Joint Learning of Policy with Unknown Temporal Constraints for Safe Reinforcement Learning

Lunet Yifru,[1] **Ali Baheri** [2]

[1] West Virginia University
[2] Rochester Institute of Technology
lay0005@mix.wvu.edu, akbeme@rit.edu

Abstract

In many real-world applications, safety constraints for reinforcement learning (RL) algorithms are either unknown or not explicitly defined. We propose a framework that concurrently learns safety constraints and optimal RL policies in such environments. Our approach merges a logically-constrained RL algorithm with an evolutionary algorithm to synthesize signal temporal logic (STL) specifications. We showcased our framework in grid-world environments, successfully identifying both acceptable safety constraints and RL policies.

Introduction

RL has emerged as a powerful computational approach for training agents to achieve complex objectives through interactions within stochastic environments (Sutton and Barto 2018). RL algorithms have demonstrated significant success in a wide range of applications and domains (Kober, Bagnell, and Peters 2013; Razzaghi et al. 2022). However, when deploying RL policies in real-world scenarios, particularly those involving safety-critical operations, ensuring the safety of the learning process becomes a paramount concern. Traditional RL algorithms tend to focus on reward maximization, which may inadvertently lead to violation of safety constraints. Safe RL aims to address this challenge by learning policies that not only maximize the expected return but also respect safety constraints throughout the learning process. One promising avenue of research in safe RL involves the use of formal methods, such as temporal logic, for specifying safety constraints in a mathematically rigorous manner. By incorporating temporal logic constraints into the reward function, RL agents can learn policies that are both high-performing and safe. However, this approach assumes the availability of accurate temporal logic specifications, which may not always be the case, especially in complex real-world environments. In this brief, we propose a novel framework for jointly learning RL policies and safety specifications. Our approach combines the strengths of RL for policy optimization with computational techniques for discovering temporal logic constraints from data. This joint learning strategy allows us to efficiently derive an optimal policy and a suitable safety constraint for a given environment, even in situations where the safety constraints are not explicitly provided in advance.

Related Work

Safe RL. Safe RL has garnered significant attention in recent years, as researchers aim to address safety concerns associated with deploying RL agents in safety-critical domains (Garcıa and Fernández 2015; Gu et al. 2022). A prevalent approach to safe RL involves formulating the problem as a constrained optimization task, where the primary objective is to maximize the expected return while satisfying given safety constraints (Achiam et al. 2017). Another direction in safe RL is risk-sensitive RL, which aims to balance the trade-off between exploration, exploitation, and risk management (Mihatsch and Neuneier 2002). Risk-sensitive RL algorithms incorporate risk measures, such as conditional value-at-risk (CVaR) (Tamar, Glassner, and Mannor 2014) and risk envelope (Majumdar et al. 2017), to guide the learning process. An additional approach to ensure safety in RL is through shielding, which intervenes in the agent's actions when it might violate safety constraints (Alshiekh et al. 2018). Integrating formal methods, like temporal logic and Lyapunov-based techniques, into RL algorithms has emerged as a promising direction for safe RL (Hasanbeig, Abate, and Kroening 2018; Alur et al. 2023; Chow et al. 2018).

STL Mining. STL has emerged as an essential formalism for specifying complex temporal properties and constraints in various applications, including robotics and cyber-physical systems. In recent years, researchers have focused on inferring or mining STL specifications from data, to facilitate the development of safe and robust systems. A key approach to mining STL from data is the use of algorithmic techniques, such as optimization-based algorithms (Abbas et al. 2014) and machine learning methods (Fronda and Abbas 2022). Optimization-based techniques seek to minimize an objective function that captures the distance between the candidate STL formulas and the given data traces. Data-driven techniques have shown promise in learning STL specifications from data. Another direction in mining STL from data is the development of automated, scalable, and robust techniques for the discovery of interpretable STL specifications (Mohammadinejad et al. 2020). (Bartocci et al.

2022) provides a comprehensive survey of the various techniques for mining STL specifications from data.

