Understanding Reward Shaping in Planning: A Theoretical and Empirical Analysis

Oleksii Shuhailo[1*], Mohsen Ghaffari[1*], Karel Chvalovský[2], Tomáš Pevný[1]

[1]Artificial Intelligence Center, Czech Technical University. Czech Republic
[2]Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague
shuhaole@fel.cvut.cz, mohsen.ghaffari.10@gmail.com, karel@chvalovsky.cz, pevnytom@fel.cvut.cz

Abstract

Reward design is a critical yet often underexplored component of reinforcement learning for planning problems. In this work we analyze how different reward shaping strategies affect learning performance when using Proximal Policy Optimization for planning tasks maximizing sub-goals and minimizing plan length. We evaluate several reward formulations derived from the objective function while keeping the learning algorithm and model architecture fixed. Experiments on benchmarks from the International Planning Competition 2023 show that commonly used shaping strategies do not consistently outperform sparse rewards. While some formulations provide competitive performance, our results highlight the difficulty of designing effective reward signals for policy gradient methods in planning domains.

Introduction

Planning problems are fundamental in robotics, scheduling, and decision-making (Ghallab, Nau, and Traverso 2004). While classical symbolic planning dominates, reinforcement learning (RL) has emerged as a promising complementary approach (Chen and Tian 2019; Ståhlberg, Bonet, and Geffner 2022, 2023). However, RL success depends critically on reward design. In planning domains with completely known deterministic transitions, the natural reward signal is sparse: agents receive feedback only upon reaching a goal. This sparsity severely hampers learning, motivating *reward shaping*—reformulating rewards to provide denser feedback during training.

Practitioners have proposed many approaches to address sparse rewards in planning: incremental improvements (Chen and Tian 2019), best-state rewards (Sutton and Barto 2018), potential-based approaches (Gehring et al. 2022; Wiewiora 2003) using heuristics, and others. Yet a fundamental question remains unanswered: *do these reward structures actually motivate agents to maximize sub-goal satisfaction while minimizing plan length?* Without this understanding, practitioners resort to ad-hoc reward choices, and we lack data on whether theoretically-motivated designs translate to learning performance in practice.

In this work, we systematically analyze existing reward formulations for planning-based RL. We define a *natural order* property capturing when rewards align with planning objectives: preferring solutions that satisfy more sub-goals and favoring shorter trajectories when quality is tied. We evaluate common reward formulations against this property through both theoretical analysis and empirical study. Specifically, we train Proximal Policy Optimization (PPO) (Schulman et al. 2017) under each reward scheme on diverse planning tasks from the 2023 International Planning Competition, keeping the algorithm, architecture, and hyperparameters fixed to isolate reward effects. The results should be interpreted within this setting and are not intended as general claims about RL or planning methods.

Our results reveal that reward formulations with desirable theoretical properties do not consistently outperform others in empirical learning. The findings highlight the difficulty of designing effective reward signals for policy gradient methods in deterministic planning domains and suggest that the relationship between reward structure and learning performance is more complex than existing heuristics assume.

Background

We consider deterministic planning problems defined over a structured state space. A state $s \in \mathcal{S}$ represents a configuration of the environment, and from each state the agent can take any applicable action $a \in \mathcal{A}(s)$. The application of an action deterministically produces a successor state $s' = T(s, a)$ through a transition function T. The number of applicable actions varies across states, resulting in a non-uniform branching factor, which is a key challenge for learning and search in planning domains.

Planning problems can be naturally cast as Markov Decision Processes (MDPs), enabling the use of reinforcement learning (RL) methods (Sutton and Barto 2018). An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the transition function, R is a reward function, and $\gamma \in (0, 1]$ is a discount factor. In our setting, transitions are deterministic, but the reward function may be shaped to provide informative feedback during search. Casting planning as an MDP allows us to interpret the search process as sequential decision making and to optimize decision policies using RL techniques.

