Make Use of Your Search Effort: Data Augmentation for Learning Effective Planning Heuristics from Optimal Plans

Shanhe Zhong, Eldan Cohen

Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada
jamessh.zhong@mail.utoronto.ca, eldan.cohen@utoronto.ca

Abstract

Learning planning heuristics to guide search has proven to be an effective way to improve satisficing planning with data, while preserving the soundness and completeness guarantees of the planner. Prior work has shown that learning from optimal plans can yield more effective heuristics, since optimal plans provide high-quality supervision about promising search directions. However, generating training data is computationally expensive, as it requires solving challenging planning tasks optimally. We therefore propose *search-free* data augmentation methods that can substantially increase the size of training data from existing optimal plans and corresponding search traces, without solving any additional planning problems. In particular, we consider three complementary data augmentations with optimality guarantees on the augmented data: (i) *Intermediate Goals*, which converts subplans of an optimal plan into new optimal plans to alternative, intermediate goal states; (ii) *Closed-List Trajectories*, which leverages the fact that under $A^\star$ with a consistent heuristic each expanded state is reached by an optimal path; and (iii) *Partial-Order Plan*, which samples multiple linearizations from the underlying POP, which are also optimal plans under state-independent action costs. Experiments on the IPC 2023 Learning Track demonstrate that our data augmentations significantly improve planning performance across many domains for two state-of-the-art frameworks, GOOSE (WLF+ranking) and Distincter (GNN+regression). For the first time, our data augmentation techniques enable GOOSE to surpass the coverage achieved by the first iteration of the LAMA planner.

1 Introduction

Automated planning provides a model-based framework for solving sequential decision-making tasks by automatically generating action sequences from declarative models of actions and goals (Geffner and Bonet 2013). Over the past decades, heuristic search has been enormously successful in various planning domains and has emerged as the predominant approach to solving planning problems (Bonet and Geffner 2001). Despite its empirical success, heuristic search still faces scalability challenges due to the inherent computational complexity of planning (Bäckström and Nebel 1995). In many planning applications, such as logistics (Sousa and Tavares 2013; Pedersen and Krüger 2015) and robotics (Barrios et al. 2011; Karpas and Magazzeni 2020), planning problems are routinely generated and solved under similar settings. This repeated process yields valuable by-products, such as search traces and plans, and naturally motivates learning from experience to amortize prior computation and improve planning efficiency over time.

Alongside recent advances in machine learning, *learning-to-plan* has attracted significant attention. Existing approaches broadly fall into two categories: learning policies for plan generation (Shen et al. 2019; Groshev et al. 2018; Garg, Bajpai et al. 2019; Silver et al. 2024), and learning planning heuristics to guide search (Garrett, Kaelbling, and Lozano-Pérez 2016; Shen, Trevizan, and Thiébaux 2020; Ferber, Helmert, and Hoffmann 2020; Chrestien et al. 2023; Chen, Thiébaux, and Trevizan 2024; Chen and Thiébaux 2024; Chen, Trevizan, and Thiébaux 2024). Among these methods, learning planning heuristics from optimal plans has emerged as an effective approach for satisficing planning (Chrestien et al. 2023; Chen, Thiébaux, and Trevizan 2024; Chen and Thiébaux 2024; Chen, Trevizan, and Thiébaux 2024; Bai, Thiébaux, and Trevizan 2025). In particular, optimal plans can provide accurate signals of promising search directions to train more effective heuristics, and search algorithms can offer soundness and completeness guarantees, which are beneficial for satisficing planning.

At the same time, successes in deep learning have also shown that training with more high-quality data can lead to more capable models (Halevy, Norvig, and Pereira 2009; Sun et al. 2017; Hestness et al. 2017). However, obtaining high-quality data (i.e., optimal plans) for learning-to-plan is often costly. In particular, because many planning problems are PSPACE-complete (Bylander 1992; Bäckström and Nebel 1995), generating data by solving such problems to optimality can be computationally expensive. Once a plan is found, the search trace (i.e., the closed list) is often discarded. In this work, we investigate *search-free* data augmentation techniques to substantially enrich and diversify the training data from existing optimal plans and their corresponding search traces, thereby making better use of the costly search effort.

Concretely, we consider three search-free data augmentation techniques compatible with both ranking-based and

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

regression-based objective functions for learning heuristics, which (i) exploit *intermediate goals* along an optimal trajectory, (ii) treat *closed-list trajectories* as additional plans generated by $A^\star$ with a consistent heuristic, and (iii) deorder the original plan into its underlying *partial-order plan* to generate additional linearizations. In addition, we show that all three techniques have formal guarantees that the augmented plans (and the induced $h^\star$ labels) remain optimal, which enables learning more effective heuristics for satisficing planning. We also empirically validate the proposed augmentation techniques on the IPC 2023 Learning Track using two state-of-the-art heuristic learning frameworks: GOOSE (Chen, Trevizan, and Thiébaux 2024) and Distincter (Bai, Thiébaux, and Trevizan 2025), showing significant improvements in coverage and, in some cases, additional gains when combining augmentations. Further, we show that our data augmentation techniques enable GOOSE, for the first time, to outperform LAMA (Richter and Westphal 2010), in terms of coverage.

2 Background

2.1 Classical Planning

A classical planning task (Geffner and Bonet 2013) is characterized by a state transition system $\langle S, A, s_0, G\rangle$, where S is the set of *states*, A is the set of *actions*, $s_0 \in S$ is the *initial state*, and $G \subseteq S$ is a set of goal states. Typically, G is specified as a *partial state* that assigns values only to the variables relevant to the goal, so any full state s_g consistent with those assignments is a goal state. An action $a \in A$ is defined as a state-transition function $a : S \to S \cup \bot$, where for a state $s \in S$, $a(s) = \bot$ indicates action a is not *applicable* at state s. Whenever $a(s) \neq \bot$, we call $a(s) \in S$ the successor state of s given action a. An action sequence $\pi = \langle a_1, ..., a_k\rangle$ is applicable in s if there exist states $s_0, s_1, \ldots, s_k$ such that (i) $s_0 = s$, and (ii) for each $1 \leq i \leq k$, a_i is applicable in s_{i-1} and $s_i = a_i(s_{i-1})$. We define applying an action sequence $\pi = \langle a_1, ..., a_k\rangle$ in a state s as $\pi(s) = s_k$. An action sequence π is a *plan* if and only if π is applicable in s_0 and that applying π in s_0 leads to a goal state $s_g \in G$ (i.e., $\pi(s_0) = s_g \in G$). The cost of a plan is defined as the sum of all action costs in the plan: $c(\pi) = \sum_{i=1}^{k} c(a_i)$. A planning problem is considered *solvable* if at least one plan exists. A plan $\pi^\star$ is *optimal* if for all plans π, $c(\pi^\star) \leq c(\pi)$.

