Using a Search-Based Safety Oracle for RL Policy Fine-Tuning

Anonymous submission

Abstract

Policies represented as neural networks (NNs), in particular ones learned by reinforcement learning (RL), are drawing increasing attention for planning. Yet they do not offer any formal guarantees, a major issue in particular when safety is a concern. Safe RL research has come up with an arsenal of techniques to improve policy safety. Here we introduce a new approach to this problem, training an initial policy using RL and subsequently doing limited safety fine-tuning. The latter is enabled by plugging in a recently introduced search-based procedure deciding state safety. Our experimental results show that, compared to standard RL and safe RL approaches, our method effectively improves worst-case policy safety while maintaining average-reward performance.

Introduction

Automated planning is concerned with the task of finding a sequence of actions that achieves a goal. It is a widely used decision-making framework, which can model a variety of complex real-world scenarios like production line logistics (Helmert and Lasinger 2010), robotics (e.g., integrated task and motion planning (Garrett et al. 2021)), and Mars exploration (Ai-Chang et al. 2004). Here we address planning under initial state uncertainty and non-deterministic effects, a notoriously challenging problem as plans need to specify a reaction to every state that may be encountered.

Neural network (NN) *policies* are drawning increasing attention in planning (e.g. Toyer et al.; Wang and Thiébaux; Ståhlberg, Bonet, and Geffner (2020; 2024; 2025)). A policy maps states to actions, leveraging the generalization capabilities of NNs to provide action choices for (in principle) arbitrary states. A wealth of reinforcement learning (RL) methods are available to train such policies (Mnih et al. 2013; Schulman et al. 2017; Haarnoja et al. 2018).

However, NN policies and RL are of course not silver bullets. Apart from wide-spread practical issues in getting them to work well, they do not come with any inherent formal guarantees. This is a major issue in particular when safety is a concern. Safe RL research has come up with an arsenal of techniques to improve policy safety (e.g. (García and Fernández 2015; Ray, Achiam, and Amodei 2019; Thananjeyan et al. 2021)).

Here we introduce a new approach to this problem, leveraging recent work on search-based safety analysis (Jain et al. 2025) to provide an oracle for learning. We train an initial policy using standard RL methods. Subsequently, we do limited fine-tuning using a well-known framework called DAgger (Ross, Gordon, and Bagnell 2011). DAgger samples policy runs, and queries a reference strategy to *relabel* the actions taken by the policy. The relabeled actions should be near-optimal, and have been provided in prior work by human experts or domain-dependent planning mechanisms. Here, we instead use search-based safety analysis as the reference strategy.

Jain et al. (2025)'s search procedure decides whether a given state s is safe, using a depth-first search inspired by Tarjan's algorithm to find strongly connected components. The search is (of course) worst-case exponential, but being focused on a single state s it can be more feasible than standard methods characterizing the entire safe region (Baier and Katoen 2008). If the search determines a state s to be safe, it in particular determines a safe action a for s. We use that action for relabelling in DAgger.

We run experiments on the benchmark set previously used to evaluate the safety oracle (Jain et al. 2025). We measure policy safety from a worst-case perspective, through the fraction of sampled initial states that are worst-case unsafe, i.e., from which an unsafe policy run exists. We refer to this criterion as *initial-state unsafety*. Our results show that, compared to standard RL and safe RL approaches, our method effectively improves initial-state safety while maintaining average-reward performance.

Background

We consider a formalism of automated planning tasks which is a generalization of *classical planning* (Ghallab, Nau, and Traverso 2004) and of *numerical planning* (Gerevini, Saetti, and Serina 2008).

