Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Per-Domain Generalizing Policies: On Learning Efficient and Robust Q-Value Functions

Nicola J. Müller[1,2,3], Moritz Oster[1,2], Isabel Valera[2], Jörg Hoffmann[1,2], Timo P. Gros[1,2,3]
[1]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
[2]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[3]Center for European Research in Trusted Artificial Intelligence (CERTAIN)
{Nicola.Mueller, Moritz.Oster, Timo_Phillip.Gros}@dfki.de, {ivalera,hoffmann}@cs.uni-saarland.de

Abstract

Learning per-domain generalizing policies is a key challenge in learning for planning. Standard approaches learn state-value functions represented as graph neural networks using supervised learning on optimal plans generated by a teacher planner. In this work, we advocate for learning Q-value functions instead. Such policies are drastically cheaper to evaluate for a given state, as they need to process only the current state rather than every successor. Surprisingly, vanilla supervised learning of Q-values performs poorly as it does not learn to distinguish between the actions taken and those not taken by the teacher. We address this by using regularization terms that enforce this distinction, resulting in Q-value policies that consistently outperform state-value policies across a range of 10 domains and are competitive with the planner LAMA-first.

Code — https://doi.org/10.5281/zenodo.19069966

1 Introduction

Learning per-domain generalizing policies using neural networks is gaining popularity in planning (Toyer et al. 2020; Ståhlberg, Bonet, and Geffner 2022a,b, 2025; Rossetti et al. 2024; Wang and Thiébaux 2024; Chen et al. 2025a,b). The key challenge is to foster scaling behavior, i.e., generalizing from small training instances to large test instances.

Prior work used *graph neural networks* (GNNs) to learn *state-value functions* $V : \mathcal{S} \to \mathbb{R}^+$ for classical planning domains, which approximate the optimal heuristic h^* (Ståhlberg, Bonet, and Geffner 2022a,b, 2025; Gros et al. 2025). Given V, a per-domain generalizing policy is computed as $\pi(s) = \arg\min_a \text{cost}(a) + V(s')$, where s' is the respective successor reached through action a.

Observe that choosing an action this way requires the GNN to process every successor state. This can be very ineffective and impede scaling performance, even when using batch processing. This motivates learning a Q-value function $Q : \mathcal{S} \times \mathit{Act} \to \mathbb{R}^+$ instead, which approximates the cost of an optimal plan starting with a given action a from a given state s (Sutton, Barto et al. 1998). A policy can then be computed as $\pi(s) = \arg\min_a Q(s, a)$, which only requires the current state to compute the Q-values of all applicable actions with a single GNN forward pass. Figure 1 (left) shows the runtime per instance size on random states for the IPC'23 domain Rovers (Taitler et al. 2024), where a Q-value policy is 5 times faster than a state-value policy.

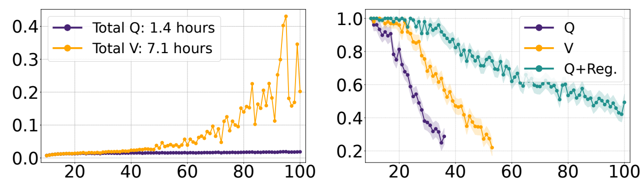

Figure 1: (Left) Total runtime of forward passes in hours per instance size on random states. (Right) Average coverage and confidence interval per instance size for trained policies.

We leverage existing planners to train our Q-value functions using *supervised learning* (SL), where the training data consist of optimal plans (Ståhlberg, Bonet, and Geffner 2022a; Rossetti et al. 2024; Gros et al. 2025). To the best of our knowledge, this has not been tried before for learning per-domain generalizing policies. Surprisingly, we show that the vanilla use of SL for Q-values yields policies that generalize poorly, despite fitting the training data well. Consider Figure 1 (right), which shows the average coverage per instance size of state (orange) vs. Q-value (purple) policies. Both policies were trained on the same optimal plans using the same loss; yet, the state-value policy generalizes much better than the Q-value policy. This is because the Q-values are often identical for all actions, leading to random action selection. In other words, the Q-values fail to distinguish between the actions taken and not taken by the teacher planner.

Our key contribution is showing that this can be fixed. We extend the SL objective with a regularization term that enforces the Q-values of non-teacher actions to be larger than those of teacher actions. This approach is shown in green in Figure 1 (right), and it vastly outperforms the state-value policy. Importantly, this difference is strictly due to better action selection, as policy runs are only limited by the number of actions, without any runtime limit.

