{(t)}\}. \tag{2}$$

This state serves as the input for both RL agents, which use it to decide on the next action. If the solution is integral, the process terminates; otherwise, the environment transitions to a new state based on the updated LP solution after applying a cut.

Action Space: The action space $\mathcal{A}$ in this environment consists of all possible cutting planes (e.g., Gomory cuts) that can be added to the LP formulation in the next iteration. Each action a_t represents a specific cut defined

by a vector $\alpha_i \in \mathbb{R}^n$ and a scalar $\beta_i \in \mathbb{R}$, forming an inequality constraint $\alpha_i^T x \leq \beta_i$. Thus, an action can be expressed as:

$$a_t = \{\alpha_i^T x \leq \beta_i\}. \tag{3}$$

The RL agents select an action from this space to modify the current LP formulation, aiming to drive the solution closer to integrality.

Reward Function: The reward function guides the RL agent in selecting effective cuts while penalizing excessive or ineffective cuts. It combines objective improvement and a penalty for the number of cuts introduced. The immediate reward at iteration t is defined as:

$$r_t = (z_{\text{LP}}^{(0)} - z_{\text{LP}}^{(t)}) - n_c \times f, \tag{4}$$

where $z_{\text{LP}}^{(0)}$ is the initial objective value, $z_{\text{LP}}^{(t)}$ is the current value, n_c is the number of cuts introduced, and f is a hyper-parameter controlling the penalty magnitude. This structure balances exploration and refinement by rewarding substantial objective improvement while discouraging unnecessary cuts. The use of $z_{\text{LP}}^{(0)}$ (instead of incremental gains like $z_{\text{LP}}^{(t-1)} - z_{\text{LP}}^{(t)}$) encourages cumulative progress rather than incremental gains, avoiding local optima.

Transitions: The transition dynamics in this environment follow a straightforward iterative process. Starting from an initial state s_t, the agent selects an action a_t, which corresponds to adding a specific cut $\alpha_i^T x \leq \beta_i$ to the LP. This results in a new LP formulation, which is then solved to yield an updated solution vector $x_{\text{LP}}^{(t+1)}$ and a new objective value $z_{\text{LP}}^{(t+1)}$. Consequently, the environment transitions to a new state $s_{t+1} = \{x_{\text{LP}}^{(t+1)}, z_{\text{LP}}^{(t+1)}\}$. This iterative process continues, with the agent refining its strategy for selecting effective cuts through learning.

Figure 1 presents a flowchart illustrating a typical MARL flowchart. With the problem environment defined, we next describe the MARL frameworks used to generate effective cutting planes.

3. PROPOSED MARL FRAMEWORK FOR CUTTING PLANE SELECTION

In this study, we propose a MARL framework that combines the strengths of the TD3 and PPO algorithms. The framework is designed to improve the efficiency of generating and selecting cutting planes in IP problems by leveraging the exploration-exploitation trade-offs inherent to both algorithms. Below, we detail the configurations of the TD3-TD3, PPO-PPO, and TD3-PPO frameworks, including specific equations and notations for each agent.

3.1 TD3-TD3 Framework

The TD3-TD3 framework employs two agents, denoted $TD3_1$ and $TD3_2$, each implementing the TD3 algorithm. TD3 is an off-policy actor-critic method tailored for continuous action spaces, which mitigates overestimation bias through double Q-learning and delayed policy updates. Given a state s_t at time step t, the actor network of agent $TD3_i$ (where $i = 1, 2$) outputs an action $a_{t,i}^{TD3} = \pi_{\theta_i}(s_t)$, where θ_i denotes the parameters of the actor network for agent $TD3_i$. After execution, the action $a_{t,i}^{TD3}$ yields a new state s_{t+1} and reward r_t. The critic networks estimate the Q-value for each state-action pair $(s_t, a_{t,i}^{TD3})$ as $Q_{\phi_{i,1}}(s_t, a_{t,i}^{TD3})$ and $Q_{\phi_{i,2}}(s_t, a_{t,i}^{TD3})$, where $\phi_{i,1}$ and $\phi_{i,2}$ are the parameters of the critic networks for agent $TD3_i$. The target Q-value $\hat{Q}_i(s_t, a_{t,i}^{TD3})$ is computed as:

$$\begin{aligned}\hat{Q}_i(s_t, a_{t,i}^{TD3}) =& r_t + \gamma \min(Q_{\phi_{i,1}}(s_{t+1}, a_{t+1,i}^{TD3}),\\ & Q_{\phi_{i,2}}(s_{t+1}, a_{t+1,i}^{TD3}),\end{aligned} \tag{5}$$

where γ is the discount factor, and $a_{t+1,i}^{TD3} = \pi_{\theta_i}(s_{t+1}) + \epsilon$ includes exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$.

The loss for the actor network in agent $TD3_i$ is defined as:

$$\mathcal{L}_{\text{actor},i} = -\mathbb{E}_{s\sim B}[Q_{\phi_{i,1}}(s, \pi_{\theta_i}(s))], \tag{6}$$

and the critic network loss is minimized by:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s,a,r,s')\sim B}[\Big(Q_{\phi_{i,j}}(s,a) - \hat{Q}_i(s,a)\Big)^2, \tag{7}$$

where B is the replay buffer shared between $TD3_1$ and $TD3_2$. In the TD3-TD3 framework, both agents share a centralized replay buffer, allowing them to learn from each other's experiences. This shared buffer contains tuples of (s_t, a_t, r_t, s_{t+1}) collected during the interaction with the environment. The agents independently select actions, and the best action is chosen based on its Q-value, which ensures that the most promising cutting planes are selected for inclusion in the IP problem.

3.2 PPO-PPO Framework

The PPO-PPO framework consists of two agents, PPO_1 and PPO_2, applying the on-policy actor-critic PPO algorithm. PPO constrains policy updates to prevent excessive divergence from prior policies, stabilizing training while encouraging exploration. Each PPO agent computes an action a_t^{PPO} by sampling from the probability distribution $\pi_\theta^{PPO}(a_t|s_t)$, where θ represents the parameters of the actor network. The advantage function, $A_t = Q(s_t, a_t^{PPO}) - V(s_t)$, where $V(s_t)$ is the state value, guides policy updates.

The PPO objective function for each agent is given by:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)A_t\right)\right], \tag{8}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and ϵ is the clipping parameter.

The value function update minimizes:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_t\left[\left(V_\psi(s_t) - R_t\right)^2\right], \tag{9}$$

where V_ψ represents the predicted state value, and R_t is the cumulative reward at time t. Similar to the TD3-TD3 framework, the PPO agents share a centralized buffer that stores trajectories, allowing both agents to leverage each other's experiences. This shared buffer enables the agents