We define a planning task Π as a tuple $\langle \mathcal{V}, \mathcal{A}, \mathcal{C}_I, \mathcal{C}_g, \mathcal{C}_f \rangle$ where $\mathcal{V}$ is a finite set of *variables* and $\mathcal{A}$ a finite set of *actions*. $\mathcal{C}_I$, $\mathcal{C}_g$, and $\mathcal{C}_f$ are *constraints* over $\mathcal{V}$ that specify the *possible* initial states, the *goal* condition, and the *failure* condition, respectively. Each variable $v \in \mathcal{V}$ could take values from a potentially *infinite* set called its domain, denoted as $\mathcal{D}(v)$. A state s is an assignment of values to all variables in $\mathcal{V}$, i.e., $s : \mathcal{V} \to \bigcup_{v \in \mathcal{V}} \mathcal{D}(v)$ is a function such that $s(v) \in \mathcal{D}(v)$ for every $v \in \mathcal{V}$. We restrict the domain of each variable to be a subset of the real numbers $\mathbb{R}$.

The core of the framework is the notion of *expressions* φ defined inductively over variables as follows:

- A term t with $t \in \mathcal{V} \cup \bigcup_{v \in \mathcal{V}} \mathcal{D}(v)$ is an expression.
- If φ is an expression, then $\neg\varphi$ is an expression, where $\neg$ is the logical negation operator.
- If φ_1 and φ_2 are expressions, then $\varphi_1 \circ \varphi_2$ is also an expression, where $\circ \in \{+, -, \times, /, =, <, \leq, \wedge, \vee\}$ is a binary arithmetic or logical operator.

An expression can be evaluated to either a boolean value or a real number given a state s. We use $\mathcal{E}[\![\varphi|s]\!]$ to denote the evaluation of an expression φ under a state s. The evaluation rule $\mathcal{E}[\![\cdot|s]\!]$ in a state s is defined inductively as follows:

- $\mathcal{E}[\![t|s]\!] \mapsto s(t)$ if $t \in \mathcal{V}$. $\mathcal{E}[\![t|s]\!] \mapsto t$ if $t \in \bigcup_{v \in \mathcal{V}} \mathcal{D}(v)$.
- $\mathcal{E}[\![\neg\varphi|s]\!] \mapsto \neg\mathcal{E}[\![\varphi|s]\!]$ if φ is an expression and $\mathcal{E}[\![\varphi|s]\!]$ a boolean value.
- $\mathcal{E}[\![\varphi_1 \circ \varphi_2|s]\!] \mapsto \mathcal{E}[\![\varphi_1|s]\!] \circ \mathcal{E}[\![\varphi_2|s]\!]$ if φ_1 and φ_2 are expressions and $\mathcal{E}[\![\varphi_1|s]\!]$ and $\mathcal{E}[\![\varphi_2|s]\!]$ are both real numbers or both boolean values.

An action $a \in \mathcal{A}$ has a precondition written as $\texttt{prec}[\![a]\!] = \phi$ which is an expression evaluated to a boolean value. We say that the action a is *applicable* in a state s if $\mathcal{E}[\![\phi|s]\!] \mapsto \texttt{true}$, i.e., the state satisfies the precondition of a. Applying a in a state s will modify the values of some variables in s by the *effects* of a denoted as $\texttt{eff}[\![a]\!]$ and henceforth result in a new state s'. In this paper, we consider *non-deterministic* effects, that is, when applying an action, *one* of several effects might happen. An effect is a *set* of tuples $\langle v, \varphi\rangle$ where $v \in \mathcal{V}$ and φ is an expression. Each tuple $\langle v, \varphi\rangle$ specifies the value of the variable v in the new state s' after applying the action a in a state s, that is, $s'(v) = \mathcal{E}[\![\varphi|s]\!]$.

The last three components $\mathcal{C}_I$, $\mathcal{C}_g$, and $\mathcal{C}_f$ in our formalism are all expressions that evaluate to a boolean value. An *initial* state s_I is *any* state such that $\mathcal{E}[\![\mathcal{C}_I|s_I]\!] \mapsto \texttt{true}$. Similarly, a state s_g is a *goal* state if $\mathcal{E}[\![\mathcal{C}_g|s_g]\!] \mapsto \texttt{true}$, and a state s_f is a *failure* state if $\mathcal{E}[\![\mathcal{C}_f|s_f]\!] \mapsto \texttt{true}$. A *policy* π is a function mapping a state s to an action a, i.e., it decides which action to take in each state. We want to obtain a policy π such that for an *arbitrary* initial state s_I, we can reach a goal state s_g while avoiding reaching *any* failure state s_f by following π.