Currently, the predominant approach to solving planning problems is heuristic search (Bonet and Geffner 2001). A heuristic $h : S \to \mathbb{R} \cup \{\infty\}$ is a function that maps planning states to real-valued estimates of the optimal cost to reach a goal state from a state s, or ∞ to indicate state s is a dead end (i.e., cannot reach any goal state). A *perfect heuristic* $h^\star(s)$ similarly returns ∞ for dead-end states, but always returns the optimal cost-to-goal for any other state s. For brevity, we refer to the optimal cost-to-goal as $h^\star$ for the remainder of the paper. A heuristic is *admissible* if $h(s) \leq h^\star(s)$ for all states $s \in S$. A heuristic h is *consistent* if for every successor state $s' = a(s)$ of any state $s \in S$, $h(s) \leq c(a) + h(s')$, and $h(s_g) = 0$ for all goal states $s_g \in G$.

2.2 Learning Planning Heuristics

In prior work on learning domain-specific planning heuristics, most studies treat *satisficing planning* as the primary use case (Garrett, Kaelbling, and Lozano-Pérez 2016; Chrestien et al. 2023; Chen, Thiébaux, and Trevizan 2024; Chen and Thiébaux 2024; Chen, Trevizan, and Thiébaux 2024), which aims to find a plan (regardless of its quality) within a given time and memory limit. Accordingly, learned heuristics are usually evaluated for search efficiency (e.g., higher coverage). Most work focuses on learning domain-specific planning heuristics, as learning strong, domain-independent heuristics has proven more difficult (Chen, Thiébaux, and Trevizan 2024). Consistent with previous work, we focus on learning a conditional machine learning model h that takes a state representation s and the goal condition G as input and outputs a heuristic value $h_G(s)$.

Most approaches follow a supervised learning scheme, in which an optimal planner, such as Scorpion (Seipp, Keller, and Helmert 2020), first generates optimal plans for a given set of planning problems. Then, the true $h^\star$ labels can be computed for each traversed state, given the optimality of the generated plans. Models representing the heuristic function are typically trained by optimizing an objective based on either *regression* or *pairwise ranking*. When optimizing regression-based objectives such as RMSE, the model learns to predict the exact $h^\star_G(s)$ values (Chen, Trevizan, and Thiébaux 2024; Bai, Thiébaux, and Trevizan 2025). When optimizing pairwise ranking-based objectives, the model learns to rank states by their $h^\star$ values (Chrestien et al. 2023; Chen and Thiébaux 2024; Hao et al. 2024b,a), denoted as $s_i \prec s_j$ if $h^\star_G(s_i) < h^\star_G(s_j)$ and $s_j \preceq s_i$ if $h^\star_G(s_j) \leq h^\star_G(s_i)$. In turn, this makes optimizing a pairwise ranking function a classification problem, penalizing the model for misclassifications.

In this work, we demonstrate the effectiveness of our proposed techniques by augmenting the training data for two recent state-of-the-art frameworks designed for learning planning heuristics: (i) GOOSE (Chen, Trevizan, and Thiébaux 2024), which uses the Weisfeiler-Leman (WL) algorithm to embed the graphs into WL features. Then, the WL features are fed into a support vector machine to predict the heuristic value, trained to optimize a pairwise ranking[1] loss function; (ii) Distincter (Bai, Thiébaux, and Trevizan 2025), which uses a graph neural network to embed the graph information and uses a fully-connected linear layer to predict the heuristic value, trained to optimize RMSE, which is a regression loss function. Note that both GOOSE and Distincter utilize the learned heuristic in a single-queue GBFS, while Distincter also does symmetry pruning when generating successors (Bai, Thiébaux, and Trevizan 2025).

2.3 Existing Data Augmentation Techniques

The topic of data augmentation for learning search heuristics has received limited attention in the literature, and existing approaches to augment training data in learning heuristics

[1]The original GOOSE used a regression objective; later work (Hao et al. 2024a,b; Chen and Thiébaux 2024; Chen 2025) showed that pairwise ranking objectives can lead to better coverage.

typically expand the set of supervised examples by extracting additional ranking relations (Hao et al. 2025) or solving additional problems (Bai, Thiébaux, and Trevizan 2025).

Hao et al. (2025) considered *search space ranking*, which uses additional states from the $A^\star$ search tree for pairwise ranking-based objectives. Notably, Hao et al. (2025) uses the h value of an admissible heuristic for states on the search tree to derive significantly more ranking relations with the states on the optimal trajectory. By admissibility, if any pair of states s, s' that satisfies $h^\star(s') \leq h(s)$, we can see $s' \preceq s$ as $h(s) \leq h^\star(s)$. Since no additional search effort is required, search space ranking can also be considered search-free data augmentation, similar to our approach. Note that this approach is only applicable in ranking-based objectives; beyond the optimal trajectory, exact $h^\star$ labels cannot be derived from the search tree for regression-based objectives. Consequently, we implement this approach for GOOSE, which also uses a ranking-based objective.

Bai, Thiébaux, and Trevizan (2025) note that a bottleneck in generating more training data lies in the limited coverage of optimal planners on the training problems. Therefore, they propose to augment the training data by deriving relaxed subproblems from each task: if the original goal has n subgoals, they create $n-1$ new tasks where the goal for the k-th new problem consists of the first k subgoals of the sequence. These subproblems tend to be easier to solve and can significantly increase training data after solving. However, this approach requires generating and solving additional problems, rather than directly extracting additional data from existing plans and traces. Consequently, this approach is not search-free and is orthogonal to our approach.

