Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2025)

Learning Efficiency Meets Symmetry Breaking

Yingbin Bai[1], **Sylvie Thiébaux**[1, 2], **Felipe Trevizan**[1]

[1]School of Computing, The Australian National University
[2]LAAS-CNRS, Université de Toulouse
yingbin.bai@anu.edu.au, sylvie.thiebaux@anu.edu.au, felipe.trevizan@anu.edu.au

Abstract

Learning-based planners leveraging Graph Neural Networks can learn search guidance applicable to large search spaces, yet their potential to address symmetries remains largely unexplored. In this paper, we introduce a graph representation of planning problems allying learning efficiency with the ability to detect symmetries, along with two pruning methods, action pruning and state pruning, designed to manage symmetries during search. The integration of these techniques into Fast Downward achieves a first-time success over LAMA on the latest IPC learning track dataset.

Extended version — https://arxiv.org/abs/2504.19738
Code — https://github.com/bybeye/Distincter

Introduction

Over the past two decades, heuristic search has achieved significant success across a variety of planning problems, and has become the standard approach in the field (Richter and Westphal 2010; Höller et al. 2020; Corrêa et al. 2022; Geißer et al. 2022; Klößner, Seipp, and Steinmetz 2023). Nevertheless, even in classical planning, scalability remains a significant challenge for these methods. This has led a growing number of researchers to turn to learning-based methods, particularly using Graph Neural Networks (GNNs) (Toyer et al. 2020; Shen, Trevizan, and Thiébaux 2020; Ståhlberg, Bonet, and Geffner 2022; Chen, Trevizan, and Thiébaux 2024; Horcík and Sír 2024; Hao et al. 2024; Drexler et al. 2024). Unlike traditional model-based methods, which are reliant solely on analysing planning domain and problem definitions, GNNs are capable of learning patterns and strategies from existing plans to enhance search efficiency and adaptability.

However, learning efficiency alone is insufficient to address the challenges inherent in large-scale planning, which often involves a substantial number of symmetrical states (Wehrle et al. 2015; Sievers et al. 2019). Although these states do not affect plan quality, they consume significant computational resources and can considerably slow down the search process. In this paper, we use NNs and GNNs with permutation invariant activation functions to learn a permutation invariant function allowing them to produce consistent outputs for symmetrical inputs. Despite this advantage, the full potential of this feature has not yet been effectively harnessed to detect and break symmetries during the search process.

In this paper, we remedy this by introducing a graph representation designed to achieve two key objectives: learning efficiency and symmetry reduction. Leveraging the strengths of this representation, we propose two pruning methodologies: action pruning and state pruning. Action pruning infers symmetries by analyzing object involvement in action parameters, without generating child states nor computing their heuristic value. Additionally, since GNNs can retain invariant outputs for symmetrical inputs, state pruning exploits this property to efficiently identify symmetries between states.

To evaluate the proposed techniques, we implemented them on top of Fast Downward (Helmert 2006) in a planner called Distincter and carried out experiments on the 2023 International Planning Competition Learning Track. The overall coverage of Distincter surpasses that of the traditional SOTA method, LAMA (Richter and Westphal 2010), for the first time in the recent literature on learning planning heuristics, marking a significant milestone for learning-based methods.

In terms of related work, recent independent work by (Drexler et al. 2024) removes symmetries in the training set in offline mode, thereby improving training effectiveness. In contrast, our approach focuses on removing symmetries during the search process, so as to enhance search efficiency and scale to large planning problems.

Background and Notation

A lifted planning problem is defined as a tuple $\Pi = \langle \mathcal{O}, \mathcal{T}, \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{O}$ denotes a set of objects, $\mathcal{T}$ is a set of object types, $\mathcal{P}$ consists of first-order predicates, $\mathcal{A}$ comprises action schemas, $\mathcal{I}$ specifies the current (or initial) state, and $\mathcal{G}$ delineates the goal.