We evaluate our approach with three GNN architectures on 10 domains and find that the trends from Figure 1 (right) apply consistently, yielding regularized Q-value policies that outperform state-value policies and are competitive with the planner LAMA-first.

2 Efficiency of State and Q-Value Policies

Here, we show that Q-value policies are more efficient than state-value policies using three *graph neural network* (GNN) architectures for per-domain generalizing policies.

GNN architectures. The *relational graph neural network* (R-GNN) is a specialized GNN architecture that operates directly on the relational structure induced by classical planning states, maintaining embeddings of a state's objects and passing information between them according to the ground atoms of which they are arguments (Ståhlberg, Bonet, and Geffner 2022a). Conversely, the *object encoding* (OE) and *object-atom encoding* (OAE) define graph representations of states that can be processed by off-the-shelf GNN architectures (Horčík and Šír 2024); we use the *relational graph convolutional network* (RGCN) (Schlichtkrull et al. 2018), as in prior work (Chen, Thiébaux, and Trevizan 2024). In a nutshell, OE defines nodes for a state's objects and adds edges between objects occurring in the same ground atoms, whereas OAE defines nodes for both objects and atoms, adding edges between atoms and their arguments.

All three GNN architectures have in common that they iteratively compute embeddings for every object in a given state. Independent of the architecture, the object embeddings can be aggregated into a state embedding, which is then used to predict either a single state-value or the Q-values of all applicable actions. We use the approach of Ståhlberg and Geffner (2025). For the sake of brevity, we omit the details here and provide a description in the technical appendix (Müller et al. 2026).

Policy efficiency. To compute a policy using state-values, i.e., $\pi(s) = \arg\min_a \mathrm{cost}(a) + V(s')$, we need to predict the values of all successor states s'. This requires expanding the current state, translating the successors into GNN input, combining them into a single batch, and passing it to the GNN. To compute a policy using Q-values, i.e., $\pi(s) = \arg\min_a Q(s, a)$, we need to predict the values of all applicable actions in the current state s. This only requires translating the current state into GNN input and passing it to the GNN. Hence, state-value policies are less efficient than Q-value policies because they require processing every successor of the current state instead of the current state only. When scaling the instance size, the overhead for computing state-values grows drastically, as a larger instance size typically coincides with a larger branching factor, and, thus, more successors to process at every state.

Consider Table 1, which compares the runtimes of state and Q-value policies on the 10 domains used in Section 4. For each domain, we generate a set of states using random walks of length 25 for 25 instances per size, ranging from the smallest possible instance size up to 100. For each state, we predict an action using a randomly initialized state or Q-value policy and record the total runtime. All policies are executed on an NVIDIA RTX A6000 GPU. Across all architectures, we see that the average runtimes of the state-value policies are between 4 and 18 times higher than those of the Q-value policies. We conclude that Q-values should be preferred over state-values for per-domain generalizing policies, as they scale much more efficiently with instance size.

	R-GNN		OE		OAE	
Domain	V	Q	V	Q	V	Q
Blocksworld	0.6	0.5	1.8	2.0	0.7	0.4
Childsnack	7.2	0.9	21.7	4.3	28.6	0.7
Ferry	0.8	0.5	2.5	2.0	1.6	0.4
Floortile	1.9	1.0	7.9	4.0	4.0	0.6
Gripper	1.0	0.6	3.1	2.2	1.9	0.5
Logistics	2.7	0.7	7.3	2.3	10.2	0.6
Rovers	7.1	1.4	39.8	7.5	25.4	1.1
Satellite	4.6	0.8	14.7	3.5	16.0	0.7
Transport	6.5	0.7	22.8	2.7	18.0	0.7
Visitall	0.4	0.2	1.1	0.6	0.9	0.3
Average	3.3	0.7	12.3	3.1	10.7	0.6

Table 1: Total runtime in hours for randomly initialized state and Q-value policies on sets of random states.

3 Regularization Terms for Supervised Learning of Q-Value Functions

We now show why vanilla supervised learning (SL) fails to learn Q-value functions for per-domain generalizing policies, and then fix this by introducing regularization terms.