3 Search-Free Data Augmentation for Extracting Additional Optimal Plans

In this section, we propose approaches for generating additional optimal plans from previously generated optimal plans and their corresponding search traces. In particular, we propose three *search-free data augmentation* techniques that work for both ranking and regression settings, without requiring to solve any additional planning problems.

3.1 Intermediate Goals

A key observation is that optimal plans are optimal at every step: every contiguous subplan of an optimal plan is itself optimal between the states it connects. We formally state this result in Proposition 1.

Proposition 1. *Let $\pi^\star = \langle a_1, ..., a_k \rangle$ be an optimal plan for the planning problem $\langle S, A, s_0, G \rangle$ which traverses the states $s_0, ..., s_k$ where $s_k \in G$. Then, any subplan $\pi_i = \langle a_1, ..., a_i \rangle$ for $1 \leq i < k$ is also necessarily an optimal plan for the planning problem $\langle S, A, s_0, s_i \rangle$.*

Proof. For convenience, we define $\pi_i^c = \langle a_{i+1}, ..., a_k \rangle$ so that $\pi^\star = \pi_i \oplus \pi_i^c$, where $\oplus$ represents the concatenation of the action sequences. Assume, for the sake of contradiction, there exists some π_i' also connecting s_0 and s_i with $c(\pi_i') < c(\pi_i)$. Then, the plan $\pi' = \pi_i' \oplus \pi_i^c$ satisfies $c(\pi') = c(\pi_i') + c(\pi_i^c) < c(\pi_i) + c(\pi_i^c) = c(\pi^\star)$, contradicting the optimality of $\pi^\star$. □

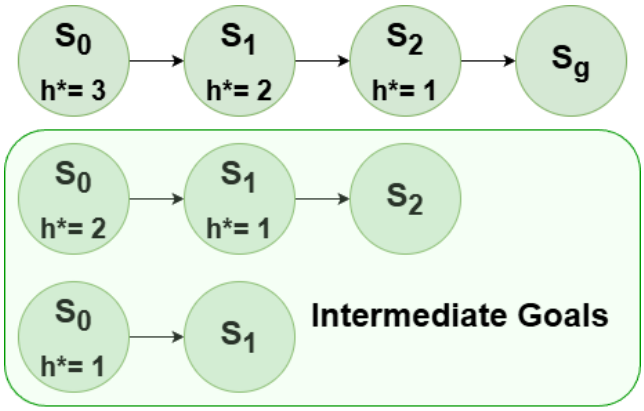

Figure 1: An illustration of data augmentation via the intermediate goals. The nodes on the optimal trajectory $[s_0, s_1, s_2, s_g]$ are coloured green; we observe that this optimal plan yields 3 data points (indicated by $h^\star$). By setting the full states s_2 and s_1 as the new goals, we obtain 2 additional optimal plans.

Proposition 1 yields a simple augmentation scheme based on alternative goal states $s_1, \ldots, s_{k-1}$, which we call *intermediate goals*. Given an optimal plan $\pi^\star = \langle a_1, \ldots, a_k \rangle$ and its state trajectory $s_0, \ldots, s_k$, for each $1 \leq i < k$, we treat s_i as a new goal state and obtain an additional optimal plan without any new search: the subplan $\pi_i = \langle a_1, \ldots, a_i \rangle$ is an optimal plan from s_0 to s_i by Proposition 1.

We demonstrate the idea of intermediate goals in Figure 1. In practice, we construct new problems by simulating $\pi^\star$ from s_0 to obtain each intermediate goal s_i, and then instantiating a new problem where the goal state corresponds to s_i. For regression objectives, we label additional input state pairs (s_j, s_i) for all $j \leq i$ with $h^\star_{s_i}(s_j) = \sum_{m=j+1}^{i} c(a_m)$. For ranking-based objectives, since state pairs s_j and s_{j+1} satisfy $s_{j+1} \prec s_j$ when conditioned on the goal G, then we also have $s_{j+1} \prec s_j$ when conditioned on the intermediate goal s_i for all $i > j$.

By obtaining optimal plans for additional problems with alternative goal states, intermediate goals can enhance the diversity of the training data. Concretely, they provide additional labelled data points for states and their corresponding $h^\star$ values to alternative goals. This augmentation requires no additional search effort and can be applied efficiently to any existing plan using a plan simulator. Note that intermediate goals are instantiated as full states, which may not match the standard goal specification of the domain. This, in turn, may introduce discrepancies in the distribution of goal conditions for the augmented data, potentially degrading the model's effectiveness. Nevertheless, the resulting supervision can still be useful: it exposes the learner to a more diverse set of goal conditions and helps it learn more general patterns of progress toward goals.

3.2 Closed List Trajectories

When solving a planning problem with best-first search algorithms such as $A^\star$, the planner iteratively expands the most promising state in the open list and adds it to the closed list. Over time, the closed list can become very large and, importantly, contain many states far from the initial state. By tracing the parent pointers of an expanded state s, we can define a state trajectory and its corresponding action sequence from the initial state s_0. A well-known result states that planning with $A^\star$ and a *consistent* heuristic in a domain with non-negative action costs guarantees that all expanded states are reached optimally, since they will never be re-expanded at a lower cost (Hart, Nilsson, and Raphael 1968). Consequently, all state trajectories on the closed list generated by $A^\star$ with a consistent heuristic are necessarily optimal. We formally state this result in Proposition 2.