A stochastic policy π_θ parameterized by θ defines a distri-

*These authors contributed equally.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

bution over decisions at each state. Given deterministic transitions, it is convenient to express the policy directly over successor states, $\pi_\theta(s' \mid s)$, which implicitly induces a distribution over actions via $s' = T(s, a)$. A trajectory is a sequence of visited states $\tau = (s_0, s_1, \ldots, s_{|\tau|-1})$. In our planning setting, solution quality is given by the number of sub-goals satisfied while executing the trajectory, and trajectory efficiency by plan length. The quality of a trajectory is measured by its discounted return

$$G(\tau) = \sum_{t=0}^{|\tau|-2} \gamma^t R(s_t, s_{t+1}). \tag{1}$$

In planning domains, the policy needs to be flexible enough to define a distribution over a variable number of actions. We use a successor-based parametrization of the policy. Given a state s with successors $\mathrm{Succ}(s) = \{T(s, a) : a \in \mathcal{A}(s)\}$, the corresponding policy is defined as

$$\pi_\theta(s' \mid s) = \frac{\exp(z_\theta(s'))}{\sum_{\tilde{s} \in \mathrm{Succ}(s)} \exp(z_\theta(\tilde{s}))}.$$

Problem Definition

In our settings, each state $s \in \mathcal{S}$ is associated with a scalar objective value

$$o \colon \mathcal{S} \to \mathbb{R},$$

measuring the quality of the state with respect to the underlying planning goal. Specifically, $o(s)$ measures the degree of goal satisfaction achieved in state s, such as the number of satisfied sub-goals. Rather than evaluating only the final state of a trajectory, we adopt the common perspective in search that the quality of a trajectory is determined by the best state encountered along it. Accordingly, we define the trajectory-level objective as

$$\mathcal{O}(\tau) = \max_{s \in \tau} o(s). \tag{2}$$

This formulation reflects the fact that search procedures aim to discover high-quality states at any point during exploration, rather than only at termination.

While reinforcement learning optimizes policies with respect to cumulative reward, the ultimate objective in planning is to maximize $\mathcal{O}(\tau)$. Therefore, the designed reward function should prioritize trajectory returns that are consistent with the ordering defined by the planning objective.

Concretely, we adopt the standard trajectory preference in planning, called the *natural order* here: trajectories are ranked primarily by the maximum goal satisfaction over their states, and ties are broken in favor of shorter trajectories to reflect a bias toward efficient solutions.

Note that designing a reward function that satisfies the natural order, a lexicographic ordering on pairs $(\max_{s \in \tau} o(s), -|\tau|)$, is tricky. The best objective value encountered along a trajectory is inherently non-additive, whereas any cumulative return based on step-wise reward is additive by definition. Even formulations that track the best value so far cannot simultaneously prefer shorter trajectories without access to the actual trajectory length, and naively penalizing length can violate the quality ordering: a discount factor $\gamma < 1$ suppresses late rewards, potentially undervaluing trajectories whose best state appears late, while a constant step penalty can cause accumulated costs to overwhelm the objective value of long but high-quality trajectories.

Reward Design Strategies

We discuss different reward constructions and briefly comment on their alignment with the natural order (goal satisfaction first, then trajectory length). All time- or history-dependent variants (including OPD and MSF) use standard state augmentation to preserve the Markov property.

Terminal Reward (TR). Terminal rewards are common in combinatorial optimization and search because they avoid shaping intermediate behavior and only score at the end of a trajectory (Bello et al. 2017; Kool, van Hoof, and Welling 2019). Let $\mathcal{T} \subseteq \mathcal{S}$ be the set of terminal states and let $\tau = (s_0, s_1, \ldots, s_n)$ terminate at $s_n \in \mathcal{T}$. Define

$$R(s_t, a_t) = \begin{cases} o(s_n), & \text{if } t = n - 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

In general, this construction violates the natural order: trajectories with better terminal states get higher returns, but those with the same terminal outcome receive equal returns regardless of length, so shorter solutions are not preferred.