3.1 Supervised Learning of Q-Value Functions

Prior work learned state-value functions using SL, where the training data consist of optimal plans for small domain instances (Ståhlberg, Bonet, and Geffner 2022a; Gros et al. 2025). However, we find that learning Q-value functions using this approach yields policies that fail to generalize. This is because the Q-value functions tend to predict the same Q-values for all actions applicable in a given state, leading to random action selection.

We investigate this behavior by comparing Q-value functions successfully learned for 10 domains using the R-GNN, OE, and OAE architectures. We refer to Section 4 for training details. For each model, we compute the average prediction error for the Q-values of teacher actions a^* in the training set (Err), i.e., $|h^*(s) - Q(s, a^*)|$. We also compute the average difference in predicted Q-values between teacher actions a^* and non-teacher actions $a_i \in Act(s) \setminus \{a^*\}$ in the training set (Diff), i.e., $|Q(s, a^*) - Q(s, a_i)|$. We present the averages of both scores over 10 domains. Consider the left column (Q) in Table 2. All models fit their training data well, as the average prediction errors for teacher actions (Err) are small, with values between 0.54 and 2.15. However, the models tend to predict the same Q-values for teacher and non-teacher actions, as the average Q-value differences (Diff) are between 0.71 and 0.96. Hence, Q-value policies learned using vanilla SL perform poorly because they fail to generalize to non-teacher actions.

3.2 Regularizing Q-Value Functions

To ensure that learned Q-value functions distinguish between teacher and non-teacher actions, we extend the SL objective with a regularization term. The extended learning objective is defined as

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(s,a^*)\sim D} \left[\mathcal{L}(s, a^*) + \lambda \cdot \Omega(s, a^*)\right],$$

where θ^* are the parameters of the Q-value function Q that minimizes the loss $\mathcal{L}$ subject to the regularizer Ω, and λ

	Q						$\Omega^{\text{Exp.}}$						$\Omega^{\text{Heu.}}$					
	R-GNN		OE		OAE		R-GNN		OE		OAE		R-GNN		OE		OAE	
	Err	Diff	Err	Diff	Err	Diff	Err	Diff	Err	Diff	Err	Diff	Err	Diff	Err	Diff	Err	Diff
Average	0.54	0.71	1.37	0.96	2.15	0.84	0.70	4.69	0.70	7.66	0.91	6.66	0.73	97.55	1.25	46.25	0.83	152.52

Table 2: Prediction errors for teacher actions (Err), and Q-value differences between teacher and non-teacher actions (Diff).

is the regularization coefficient. The role of $\mathcal{L}$ is to ensure that Q predicts the Q-values of teacher actions a^* accurately, whereas Ω ensures that the Q-values of non-teacher actions a_i are larger than those of teacher actions a^*, i.e., $\forall a_i \in Act(s)\setminus\{a^*\} : Q(s, a_i) > Q(s, a^*)$. We define the regularization term in general as

$$\Omega(s, a^*) = \sum_{a_i \in Act(s)\setminus\{a^*\}} \max\{0, B - Q(s, a_i)\},$$

where B is a lower bound for the Q-values of non-teacher actions a_i. The penalty induced by Ω increases linearly if $Q(s, a_i)$ is smaller than B, whereas there is no penalty if $Q(s, a_i) \geq B$ holds. We now introduce two regularizers Ω that differ in how they compute the lower bound B.

Explicit regularizer. The *explicit regularizer* $\Omega^{\text{Exp.}}$ computes B using the h^* values of the states s in the training data, which is feasible as they are on optimal trajectories. We define the lower bound as

$$B^{\text{Exp.}}(s) = h^*(s) + 1,$$

ensuring that the Q-values of non-teacher actions $Q(s, a_i)$ are strictly larger than the target values of teacher actions $Q(s, a^*) = h^*(s)$. While this biases learning towards an inadmissible Q-value function, as there might be multiple optimal actions, we can still converge to an optimal policy since teacher actions always have the lowest Q-values. We note that, in general, admissibility cannot be guaranteed for a learned Q-value function, even without this bias.