Proposition 2. *For a planning problem $\langle S, A, s_0, G\rangle$ where $c(a) \geq 0$ for all $a \in A$, if a closed list state $s_t \in S$ on the trajectory $s_0, ..., s_t$ is reached by the action sequence $\pi_t = \langle a_1, ..., a_t\rangle$ generated by $A^\star$ using a consistent heuristic, then π_t is an optimal plan for the planning problem $\langle S, A, s_0, s_t\rangle$.*

Proof. As s_t is expanded by $A^\star$ with a consistent heuristic, this result follows directly from Lemma 2 (Hart, Nilsson, and Raphael 1968), which indicates that the cost to reach s_t from s_0 is equal to the optimal cost to reach s_t from s_0. □

Proposition 2 implies that *every expanded state* can be treated as the goal state of a new planning instance, with an optimal plan obtained by recursively tracing its parent pointers back to the root node. Interestingly, this augmentation can be applied even if the original planning run terminates without a solution, since the optimality claim does not require that $A^\star$ ever reach a goal state of the original task. Note that unlike Hao et al. (2025), our approach is able to provide exact $h^\star$ labels through conditioning on alternative goals, which consequently enables our approach in regression-based methods.

We illustrate an example of using closed-list trajectories in Figure 2. However, because the closed list can contain an enormous number of trajectories across many training problems, training on all such trajectories can be computationally expensive. Moreover, as more trajectories are included in the dataset, the gains in the number of unique data points typically diminish. Consequently, it is often more cost-effective to train on only a subset of trajectories. In practice, when training with a limited number of trajectories, we observed that selecting the k longest trajectories from the closed list generally leads to better coverage than randomly sampling k trajectories.

As with intermediate goals, closed-list trajectories can substantially diversify the goal conditions seen during training. In fact, because every expanded state can serve as a derived goal, this augmentation often induces a much broader range of goal states than intermediate goals alone. This broader supervision can be beneficial, as it exposes the learner to many additional state-goal pairs, potentially enabling the model to learn a more general pattern. However, it can also be a drawback in domains where test problems only consist of highly structured goal conditions. In such cases, treating arbitrarily expanded full states as goals may introduce a strong mismatch between the goal distributions in the training and test sets, potentially hurting the effectiveness of the learned heuristic.

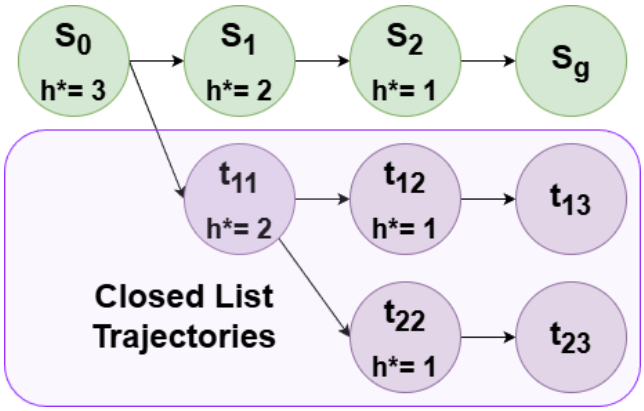

Figure 2: An illustration of data augmentation via the closed list trajectories. The nodes on the optimal trajectory are coloured green, and the states on the closed list are coloured purple. We observe that by taking states t_{13} and t_{23} as the new goals, we can obtain 2 additional optimal plans.

3.3 Partial-Order Plan

A sequential plan is characterized by a set of actions $\mathcal{A}$ together with a set of total-order constraints $\mathcal{O}$ over those actions. Such a plan can often be viewed as a linearization of an underlying partial-order plan (POP), which relaxes some of the constraints in $\mathcal{O}$ to allow greater flexibility during execution (Veloso, Pollack, and Cox 1998; Backstrom 1998; Muise, Beck, and McIlraith 2016). A valid POP may admit many sequential linearizations, which can in turn traverse different states and therefore yield substantially more training data. For an optimal sequential plan $\pi^\star$ and its corresponding POP $P^\star$, any linearization $\tilde{\pi}$ of $P^\star$ preserves the same action set and differs only in the ordering of actions; therefore, under state-independent action costs, it must also be optimal. We formalize this observation in Proposition 3.

Proposition 3. *Let $\pi^\star = \langle a_1, ..., a_k\rangle$ be an optimal plan for the planning problem $\langle S, A, s_0, G\rangle$ which traverses states $s_0, ..., s_k$. If action costs are independent of states, then all valid linearizations $\tilde{\pi}$ of the underlying POP of $\pi^\star$ are also necessarily optimal plans for the planning problem $\langle S, A, s_0, G\rangle$.*

Proof. Due to the state-independent action cost assumption, we note that any valid plan linearization $\tilde{\pi}$ necessarily has $c(\tilde{\pi}) = c(\pi^\star)$. By optimality of $\pi^\star$ and $c(\tilde{\pi}) = c(\pi^\star)$, we also have $c(\tilde{\pi}) \leq c(\pi)$ for any plan π for the problem $\langle S, A, s_0, G\rangle$. □

Proposition 3 provides a data augmentation technique that does not introduce alternative goal conditions. Instead, $\tilde{\pi}$

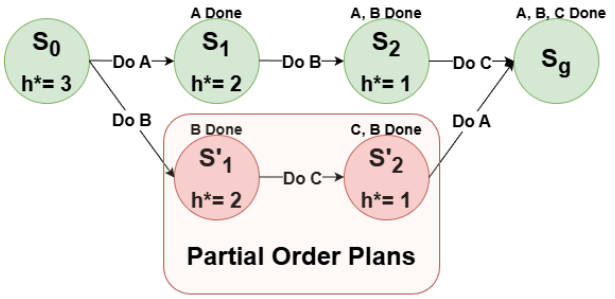

Figure 3: An illustration of data augmentation via the partial-order plans. The figure demonstrates linearizations of a simple POP $\mathcal{P} = \{\mathcal{A}, \mathcal{O}\}$ where $\mathcal{A}$: {Do A, Do B, Do C} and $\mathcal{O}$: {Do B $\prec$ Do C} meaning Do B must occur before Do C. The nodes on the optimal trajectory are coloured green, and the states on another linearization are coloured red. We observe that this new linearization provides 2 additional state labels.

reaches alternative intermediate states because the same actions are executed in a different order. As such, POP-based augmentation would not be susceptible to mismatches in goal conditions. Moreover, this augmentation can also be applied to plans obtained from closed-list trajectories or intermediate goals, further increasing the yield of those techniques.