Incremental Objective Difference (IOD). Incremental objective-difference rewards provide dense feedback by rewarding local progress in the underlying objective, which is a standard trick for reducing sparsity in learning-based planning and control (Chen and Tian 2018). Formally, define

$$R(s_t, a_t) = o\big(T(s_t, a_t)\big) - o(s_t). \tag{4}$$

In general, this reward violates the natural order: cumulative local improvements can rank trajectories differently from $\max_{s \in \tau} o(s)$, making returns depend on the path rather than the best achieved quality and solution length.

Step incremental. A common motivation for incremental shaping methods is to avoid sensitivity to the discount factor by using undiscounted returns and encoding preferences directly in the per-step reward (Sutton and Barto 2018).

$$R'(s_t, a_t) = R(s_t, a_t) - c, \tag{5}$$

where $c > 0$ is a constant, and $R(s_t, a_t)$ can be instantiated using IOD and TR, yielding SIOD and STR, respectively. In terms of natural-order alignment, the conclusion is the same as for IOD rewards.

Maximum-So-Far (MSF). This reward mirrors improvement-based criteria such as expected improvement in Bayesian optimization (Bergstra et al. 2011). To preserve the Markov property, the MDP is extended to track the maximum objective value seen so far, with state $s_t = (v_t, m_t)$ and $m_t = \max_{0 \le i \le t} o(v_i)$. The transition function is defined as

$$T'\big((v_t, m_t), a_t\big) = \big(T(v_t, a_t),\, \max\{m_t,\, o(T(v_t, a_t))\}\big),$$

and the reward function is given by

$$R\big((v_t, m_t), a_t\big) = \max(0,\, m_{t+1} - m_t), \tag{6}$$

where $(v_{t+1}, m_{t+1}) = T'((v_t, m_t), a_t)$. This construction violates the natural order by ignoring solution length and depending on the distribution of improvements.

Potential-based shaping from heuristics (POT). POT is a principled way to densify rewards while preserving optimal policies under mild conditions (Ng, Harada, and Russell 1999), and it can be instantiated from planning heuristics by setting the potential to a heuristic estimate (Gehring et al. 2022). Formally, for $s_{t+1} = T(s_t, a_t)$ define

$$R'(s_t, a_t) = R(s_t, a_t) + \gamma\,\phi(s_{t+1}) - \phi(s_t), \qquad (7)$$

where typically $\phi(s) = -h(s)$ for a cost-to-go heuristic h and $R(s_t, a_t)$ is the reward based on TR. In general, this shaped reward breaks the natural order: it guides policy optimization rather than enforcing $(\max_{s\in\tau} o(s), -|\tau|)$, so equal-quality trajectories may be ranked inconsistently.

Relative Objective Difference (ROD). Relative (normalized) improvements are common in numerical optimization and benchmarking because they measure progress on a scale-free basis (Nocedal and Wright 2006; Moré and Wild 2009). Formally, assuming $o(s_t) \neq 0$ define

$$R(s_t, a_t) = \frac{o(T(s_t, a_t)) - o(s_t)}{o(s_t)}.$$

This construction violates the natural order since step-wise aggregation can misrank trajectories and ignores length.

Direct Preference Optimization (DPO). DPO optimizes a policy directly from pairwise trajectory preferences, avoiding explicit reward modeling and a separate RL fine-tuning stage (Rafailov et al. 2023). Formally, given a preferred trajectory τ_1 and a dispreferred trajectory τ_2, DPO minimizes

$$\mathrm{L} = -\log\sigma\left(\beta\log\frac{\pi_\theta(\tau_1)}{\pi_{ref}(\tau_1)} - \beta\log\frac{\pi_\theta(\tau_2)}{\pi_{ref}(\tau_2)}\right), \qquad (8)$$

where π_θ is the policy, π_{ref} a fixed reference policy, and $\beta > 0$ a temperature parameter. Binary preferences may violate the lexicographic order, so the natural order does not hold.