A predicate $p \in \mathcal{P}$ has parameters $x_{p_1}, \ldots, x_{p_n}$ for $p_n \in \mathbb{N}$, where each parameter requires a specific type of object. A predicate can be instantiated by assigning each x_i to an object from $\mathcal{O}$, resulting in a proposition ρ. A state is an assignment of truth value to the propositions.

An action schema $a = \langle X_a, \mathit{pre}(a), \mathit{add}(a), \mathit{del}(a) \rangle$ is

defined as a tuple comprising a list of typed parameters $X_a = (x_{a_1}, \dots x_{a_n})$, along with sets of preconditions, add effects, and delete effects, all of which are predicates in $\mathcal{P}$ with parameters from X_a. When all parameters of an action schema are instantiated with objects of the required types, the action is referred to as a ground action. A ground action a is applicable in a state s if $pre(a) \subseteq s$. When a is applied to s, the resulting state s' is given by $(s \setminus del(a)) \cup add(a)$. In this context, the state s is referred to as the parent state, and s' is known as the child state. Since the set of applicable actions for a parent state is typically not a singleton, expanding a parent state usually generates a set of child states.

A sequence of actions $a_1, \dots, a_n$ is applicable in a state s if there exists a sequence of states $s_0, \dots, s_n$ such that $s_0 = s$, and for each $i \in \{1, \dots, n\}$, the state s_i is the result of applying a_i in s_{i-1}. The aim is to find a plan for a given planning problem Π, which is a sequence of ground actions that is applicable in the initial state $\mathcal{I}$ and results in a state s_n such that $\mathcal{G} \subseteq s_n$.

A colored (or labelled) graph is a tuple $G = \langle V, E, c, l \rangle$ where V is the set of vertices, E is the set of edges, and c (resp. l) maps vertices (resp. edges) to their color. Two graphs $G = \langle V, E, c, l \rangle$ and $G' = \langle V', E', c', l' \rangle$ are isomorphic, denoted by $G \cong G'$, if there exists a bijection $\tau : V \to V'$ such that $(u, v) \in E$ iff $(\tau(u), \tau(v)) \in E'$, $c'(\tau(v)) = c(v)$ for all $v \in V$, and $l'((\tau(u), \tau(v))) = l((u, v))$ for all $(u, v) \in E$.

An automorphism of G is defined as an equivalence relation σ representing an isomorphism between G and itself. The set of all automorphisms of G forms a group under the operation of composition, known as the automorphism group $\mathrm{Aut}(G)$ of the graph. The orbit of a vertex v in a graph consists of all vertices that can be transformed into v by any automorphism in $\mathrm{Aut}(G)$. This implies that any two vertices within the same orbit are structurally equivalent in the graph, maintaining the same connections and roles relative to other vertices and edges.

Distincter

Typed Instance Learning Graph (TILG)

Our graph representation extends the Instance Learning Graph (ILG) (Chen, Trevizan, and Thiébaux 2024), maintaining similar structures but offering additional information for learning and symmetry detection. The graph's vertices represent objects and propositions in the initial (current) state and the goal, and edges exist between propositions and the objects in their parameter list.[1] Vertex features capture the object types, the predicates instanciated by the propositions, and whether goal propositions have been achieved. Edge features capture the index of objects in proposition parameter lists. Formally:

Definition 1 *Let $\Pi = \langle \mathcal{O}, \mathcal{T}, \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ represent a lifted planning problem. The typed instance learning graph (TILG) for Π is the undirected graph $G_\Pi = \langle V, E, f, l \rangle$, such that:*

- $V = \mathcal{O} \cup \mathcal{I} \cup \mathcal{G}$
- $E = \{(o, p(o_1, ..., o_n)) | o \in \mathcal{O}, \exists i \; o = o_i, p(o_1, ..., o_n) \in \mathcal{I} \cup \mathcal{G}\}$
- $c : V \to \{(status, class) \mid status \in \{0, 1, 2, 3\}, class \in \mathcal{T} \cup \mathcal{P}\}$*, maps each vertex to a tuple where:*
  - *$status$ indicates the goal status of propositions: 0 for non-goal propositions in $\mathcal{I} \setminus \mathcal{G}$, 1 for unachieved goal propositions in $\mathcal{G} \setminus \mathcal{I}$, and 2 for achieved goal propositions in $\mathcal{I} \cap \mathcal{G}$. $status = 3$ for object vertices.*
  - *$class$ refers to the object type for object vertices, and for proposition vertices, it denotes the predicate of which the proposition is an instance.*
- $l : E \to \mathbb{N}$*, where for each edge $e \in E$, $l(e)$ indicates the index of the object in the proposition parameters.*

