Symmetry-Invariant Novelty Heuristics via Unsupervised Weisfeiler-Leman Features

Dillon Z. Chen

LAAS-CNRS, University of Toulouse

Abstract

Novelty heuristics aid heuristic search by exploring states that exhibit novel atoms. However, novelty heuristics are not symmetry invariant and hence may sometimes lead to redundant exploration. In this preliminary report, we propose to use Weisfeiler-Leman Features for planning (WLFs) in place of atoms for detecting novelty. WLFs are recently introduced features for learning domain-dependent heuristics for generalised planning problems. We explore an unsupervised usage of WLFs for synthesising lifted, domain-independent novelty heuristics that are invariant to symmetric states. Experiments on the classical International Planning Competition and Hard To Ground benchmark suites yield promising results for novelty heuristics synthesised from WLFs.

1 Introduction

Novelty heuristics (Lipovetzky and Geffner 2017; Katz et al. 2017), which originated from the notion of width for propositional planning (Lipovetzky and Geffner 2012), are a simple yet powerful domain-independent technique for enhancing any given heuristic by helping guide exploration in search. Novelty heuristics have later been extended to handle numeric planning (Ramírez et al. 2018; Teichteil-Königsbuch, Ramírez, and Lipovetzky 2020; Chen and Thiébaux 2024b), lifted planning (Corrêa and Seipp 2022), to guide policy search in generalised planning (Lei, Lipovetzky, and Ehinger 2023), and enhancing the LAMA planner (Corrêa and Seipp 2024).

Weisfeiler-Leman Features for planning (WLFs) are a recently introduced, efficient-to-compute feature generator for planning problems (Chen, Trevizan, and Thiébaux 2024). We introduce the usage of WLFs for generating domain-independent novelty heuristics, which we name *Weisfeiler-Leman (WL) novelty heuristics*. We do so by generalising the quantified-both novelty heuristic framework introduced by Katz et al. (2017) to support arbitrary planning feature generators. We show that WL novelty heuristics are symmetry-invariant, a first of its kind for novelty heuristics. Preliminary experimental evaluation on the International Planning Competition and Hard To Ground benchmark suites show promising results for WL novelty heuristics.

2 Background

This section formalises lifted STRIPS problems, the Weisfeiler-Leman Features for planning, and Quantified Both novelty heuristics. Let $[\![n]\!]$ denote the set of integers $\{1, \dots, n\}$ and $|S|$ the size of a set S. The minimum of an empty set is ∞.

Planning Problem and Lifted Representation A classical planning problem is a deterministic state transition model (Geffner and Bonet 2013) given by a tuple $\mathbf{P} = \langle S, A, s_0, G\rangle$ where S is a set of states, A a set of actions, $s_0 \in S$ an initial state, and $G \subseteq S$ a set of goal states. Each action $a \in A$ is a function $a : S \to S \cup \{\bot\}$ where $a(s) = \bot$ if a is not applicable in s, and $a(s) \in S$ is the successor state when a is applied to s. A solution for a planning problem is a plan: a sequence of actions $\pi = a_1, \dots, a_n$ where $s_i = a_i(s_{i-1}) \neq \bot$ for $i \in [\![n]\!]$ and $s_n \in G$. A state s in a planning problem $\mathbf{P}$ induces a new planning problem $\mathbf{P}' = \langle S, A, s, G\rangle$. A planning problem is solvable if there exists at least one plan. Satisficing planning refers to the task of finding any plan for a planning problem if it exists.

Planning problems are often compactly formalised in a lifted representation using predicate logic, such as via PDDL (Ghallab et al. 1998; Haslum et al. 2019). More specifically, a *lifted (STRIPS) planning problem* is a tuple $\mathbf{P} = \langle \mathcal{O}, \mathcal{P}, \mathcal{A}, s_0, \mathcal{G}\rangle$, where $\mathcal{O}$ denotes a set of objects, $\mathcal{P}$ a set of predicate symbols, $\mathcal{A}$ a set of action schemata, s_0 the initial state, and $\mathcal{G}$ now the goal condition. Understanding of the transition system induced by $\mathcal{A}$ is not necessary for this paper, as the proposed novelty heuristics are applicable when the action model is not known (Francès et al. 2017). We next focus on state and goal condition representations.