Order-Preserving Decay Shaping (OPD). This is a modification of IOD to also preserve the natural order by incorporating a step-dependent decay term

$$R(s_t, a_t) = o\big(T(s_t, a_t)\big) - o(s_t) + \frac{\delta}{t+2} - \frac{\delta}{t+1}, \qquad (9)$$

where $\delta > 0$, and $\forall s, s' \in \mathcal{S},\ o(s) \neq o(s') \Rightarrow |o(s) – o(s')| > \delta$.

Experiments

Experimental settings. The experimental protocol follows the bootstrap process (Arfaee, Zilles, and Holte 2011). For each domain, we iterate sequentially over the available IPC'23 Learning Track instances (Taitler et al. 2024). For each instance, we (i) run PPO search with the current policy until a goal is reached, the 30s time limit is exceeded, or a dead end is reached (no applicable actions), (ii) extract learning samples according to the corresponding reward construction, and (iii) update the model parameters using the collected samples. Training proceeds online and sequentially across instances without resetting the model.

All methods use the same PPO procedure, network architecture, and optimizer settings. For data collection, we split each linear rollout into trajectory windows of length 100. For rollouts longer than 100, we additionally add windows starting every 50 steps. PPO then performs one actor update epoch and five critic update epochs per minibatch (using the same minibatches for both). We use GAE with $\lambda = 0.95$, clipping parameter $\epsilon = 0.2$ (clip coefficient 0.2), and normalize advantages per minibatch. The critic is trained with an MSE loss. We optimize both actor and critic with Adam (learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$), with learning-rate decay by a factor 0.8 every 5 updates. We train for 10 outer epochs and evaluate both $\gamma \in \{0.99, 1.0\}$. The actor and critic are implemented as separate networks and optimized independently.

We use the ObjectBinary encoding with edge features as proposed in (Horčík et al. 2025) with following settings: a single message-passing layer (hidden dimension 32), `sum` and `max` pooling, and a two-layer MLP (hidden dimension 32, output dimension 1) with ReLU nonlinearities.

We define a state scoring function $o(s)$ as the number of achieved goal atoms in s (size of the intersection of the state's atoms with the goal atoms).

Learning is performed as proposed in (Orseau and Lelis 2021), where the goal is to solve a set of problem instances in a given time, which is in case of this paper 10 epochs. The goal is not to evaluate generalization to unseen instances, but to compare the effect of different reward shaping strategies on learning within a fixed set of problems. All chosen instances in each domain are used for training, and results are reported as mean and standard deviation over three independent runs for each method on each problem instance. The experiments were implemented in Julia 1.11.7 on top of `PDDL.jl` and `SymbolicPlanners.jl` (Zhi-Xuan 2022). We use the official implementation of the ObjectBinary state encoding from (Horčík et al. 2025).

Results. According to Table 1, IOD and TR—two commonly used reward designs in planning domains—exhibit very similar behavior and overall strong performance. Both approaches provide stable and effective guidance for the learning process. As shown in Figure 1, IOD tends to produce the shortest solutions for many of the problem instances and is the second best (DPO is the best) on average. In contrast, MSF, which directly reflects the behavior of available solvers, performs noticeably worse than the other approaches. This can be attributed to the nature of its reward signal, which does not sufficiently emphasize global optimality and instead tends to favor local objective satisfaction (see Figure 2). The results further indicate that ROD is not a suitable choice for reward design in planning domains, as it consistently underperforms compared to the alternatives. Although DPO does not demonstrate competitive performance in our experiments, we believe this is primarily due to the limited number of trajectories provided during training, rather than an inherent weakness of the method itself. SIOD and STR represent two variations of the same underlying idea, and thus their similar performance is expected. However, both methods are consistently inferior to IOD and TR,