Heuristic regularizer. The *heuristic regularizer* $\Omega^{\text{Heu.}}$ uses an admissible heuristic to compute *action-dependent* lower bounds B_i, which provide more information about the true Q-values of non-teacher actions a_i. First, we compute a lower bound $\text{cost}(a_i) + h(s'_i)$ for every a_i, where s'_i is the successor reached through action a_i, and h is an admissible heuristic; in this paper, h^{LMcut} (Helmert and Domshlak 2009)[1]. However, this lower bound may not be strictly larger than the target Q-value of the teacher action, i.e., $h^*(s)$, preventing us from distinguishing between non-teacher and teacher actions. Thus, we compute the final lower bound as

$$B^{\text{Heu.}}(s, a_i) = \max\{h^*(s) + 1, \text{cost}(a_i) + h(s'_i)\}.$$

On average, $B^{\text{Heu.}}$ is tighter than $B^{\text{Exp.}}$ for 17% of the non-teacher actions in the training sets used in Section 4.

Comparing regularizers. Consider again Table 2, where the middle ($\Omega^{\text{Exp.}}$) and right ($\Omega^{\text{Heu.}}$) columns show the prediction errors (Err) and Q-value differences (Diff) for Q-value functions trained using the explicit and heuristic regularizers, respectively. Compared to vanilla SL on the left (Q), the Q-value differences increase while the prediction errors mostly decrease, suggesting that the regularizers not only help to distinguish between non-teacher and teacher actions but can also help to predict the Q-values of teacher actions more accurately. The $\Omega^{\text{Heu.}}$ models also have considerably larger Q-value differences than the $\Omega^{\text{Exp.}}$ models, which is due to the high predicted value for actions leading to dead-ends.

4 Experiments

In this section, we outline our data set construction and training setup, and then evaluate our approaches with respect to scaling behavior and IPC test set performance.

Data sets. We consider the domains from the IPC'23 learning track (Taitler et al. 2024), where we omit Miconic and Spanner because all policies immediately achieve 100% coverage, and Sokoban because its generator requires unfeasibly many runs until a solvable instance is generated. We also consider the Gripper, Logistics, and Visitall domains from the FF domain collection[2].

To construct training sets for each domain, we use the publicly available generators to uniformly sample up to 100 unique instances per size, ranging from size 2 to 100. We use the *seq-opt-merge-and-shrink* configuration of Fast Downward (Helmert 2006) with limits of 20 minutes and 64 GB as the optimal teacher planner, deleting instances that were not solved. We provide full details about the training sets in the technical appendix (Müller et al. 2026). For validation, we use Gros et al.'s dynamic coverage validation, which does not require a precomputed validation set, as it runs policies on increasingly large instances generated on-the-fly (2025).

Training setup. We train state, Q, and regularized Q-value functions using the R-GNN, OE, and OAE architectures. As the loss function $\mathcal{L}$, we use the *mean absolute error* (MAE), which is defined as $\text{MAE}(s) = |h^*(s) - V(s)|$ for state-values and as $\text{MAE}(s, a^*) = |h^*(s) - Q(s, a^*)|$ for Q-values. In preliminary experiments, we also tested the mean squared error and temporal difference error losses, but found that MAE yields the best policies. For the regularized Q-value functions, we use a regularization coefficient of $\lambda = 1.0$, giving equal weight to $\mathcal{L}$ and Ω. We repeat the training using three random seeds and return the policy with the best validation performance. When running the policies, we forbid revisiting states to prevent cycles, as in prior work (Ståhlberg, Bonet, and Geffner 2022a,b, 2023).

Scaling behavior evaluation. We use Gros et al.'s scaling behavior evaluation to compare how well the trained policies

[1]For dead-ends, i.e., $h^{\text{LMcut}}(s'_i) = \infty$, we use a value of 1120, which is drastically larger than all plans in our training data.