Each symbol $P \in \mathcal{P}$ is associated with an arity $\mathrm{ar}(P) \in \mathbb{N} \cup \{0\}$. Predicates take the form $P(x_1, \dots, x_{\mathrm{ar}(P)})$, where the x_is denote their arguments. Atoms are defined by substituting objects into predicate arguments, e.g. $p = P(o_1, \dots, o_{\mathrm{ar}(P)})$. More specifically, given $P \in \mathcal{P}$, and a tuple of objects $\mathbf{o} = \langle \mathbf{o}_1, \dots, \mathbf{o}_{\mathrm{ar}(P)}\rangle$, we denote $P(\mathbf{o})$ as the atom defined by substituting $\mathbf{o}$ into arguments of P. A state s is a set of atoms. The goal condition $\mathcal{G}$ also consists of a set of atoms, and a state s is a goal state if $s \supseteq \mathcal{G}$.

We introduce the notational shorthand $\mathbf{P}[\omega]$ as the ω component of a problem $\mathbf{P}$; e.g. $\mathbf{P}[s_0]$ is the initial state of $\mathbf{P}$; and given a state s, let $\mathbf{P}_s = \langle \mathbf{P}[\mathcal{O}], \mathbf{P}[\mathcal{P}], \mathbf{P}[\mathcal{A}], s, \mathbf{P}[\mathcal{G}]\rangle$.

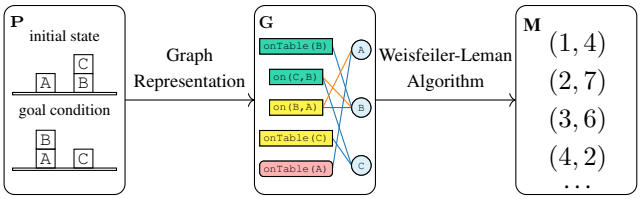

Figure 1: The pipeline for generating WL Features for a Blocksworld problem (`clear` atoms omitted).

Algorithm 1: The Weisfeiler-Leman Algorithm

Data: A graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{F}, \mathbf{L} \rangle$, injective HASH function, and number of iterations L.
Result: Multiset of colours $\mathbf{M}$.

1 $c^0(v) \leftarrow \mathbf{F}(v), \forall v \in \mathbf{V}$
2 **for** $l = 1, \ldots, L$ **do for** $v \in \mathbf{V}$ **do**
3 $\quad c^l(v) \leftarrow$ HASH $\left(c^{l-1}(v), \bigcup_{\iota \in \Sigma_\mathrm{E}} \{\!\{ (c^{l-1}(u), \iota) \mid u \in \mathbf{N}_\iota(v) \}\!\}\right)$
4 **return** $\bigcup_{l=0,\ldots,L} \{\!\{ c^l(v) \mid v \in \mathbf{V} \}\!\}$

Weisfeiler-Leman Features for Planning Weisfeiler-Leman Features (WLFs) are introduced as state-centric feature generators for planning problems for use with learning heuristics for search (Chen, Trevizan, and Thiébaux 2024). WLFs have been extended to handle numeric planning problems (Chen and Thiébaux 2024a), probabilistic planning problems (Zhang 2024), and for learning action set heuristics (Wang and Trevizan 2025). As summarised in Figure 1, WLFs are generated via a 2 step process, consisting of (1) transforming a planning problem $\mathbf{P}$ into a graph with edge labels[1] $\mathbf{G}$, and (2) running the Weisfeiler-Leman (WL) algorithm on the graph to generate a set of features $\mathbf{M}$ that takes the form of a set of tuples $(f_i, v_i) \in \mathbb{N} \times \mathbb{N}$.

Step (1) may use any graph representation of the planning problem. Formally, we denote a graph with categorical node features and edge labels by a tuple $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{F}, \mathbf{L} \rangle$ where $\mathbf{V}$ is a set of nodes, $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ a set of edges, $\mathbf{F} : \mathbf{V} \to \Sigma_\mathrm{V}$ the discrete node features, and $\mathbf{L} : \mathbf{E} \to \Sigma_\mathrm{E}$ the edge labels, where Σ_V and Σ_E are finite sets of symbols. The neighbourhood of a node $u \in \mathbf{V}$ with respect to an edge label ι is defined by $\mathbf{N}_\iota(u) = \{v \in \mathbf{V} \mid e = \langle u, v \rangle \in \mathbf{E} \wedge \mathbf{L}(e) = \iota\}$. Let $\mathcal{G} : \{\mathbf{P}\} \to \{\mathbf{G}\}$ denote a *graph representation function* that maps planning problems into graphs.