Methodology

We cast the joint learning of policy with unknown specifications as a bi-level optimization problem (Sinha, Malo, and Deb 2017). In this formulation, the upper level optimization aims to infer the correct STL safety constraint, while the lower level optimization focuses on learning the optimal policy under the inferred constraint. A human expert assists the learning by labeling trajectories based on their safety. In this context, safety is attained when a trajectory achieves the environmental objective without violating any safety constraints, i.e., the trace should have a positive robustness value against the true safety constraint. These components are iteratively called upon to simultaneously identify the optimal policy and the suitable STL constraint with the aid of the human expert. The outer loop, an evolutionary algorithm, is designed to infer both the template and the parameters of an STL specification that can classify the labeled dataset. This method is inspired by the work in (Nenzi et al. 2018), which has been shown to result in simpler, more interpretable outputs, as well as an improved misclassification rate compared to those in (Bombara and Belta 2021; Kong et al. 2014). The algorithm implements the following procedures: random generation of the initial STL population, evaluation of fitness, $\mathcal{F}$, following the Eq. 1, ranking population members based on fitness, discarding the bottom half of the population, and applying genetic alterations such as mutation and crossover. For a positively labeled trace, X_p, and a negatively labeled trace, X_n, in their respective positive and negative datasets, D_p and D_n, the fitness function is,

$$\mathcal{F}(\phi) = N_{\rho(\phi)^{+}|X_p} + N_{\rho(\phi)^{-}|X_n} + \mid \overline{\rho}(\phi)_{D_p} - \overline{\rho}(\phi)_{D_n} \mid \quad (1)$$

where, $\mathcal{F}$ is the fitness function for STL ϕ. The first term in Eq. 1 represents the number of true positive classifications for the positive samples, the second term represents the number of true negative classifications for the negative samples, and the third term computes the absolute value difference between the average of the robustness values, $\overline{\rho}(\phi)$, for samples in D_p and D_n.

The inner loop is comprised of a logically-constrained Q-learning in which the reward is based solely on the robustness of a trajectory throughout an episode against a given STL constraint. The definition of the reward function is shown in Eq. 2.

$$\mathcal{R} = \begin{cases} \rho(\phi_{[0:T]}), & \text{if } \rho(\phi_{[0:T]}) < 0 \\ \rho(\phi_{[0:T]}) + 100, & \text{if } \rho(\phi_{[0:T]}) \geq 0 \end{cases} \quad (2)$$

where, $\mathcal{R}$ is the reward value determined by the robustness degree $\rho(\phi)$ of the sample s over the horizon of the STL, T.

The reward is sparse because it is given at the end of an episode, and not at each step. This is due to the fact that, with timed STL constraints, the robustness value cannot be quantified at every step and can only be computed over a signal at least as long as the horizon, T, of the STL. After training, the algorithm generates a certain number of rollout traces that are presented to the human expert for labeling based on their safety, which is our final component. In our experiments, we have automated the human labeling process by computing the robustness value of the traces against the true safety constraint of the environment. However, it is important to note that this is only done for automation purposes, and as per the basis of our problem, this true safety constraint is unknown, and traces should actually be labeled by an expert. The labeled traces are then used as input to the evolutionary algorithm. This process is repeated iteratively until convergence, which, in this case, is defined by the number of rollout traces that are labeled safe by the human expert. This metric is chosen because the safety of the rollout traces reflects the quality of the STL used as a safety constraint as well as the quality of the policy generated using that constraint. The framework is depicted graphically in Fig. 1.

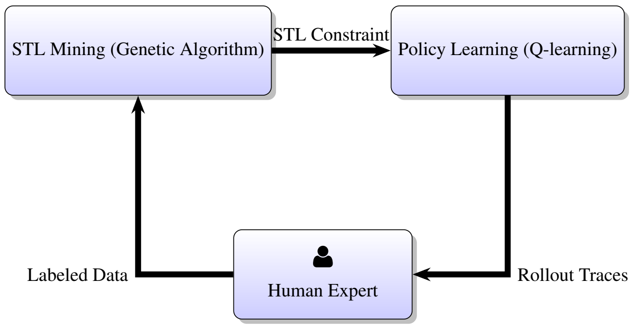

Figure 1: Overview of the proposed framework.