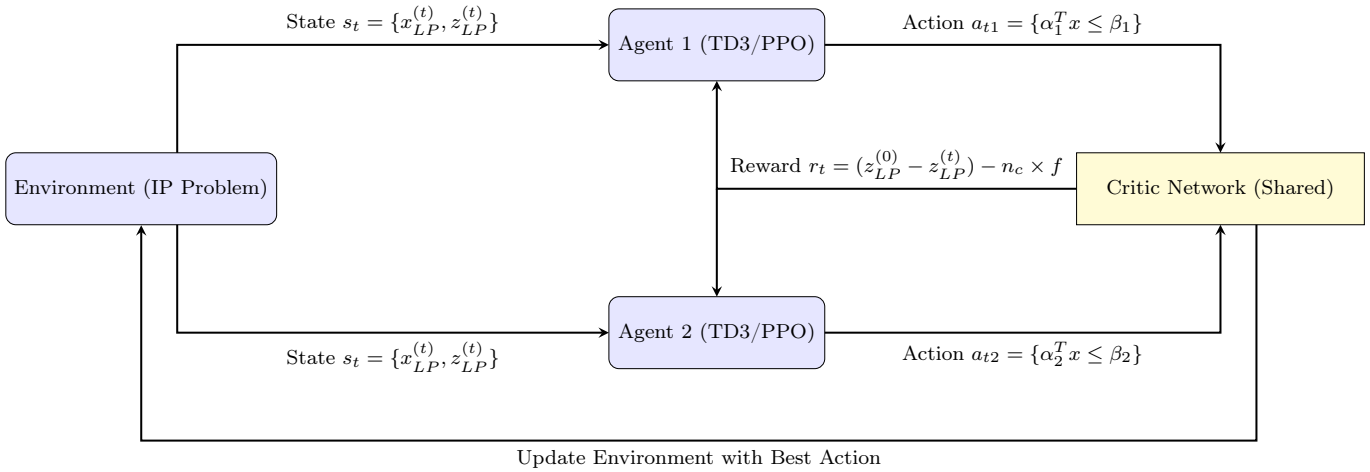

Fig. 1. Flowchart of MARL framework for IP solutions using shared experiences from two agents (TD3/PPO) to optimize cut selection

to optimize their policies based on a more diverse set of trajectories, leading to better performance in selecting effective cutting planes.

3.3 TD3-PPO Hybrid Framework

The TD3-PPO framework leverages the complementary strengths of TD3's stability in continuous action spaces and PPO's robust exploration in dynamic environments. By combining these two agents, the framework balances exploitation and exploration, critical for optimizing cut selection in IP problems that involve mixed discrete-continuous decision spaces.

TD3 operates off-policy, updating its policy using past experiences stored in a shared replay buffer, while PPO works on-policy, sampling recent trajectories tied closely to its current policy iteration. Replay sampling is limited to recent trajectories to maintain PPO's on-policy integrity and prevent policy divergence. This coordination allows PPO to benefit from TD3's exploratory experiences while still selecting diverse, effective cuts. Simultaneously, TD3 refines its policies using both historical and recent data, ensuring stable exploitation. This synergistic interaction improves adaptability in dynamic optimization scenarios as both agents collaboratively explore and refine optimal cutting planes, enhancing learning efficiency and accelerating convergence.

To select the optimal action at iteration t, a Q-value comparison mechanism evaluates the actions proposed by both agents:

$$a_t^* = \begin{cases} a_t^{TD3} & \text{if } Q_{\phi_1}(s_t, a_t^{TD3}) \geq V_\psi(s_t), \\ a_t^{PPO} & \text{otherwise,} \end{cases} \tag{10}$$

where Q_{ϕ_1} is the TD3 critic's Q-value, and V_ψ is PPO's state-value estimate. This mechanism prioritizes TD3's stable exploitation while allowing PPO's exploration when advantageous. The shared buffer enables mutual learning: high-reward actions identified by TD3 are stored and later inform PPO's policy updates, while PPO's exploration enriches TD3's decision-making. This collaborative learning accelerates convergence and improves the diversity and effectiveness of the generated cuts, addressing the challenges of IP optimization.

4. RESULTS AND DISCUSSIONS

This section discusses the comparative performance of three MARL frameworks used to generate cutting planes. The analysis focuses on evaluating the cumulative rewards and success rates for both cover cuts and single cuts across various problem scenarios. The provided results are based on the convergence and effectiveness of each approach in the sensor network design problem. Training of the RL agents was performed on a workstation equipped with an NVIDIA RTX 3080 GPU. The implementation was developed in Python 3.12.1, utilizing the PyTorch library for neural network training and the PuLP package for solving LP problems.

4.1 Sensor Network Design Problem

In a practical application, the MARL framework was tested on a sensor network design problem involving a flow network, as shown in Figure 2 (M and Magbool Jan, 2023). The objective was to minimize cost, subject to constraints on observability, a cost cap, and a minimum number of sensors. The problem was formulated as an IP problem, and the RL algorithms were used to find optimal solutions.