Connection to MDPs One way to obtain such a policy is via Reinforcement Learning (RL). In particular, the planning framework we consider has a natural connection to Markov Decision Processes (MDPs) on top of which RL algorithms are deployed. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{P}_I\rangle$. $\mathcal{S}$ is a (potentially infinite) set of states and $\mathcal{A}$ is a (potentially infinite) set of actions. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathbb{D}(\mathcal{S})$ with $\mathbb{D}(\mathcal{S})$ being the set of *probability distributions* over $\mathcal{S}$ is the state transition function. Concretely, $\mathcal{T}(s, a)$ returns a probability distribution over the next states when taking action a in state s. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function which defines the reward for taking an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$. $\mathcal{P}_I$ is a probability distribution over $\mathcal{S}$ which defines the probability of a state being the initial state.

Our planning framework can be naturally converted to an MDP. The set of all assignments over $\mathcal{V}$ (i.e., the set of all functions $s : \mathcal{V} \to \bigcup_{v \in \mathcal{V}} \mathcal{D}(v)$) forms the state space $\mathcal{S}$ of the MDP. The MDP and the planning task share the same set of actions. The actions' preconditions and effects induce the transition function $\mathcal{T}$ of the MDP. The initial state distribution $\mathcal{P}_I$ is thus a uniform distribution over all states satisfying $\mathcal{C}_I$. A reward (e.g., $+1$) is given for reaching a goal state and 0 for all other states.

Safe RL Our planning framework is also concerned with failure conditions. This additional property can also be captured by an extended version of MDPs called *Constrained MDPs* (CMDPs) (Altman 1999). A CMDP features an additional component $\mathcal{J}$, which is a set of constraints (also called cost functions) c. The constraint set $\mathcal{J}$ defines the feasible set of policies. Concretely, any feasible policy π must hold that $J_c(\pi) \leq d_c$ for every $c \in \mathcal{J}$, where $J_c(\pi)$ is the expected cost of π under the constraint c and d_c is a threshold for c. Note that the concrete formalization of a cost function c can vary from different scenarios (and so does J_c). In our case, we can define a cost function $c : \mathcal{S} \to \{0, 1\}$ such that $c(s) = 1$ if s is a failure state and $c(s) = 0$ otherwise. Consequently, the expected cost $J_c(\pi)$ of a policy π captures the probability of reaching a failure state by following π.

Solving a CMDP is to maximize the expected return of a policy $J_r(\pi)$ subject to the constraint that $J_c(\pi) \leq d_c$ for all $c \in \mathcal{J}$. One standard way to achieve this is via *Lagrangian multipliers*, i.e., solving the following problem:

$$\max_{\pi} \min_{\substack{c \in \mathcal{J} \\ \lambda_c \geq 0}} J_r(\pi) - \sum_{c \in \mathcal{J}} \lambda_c (J_c(\pi) - d_c) \tag{1}$$

which results in algorithms like PPO-Lag (Ray, Achiam, and Amodei 2019).

In our context, the cost function and the reward are sparse and mutually exclusive. Hence, solving the above lagrangian problem can simply be done by *reward shaping* (i.e., by incorporating the cost into the reward function) (Thananjeyan et al. 2021). Concretely, consider a state transition (s, a, s'), we define the shaped reward to be $\mathcal{R}(s, a) - c(s')$. In other words, we give a penalty of -1 for reaching a failure state. By doing so, we construct a new MDP without the constraint set and can apply any RL algorithm to obtain a safe policy.

Improving Worst-Case Safety

Basically all RL and safe RL algorithms optimizing policies by *sampling* state-action trajectories. As a result, they struggle to recognize actions that have a small chance of leading to failure states and may even learn to take such actions during training, violating worst-case safety.

State and Action Safety We say a state s is *safe* if there exists a policy π such that starting from s and following π, it is guaranteed that we will *not* reach a failure state. Similarly, we say that an action a is *safe* with respect to a state s if all successor states obtained by taking a in s are safe (due to non-deterministic effects, there may be more than one successor state). Note that an *unsafe* action in a state could still result in a safe successor state but, in the worst case, can lead to a failure state. Hence, it is a problematic action that the policy should avoid in that state.