In ILG, the object type information is absent, whereas TILG embeds it within each object vertex. This may seem minor, but it adds valuable information to each object vertex significantly enriching the information available. Moreover, Fast Downward omits static propositions during search, which causes them to be missing in existing ILG implementations as well. While this omission does not affect traditional heuristic methods, it significantly impacts learning methods, which estimate heuristics based on the graph. Without static propositions, crucial information is lost, leading to blind guesses for some actions. For instance, "waiting" propositions in the childsnack domain are static, and without this information, planners are unable to determine which table the tray should be moved to. Therefore, TILG includes static propositions.

In the following, all elements of the problem Π are fixed, except for the current state s. **We shall therefore identify Π with s and will refer to the TILG G_π as G_s.**

Action Pruning

With Greedy Best-First Search (GBFS), planners select the state with the smallest heuristic value to expand, which requires computing the heuristic value of all child states. When child states contain a large number of symmetries, this can result in significant time wasted on redundant calculations.

Shallow pruning was designed to address this challenge (Pochter, Zohar, and Rosenschein 2011). However, the problem description graph (PDG) used in shallow pruning requires instantiating all predicates and action schemas within the graph, resulting in significant computational overhead for each state. To improve efficiency, we introduce Action Pruning, which replaces the PDG with TILG. A key innovation of action pruning is its ability to **infer symmetrical child states from the parent state**, eliminating the need for action preconditions and effects in the graph. By leveraging the much more compact TILG representation and its inference capability, action pruning enables faster automorphism calculations.

Definition 2 *(Object Tuples Equivalence) Let $\langle A_1, \dots, A_n \rangle$ and $\langle B_1, \dots, B_n \rangle$ be two tuples of objects s.t. A_i and B_i in $\mathcal{O}$ with corresponding vertices u_i and v_i in the TILG G_s. We say that $\langle A_1, \dots, A_n \rangle$ is equivalent to $\langle B_1, \dots, B_n \rangle$ in s, denoted as $\langle A_1, \dots, A_n \rangle \simeq \langle B_1, \dots, B_n \rangle$, iff there exist an*

[1] In the following, we will use the word *symmetric* to refer to states represented by isomorphic TILG and to objects or propositions that are related via τ (or σ depending on the context).

Algorithm 1: Action pruning algorithm

Input: Planning problem with current state s
Input: Set A_s of actions applicable in s
Output: Pruned action set $A_p \subseteq A_s$

$K \leftarrow \emptyset$, $A_p \leftarrow \emptyset$
Graph $G_s \leftarrow$ TILG(s) *with encoding in Eq.* 1
Orbits $O_s \leftarrow$ Nauty(G_s)
for a **in** A_s **do**
  $K_a \leftarrow$ Replace_params_with_orbits(a, O_s)
  if K_a **not in** K **then**
    $K \leftarrow K \cup \{K_a\}$
    $A_p \leftarrow A_p \cup \{a\}$
  end if
end for
return A_p

automorphism of G_s represented by the bijective function σ s.t. $\sigma(u_i) = v_i$ for all $i \in \{1, \ldots, n\}$.

Theorem 1 *Let A_i and B_i be objects in $\mathcal{O}$ for $i \in \{1, \ldots, n\}$ with $A_i \neq A_j$ and $B_i \neq B_j$ for all $i \neq j$. Let an action schema $\alpha \in \mathcal{A}$, and consider two ground actions, $a = \alpha(A_1, A_2, \ldots, A_n)$ and $b = \alpha(B_1, B_2, \ldots, B_n)$, applicable in a state s, resulting in successor states s_a and s_b respectively. If $\langle A_1, \ldots, A_n \rangle \simeq \langle B_1, \ldots, B_n \rangle$ in s then the TILGs G_{s_a} and G_{s_b} are isomorphic, i.e., $G_{s_a} \cong G_{s_b}$.*

The proof of Theorem 1 is in the supplementary material (Bai, Thiebaux, and Trevizan 2025).