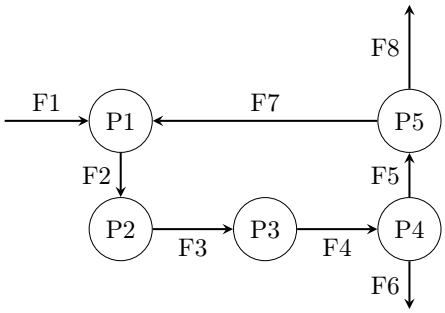

Fig. 2. Representative flow network

Figures 3 and 4 illustrate the cumulative reward progression for each framework (TD3-TD3, PPO-PPO, and TD3-

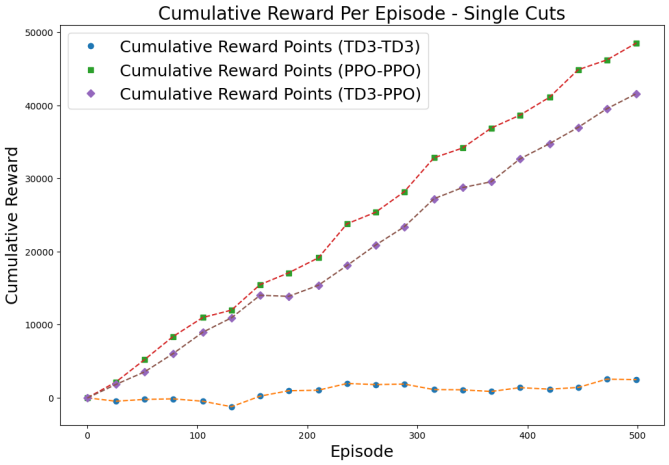

Fig. 3. Cumulative rewards comparison of single Gomory cuts in the sensor network design problem across 500 episodes

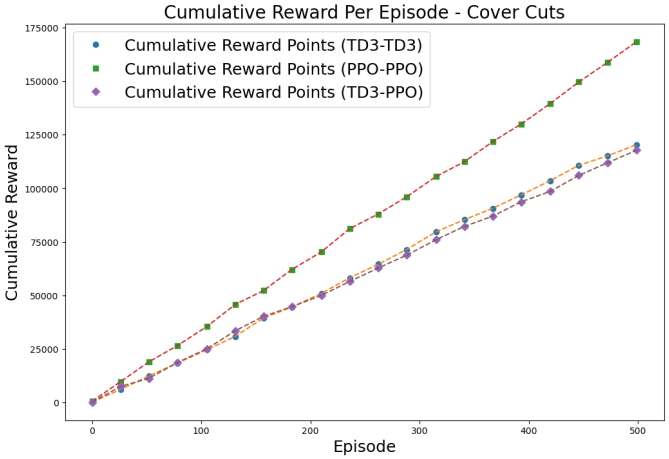

Fig. 4. Cumulative rewards comparison of cover cuts in the sensor network design problem across 500 episodes

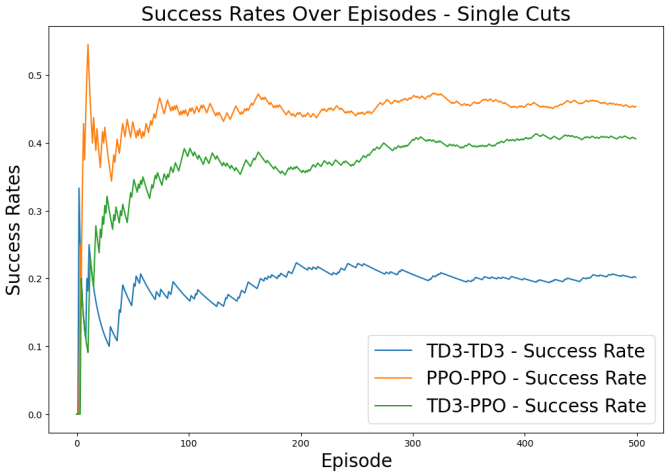

Fig. 5. Success rate comparison of single Gomory cuts in the sensor network design problem across 500 episodes