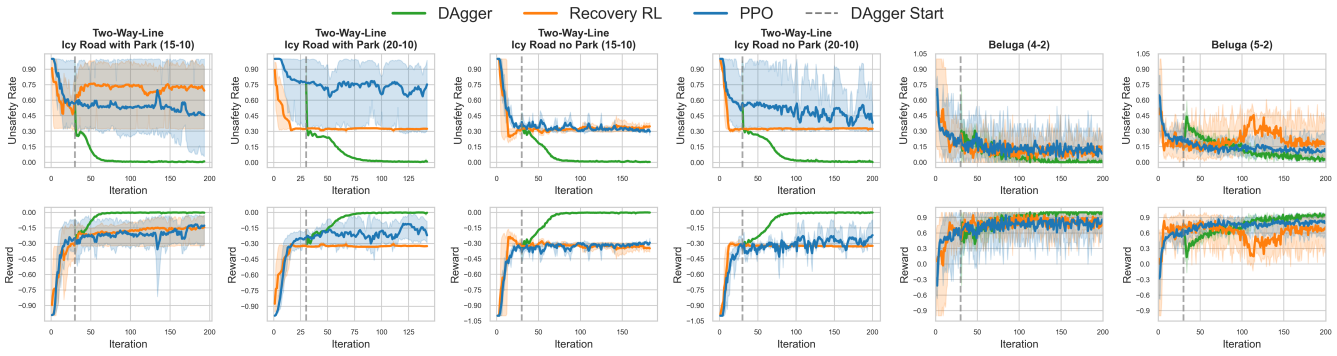

Figure 1: The initial state unsafety and the average reward achieved by refined DAgger (ours), Recover RL, and PPO during training. The top row is the unsafety rate (the lower the better) and the bottom row shows the average reward. DAgger with the safety oracle achieves lowest initial state unsafety and has less variance in performance.

Algorithm 1: DAgger with a safety oracle

▷ Let π_0 be the initial policy ◁
$\mathbb{D}^+ \leftarrow \emptyset$; $\mathbb{D}^- \leftarrow \emptyset$ ▷ Initialize the buffers
for $i = 1$ to N **do**
  ▷ Sample M trajectories by running π_{i-1} ◁
  $\{\tau_i\}_{i=1}^{M} \leftarrow$ execute π_{i-1}
  for all (s, a) in $\{\tau_i\}_{i=1}^{M}$ **do**
    `is_safe`, $a' \leftarrow \textsc{Oracle}(s, a)$ ▷ Run the oracle
    if `is_safe` **then**
      $\mathbb{D}^+ \leftarrow \mathbb{D}^+ \cup \{(s, a)\}$
    else if $a' \neq$ `null` **then**
      $\mathbb{D}^- \leftarrow \mathbb{D}^- \cup \{(s, a')\}$
  for B batches **do**
    ▷ Let b be the batch size ◁
    $\mathbb{B} \leftarrow$ sample $\frac{b}{2}$ from $\mathbb{D}^+$ and $\frac{b}{2}$ from $\mathbb{D}^-$
    ▷ Supervised learning on the batch $\mathbb{B}$ ◁
    $\pi_{i-1} \leftarrow \arg\max_\pi \sum_{(s,a)\in\mathbb{B}} \log \pi(a|s)$
  $\pi_i \leftarrow \pi_{i-1}$

Fine-tuning Policies Given a policy π (which could be obtained via any RL or safe RL method), we now demonstrate how to improve its worst-case safety by fine-tuning it online with a simple yet effective policy fine-tuning method called DAgger (Ross, Gordon, and Bagnell 2011). The key idea of DAgger is straightforward: It iteratively updates the policy by executing the current policy to collect trajectories, querying a *human expert* to relabel the actions in the visited states, and aggregating the resulting state-action pairs into a buffer. This buffer then serves as training data for supervised updates of the policy.