We illustrate the use of POPs for data augmentation in Figure 3. To obtain the underlying POP from a sequential plan, two broad approaches exist: (i) *deordering*, which relaxes ordering constraints from $\mathcal{O}$; and (ii) *reordering*, which may add some constraints in order to relax others and achieve greater flexibility overall. Relaxing more ordering constraints generally increases the number of possible linearizations (Muise, Beck, and McIlraith 2016). While reordering can potentially achieve greater flexibility, the problem is NP-hard and difficult to approximate (Backstrom 1998). We therefore focus on deordering in this work. In particular, we use the efficient yet effective polynomial-time KK algorithm (Kambhampati and Kedar 1994) to deorder plans by systematically removing ordering constraints, while keeping those needed to preserve causal support for actions. We then sample linearizations of the resulting deordered POP using Kahn's algorithm (Kahn 1962).

Note that deordering and linearizing can yield a large number of optimal plans in some domains. As with closed-list trajectories, training on all possible linearizations can be computationally expensive and may yield diminishing returns in terms of additional unique data points. We hypothesize that a more diverse set of linearizations would provide a broader spectrum of information, which may be beneficial for learning more effective heuristics. Accordingly, we use Zhong, Shati, and Cohen's (2024) algorithm for selecting a set of k diverse plans that maximizes the minimum pairwise state dissimilarity. A potential limitation of this augmentation is that some domains with a single shared bottleneck resource (e.g., a single arm in Blocksworld) may allow little to no relaxation of ordering constraints, thereby limiting the number of distinct linearizations available.

4 Experimental Results

4.1 Experimental Setup

Data generation. We train and evaluate the models on the International Planning Competition 2023 Learning Track (Taitler et al. 2024). For each domain, the training set consists of 99 problems, and the test set consists of 90 problems across three difficulty levels. To generate the training data, we follow prior work (Chen, Trevizan, and Thiébaux 2024; Bai, Thiébaux, and Trevizan 2025) and run the Scorpion planner with the offline Saturated Cost Partitioning (SCP) heuristic (Seipp, Keller, and Helmert 2020), a consistent heuristic, under a 30-minute overall time limit and a 16 GB memory limit. From the terminal nodes (i.e., nodes without successors) of the closed list, we save up to 20 longest trajectories and, as a baseline, 20 random trajectories. Then, we create an intermediate goals dataset from the generated optimal plans using the plan simulator in Unified Planning (Micheli et al. 2025). To generate the POP, we deorder the optimal plans using the KK algorithm (Kambhampati and Kedar 1994). We randomly sample up to 100 unique trajectories from the POP using Kahn's algorithm (Kahn 1962). For each domain, we use beam search to select up to 5 unique linearizations; as a baseline, we also randomly sample 5 linearizations. We also analyze the impact of doubling the number of trajectories and linearizations.

Training and evaluation. For GOOSE, we use the recommended feature pruning behaviour from a recent paper studying its hyperparameters (Chen 2025): i-mf feature pruning (Hao et al. 2025) for all domains except Blocksworld, for which no feature pruning is used. For Distincter, we use the GNN settings from the original paper (Bai, Thiébaux, and Trevizan 2025): 4 layers for Spanner, 7 layers for Sokoban, and 3 layers for every other domain. We utilize the original training procedure, which trains the model with RMSE loss for 30 epochs with cosine annealing and performs early stopping on a validation set as per Bai, Thiébaux, and Trevizan (2025). We note that training has converged for all models in terms of validation accuracy before the maximum number of epochs. For evaluation, we follow the same guidelines as prior work, using a 30-minute time limit and an 8 GB memory limit for all the planners. For reference, we also report the performance of LAMA-First (Richter and Westphal 2010) following prior work (Bai, Thiébaux, and Trevizan 2025). For fairness among the planners[2], we run all evaluations on an Intel i7-13700K CPU.

We compare our data augmentation techniques to the original GOOSE dataset (Chen, Trevizan, and Thiébaux 2024) (labelled None) and those considered by Hao et al. (2025) (labelled Search Space). For brevity, we abbreviate our data augmentation as IG (Intermediate Goals), CLT (Closed List Trajectories), POP (Partial-Order Plans) and domain names in the reported tables as BL (Blocksworld), CH (Childsnack), FE (Ferry), FL (Floortile), MI (Miconic),

[2]Note that Distincter is a neural model; hence, we expect its coverage to improve significantly when run on a strong GPU.

Methodology	Augmentation	BL	CH	FE	FL	MI	RO	SA	SO	SP	TR	Σ
GOOSE	None	81	23	**77**	3	**90**	53	57	**37**	**69**	54	544
	Search Space	82	23	**77**	4	**90**	**55**	61	**37**	**69**	55	553
	IG	85	35	**77**	**7**	**90**	47	61	**37**	**69**	55	563
	CLT	80	18	60	4	**90**	52	45	31	**69**	57	506
	POP	81	23	**77**	3	**90**	54	**63**	**37**	**69**	54	551
	IG, CLT	85	35	**77**	**7**	**90**	39	39	31	**69**	**59**	531
	IG, POP	**86**	35	**77**	6	**90**	51	57	**37**	**69**	54	562
	CLT, POP	83	38	60	6	**90**	52	55	31	**69**	56	540
	IG, CLT, POP	**86**	**60**	**77**	6	**90**	52	55	31	**69**	58	**584**
	Per-Domain Best	**86**	**60**	**77**	**7**	**90**	54	**63**	**37**	**69**	**59**	**602**
Distincter	None	68	34	64	**2**	32	34	39	**31**	30	14	348
	IG	**69**	64	61	**2**	30	30	36	25	30	19	366
	CLT	**69**	42	61	**2**	46	30	37	26	30	30	372
	POP	68	64	64	**2**	48	**36**	**40**	**31**	30	**42**	**425**
	IG, CLT	68	**67**	**68**	**2**	48	27	36	**31**	30	29	406
	IG, POP	**69**	65	61	**2**	**51**	31	27	25	**52**	29	412
	CLT, POP	65	37	62	1	37	29	34	30	30	26	351
	IG, CLT, POP	68	61	**68**	**2**	37	29	34	**31**	30	32	392
	Per-Domain Best	**69**	**67**	**68**	**2**	**51**	**36**	**40**	**31**	**52**	**42**	**458**
Per-Domain Best	-	**86**	**67**	**77**	7	**90**	54	63	37	**69**	59	**609**
LAMA-First	-	56	33	70	**11**	**90**	**70**	**89**	**43**	30	**64**	556

Table 1: Coverage of GOOSE and Distincter trained on training datasets obtained from various search-free data augmentations. The top three configurations for each methodology and domain are indicated by the cell colour intensity, and the top configuration for each methodology is also **bolded**. The per-domain best rows represent an oracle upper bound in coverage per domain across our augmentations.