Unfortunately, identifying all isomorphic successor states in order to prune actions requires testing an exponential number of tuples against the automorphisms of G_s. Therefore, we resort to a simpler method that over-approximates the set of equivalent tuples and isomorphic successor states, and consequently does not preserve the completeness of the search process. It relaxes the conditions of the theorem by checking the equivalence of all individual pairs A_i and B_i in s, i.e., the condition that $\langle A_1, \ldots, A_n \rangle \simeq \langle B_1, \ldots, B_n \rangle$ is replaced with $\langle A_i \rangle \simeq \langle B_i \rangle \; \forall i$, or in other terms that A_i and B_i are in the same orbit of G_s. In our experiments, we find that this over-approximation yields good results in practice. Moreover, we did not observe any failure due to incompleteness, and therefore do not currently employ any fallback mechanism.

This process of action pruning is outlined in Algorithm 1. First, the planning problem with current state s is converted into a TILG G_s. The Nauty library (McKay and Piperno 2014) is then utilized to compute the orbits O_s of G_s. Since Nauty lacks support for feature vectors, we aggregate vertex features into an unique vertex color to detect automorphisms. This color-coding strategy is detailed in Equation 1:

$$\text{color} = \sum_{i=1}^{N} 10^{\beta_i} \times F_i \quad \text{with} \qquad \beta_i = \begin{cases} \sum_{n=1}^{i-1} \lceil \log_{10} M_n \rceil & i \geq 2 \\ 0 & i = 1 \end{cases}, \tag{1}$$

where N denotes the number of features, F_i represents the value of feature i, and M_n is the maximum possible value of feature i.

After obtaining the orbits O_s of G_s, the parameters of each applicable action a are substituted with their respective orbit IDs, generating a unique hash key K_a. This hash key is subsequently used to identify and eliminate symmetric actions, ensuring that only distinct actions are retained in A_p for further processing.

State Pruning

Symmetries arise not only between child states but also across states from different parents. Many state pruning approaches have been proposed and proven useful in classical planning (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012). However, the main issue limiting their widespread use in planning problems is their high computational cost. To address this issue, we propose a novel method that performs state pruning with negligible additional overhead. Specifically, building on the permutation invariance property of GNNs, we use the embeddings from the second-to-last layer of the network as hash keys to efficiently detect and eliminate symmetries across states.

The idea of using neural network outputs to check similarity is not new, having been employed in Siamese networks since early work in deep learning (Bromley et al. 1993). These identical architecture, weight-shared networks are specifically designed to assess and compare the similarity between two inputs. This approach has proven effective across various fields, including fingerprint identification (Li et al. 2021) and anomaly detection (Zhou et al. 2021). For GNNs, Chen et al. (2019) highlight the equivalence between graph isomorphism testing and approximating permutation-invariant functions. Moreover, standard GNNs have been shown to possess an expressive power, comparable to that of the 1-LWL test (Xu et al. 2019). While this implies GNNs may be unable to distinguish some non-isomorphic graphs, compromising the completeness of the search when state pruning is used, our results demonstrate that GNNs based on TILG can be highly effective in both heuristic prediction and state pruning.

The TILG G_{s_i} for the current state s_i is fed through a graph network ϕ_θ to encode an embedding z_i. Subsequently, z_i is processed by a fully connected linear layer φ_θ to generate a heuristic value $\hat{h}_i \in \mathbb{R}$. This process is represented by $z_i = \phi_\theta(G_{s_i})$ and $\hat{h}_i = \varphi_\theta(z_i)$.