Results

We consider the problem of implementing RL algorithms in an environment where the safety constraint is unknown in advance. Specifically, our goal is to simultaneously infer the correct STL safety constraint and an optimal policy. To evaluate our framework, we have designed grid-world environments of varying sizes for an agent to navigate through to reach a goal at a specific location, under temporal constraints. Initially, neither the goal location nor the time constraint are known, making it impossible to design a traditional reward function. The problem was initiated with 1000 random 2-dimensional coordinate traces within the environment, which were then labeled by a human to create a dataset for the evolutionary algorithm. The algorithm proceeds with the steps outlined in the methodology until the number of safe traces, as labeled by the human expert, meets a certain threshold. The experiment was performed on 6×6, 8×8, and 10×10 grid environments. The outputs were evaluated based on two metrics: (i) the change in the percentage of the number of unsafe traces from the first batch of rollout traces to the last batch and (ii) the average misclassification rate (MCR) of the inferred STL specification against a dataset labeled by the human expert. The first metric evaluates how closely the inferred STL specification is able to capture the true environmental constraints by assessing how the number of unsafe samples in the rollout traces has decreased over the iterations, indicating the STL specification is getting closer to the true (but unknown) constraint. The second metric conveys the classification capability of the inferred STL against labeled datasets. It quantifies how well the STL distinguishes between safe and unsafe trajectories, as compared to that of a human expert. The results are given in Table 1.

Table 1: Percentage of unsafe traces (per rollout sample) at the beginning and the end of the process, and average MCR for inferred STL.

Size	% of Unsafe Traces		Inferred STL
	First rollout	Last rollout	MCR
6×6	73.2%	1.2%	0.02±0.0013
8×8	86.7%	4.3%	0.04±0.001
10×10	91.4%	11.1 %	0.06±0.009

Conclusions

In this paper, we have studied a joint learning framework for the safety constraint and the RL policy of an environment where the safety constraints are unknown *a priori*. We have implemented an algorithm that optimizes the safety constraint and the RL policy simultaneously. Our preliminary results have shown that our framework is capable of identifying safety constraints that are suitable for the environment and an optimal RL policy that results in safe behavior. Future directions for this work will include testing our algorithm in complex environments, assessing adaptability, and improving the algorithm's computational efficiency.

References

Abbas, H.; Winn, A.; Fainekos, G.; and Julius, A. A. 2014. Functional gradient descent method for Metric Temporal Logic specifications. In *2014 American Control Conference*, 2312–2317.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International conference on machine learning*, 22–31. PMLR.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Alur, R.; Bansal, S.; Bastani, O.; and Jothimurugan, K. 2023. Specification-Guided Reinforcement Learning.

Bartocci, E.; Mateis, C.; Nesterini, E.; and Nickovic, D. 2022. Survey on mining signal temporal logic specifications. *Information and Computation*, 289: 104957.

Bombara, G.; and Belta, C. 2021. Offline and Online Learning of Signal Temporal Logic Formulae Using Decision Trees. *ACM Transactions on Cyber-Physical Systems*, 5: 1–23.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31.

Fronda, N.; and Abbas, H. 2022. Differentiable Inference of Temporal Logic Formulas. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41: 1–1.

Garcıa, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1): 1437–1480.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; and Knoll, A. 2022. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*.

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

Kong, Z.; Jones, A.; Ayala, A.; Gol, E.; and Belta, C. 2014. Temporal logic inference for classification and prediction from data. 273–282.

Majumdar, A.; Singh, S.; Mandlekar, A.; and Pavone, M. 2017. Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models. In *Robotics: science and systems*, volume 16, 117.

Mihatsch, O.; and Neuneier, R. 2002. Risk-sensitive reinforcement learning. *Machine learning*, 49: 267–290.