RO (Rovers), SA (Satellite), SO (Sokoban), SP (Spanner) and TR (Transport). We also evaluate combinations of augmentations by concatenating the datasets produced by applying different augmentations to the GOOSE dataset.

4.2 Results on Search-Free Data Augmentation

We report the main coverage results in Table 1. We analyze the results for GOOSE and Distincter separately.

Coverage Results for GOOSE. Table 1 shows that our search-free augmentation techniques substantially strengthen GOOSE: the baseline solves 544 instances, while using all three data augmentations solves 584 (+40), surpassing previous data augmentation methods by 31 plans. Overall, the largest gain appears in Childsnack, where using all three augmentations improves coverage from 23 to 60 (+37), suggesting the three augmentations can sometimes provide complementary supervision for the model. We also observe an increase of four or five instances in each of Blocksworld, Floortile, Satellite, and Transport. In contrast, performance in Ferry and Sokoban sometimes degrade in some configurations, while degradation is seen across all augmentations except for POP (+1) for Rovers. Finally, we observe that using CLT alone often degrades planning performance (-38), likely due to a distribution shift from treating expanded full states as alternative goals; its benefits emerge mainly when combined with IG and/or POP.

Coverage Results for Distincter. Table 1 shows that Distincter benefits strongly from our augmentations: the baseline solves 348 instances, while the best single configuration (POP) solves 425 (+77). We note that the main gains concentrate in a few domains. The largest improvements are in Childsnack (+33; best with IG and CLT), Spanner (+22; best with IG and POP), Transport (+28; best with POP), and Miconic (+19; best with IG and POP). In contrast, Floortile, Satellite, and Sokoban show degradation in some configurations (the best configurations match the baseline), and Rovers improves only slightly in one of the configurations (+2). Overall, we see a different pattern with the effect of our proposed augmentations compared to GOOSE, as the state diversity provided by POP seems to be more effective than the goal diversity provided by IG and CLT.

Number of Expansions and Plan Length. We report the average plan length and number of expansions for problems solved by both the baseline and the best configuration (by coverage) for GOOSE and Distincter, respectively, in Table 2. On average, we note that the augmented GOOSE expands fewer or equal states in 9 out of 10 domains (strictly fewer in 6), and the augmented Distincter expands strictly fewer states in 8 out of 10 domains. In fact, we observe that the average number of expansions in some domains, such as Childsnack, is reduced by orders of magnitude. We also observe that, on average, the best augmentation finds shorter or

	GOOSE				Distincter			
	Baseline		Best		Baseline		Best	
Domain	**Expansions**	**Length**	**Expansions**	**Length**	**Expansions**	**Length**	**Expansions**	**Length**
BL	3,551.7	494.8	**757.7**	**484.6**	1,913.6	246.4	**1,900.3**	**227.0**
CH	2,572,660.4	26.2	**32.1**	**25.0**	41,301.3	29.7	**29.7**	**27.0**
FE	**10,288.1**	411.1	10,388.6	**402.4**	**1,421.5**	168.3	1,473.5	**160.8**
FL	3,121,304.3	**34.7**	**113,090.7**	**34.7**	991,099.0	33.0	**448,087.5**	**32.0**
MI	**1,971.0**	**299.6**	**1,971.0**	**299.6**	15,030.1	25.0	**1,031.9**	**24.3**
RO	7,147.7	251.8	**1,531.2**	**239.3**	889.5	89.3	**561.0**	**87.9**
SA	876.1	**74.1**	**484.2**	76.8	**275.7**	39.8	778.8	**38.2**
SO	**62,741.8**	**69.9**	**62,741.8**	**69.9**	26,284.7	40.5	**2,147.4**	**34.6**
SP	**98.3**	**97.3**	**98.3**	**97.3**	208.2	**13.9**	**21.6**	15.8
TR	2,230.3	**64.1**	**1,714.8**	68.1	41,605.3	**16.5**	**300.9**	17.9
Avg.	578,287.0	182.4	**19,281.1**	**179.8**	112,002.9	70.2	**45,633.3**	**66.5**

Table 2: Comparison of average number of expansions and average plan lengths for the baseline and the best data augmentation configuration from GOOSE and Distincter. Best refers to the augmentation configuration with the highest coverage, for each domain. **Bold** indicates the better value for each domain and method (lower is better; ties are both bolded). The Avg. row reports the arithmetic mean across domains.

	Dataset Statistics			GOOSE		Distincter	
Augmentation	**Plan Count**	**Data Points**	**Solve Time**	**Train Time**	**Total Time**	**Train Time**	**Total Time**
None	71.4	1,409.6	9,032.3	30.1	9,062.4	223.8	9,256.1
IG	1,418.3	27,189.2	9,204.5	935.6	10,140.1	371.7	9,576.2
CLT	880.1	19,121.4	9,057.3	412.8	9,470.1	395.3	9,452.6
POP	297.4	2,732.7	9,131.6	173.6	9,305.2	243.9	9,375.5
IG, CLT	2,227.0	47,639.9	9,229.5	1,318.3	10,547.8	469.1	9,698.6
IG, POP	1,644.3	29,343.5	9,303.8	1,079.1	10,382.9	322.1	9,625.9
CLT, POP	1,106.1	20,054.6	9,156.6	556.3	9,712.9	312.3	9,468.9
IG, CLT, POP	2,453.0	49,425.6	9,328.8	1,461.8	10,790.6	499.1	9,827.9

Table 3: Average dataset sizes and runtimes (measured in seconds) per domain across all evaluated data augmentations for GOOSE and Distincter. The original GOOSE dataset (labelled None) is reported as a reference. The solve time column reports the time spent on data generation (i.e., planning) and data augmentation. The total time column reports the total time used in generating data and training the model.

equal-length plans in 8 domains for both GOOSE and Distincter. Overall, our augmentation can substantially reduce search effort relative to the baseline (thus achieving better coverage) while preserving plan quality.