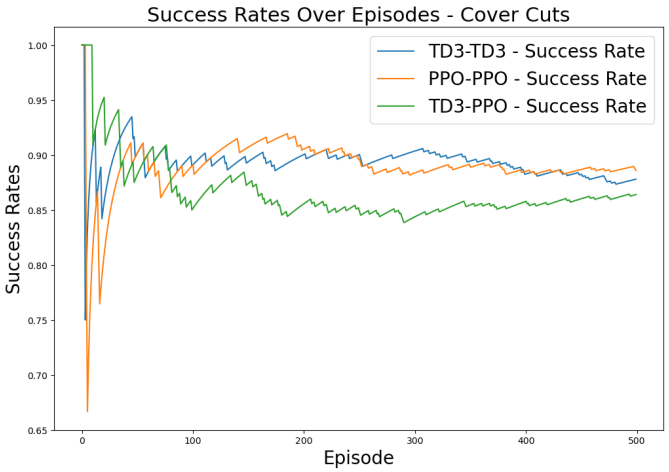

Fig. 6. Success rate comparison of cover cuts in the sensor network design problem across 500 episodes

PPO) when applied to the sensor network design problem using both single and cover cuts. As seen in Figure 3, all three frameworks show a steady increase in cumulative rewards, indicating that learning is progressing. PPO-PPO consistently achieves the highest cumulative rewards, followed by TD3-PPO and TD3-TD3. The superior performance of PPO-PPO indicates a stronger ability to explore the solution space, selecting more effective single cuts that improve the feasible region more efficiently. TD3-PPO performs reasonably well, demonstrating a balanced approach between exploration and exploitation, while TD3-TD3 lags slightly behind in this scenario.

In Figure 4, PPO-PPO once again outperforms the other frameworks, showing the highest cumulative rewards. TD3-TD3 and TD3-PPO both follow a similar trajectory, with TD3-TD3 performing slightly better than TD3-PPO. These results suggest that PPO-PPO utilizes cover cuts to tighten the feasible region in the sensor network design problem more effectively.

The success rates for all frameworks in the sensor network design problem are shown in Figures 5 and 6. As seen in Figure 5, PPO-PPO achieves the highest success rate for single cuts, stabilizing around 0.45, followed by TD3-PPO slightly above 0.4, and TD3-TD3 around 0.2. This indicates that PPO-PPO is the most effective at consistently selecting valid cuts, with TD3-PPO performing reasonably well. Although the success rates are below 50%, this is expected due to the complexity of selecting effective cuts in IP problems. Despite this, the cumulative reward trends show steady improvement in overall solution quality, suggesting that the framework effectively identifies and prioritizes useful cuts over time.

Figure 6 shows the cover-cut success rates. PPO-PPO shows the best balance of speed and effectiveness, achieving the highest success rates of 90%. TD3-TD3, though slower, is highly stable, while TD3-PPO offers a middle-ground solution. These results indicate that PPO-PPO is more effective at consistently selecting successful cover cuts in the sensor network design problem, with TD3-PPO and TD3-TD3 performing slightly below but very close in this case.

Table 1 compares the run times and success rates of TD3-TD3, PPO-PPO, and TD3-PPO frameworks for single Gomory and lifted cover cuts in a flow network problem.

PPO-PPO consistently showed the fastest performance, with a run time of 57.5 seconds for single Gomory cuts and 36.2 seconds for lifted cover cuts. TD3-TD3 had the longest times across both problems, while TD3-PPO offered a middle ground. Overall, PPO-PPO demonstrated superior efficiency, with TD3-PPO being a balanced option and TD3-TD3 slower across scenarios.

Table 1. Summary of run times (in seconds) and success rate of different MARL frameworks in representative flow network problem for single and cover cut experiments

	TD3-TD3		PPO-PPO		TD3-PPO	
Cut type/Agent	Runtime	Success rate	Runtime	Success rate	Runtime	Success rate
Single Gomory cut	73.5	0.2	**57.5**	**0.45**	69.3	0.4
Lifted cover cut	39.9	0.87	**36.2**	**0.9**	40.1	0.85

The results highlight distinct differences in how the three RL configurations – TD3-TD3, PPO-PPO, and TD3-PPO perform in selecting effective cutting planes.