Mohammadinejad, S.; Deshmukh, J. V.; Puranic, A. G.; Vazquez-Chanlatte, M.; and Donzé, A. 2020. Interpretable classification of time-series data using efficient enumerative techniques. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 1–10.

Nenzi, L.; Silvetti, S.; Bartocci, E.; and Bortolussi, L. 2018. A Robust Genetic Algorithm for Learning Temporal Specifications from Data.

Razzaghi, P.; Tabrizian, A.; Guo, W.; Chen, S.; Taye, A.; Thompson, E.; Bregeon, A.; Baheri, A.; and Wei, P. 2022. A Survey on Reinforcement Learning in Aviation Applications. *arXiv preprint arXiv:2211.02147*.

Sinha, A.; Malo, P.; and Deb, K. 2017. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2): 276–295.

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

Tamar, A.; Glassner, Y.; and Mannor, S. 2014. Policy gradients beyond expectations: Conditional value-at-risk. *arXiv preprint arXiv:1404.3862*.

Appendix

Joint Convergence of the Inner Loop and Outer Loop

The goal is to establish that the joint convergence of the inner and outer loops leads to the overall convergence of the framework. In other words, we want to show that if the inner loop (Q-learning) converges to an optimal policy and the outer loop (evolutionary algorithm for STL synthesis) converges to an optimal STL constraint, then the entire framework converges to a stable solution. By proving these two implications, we demonstrate that the proposed framework is effective in solving the given problem and that it converges to a solution that meets both the optimality of the policy and the optimality of the STL constraint. To prove the joint convergence, we need to establish two implications:

- Convergence of the inner loop (Q-learning) to an optimal policy π^* implies the convergence of the outer loop (evolutionary algorithm) to an optimal STL constraint ϕ^*.
- Convergence of the outer loop (evolutionary algorithm) to an optimal STL constraint ϕ^* implies the convergence of the inner loop (Q-learning) to an optimal policy π^*.

Implication 1. This implication aims to show that if the inner loop converges to an optimal policy, the outer loop converges to an optimal STL constraint. This is achieved by demonstrating that the fitness function is maximized for the optimal STL constraint and that the evolutionary algorithm converges to this optimal constraint under certain conditions.

Assume that the inner loop converges to an optimal policy π^*. Under this assumption, we need to prove that the outer loop converges to an optimal STL constraint ϕ^*. We approach this by showing that:

- The fitness function $\mathcal{F}(\pi^*, \phi)$ is maximized for the optimal STL constraint π^*. This can be done by analyzing the properties of the fitness function and the reward function $\mathcal{R}(\pi, \phi)$ under the optimal policy.
- The evolutionary algorithm for STL synthesis converges to the optimal STL constraint π^* under certain conditions, such as proper selection pressure, sufficient exploration, and well-defined mutation and crossover operators.

By showing that the fitness function is maximized for the optimal STL constraint, and that the evolutionary algorithm converges to this optimal constraint, we establish the convergence of the outer loop under the assumption that the inner loop converges to the optimal policy.

Implication 2. This implication aims to show that if the outer loop converges to an optimal STL constraint, the inner loop converges to an optimal policy. This is achieved by demonstrating that the reward function provides the necessary guidance under the optimal STL constraint, and that the Q-learning algorithm converges to the optimal policy under this guidance.

Assume that the outer loop converges to an optimal STL constraint π^*. Under this assumption, we need to prove that the inner loop converges to an optimal policy π^*. We approach this by showing that:

- The reward function $\mathcal{R}(\pi, \phi^*)$ has the necessary properties to guide the Q-learning algorithm towards the optimal policy. This can be done by analyzing the reward function under the optimal STL constraint and ensuring that it provides proper guidance and exploration-exploitation trade-off.
- The Q-learning algorithm converges to the optimal policy π^* under certain conditions, such as proper learning rates, sufficient exploration, and well-defined state and action spaces.

By showing that the reward function provides the necessary guidance under the optimal STL constraint, and that the Q-learning algorithm converges to the optimal policy under this guidance, we establish the convergence of the inner loop under the assumption that the outer loop converges to the optimal STL constraint.