Instead of involving a human expert (which is often costly or even infeasible), we employ the *safety oracle* developed by Jain et al. (2025). The oracle takes as input a state s and an action a and returns whether a is safe with respect to s. If a is unsafe, the oracle also returns a safe action a' if such an action exists (i.e., if s is a safe state). The refined DAgger framework with the safety oracle is shown in Alg. 1.

In contrast to the original version of DAgger, we maintain two buffers $\mathbb{D}^+$ and $\mathbb{D}^-$. $\mathbb{D}^+$ stores the state-action pairs where the action (given by the policy) is safe. $\mathbb{D}^-$ stores the pairs where the oracle corrects an unsafe action taken by the policy with a safe one. The purpose of having this additional buffer $\mathbb{D}^+$ is to restrict the update of the policy from deviating too far away from the original one, i.e., from being dominated by the oracle-generated data. This is similar to what has been done by Ball et al. (2023). The hyperparameters in the refined framework include the number of iterations N, the number of sampled trajectories M in each iteration and the number and sizes of batches B and b for supervised learning, respectively.

Empirical Evaluation

We based our empirical evaluation on the benchmarks used by Jain et al. (2024; 2025). The benchmark suite consists of Beluga, which is a non-deterministic domain abstracted from a logistic scenario within Airbus, and different variants of Transport (One Way Line and Two Way Lines) where non-determinism is introduced due to icy roads. The experiments illustrate the effectiveness of combining DAgger with the safety oracle in improving the worst-case safety and, in many cases, the average cumulative reward of a given policy.

Configurations We measured the worst-case safety of a policy in terms of *initial state unsafety*. Concretely, We sampled a set of 5000 initial states from the initial state distribution of each benchmark and keep it fixed across experiments. We evaluate a given policy by leveraging the safety oracle to check, for each of these initial states s, whether the policy ever enters an unsafe state when starting from s (equivalently, whether any run of the policy on s is unsafe). Initial state unsafety is then measured as the percentage of the initial states on which the policy was unsafe.

We measure policy performance by the average cumulative reward the policy achieves on the 5000 initial states.

For initial training of policies, we used a standard RL algorithm on each benchmark, with reward $+1$ for reaching the goal, -1 for reaching a failure state, and 0 for all other cases. We tried both PPO (Schulman et al. 2017) and SAC (Haarnoja et al. 2018), and found that PPO consistently per-

forms better. We ran PPO for 30 iterations and then started the fine-tuning process. As a comparison, we kept running PPO with the same number of iterations.

As another baseline, we also ran an advanced Safe RL approach called Recovery RL (Thananjeyan et al. 2021). The fundamental idea of Recovery RL is to first learn a recovery policy and a risk value function (which estimates the risk of taking an action in a state) *offline*. During the online training, a task policy is trained as usual while the recovery policy and the risk value function are fine-tuned in parallel. The final policy is a combination of both the task policy and the recovery policy: if the risk value of an action selected by the task policy is above a certain threshold, the recovery policy is executed instead. In our experiments, we used Implict Q-Learning (IQL) (Kostrikov, Nair, and Levine 2022) to learn the recovery policy and the risk function offline, and during online training, we stuck with PPO to keep consistency. We tuned hyperparameters using Optuna (Akiba et al. 2019) and selected the best configurations.

Experimental Results The experimental evaluation were run on 17 benchmarks from 6 domains. Due to the space limit, we only report representative results for improving worst-case safety on 6 benchmarks drawn from 3 domains. The results for the other benchmarks are similar. We ran each approach (DAgger, Recovery RL, and PPO) with 3 different seeds and plotted the average results. The results are shown in Fig. 1. The shadow area around each line represents the 95% confidence interval.

Overall, DAgger with the oracle (the green line) achieved the lowest initial state unsafety, and in many cases, also improved the average reward. In particular, for the two variants of Two Way Line (the first four columns in Fig. 1), initial state unsafety for PPO and Recovery RL were still high when the average rewards achieved by the policies had converged. In contrast, with the help of the safety oracle, DAgger significantly improves worst-case safety. For Beluga, the performance gap between DAgger and the other two approaches is not significant, but DAgger still achieves better worst-case safety (and higher reward). Fig. 1 also shows that DAgger is significantly more stable than the other two approaches, as reflected by the thin confidence intervals.