For our experiments, we use the *Instance Learning Graph* (ILG) (Chen, Trevizan, and Thiébaux 2024) representation function. Given a problem $\mathbf{P} = \langle \mathcal{O}, \mathcal{P}, \mathcal{A}, s_0, \mathcal{G} \rangle$ we define $\mathrm{ILG}(\mathbf{P}) = \langle \mathbf{V}, \mathbf{E}, \mathbf{F}, \mathbf{L} \rangle$ with $\mathbf{V} = \mathcal{O} \cup s_0 \cup \mathcal{G}$, $\mathbf{E} = \bigcup_{p=P(\mathbf{o}) \in s_0 \cup \mathcal{G}} \{\langle p, \mathbf{o}_1 \rangle, \langle \mathbf{o}_1, p \rangle, \ldots, \langle p_{\mathrm{ar}(P)}, \mathbf{o} \rangle, \langle \mathbf{o}_{\mathrm{ar}(P)}, p \rangle\}$, $\mathbf{F} : \mathbf{V} \to (\{\mathtt{ap}, \mathtt{ug}, \mathtt{ag}\} \times \mathcal{P}) \cup \{\mathtt{ob}\}$ defined by $u \mapsto \mathtt{ob}$ if $u \in \mathcal{O}$, $u \mapsto (\mathtt{ag}, P)$ if $u = P(\mathbf{o}) \in s_0 \cap \mathcal{G}$, $u \mapsto (\mathtt{ap}, P)$ if $u = P(\mathbf{o}) \in s_0 \setminus \mathcal{G}$, and $u \mapsto (\mathtt{ug}, P)$ if $u = P(\mathbf{o}) \in \mathcal{G} \setminus s_0$, and $\mathbf{L} : \mathbf{E} \to \mathbb{N}$ defined by $\langle p, \mathbf{o}_i \rangle \mapsto i$.

[1] Also known as 'relational structures' in other communities.

Step (2) transforms the graph into a multiset $\mathbf{M} = \{(f_i, v_i) \in \mathbb{N} \times \mathbb{N} \mid i \in [\![n]\!]\}$ that consists of a set of elements f_i and their counts v_i. The transformation is performed by the WL algorithm described in Algorithm 1 where the input is a graph $\mathbf{G}$ and a hyperparameter $L \in \mathbb{N}$. The underlying concept of the WL algorithm iteratively updates node colours based on the colours of their neighbours. Line 1 initialises node graph colours as their categorical node features. Lines 2 to 3 iteratively update the colour of each node v in the graph by collecting all its neighbours and the corresponding edge label (u, ι) into a multiset. This multiset is then hashed alongside v's current colour with an injective function to produce a new refined colour. In practice, the injective hash function is built lazily, where every time a new multiset is encountered, it is mapped to a new, unseen hash value. After L iterations, the multiset of all node colours seen throughout the algorithm is returned. Given a graph representation $\mathcal{G}$, we denote the set of WLF features of a given problem by

$$\mathrm{WL}^{\mathcal{G}}(\mathbf{P}) = \mathbf{M}. \tag{1}$$

Quantified-Both Novelty Heuristics A heuristic (function) is a function[2] $h : S \to \mathbb{R}$, where lower values are favoured for search. The *Quantified Both* (QB) novelty heuristic (Katz et al. 2017) creates a new heuristic h_QB from an existing heuristic h that favours novel states and distinguishes both novel and non-novel states using h. Given a problem and a set of states seen during search C, the QB heuristic is defined by

$$h_\mathrm{QB}(s) = \begin{cases} -\left|F_\mathrm{new}(s)\right|, & \text{if } \left|F_\mathrm{new}(s)\right| > 0, \\ +\left|F_\mathrm{old}(s)\right|, & \text{otherwise,} \end{cases} \tag{2}$$

$$F_\mathrm{new}(s) = \{p \in s \mid h(s) < \min_{t \in C, p \in t} h(t)\} \tag{3}$$

$$F_\mathrm{old}(s) = \{p \in s \mid h(s) > \min_{t \in C, p \in t} h(t)\}. \tag{4}$$

As discussed by Katz et al. (2017, page 7), the QB novelty heuristic differs from the Partition Novelty (PN) measure heuristic (Lipovetzky and Geffner 2017; Corrêa and Seipp 2022) in that PN does not favour states with lower h values, and does not try to distinguish states that are not novel. For example, two states with original h values of 9 and 7 that have the same PN values are distinguished by QB.