[2]https://fai.cs.uni-saarland.de/hoffmann/ff-domains.html

	R-GNN								OE								OAE							
	V		Q		$\Omega^{\text{Exp.}}$		$\Omega^{\text{Heu.}}$		V		Q		$\Omega^{\text{Exp.}}$		$\Omega^{\text{Heu.}}$		V		Q		$\Omega^{\text{Exp.}}$		$\Omega^{\text{Heu.}}$	
Domain	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov	Scale	SCov
Blocksworld	52	44.5	5	1.6	54	47.8	**55**	**48.2**	14	9.2	5	1.2	48	40.5	**50**	**42.4**	37	27.5	4	1.3	**52**	**44.9**	**52**	44.6
Childsnack	71	43.4	16	4.2	**100**	**93.0**	**100**	**93.0**	26	12.0	9	0.2	82	43.1	**98**	**46.9**	31	15.9	36	14.9	**100**	**65.8**	83	41.5
Ferry	**100**	84.5	30	12.9	**100**	**85.2**	**100**	84.5	33	19.8	21	10.9	**100**	**80.2**	**100**	80.1	45	26.4	7	2.1	**100**	79.5	**100**	**80.1**
Floortile	8	0	8	0	60	44.5	**62**	**45.6**	8	0	8	0	50	28.8	**56**	**43.2**	8	0	8	0	59	42.0	**62**	**45.9**
Rovers	53	29.8	36	17.1	**100**	**66.5**	**100**	64.4	25	9.8	33	13.3	**48**	**24.6**	46	23.0	22	8.3	28	9.8	60	**30.1**	**61**	29.8
Satellite	50	28.2	37	21.5	**100**	**69.4**	**100**	69.1	20	9.8	25	12.9	**73**	**45.3**	**73**	45.2	24	9.7	6	0.2	**92**	55.1	90	**55.5**
Transport	72	46.6	46	24.7	**100**	90.4	**100**	**92.7**	38	20.1	42	23.1	**100**	88.6	**100**	**89.7**	19	9.3	57	32.1	**100**	**89.7**	**100**	**89.6**
Gripper	**100**	77.5	36	15.2	**100**	88.8	**100**	**88.9**	**100**	83.0	63	33.4	**100**	**88.9**	**100**	88.4	**100**	**88.6**	**100**	73.8	**100**	88.5	**100**	**88.6**
Logistics	47	24.9	50	25.9	**100**	82.4	**100**	**84.3**	36	19.8	52	27.0	**100**	**86.2**	**100**	85.7	38	22.6	57	31.2	**100**	**77.0**	**100**	75.6
Visitall	256	11.4	256	11.4	**961**	**25.1**	**961**	22.8	196	9.5	144	5.7	**961**	**22.7**	**961**	22.1	529	13.4	484	14.9	**961**	**24.8**	676	16.6

Table 3: Scale and SCov scores of the scaling behavior evaluation. Bold scores are the best for each architecture, and highlighted scores are the best overall.

	R-GNN						LAMA-first	
	V		$\Omega^{\text{Exp.}}$		$\Omega^{\text{Heu.}}$			
Domain	Cov	Len	Cov	Len	Cov	Len	Cov	Len
Blocksworld	47.8	90.0	**78.9**	233.1	**78.9**	232.1	56.7	288.7
Childsnack	5.6	41.2	51.1	41.4	**54.4**	41.5	35.2	41.8
Ferry	54.4	73.7	**67.8**	119.6	**67.8**	119.6	**67.8**	150.3
Floortile	0.0	—	36.7	93.5	**37.8**	109.3	16.9	51.0
Rovers	24.4	49.0	36.7	66.2	37.8	80.3	**57.8**	171.0
Satellite	32.2	36.9	64.4	71.2	66.7	68.2	**85.6**	104.4
Transport	24.4	69.6	70.0	91.3	**74.4**	96.8	64.4	70.7
Average	27.0	60.1	57.9	102.3	**59.7**	106.8	54.9	125.4

Table 4: Coverage in percent (Cov) and average plan length (Len) on IPC'23 test sets. Bold scores are the best coverages for each domain.

generalize (2025). For a given instance size n, the evaluation runs a policy on uniformly generated instances until the estimate of the average coverage $\hat{C}_n$ for size n is within a 10% confidence interval with 90% probability. This process repeats with an increasing size n until the coverage $\hat{C}_n$ drops below 30% or the maximum size of 100 (1000 for Visitall) is reached. Policy runs have a step limit of $100 + n$ that increases with the instance size n. The evaluation results are reported using two summary scores: *Scale* is the largest instance size n before termination, which measures how far the policy generalizes, and *SCov* is the sum of coverages $\hat{C}_n$ per size n, which measures overall performance.

Table 3 shows the summary scores for the scaling behavior evaluation of all policies. We observe the following trends across all architectures: The vanilla Q-value policies (Q) perform the worst overall. This is due to failing to distinguish teacher and non-teacher actions, as shown in Section 3. Importantly, we see that the state-value policies (V) are consistently outperformed by the regularized Q-value policies ($\Omega^{\text{Exp.}}$ & $\Omega^{\text{Heu.}}$). This shows that the additional learning signals provided by regularizing the Q-values of non-teacher actions lead to vastly improved generalization. Lastly, we see that the policies trained using the heuristic regularizer ($\Omega^{\text{Heu.}}$) outperform the explicit regularizer policies ($\Omega^{\text{Exp.}}$) on some domains. This shows that providing tighter lower bounds for the Q-values of non-teacher actions can further improve generalization.