Next, z_i is rounded up and encoded using MD5 to shorten its length, serving as a key in a hash map for state matching. Since z_i is efficiently captured during the network's forward pass, there is no need to generate keys through computationally expensive methods like calculating isomorphisms in PDG (Pochter, Zohar, and Rosenschein 2011), resulting in minimal additional cost for state pruning.

Experiments

Datasets. We evaluate our framework, Distincter, on the 2023 International Planning Competition Learning Track (Seipp and Segovia-Aguas 2023), which includes ten domains. In ablation experiments, to assess the effectiveness

Domain	h^{FF}	LAMA	GOOSE	OptRank	GPR	Distincter
blocksworld	28	61	61±10	44±11	69	**88±4**
childsnack	26	34	16±4	32±1	20	**64±5**
ferry	71	70	70±0	64±4	82	**83±1**
floortile	**10**	**10**	1±0	1±0	2	2±0
miconic	**90**	**90**	89±1	88±4	**90**	**90±0**
rovers	29	**70**	28±1	31±2	36	42±2
satellite	64	**90**	29±2	29±3	39	48±17
sokoban	36	**40**	34±0	32±1	38	32±2
spanner	30	30	39±16	65±0	74	**90±0**
transport	41	**68**	37±4	42±5	28	50±3
Sum	425	563	405	429	478	**589**

Table 1: Coverage comparison with SOTA methods on the 2023 International Planning Competition Learning Track.

Domain	LAMA	Distincter
blocksworld	390	**198±10**
childsnack	45	**34±3**
ferry	257	**206±2**
floortile	34	**32±0**
miconic	324	**273±8**
rovers	**72**	106±11
satellite	**18**	27±8
sokoban	**46**	49±14
spanner	**14**	16±0
transport	49	**45±3**
Sum	1249	**987**

Table 2: Average plan lengths over problems solved by both LAMA and Distincter.

of two proposed pruning methods, we further employ six domains that contain a lot of symmetrical states.

Network Structure. Our graph network consists of RGCN layers with a hidden dimension of 64 (Schlichtkrull et al. 2018), followed by global add pooling, and a linear layer producing a one-dimensional output. The network is implemented using the standard PyTorch Geometric package (Fey and Lenssen 2019). For further setting information, please see the supplementary material (Bai, Thiebaux, and Trevizan 2025).

Training and Evaluation. For each domain, we train a GNN using the RMSE loss function for 30 epochs, including 10 warm-up epochs. We use an initial learning rate of 10^{-3} and apply cosine annealing(Loshchilov and Hutter 2017) over a single cycle, with a momentum value of 0.9. To adapt to varying numbers of examples (N) across different domains, we set the number of iterations to 100 per epoch and adjust the batch size accordingly, using $\frac{N}{100}$.

For evaluation, our planner is based on Fast Downward using eager-GBFS (Helmert 2006) guided by GNN heuristic values. Upon completing the training, the model is saved in JIT format and executed using C++. To ensure result stability and mitigate dataset bias, we employ early stopping on a validation set to select optimal models (Bai et al. 2023). All experiments are conducted on a single CPU core with an NVIDIA A6000 GPU and 8GB of memory, with a 30 minute timeout per problem. The mean and standard deviation are computed from three trials.

Domain	None	Action	State	Distincter
blocksworld	79±11	79±7	88±3	88±4
childsnack	34±4	63±3	61±1	64±5
ferry	82±0	82±0	83±1	83±1
floortile	2±1	2±0	2±0	2±0
miconic	90±0	90±0	90±0	90±0
rovers	41±2	41±3	41±2	42±2
satellite	45±13	46±13	47±17	48±17
sokoban	32±2	32±2	32±2	32±2
spanner	83±0	90±0	83±0	90±0
transport	42±1	42±1	49±2	50±3
gripper	24±9	75±4	90±0	90±0
grippers	62±5	89±1	85±1	90±1
logistics	19±9	36±4	53±4	52±3
movie	90±0	74±17	90±0	75±17
tsp	76±24	78±0	90±0	90±0
tyreworld	0±0	1±1	64±20	65±21
Sum	803	920	1048	1051

Table 3: Ablation study. "None" refers to GBFS + GNN heuristic without pruning, "Action" denotes the use of action pruning, "State" represents the use of state pruning.