3 Symmetry-Invariant Weisfeiler-Leman Novelty Heuristics

One can generalise QB heuristics to take arbitrary features in place of atoms in the definitions of F_new and F_old in Equations (3) to (4). Specifically, let $\mathcal{F} : S \to 2^{\Sigma_\mathcal{F}}$ denote a *feature mapping* of states to sets of features in $\Sigma_\mathcal{F}$. An example is the feature $\mathrm{AT} : S \to S$ that maps a state to itself, the set of true atoms under the closed world assumption.

Then we define the *Generalised Quantified Both* novelty heuristic that takes as input a heuristic h, feature mapping $\mathcal{F}$

[2] Heuristic functions usually take nonnegative values. In the context of satisficing planning and greedy best first search, it is fine to exhibit negative heuristic values.

of states, and a set of states seen during search C by

$$h^{\mathcal{F}}_{\text{QB}}(s) = \begin{cases} -\left|\mathcal{F}_{\text{new}}(s)\right|, & \text{if } |\mathcal{F}_{\text{new}}(s)| > 0, \\ +\left|\mathcal{F}_{\text{old}}(s)\right|, & \text{otherwise,} \end{cases} \tag{5}$$

$$\mathcal{F}_{\text{new}}(s) = \{p \in \mathcal{F}(s) \mid h(s) < \min_{t \in C, p \in \mathcal{F}(t)} h(t)\} \tag{6}$$

$$\mathcal{F}_{\text{old}}(s) = \{p \in \mathcal{F}(s) \mid h(s) > \min_{t \in C, p \in \mathcal{F}(t)} h(t)\}. \tag{7}$$

Note that the original QB heuristic in Equation (2) is subsumed by the Generalised QB heuristic as $h_{\text{QB}} = h^{\text{AT}}_{\text{QB}}$. We will refer to $h^{\text{AT}}_{\text{QB}}$ as the *Atom Novelty Heuristic*. Similarly, we can define a QB heuristic using WL features from Equation (1) denoted by $h^{\text{WL}^{\mathcal{G}}}_{\text{QB}}$. We name $h^{\text{WL}^{\mathcal{G}}}_{\text{QB}}$ as the *Weisfeiler-Leman Novelty heuristic*. Note also that feature mappings can be combined to make new novelty heuristics, such as $h^{\text{AT;WL}^{\mathcal{G}}}_{\text{QB}}$ that uses the feature mapping $\text{AT; WL}^{\mathcal{G}} : S \to S \cup \Sigma_{\text{WL}^{\mathcal{G}}}$ defined by $s \mapsto s \cup \text{WL}^{\mathcal{G}}(\mathbf{P}_s)$.

We now sketch a proof of how WL novelty heuristics are symmetry invariant. Following Drexler et al. (2024, Definition 7) and Chen et al. (2025, Definition 16), we define an equivalence relation on states of a planning problem via bijection between objects, in contrast to work by Sievers et al. (2019) that reduce problems to graph automorphisms.

Definition 3.1 (Equivalence Relation)**.** Let $\mathbf{P}$ be a planning problem. We define a relation $\sim_U$ on states in $\mathbf{P}$ by $s_1 \sim_U s_2$ if there exists a permutation (bijective mapping from a set to itself) $f : \mathbf{P}[\mathcal{O}] \to \mathbf{P}[\mathcal{O}]$ such that $s_2 = \{P(f(o_1), \ldots, f(o_n)) \mid P(o_1, \ldots, o_n) \in s_1\}$.

Next, following Drexler et al. (2024, Theorem 12), we define a notion of symmetry-invariant graph representation for states in a planning problem.