Size of Training Data and Training Times. We report the per-domain average numbers of unique plans (plan count), the number of unique labelled state-goal pairs (data points), total time required for data generation and data augmentation (solve time), and training times for both GOOSE and Distincter in Table 3 across all tested configurations. On average, our data augmentations significantly increase the size of the training data. In particular, both IG and CLT increased the number of unique data points by an order of magnitude. POP, on the other hand, provides relatively fewer additional unique data points, as the different linearizations traverse many overlapping states. Overall, we see that each augmentation technique extends the solve time by 30–180 seconds on average, which represents a relatively small portion of the solve time. We also note that training time increases as a result of the additional training data, especially for GOOSE. This is expected as GOOSE performs feature pruning, which requires solving NP-complete problems (Hao et al. 2025). Nevertheless, we note that the total time (solving + training) increases by only about 1–6% for Distincter and about 3–19% for GOOSE when using our data augmentation techniques.

Ablations on Data Selection. We report the coverage results on different data selection methods for CLT and POP in Table 4. We observe that our diversity-based selection for POP consistently outperforms random selection for both GOOSE and Distincter, increasing total coverage from 547 to 551 for GOOSE and from 412 to 425 for Distincter. These gains are concentrated in several domains, such as Rovers and Satellite for GOOSE, and Childsnack, Satellite, and Transport for Distincter, while many other domains remain unchanged. For CLT in GOOSE, selecting the longest

Methodology	Augmentation	Selection	BL	CH	FE	FL	MI	RO	SA	SO	SP	TR	Σ
GOOSE	POP	Random	**81**	**23**	**77**	**3**	**90**	51	62	**37**	**69**	**54**	547
		Diverse	**81**	**23**	**77**	**3**	**90**	**54**	**63**	**37**	**69**	**54**	**551**
	CLT	Random	**81**	13	60	3	**90**	51	42	**31**	**69**	**57**	497
		Longest	80	**18**	**60**	**4**	**90**	**52**	**45**	**31**	**69**	**57**	**506**
Distincter	POP	Random	**68**	56	**64**	**2**	**51**	**36**	39	**31**	**30**	35	412
		Diverse	**68**	**64**	**64**	**2**	48	**36**	**40**	**31**	**30**	**42**	**425**
	CLT	Random	67	**48**	**62**	**2**	42	29	30	30	**30**	**30**	370
		Longest	**69**	42	61	1	**46**	**30**	**37**	26	**30**	**30**	**372**

Table 4: Coverage ablation for POP and CLT under different data-selection strategies. **Bold** indicates the better selection within each methodology, augmentation, and domain.

Methodology	Augmentation	Count	BL	CH	FE	FL	MI	RO	SA	SO	SP	TR	Σ
GOOSE	POP	5	**81**	**23**	**77**	**3**	**90**	54	**63**	**37**	**69**	**54**	**551**
		10	**81**	**23**	**77**	**3**	**90**	**55**	57	**37**	**69**	**54**	546
	CLT	20	**80**	**18**	60	**4**	**90**	**52**	**45**	**31**	**69**	57	506
		40	79	**18**	**77**	3	**90**	43	41	**31**	**69**	**59**	**510**
Distincter	POP	5	**68**	**64**	**64**	**2**	48	**36**	40	**31**	**30**	**42**	**425**
		10	**68**	**64**	63	**2**	**55**	35	**42**	22	**30**	**42**	423
	CLT	20	69	42	61	1	**46**	**30**	**37**	26	30	**30**	372
		40	**76**	**45**	**63**	**3**	31	27	35	**27**	**46**	**30**	**383**

Table 5: Coverage results for POP and CLT under different numbers of trajectories and linearizations. **Bold** indicates the better number of trajectories or linearizations for each methodology and domain.

trajectories also outperforms random sampling, increasing total coverage from 497 to 506. For CLT in Distincter, we observe a smaller improvement in total coverage from 370 to 372. Overall, these results indicate that effective data selection strategies can provide better supervision for heuristic learning than simple random sampling.

In our experiments, we fix CLT and POP to use 20 trajectories and 5 linearizations in order to keep the augmented datasets computationally manageable. To better understand the impact of these choices, we analyze the coverage when doubling the number of trajectories and linearizations in Table 5. For POP, we observe that increasing the number of linearizations leads to a slight degradation of -5 and -2 in overall coverage. For CLT, we see that increasing the number of trajectories seems to lead to a coverage improvement of +5 and +11, respectively. Interestingly, doubling the number of augmented plans interacts with different domains differently. For instance, using 40 trajectories yields a significant gain in Ferry (+17) for GOOSE but a significant loss in Rovers (-9). Similarly, we observe significant gains in Spanner (+16) and Blocksworld (+7) for Distincter at 40 trajectories, whereas Miconic degraded significantly (-15).

5 Conclusion and Future Works

Our results show that our proposed search-free data augmentation can substantially improve the effectiveness of the learned planning heuristics, often yielding higher coverage, fewer node expansions, and comparable or slightly shorter plan lengths. In fact, the best augmentation configuration can improve the coverage of GOOSE and Distincter by 40 and 77 problems, respectively. For the first time, we enhance GOOSE to exceed the coverage of the first iteration of the LAMA planner (Richter and Westphal 2010).