DAgger with the oracle can also be used to learn a policy from *scratch*. We demonstrated this on benchmarks where for each inital state following the initial state distribution, all actions are *unsafe* (in such cases, measuring initial state safety is not meaningful because it is always 0). Note that despite all actions being unsafe for all initial states, a state where a safe action exists can still be visited during policy execution, and hence, the safety oracle is still applicable.

We started DAgger with a *randomly initialized* policy and compared against standard PPO. We again used a fixed set of 5000 initial states and measured the average reward achieved by the two approaches over these initial states throughout training. Each approach were still run with 3 seeds and the average results were plotted in Fig. 2. Both approaches could achieve a nearly perfect reward (which is the reason why we did not put Recovery RL into this comparison), but DAgger achieved it faster and with less variance in performance.

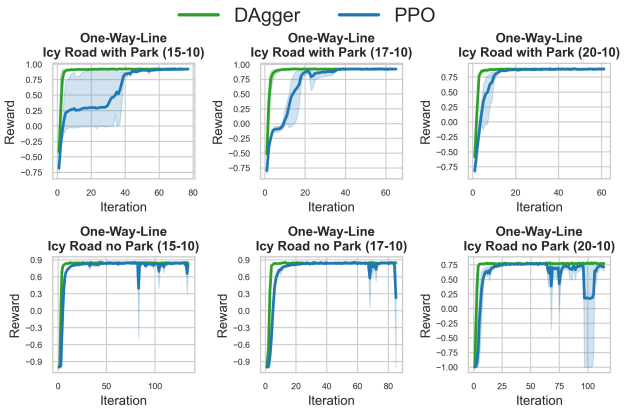

Figure 2: DAgger improves convergence compared to PPO for unsafe initial states.

Related Work Safe RL is centered around solving Eq. 1 (background section). This dual optimization problem leads to variations of standard RL algorithms like PPO-Lag, TRPO-Lag (Ray, Achiam, and Amodei 2019), and SAC-Lag (Ha et al. 2020). There are also approaches (e.g., the work by Thomas, Luo, and Ma (2021)) that avoid using Lagrangian multipliers. However, these approaches tend to struggle with sparse reward environments, which we address here.

Several works have extended DAgger (Ross, Gordon, and Bagnell 2011) to improve safety during policy learning. SafeDAgger (Zhang and Cho 2017) queries the reference strategy whenever a safety classifier predicts significant deviation between the learned and reference policies. In simulated autonomous driving, this improved safety and efficiency, as only states deemed unsafe by the safety classifier require labelling by the expert. LazyDAgger (Hoque et al. 2021) extends SafeDAgger with a re-entry threshold that reduces context switches between the expert and learned policy. EnsembleDAgger (Menda, Driggs-Campbell, and Kochenderfer 2019) follows the learned policy only when its action predictions have a low uncertainty and otherwise falls back to the expert, improving safety and learning efficiency on continuous control benchmarks. HG-DAgger (Kelly et al. 2019) introduces a human-in-the-loop approach where a human expert has uninterrupted control until they decide to give control back to the learned policy.

Conclusion

We contribute a new approach to safe RL, leveraging a search-based safety oracle for fine-tuning with DAgger. Our experiments provide evidence that this method is indeed competitive.

One topic for future work is whether the safety oracle can be replaced with a *safety classifier*, trained using data provided by the safety oracle, to reduce the computational overhead caused by the oracle. More generally, our work indicates that search-derived knowledge can be useful for safe RL, opening the question what are the best methods for doing so, under which circumstances.

References

Ai-Chang, M.; Yglesias, J.; Chafin, B.; Dias, W.; Maldague, P.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.-J.; Jonsson, A.; Kanefsky, B.; Morris, P.; and Rajan, K. 2004. MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission. *IEEE Intelligent Systems*, 19: 8–12.

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, 2623–2631. ACM.

Altman, E. 1999. *Constrained Markov Decision Processes*. CRC Press.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT.