Results

We compare Distincter with SOTA baselines, including both traditional heuristic search methods, namely LAMA (Richter and Westphal 2010) and GBFS with h^{FF} (Hoffmann and Nebel 2001), and GBFS with learnt heuristics using GOOSE (Chen, Thiébaux, and Trevizan 2024), its optimal ranking counterpart (Hao et al. 2024) and Gaussian Process Regression (GPR) from WL features (Chen, Trevizan, and Thiébaux 2024). Note that, as shown in (Hao et al. 2024), other methods such as STRIPS-HGN (Shen, Trevizan, and Thiébaux 2020) and Perfrank (Chrestien et al. 2023), are dominated by our baselines. All baselines are run on the same hardware and with the same computational requirements as Distincter. In terms of coverage, Distincter matches or surpasses all baselines across five domains. Notably, when compared to the strongest learning baseline, GPR (Chen, Trevizan, and Thiébaux 2024), Distincter achieves parity or superiority in eight of the ten domains. Additionally, the total coverage of Distincter surpasses that of model-based methods: it exceeds that of h^{FF} by a substantial margin of 164 and that of LAMA by 26. In Table 2, we report the average plan length over the problems successfully solved by both approaches. The results suggest a correlation between plan length and coverage.

In keeping with the 8GB memory requirement of the IPC learning track, we found it is insufficient for the Fast Downward translator to ground some large problems, resulting in performance loss as shown in Table 1. For instance, in the

"ferry" domain, when sufficient memory is available, Distincter can solve all 90 test problems.

Although Distincter exhibits very good performance across many domains, it struggles in others – see e.g. its low performance on "Floortile" and "Sokoban", and the large deviations observed in the "Satellite" domain. The key issue with Floortile and Sokoban is that they require path-finding and geometrical reasoning which cannot be achieved with the limited receptive field of ordinary GNNs. Dead ends in these domains are another issue, as they cannot always be captured when training with optimal plans only. In Satellite, due to the the lack of static propositions in its graph, GOOSE learns a simple strategy that works in simple problems only. On the other hand, thanks to statics being included, Distincter is able to learn to solve much larger problems. However, standard GNNs are not expressive enough to learn to distinguish all non-isomorphic Satellite states (Drexler et al. 2024), and therefore the learning procedure fails every once in a while, leading to a large variance.

Ablation Study

To assess the effectiveness of our proposed pruning techniques, we performed ablation experiments across four configurations. From Table 3, we observe that in domains with a high degree of symmetries, such as "spanner" and "childsnack", action pruning offers substantial benefits. However, in the "movie" domain, which contains a large number of redundant objects, action pruning consumes excessive time computing symmetries. In contrast, state pruning improves performance in seven domains, demonstrating its broader utility. By combining both techniques, we can leverage their strengths to achieve better overall outcomes.

Conclusion

We introduced TILG, a novel graphical representation that captures key problem-solving features and is designed for combining efficiency with symmetry detection. Leveraging the properties of TILG, we proposed two efficient pruning techniques that are suitable for large-scale planning problems. Our framework, Distincter, achieved a historic milestone by outperforming the LAMA framework on the learning track of the 2023 International Planning Competition.

In addition, both pruning methods are applicable to traditional model-based approaches. Although state pruning with GNNs can be computationally expensive, small dedicated GNNs can mitigate this issue.

Acknowledgments

The authors thank Sandra Kiefer and Brendan McKay for useful discussions. This work was supported by the Australian Research Council grant DP220103815, by the Artificial and Natural Intelligence Toulouse Institute (ANITI) under the grant agreement ANR-23-IACL-0002, and by the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No. 101070149.

References

Bai, Y.; Han, Z.; Yang, E.; Yu, J.; Han, B.; Wang, D.; and Liu, T. 2023. Subclass-Dominant Label Noise: A Counterexample for the Success of Early Stopping. In *NeurIPS*.