(1) PPO-PPO stands out across all metrics, consistently achieving higher cumulative rewards and success rates. Its on-policy approach encourages greater exploration, making it particularly effective in complex scenarios like sensor network design. This ability to quickly explore and identify effective cutting planes allows it to converge faster than the other methods.
(2) TD3-TD3 takes a more stable and cautious approach, leading to a slower accumulation of rewards but still competitive success rates. It performs well in environments where extensive exploration is less critical, such as cover cuts, but struggles to keep up with the more exploration-heavy PPO-PPO in dynamic problems like sensor network design.
(3) TD3-PPO, as a hybrid method, balances exploration and stability, performing better than TD3-TD3 in tasks that require more exploration, like single cuts. However, it doesn't achieve the rapid convergence of PPO-PPO. Its balanced nature makes it a versatile option for environments needing both stability and exploration.

In summary, the choice of MARL framework for cutting plane selection depends on the nature of the optimization problem. PPO-PPO is particularly well-suited for problems where exploration is key to success, such as in the sensor network design problem. TD3-TD3 is more appropriate for scenarios where stability and careful exploitation of past experiences are necessary, while TD3-PPO offers a flexible solution that can adapt to a variety of problem complexities.

5. CONCLUSIONS

This paper demonstrates the potential of MARL strategies to improve cutting plane selection for IP problems. Comparing TD3-TD3, PPO-PPO, and TD3-PPO, the study reveals their strengths and weaknesses. PPO-PPO excelled in tasks requiring extensive exploration, while TD3-TD3 delivered slower but consistent results, ideal for stable environments. The hybrid TD3-PPO offered a balanced, flexible approach suited for diverse challenges. These findings highlight MARL's potential in solving complex optimization problems, like the sensor network design problem, by enabling more dynamic and efficient cut selection. Future extensions of this MARL framework can include adaptive reward functions, cross-framework integrations, and applications in varied IP problem domains to further enhance MARL's robustness and utility.

REFERENCES

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2), 405–421.

Boyd, S.P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2), 156–172.

Conforti, M., Cornuejols, G., and Zambelli, G. (2014). *Integer Programming*. Springer Publishing Company, Incorporated.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Gupta, N., Anand, S., Joshi, T., Kumar, D., Ramteke, M., and Kodamana, H. (2023). Process control of mab production using multi-actor proximal policy optimization. *Digital Chemical Engineering*, 8, 100108.

Gupta, N., Anand, S., Kumar, D., Ramteke, M., Kandath, H., and Kodamana, H. (2024). A twin agent reinforcement learning framework by integrating deterministic and stochastic policies. *Industrial & Engineering Chemistry Research*.

Joshi, T., Kodamana, H., Kandath, H., and Kaisare, N. (2023). Tasac: A twin-actor reinforcement learning framework with a stochastic policy with an application to batch process control. *Control Engineering Practice*, 134, 105462.

Joshi, T., Makker, S., Kodamana, H., and Kandath, H. (2021). Twin actor twin delayed deep deterministic policy gradient (tatd3) learning for batch process control. *Computers & Chemical Engineering*, 155, 107527.

M, A. and Magbool Jan, N. (2023). Convex optimization approach to design sensor networks using information theoretic measures. *AIChE Journal*, 70(2), e18267. doi: https://doi.org/10.1002/aic.18267.

Paulus, M.B., Zarpellon, G., Krause, A., Charlin, L., and Maddison, C. (2022). Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *International conference on machine learning*, 17584–17600. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Turner, M., Koch, T., Serrano, F., and Winkler, M. (2022). Adaptive cut selection in mixed-integer linear programming. *arXiv preprint arXiv:2202.10962*.