Ball, P. J.; Smith, L.; Kostrikov, I.; and Levine, S. 2023. Efficient Online Reinforcement Learning with Offline Data. In *Proceedings of the 40th International Conference on Machine Learning, ICML 2023*, 1577–1594. PMLR.

García, J.; and Fernández, F. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16: 1437–1480.

Garrett, C. R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4: 265–293.

Gerevini, A.; Saetti, A.; and Serina, I. 2008. An Approach to Efficient Planning with Numerical Fluents and Multi-Criteria Plan Quality. *Artificial Intelligence*, 172: 899–944.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning – Theory and Practice*. Morgan Kaufmann.

Ha, S.; Xu, P.; Tan, Z.; Levine, S.; and Tan, J. 2020. Learning to Walk in the Real World with Minimal Human Effort. In *Proceedings of the 4th Conference on Robot Learning, CoRL 2020*, 1110–1120. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018,*, 1856–1865. PMLR.

Helmert, M.; and Lasinger, H. 2010. The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010*, 234–237. AAAI.

Hoque, R.; Balakrishna, A.; Putterman, C.; Luo, M.; Brown, D. S.; Seita, D.; Thananjeyan, B.; Novoseller, E. R.; and Goldberg, K. 2021. LazyDAgger: Reducing Context Switching in Interactive Imitation Learning. In *Proceedings of the 17th IEEE International Conference on Automation Science and Engineering, CASE 2021*, 502–509. IEEE.

Jain, C.; Cascioli, L.; Devos, L.; Vinzent, M.; Steinmetz, M.; Davis, J.; and Hoffmann, J. 2024. Safety Verification of Tree-Ensemble Policies via Predicate Abstraction. In *Proceedings of the 27th European Conference on Artificial Intelligence, ECAI 2024*, 1189–1197. IOS Press.

Jain, C.; Sherbakov, D.; Vinzent, M.; Steinmetz, M.; Davis, J.; and Hoffmann, J. 2025. Policy Safety Testing in Non-Deterministic Planning: Fuzzing, Test Oracles, Fault Analysis. In *Proceedings of the 28th European Conference on Artificial Intelligence, ECAI 2025*, 4798–4806. IOS.

Kelly, M.; Sidrane, C.; Driggs-Campbell, K. R.; and Kochenderfer, M. J. 2019. HG-DAgger: Interactive Imitation Learning with Human Experts. In *Proceedings of the 36th International Conference on Robotics and Automation, ICRA 2019*, 8077–8083. IEEE.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *The 10th International Conference on Learning Representations, ICLR 2022*. OpenReview.

Menda, K.; Driggs-Campbell, K. R.; and Kochenderfer, M. J. 2019. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. In *Proceedings of the 32nd IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019*, 5041–5048. IEEE.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. OpenAI.

Ross, S.; Gordon, G. J.; and Bagnell, D. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, 627–635. JMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2025. Learning More Expressive General Policies for Classical Planning Domains. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence, AAAI 2025*, 26697–26706. AAAI.

Thananjeyan, B.; Balakrishna, A.; Nair, S.; Luo, M.; Srinivasan, K.; Hwang, M.; Gonzalez, J. E.; Ibarz, J.; Finn, C.; and Goldberg, K. 2021. Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones. *IEEE Robotics and Automation Letters*, 6(3): 4915–4922.

Thomas, G.; Luo, Y.; and Ma, T. 2021. Safe Reinforcement Learning by Imagining the Near Future. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*. Curran Associates.

Toyer, S.; Thiébaux, S.; Trevizan, F. W.; and Xie, L. 2020. ASNets: Deep Learning for Generalised Planning. *Journal of Artificial Intelligence Research*, 68: 1–68.

Wang, R. X.; and Thiébaux, S. 2024. Learning Generalised Policies for Numeric Planning. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling, ICAPS 2024*, 633–642. AAAI.

Zhang, J.; and Cho, K. 2017. Query-Efficient Imitation Learning for End-to-End Simulated Driving. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence, AAAI 201*, 2891–2897. AAAI.