	IOD	TR	MSF	ROD	DPO	SIOD	STR	OPD	POT
Blocks	60.7±5.3	62.9±5.4	50.5±10.0	22.6±15.4	6.0±0.6	54.3±5.6	61.9±12.8	54.5±10.3	**70.2±5.3**
Ferry	6.6±0.3	**7.4±1.9**	6.0±1.2	6.3±1.4	5.8±1.8	5.3±0.5	5.6±1.4	6.2±1.1	5.5±0.8
Floortile	2.1±0.6	1.2±0.8	1.4±0.8	1.8±0.6	1.4±0.3	2.5±0.3	2.0±0.8	**2.8±0.8**	1.3±0.3
Miconic	**12.5±0.3**	11.5±0.8	12.3±0.6	11.1±1.4	11.5±1.1	11.1±0.0	11.3±0.3	11.5±1.1	10.4±1.1
Rovers	52.0±3.8	56.4±1.4	53.4±2.7	52.9±2.3	52.9±3.5	52.2±5.1	48.3±4.3	53.1±2.5	**60.7±1.2**
Satellite	68.9±1.3	70.4±0.0	60.6±12.7	45.3±8.4	36.3±2.0	70.0±0.8	69.5±1.4	**70.6±0.3**	69.0±1.4
Sokoban	4.4±0.3	5.0±0.3	4.8±1.9	**5.1±0.3**	4.6±1.1	**5.1±0.3**	5.0±1.4	4.8±0.6	4.6±1.3
Spanner	76.9±2.2	77.1±1.1	74.1±5.6	78.8±2.0	**81.5±1.0**	75.2±3.1	76.2±2.3	76.3±2.7	79.9±0.0
Transport	71.4±0.9	71.4±0.9	30.7±5.1	29.3±8.2	11.8±2.2	72.0±1.1	72.0±1.4	70.0±1.7	**72.7±0.3**
Total	39.5±0.1	40.4±0.4	32.6±1.3	28.1±3.0	23.5±0.7	38.6±1.4	39.1±1.0	38.9±1.3	**41.6±0.6**

Table 1: The final results for every method. Each entry is mean ± standard deviation (%) of solved problems over 3 runs. **Bold** marks the best result in each row, and **Total** is the average across domains.

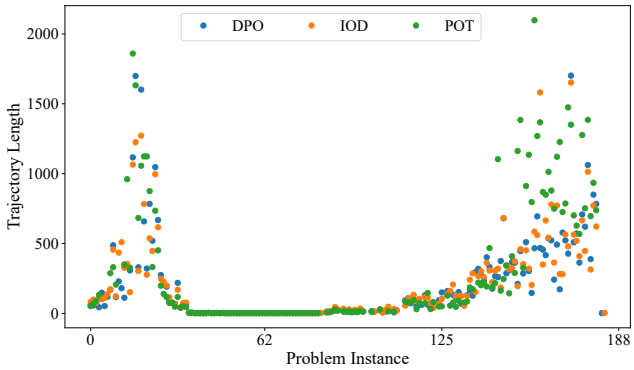

Figure 1: Solution lengths for Rovers instances (solved cases), showing top two methods and the worst.

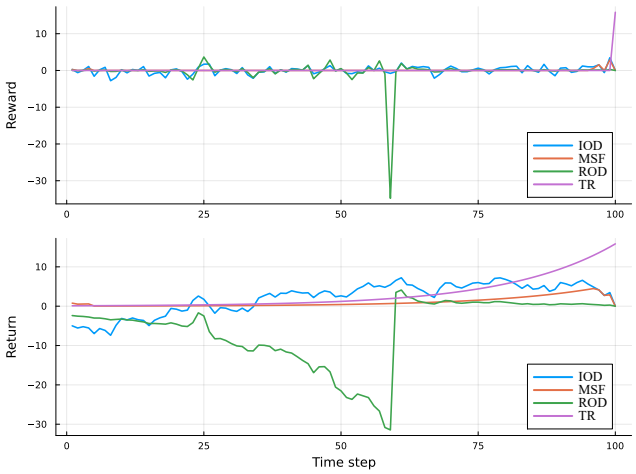

Figure 2: Comparison between the behavior of different reward designs for a given trajectory.