However, our approach has some limitations. In particular, data augmentation can greatly increase the size of the training set, thereby raising preprocessing and training costs. This is especially true for computationally complex subroutines, such as feature pruning in GOOSE, where additional features and data points can significantly increase computation cost. Nevertheless, we observed that performing data augmentation and training on the augmented data remains significantly cheaper than solving more problems. Another observation is that more data did not always translate into better coverage. In several domains, some augmentations help only marginally, and in others, they can even hurt, particularly when the augmented data introduces a distribution shift in goals relative to the test problems. In future works, developing more effective data selection strategies for the augmented dataset could possibly help mitigate this issue by retaining the most informative data points. Moreover, evaluating the empirical impact of learning from augmented datasets originating from sub-optimal plans can also be an interesting direction for future work. Finally, investigating compositional combinations of our augmentation techniques beyond a simple union of the augmented datasets could also be worthwhile, as it can potentially yield significantly more diverse training data.

Acknowledgments

We thank the anonymous reviewers whose valuable feedback helped improve the final paper. The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada.

References

Backstrom, C. 1998. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research*, 9: 99–137.

Bäckström, C.; and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4): 625–655.

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

Barrios, E. Q.; Olaya, Á. G.; Millán, D. B.; and Rebollo, F. F. 2011. Control of autonomous mobile robots with automated planning. *Journal of Physical Agents*, 5(1): 3–13.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2): 5–33.

Bylander, T. 1992. Complexity results for extended planning. In *Artificial Intelligence Planning Systems*, 20–27. Elsevier.

Chen, D.; and Thiébaux, S. 2024. Graph learning for numeric planning. *Advances in Neural Information Processing Systems*, 37: 91156–91183.

Chen, D. Z. 2025. Weisfeiler-Leman Features for Planning: A 1,000,000 Sample Size Hyperparameter Study. *arXiv preprint arXiv:2508.18515*.

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.; Trevizan, F.; and Thiébaux, S. 2024. Return to tradition: Learning reliable heuristics with classical machine learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 68–76.

Chrestien, L.; Edelkamp, S.; Komenda, A.; and Pevny, T. 2023. Optimize planning heuristics to rank, not to estimate cost-to-goal. *Advances in Neural Information Processing Systems*, 36: 25508–25527.

Ferber, P.; Helmert, M.; and Hoffmann, J. 2020. Reinforcement learning for planning heuristics. In *Proceedings of the 1st workshop on bridging the gap between AI planning and reinforcement learning (PRL)*, 119–126.

Garg, S.; Bajpai, A.; et al. 2019. Size independent neural transfer for rddl planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 631–636.

Garrett, C. R.; Kaelbling, L. P.; and Lozano-Pérez, T. 2016. Learning to rank for synthesizing planning heuristics. *arXiv preprint arXiv:1608.01302*.

Geffner, H.; and Bonet, B. 2013. *A concise introduction to models and methods for automated planning*. Morgan & Claypool Publishers.

Groshev, E.; Goldstein, M.; Tamar, A.; Srivastava, S.; and Abbeel, P. 2018. Learning generalized reactive policies using deep neural networks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 408–416.

Halevy, A.; Norvig, P.; and Pereira, F. 2009. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2): 8–12.

Hao, M.; Chen, D. Z.; Trevizan, F.; and Thiébaux, S. 2025. Effective data generation and feature selection in learning for planning. In *European Conference on Artificial Intelligence (ECAI-25)*.

Hao, M.; Trevizan, F.; Thiébaux, S.; Ferber, P.; and Hoffmann, J. 2024a. Guiding GBFS through learned pairwise rankings. In *Thirty-Third International Joint Conference on Artificial Intelligence {IJCAI-24}*, 6724–6732. International Joint Conferences on Artificial Intelligence Organization.

Hao, M.; Trevizan, F.; Thiébaux, S.; Ferber, P.; and Hoffmann, J. 2024b. Learned Pairwise Rankings for Greedy Best-First Search. In *Proc. ICAPS Workshop on Reliable Data-Driven Planning and Scheduling*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hestness, J.; Narang, S.; Ardalani, N.; Diamos, G.; Jun, H.; Kianinejad, H.; Patwary, M. M. A.; Yang, Y.; and Zhou, Y. 2017. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Kahn, A. B. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11): 558–562.

Kambhampati, S.; and Kedar, S. 1994. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence*, 67(1): 29–70.

Karpas, E.; and Magazzeni, D. 2020. Automated planning for robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3: 417–439.

Micheli, A.; Bit-Monnot, A.; Röger, G.; Scala, E.; Valentini, A.; Framba, L.; Rovetta, A.; Trapasso, A.; Bonassi, L.; Gerevini, A. E.; et al. 2025. Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29: 102012.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research*, 57: 113–149.

Pedersen, M. R.; and Krüger, V. 2015. Automated planning of industrial logistics on a skill-equipped robot. In *IROS 2015 workshop Task Planning for Intelligent Robots in Service and Manufacturing*. Citeseer.

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

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated cost partitioning for optimal classical planning. *Journal of Artificial Intelligence Research*, 67: 129–167.

Shen, W.; Trevizan, F.; and Thiébaux, S. 2020. Learning domain-independent planning heuristics with hypergraph networks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 574–584.

Shen, W.; Trevizan, F.; Toyer, S.; Thiébaux, S.; and Xie, L. 2019. Guiding search with generalized policies for probabilistic planning. In *Proceedings of the international symposium on combinatorial search*, volume 10, 97–105.

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L.; and Katz, M. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 20256–20264.

Sousa, A.; and Tavares, J. 2013. Toward automated planning algorithms applied to production and logistics. *IFAC Proceedings Volumes*, 46(24): 165–170.

Sun, C.; Shrivastava, A.; Singh, S.; and Gupta, A. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, 843–852.

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.

Veloso, M. M.; Pollack, M. E.; and Cox, M. T. 1998. Rationale-based monitoring for planning in dynamic environments. In *AIPS*, volume 171, 180.

Zhong, S.; Shati, P.; and Cohen, E. 2024. Bi-Criteria Diverse Plan Selection via Beam Search Approximation. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 188–196.