Bai, Y.; Thiebaux, S.; and Trevizan, F. 2025. Learning Efficiency Meets Symmetry Breaking. arXiv:2504.19738.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1993. Signature Verification Using a Siamese Time Delay Neural Network. In *NeurIPS*, 737–744.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. W. 2024. Learning Domain-Independent Heuristics for Grounded and Lifted Planning. In *AAAI*, 20078–20086.

Chen, D. Z.; Trevizan, F. W.; and Thiébaux, S. 2024. Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning. In *ICAPS*, 68–76.

Chen, Z.; Villar, S.; Chen, L.; and Bruna, J. 2019. On the equivalence between graph isomorphism testing and function approximation with GNNs. In *NeurIPS*, 15868–15876.

Chrestien, L.; Edelkamp, S.; Komenda, A.; and Pevný, T. 2023. Optimize Planning Heuristics to Rank, not to Estimate Cost-to-Goal. In *NeurIPS*.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2022. The FF Heuristic for Lifted Classical Planning. In *AAAI*, 9716–9723.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced Symmetry Breaking in Cost-Optimal Planning as Forward Search. In *ICAPS*, 343–347.

Drexler, D.; Ståhlberg, S.; Bonet, B.; and Geffner, H. 2024. Symmetries and Expressive Requirements for Learning General Policies. In *KR*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. *CoRR*, abs/1903.02428.

Geißer, F.; Haslum, P.; Thiébaux, S.; and Trevizan, F. W. 2022. Admissible Heuristics for Multi-Objective Planning. In *ICAPS*, 100–109.

Hao, M.; Trevizan, F. W.; Thiébaux, S.; Ferber, P.; and Hoffmann, J. 2024. Guiding GBFS through Learned Pairwise Rankings. In *IJCAI*, 6724–6732.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J. Artif. Intell. Res.*, 14: 253–302.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN Planning as Heuristic Progression Search. *J. Artif. Intell. Res.*, 67: 835–880.

Horcík, R.; and Sír, G. 2024. Expressiveness of Graph Neural Networks in Planning Domains. In *ICAPS*, 281–289.

Klößner, T.; Seipp, J.; and Steinmetz, M. 2023. Cartesian Abstractions and Saturated Cost Partitioning in Probabilistic Planning. In *ECAI*, 1272–1279.

Li, Q.; Liao, X.; Liu, M.; and Valaee, S. 2021. Indoor Localization Based on CSI Fingerprint by Siamese Convolution Neural Network. *IEEE Trans. Veh. Technol.*, 70(11): 12168–12173.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*.

McKay, B. D.; and Piperno, A. 2014. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60: 94–112.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting Problem Symmetries in State-Based Planners. In *AAAI*, 1004–1009.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Schlichtkrull, M. S.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*, volume 10843, 593–607.

Seipp, J.; and Segovia-Aguas, J. 2023. Int. Planning Competition 2023 - Learning Track.

Shen, W.; Trevizan, F. W.; and Thiébaux, S. 2020. Learning Domain-Independent Planning Heuristics with Hypergraph Networks. In *ICAPS*, 574–584.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2019. Theoretical Foundations for Structural Symmetries of Lifted PDDL Tasks. In *ICAPS*, 446–454.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022. Learning General Optimal Policies with Graph Neural Networks: Expressive Power, Transparency, and Limits. In *ICAPS*, 629–637.

Toyer, S.; Thiébaux, S.; Trevizan, F.; and Xie, L. 2020. Asnets: Deep learning for generalised planning. *J. Artif. Intell. Res.*, 68: 1–68.

Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Integrating Partial Order Reduction and Symmetry Elimination for Cost-Optimal Classical Planning. In *IJCAI*, 1712–1718.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Zhou, X.; Liang, W.; Shimizu, S.; Ma, J.; and Jin, Q. 2021. Siamese Neural Network Based Few-Shot Learning for Anomaly Detection in Industrial Cyber-Physical Systems. *IEEE Trans. Ind. Informatics*, 17(8): 5790–5798.