suggesting that they are not optimal choices for reward design in this context. From a lexicographic perspective, OPD appears to be a promising reward design, as it aims to balance optimality with shorter solution lengths. However, the results in Table 1 reveal that it can fail significantly in certain cases. We attribute this behavior to the fact that, in many planning problems, step-wise transitions do not always yield a consistent δ-improvement. As a result, OPD cannot reliably guarantee either optimality or minimal solution length. Finally, POT demonstrates strong performance in some instances. However, as shown in Figure 1, it often produces significantly longer solutions. Therefore, while POT may be a reasonable choice when optimality is the sole objective, it becomes less suitable in scenarios where solution length is also an important consideration.

A complete comparison of solved problems across different γ values, instance lengths for all domains, and reward behavior for all methods and domains is provided in the supplementary materials accompanying the paper.

Conclusion

In this work, we investigated the impact of different reward design strategies across a diverse set of IPC benchmark domains. We observed that the overall performance of the evaluated methods is similar. While certain reward formulations achieve better results on individual domains, the differences between top-performing approaches remain small. This suggests that, within the considered setting, the choice of reward shaping strategy has a limited effect on performance when combined with PPO. These findings do not generalize to all planning domains. Domains with different structural properties may exhibit stronger sensitivity to reward design.

As future work, we plan to evaluate reward design strategies across a wider range of RL methods, including alternative policy gradient methods, value-based approaches, and search-based methods such as AlphaZero. We aim to develop more informative evaluation metrics that better capture differences in learning dynamics and solution quality beyond the average of solved instances.

Overall, our results suggest that while reward shaping plays an important conceptual role, its empirical impact may be more subtle than commonly assumed.

Acknowledgments

Oleksii Shuhailo was supported by the European Space Agency (ESA) InCubed programme through the project "Enhanced Next-Generation Geospatial Intelligence with Large Language Models (ENGINE)". Tomas Pevny acknowledges support from the Czech Science Foundation (GAČR) under grant no. 25-17259K, "Tradeoffs for Information Hiding in Generated Media (DETERMINE)". Karel Chvalovský's work was supported by the Czech Science Foundation grant no. 24-12759S. The authors acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Arfaee, S. J.; Zilles, S.; and Holte, R. C. 2011. Learning heuristic functions for large state spaces. *Artif. Intell.*, 175(16-17): 2075–2098.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2017. Neural Combinatorial Optimization with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

Bergstra, J.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, 2546–2554. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781618395993.

Chen, X.; and Tian, Y. 2018. Learning to Progressively Plan. *CoRR*, abs/1810.00337.

Chen, X.; and Tian, Y. 2019. Learning to perform local rewriting for combinatorial optimization. *Advances in neural information processing systems*, 32.

Gehring, C.; B"ackstr"om, C.; Brown, D. S.; Juba, B.; and Zilles, S. 2022. Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*.

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

Horčík, R.; Šír, G.; Šimek, V.; and Pevnỳ, T. 2025. State Encodings for GNN-Based Lifted Planners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 26525–26533.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations (ICLR)*.

Moré, J. J.; and Wild, S. M. 2009. Benchmarking Derivative-Free Optimization Algorithms. *SIAM Journal on Optimization*, 20(1): 172–191.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 278–287.

Nocedal, J.; and Wright, S. J. 2006. *Numerical Optimization*. Springer, 2 edition.

Orseau, L.; and Lelis, L. H. 2021. Policy-guided heuristic search with guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12382–12390.

Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 629–637.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2023. Learning General Policies with Policy Gradient Methods. In *Proceedings of the Twenty-First International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 647–657.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fišer, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Magazine*, 45(2): 280–296.

Wiewiora, E. 2003. Potential-Based Shaping and Q-Value Initialization are Equivalent. In *Proceedings of the 12th Yale Workshop on Adaptive and Learning Systems*.

Zhi-Xuan, T. 2022. *Pddl. jl: An extensible interpreter and compiler interface for fast and flexible ai planning*. Ph.D. thesis, Massachusetts Institute of Technology.