Proof of Implication 1.

(i) Maximizing the fitness function for the optimal STL constraint:

To provide a more detailed and mathematical presentation of Implication 1, let's first define the key components of the framework: (i) policy: $\pi : S \to A$, a mapping from states to actions, (ii) reward function: $\mathcal{R}(\pi, \varphi) : \Pi \times \Phi \to \mathbb{R}$, a function that measures the reward for a given policy π and STL constraint φ, (iii) fitness function: $\mathcal{F}(\pi, \varphi) : \Pi \times \Phi \to \mathbb{R}$, a function that measures the fitness of a given policy π and STL constraint φ. Now, let's proceed with the proof of Implication 1:

Assume that the inner loop converges to an optimal policy π^*. We want to show that the fitness function $\mathcal{F}(\pi^*, \varphi)$ is maximized for the optimal STL constraint φ^*. Let $\mathcal{R}^* = \max_{\pi,\varphi} \mathcal{R}(\pi, \varphi)$ be the maximum achievable reward. We know that the reward function $\mathcal{R}(\pi, \varphi)$ is maximized for the optimal policy π^* and the optimal STL constraint φ^*, i.e., $\mathcal{R}(\pi^*, \varphi^*) = \mathcal{R}^*$. We define the fitness function $\mathcal{F}(\pi, \varphi)$ as a function of the reward function $\mathcal{R}(\pi, \varphi)$:

$$\mathcal{F}(\pi, \varphi) = \frac{R(\pi, \varphi)}{R^*}$$

since $\mathcal{R}(\pi^*, \varphi^*) = \mathcal{R}^*$, we have:

$$\mathcal{F}(\pi^*, \varphi^*) = \frac{\mathcal{R}(\pi^*, \varphi^*)}{\mathcal{R}^*} = \frac{\mathcal{R}^*}{\mathcal{R}^*} = 1$$

This result shows that the fitness function $\mathcal{F}(\pi^*, \varphi)$ is indeed maximized for the optimal STL constraint φ^*, given that the inner loop converges to the optimal policy π^*. The maximum value of the fitness function is 1, which occurs when both the policy and the STL constraint are optimal.

(ii) Convergence of the evolutionary algorithm for STL synthesis:

Let φ_i be the STL constraint at iteration i of the evolutionary algorithm. We want to show that the evolutionary algorithm converges to the optimal STL constraint φ^* under certain conditions. Let $P(\varphi_i)$ be the probability distribution of the STL constraint population at iteration i. The evolutionary algorithm updates $P(\varphi_i)$ through selection, mutation, and crossover operators. Let $P_{\text{sel}}(\varphi_i)$, $P_{\text{mut}}(\varphi_i)$, and $P_{\text{cross}}(\varphi_i)$ be the updated probability distributions after applying the selection, mutation, and crossover operators, respectively. Then, the probability distribution at iteration $i + 1$ is given by:

$$P(\varphi_{i+1}) = P_{\text{cross}}\left(P_{\text{mut}}\left(P_{\text{sel}}(\varphi_i)\right)\right)$$

Under proper selection pressure, sufficient exploration, and well-defined mutation and crossover operators, it can be shown that the evolutionary algorithm converges to the optimal STL constraint φ^* as the number of iterations approaches infinity:

$$\lim_{i\to\infty} P(\varphi_i) = \delta(\varphi - \varphi^*)$$

where $\delta(\cdot)$ is the Dirac delta function, which means that the probability distribution converges to a distribution concentrated on the optimal STL constraint φ^*. By proving both (a) and (b), we establish that the outer loop converges to the optimal STL constraint φ^* under the assumption that the inner loop converges to the optimal policy π^*. This shows that the joint convergence of the inner and outer loops leads to the overall convergence of the framework.

Proof of Implication 2.