IPC test sets. We additionally evaluate our policies on the IPC'23 test sets, which provides insights into their performance on particularly difficult instances. We consider only the R-GNN architecture, as it showed the overall best performance in the scaling behavior evaluation, and we omit the vanilla Q-value policies due to their poor performance. We ensured that the training and test sets do not overlap. All policies are executed on an AMD EPYC 9454 CPU with time and memory limits of 1 minute and 8 GB. As a baseline, we run the LAMA-first planner (Richter and Westphal 2010) using the same hardware and limits.

Consider Table 4, which shows the coverage and average plan length for the 7 considered IPC'23 domains. As before, our regularized Q-value policies consistently achieve higher coverages than the state-value policies. Impressively, the coverages and plan lengths of the regularized Q-value policies are competitive with those of LAMA-first.

5 Conclusion

Learning per-domain generalizing policies is becoming increasingly popular, and supervised learning provides a natural training framework as we can leverage optimal planners to generate rich training data. In this work, we have shown that the common approach of learning state-value functions yields policies whose runtime scales poorly with instance size. The more efficient approach of learning Q-value functions, however, fails to learn the distinction between teacher and non-teacher actions. Hence, we introduce regularization terms that use optimal and admissible heuristics to enforce the Q-values of non-teacher actions to be larger than those of teacher actions. Experiments on 10 domains using three GNN architectures show that our regularized Q-value policies are not only more efficient but also achieve vastly higher generalization than state-value policies.

A promising next step for our work would be to continue the policy training using reinforcement learning on dynamically generated instances. These instances could be generated as needed, e.g., we could gradually increase their size to improve scaling or focus on instances where the policy previously failed.

Acknowledgements

This work was partially supported by the German Federal Ministry of Education and Research (BMBF) as part of the project MAC-MERLin (Grant Agreement No. 01IW24007), by the German Research Foundation (DFG) - GRK 2853/1 "Neuroexplicit Models of Language, Vision, and Action" - project number 471607914, and by the European Regional Development Fund (ERDF) and the Saarland within the scope of (To)CERTAIN.

References

Chen, D. Z.; Hao, M.; Thiébaux, S.; and Trevizan, F. 2025a. Graph Learning for Planning: The Story Thus Far and Open Challenges. In *AAAI Workshop on Generalization in Planning*.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. 2024. Learning Domain-Independent Heuristics for Grounded and Lifted Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20078–20086.

Chen, D. Z.; Zenn, J.; Cinquin, T.; and McIlraith, S. A. 2025b. Language models for PDDL planning: Generating sound and programmatic policies. In *Eighteenth European Workshop on Reinforcement Learning*.

Gros, T. P.; Müller, N. J.; Fišer, D.; Valera, I.; Wolf, V.; and Hoffmann, J. 2025. Per-domain generalizing policies: On validation instances and scaling behavior. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, 198–203.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, 162–169.

Horčík, R.; and Šír, G. 2024. Expressiveness of graph neural networks in planning domains. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 281–289.

Müller, N. J.; Oster, M.; Valera, I.; Hoffmann, J.; and Gros, T. P. 2026. Per-Domain Generalizing Policies: On Learning Efficient and Robust Q-Value Functions (Extended Version with Technical Appendix). arXiv:2603.17544.

Richter, S.; and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Rossetti, N.; Tummolo, M.; Gerevini, A. E.; Putelli, L.; Serina, I.; Chiari, M.; and Olivato, M. 2024. Learning General Policies for Planning through GPT Models. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 500–508.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, 593–607.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022a. 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. 2022b. Learning Generalized Policies without Supervision Using GNNs. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 474–483.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2023. Learning general policies with policy gradient methods. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 647–657.

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

Ståhlberg, S.; and Geffner, H. 2025. First-Order Representation Languages for Goal-Conditioned RL. *CoRR*, abs/2512.19355.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

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

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 International Conference on Automated Planning and Scheduling*, volume 34, 633–642.