Assume that the outer loop converges to an optimal STL constraint φ^*. Under this assumption, we need to prove that the inner loop converges to an optimal policy π^*. We approach this by showing that:

- The reward function $\mathcal{R}(\pi, \varphi^*)$ has the necessary properties to guide the Q-learning algorithm towards the optimal policy. This can be done by analyzing the reward function under the optimal STL constraint and ensuring that it provides proper guidance and exploration-exploitation trade-off. Specifically, we show that $\mathcal{R}(\pi, \varphi^*)$ is Lipschitz continuous and has a unique maximum at the optimal policy π^*. Moreover, the reward function should encourage sufficient exploration of the state-action space while exploiting the knowledge acquired during the learning process.
- The Q-learning algorithm converges to the optimal policy π^* under certain conditions, such as proper learning rates, sufficient exploration, and well-defined state and action spaces. According to the Q-learning convergence theorem, the Q-learning algorithm converges to the optimal Q-function $Q^*(s, a)$ if the following conditions are satisfied:

1. Each state-action pair (s, a) is visited infinitely often, i.e., $\lim_{t\to\infty} N_t(s, a) = \infty$, where $N_t(s, a)$ is the number of visits to the state-action pair (s, a) up to time t.
2. The learning rate $\alpha_t(s, a)$ satisfies $\sum_{t=1}^{\infty} \alpha_t(s, a) = \infty$ and $\sum_{t=1}^{\infty} \alpha_t^2(s, a) < \infty$. This condition ensures that the learning rate decays slowly enough to guarantee convergence.

Under the optimal STL constraint φ^*, we assume that the state and action spaces are well-defined, and the exploration strategy (e.g., using an ϵ-greedy approach) guarantees that each state-action pair is visited infinitely often. If these conditions are met, the Q-learning algorithm converges to the optimal policy π^*. By showing that the reward function provides the necessary guidance under the optimal STL constraint, and that the Q-learning algorithm converges to the optimal policy under this guidance, we establish the convergence of the inner loop under the assumption that the outer loop converges to the optimal STL constraint.

Bounds on the Error

Deriving bounds on the error between the discovered policy and the true optimal policy involves analyzing the mathematical relationship between the error and various factors influencing it. Here's a possible way to approach this analysis: Let π^* be the true optimal policy and π' be the discovered policy. Define the error between these policies as:

$$\epsilon(\pi', \pi^*) = E\left[R(s, \pi^*(s)) - R(s, \pi'(s))\right]$$

where $E[\cdot]$ denotes the expected value, $\mathcal{R}(s, a)$ is the reward function for taking action a in state s, and the expectation is taken over all states s in the state space. Now, consider the following factors that may affect the error bounds:

- Granularity of the state abstraction (denoted by G): A coarse state abstraction may lead to a larger error between the discovered policy and the true optimal policy. The impact of state abstraction granularity on the error can be represented as:
$$\epsilon_G(G) \leq C_1 \cdot G$$
where C_1 is a constant that depends on the problem's specific characteristics.
- Quality of the learned STL specifications (denoted by Q): If the learned STL specifications are not accurate or expressive enough, the error between the discovered policy and the true optimal policy may be larger. The impact of the quality of the learned STL specifications on the error can be represented as:
$$\epsilon_Q(Q) \leq C_2 \cdot (1 - Q)$$
where C_2 is a constant that depends on the problem's specific characteristics.
- Amount of human feedback provided (denoted by H): Human feedback can help guide the learning process and reduce the error between the discovered policy and the true optimal policy. The impact of the amount of human feedback on the error can be represented as:
$$\epsilon_H(H) \leq C_3 \cdot e^{-H}$$
where C_3 is a constant that depends on the problem's specific characteristics, and e^{-H} represents the exponential decay in error with increasing human feedback.

By combining these individual error bounds, we obtain an overall error bound:

$$\epsilon(\pi', \pi^*) \leq C_1 \cdot G + C_2 \cdot (1 - Q) + C_3 \cdot e^{-H}.$$

This bound shows the relationship between the error and the factors affecting it, such as the granularity of the state abstraction, the quality of the learned STL specifications, and the amount of human feedback provided. By analyzing this error bound, we can identify the trade-offs between these factors and develop strategies to minimize the error and improve the effectiveness of the bi-level optimization framework.