Definition 3.2 (Symmetry-Invariant Graph Representation)**.** Let $\mathcal{G} : \{\mathbf{P}\} \to \{\mathbf{G}\}$ be a graph representation function. We say that $\mathcal{G}$ is a *symmetry-invariant graph representation function* if for any given problem $\mathbf{P}$, for any two states s, s' in $\mathbf{P}$, we have that $s \sim_U s'$ if and only if $\mathcal{G}(\mathbf{P}_s)$ is (graph) isomorphic to $\mathcal{G}(\mathbf{P}_{s'})$.

Now, we have a known fact that the WL algorithm is a graph invariant, meaning that the features output by Algorithm 1 are the same for isomorphic graphs.

Lemma 3.3 (Weisfeiler and Leman (1968))**.** *The WL algorithm is a graph invariant; i.e., if two graphs* $\mathbf{G}_1$ *and* $\mathbf{G}_2$ *are isomorphic, then* $\text{WL}(\mathbf{G}_1) = \text{WL}(\mathbf{G}_2)$.

We now state our main theoretical result following from the previous lemma that states that WLF Novelty heuristics are symmetry-invariant heuristics.

Proposition 3.4 (WL Novelty Heuristics are symmetry-invariant)**.** *Let* $\mathcal{G}$ *be a symmetry-invariant graph representation function. Then* $h^{\text{WL}^{\mathcal{G}}}_{\text{QB}}$ *is symmetry-invariant; i.e.* $s \sim_U s'$ *implies* $h^{\text{WL}^{\mathcal{G}}}_{\text{QB}}(s) = h^{\text{WL}^{\mathcal{G}}}_{\text{QB}}(s')$.

Proof Sketch. Let $s \sim_U s'$. Then $\mathcal{G}(\mathbf{P}_s)$ is isomorphic to $\mathcal{G}(\mathbf{P}_{s'})$ by the proposition assumption and Definition 3.2. Then by Lemma 3.3, $\text{WL}(\mathcal{G}(\mathbf{P}_s)) = \text{WL}(\mathcal{G}(\mathbf{P}_{s'}))$. Then Equation (6) is the same for s and s', and similarly for Equation (7) where $\mathcal{F}(t) := \text{WL}(\mathcal{G}(P_t))$ for all states t. □

We conclude this section with an example sketch of how the original (quantified both) novelty heuristics are not symmetry-invariant.

Example 3.5 (Atom Novelty Heuristics are not symmetry-invariant)**.** Let us consider the canonical Childsnack domain (Fuentetaja and de la Rosa 2016) where one must make sandwiches for children. Some children have hard constraints on what sandwiches they can eat, e.g. some are allergic to gluten. Gluten-free ingredients can be with non-gluten-free ingredients to make sandwiches that are not safe for children allergic to gluten. Delete-free heuristics such as h^{ff} perform badly as it thinks that making one gluten-free sandwich is sufficient for feeding all children and has to explore a large symmetric state-space to clear out deadends which occur when non-gluten-free sandwiches are made with gluten-free ingredients. For example consider the states {`exists(sw1, gluten-free)`, `exists(sw2, gluten)`, `exists(sw3, gluten)`} and {`exists(sw4, gluten-free)`, `exists(sw5, gluten)`, `exists(sw6, gluten)`} which are equivalent under the relation $\sim_U$ in Definition 3.1 if the goals are of the form $\bigwedge_{\text{c} \in \text{Children}}$ `served(c)`. However, atom novelty heuristics will see both states as novel if they have not encountered any of the facts before.

4 Experiments

We run experiments to help answer the following questions. Our preliminary results are summarised as answers in pink, but we refer to later in the text for a more detailed analysis.

- When does novelty help improve the base heuristic? When the base heuristic is informative so exploration helps.
- Are WL features useful for generating novelty heuristics? Yes in general, but not always.
- Is combining feature sets helpful for novelty heuristics? Yes in general, but not always.

Setup We experiment with two benchmark suites: the set of classical planning problems from the 1998 to 2023 International Planning Competitions (IPC), and the Hard To Ground (HTG) (Corrêa et al. 2020; Lauer et al. 2021). The latter benchmark is chosen to demonstrate that our approach is easily applicable to the lifted setting. For the IPC (resp. HTG) domains, we use heuristics and implement our approaches in Fast Downward Version 24.06[3] (resp. Powerlifted 2024[4]). We experiment with the goalcount (h^{gc}), additive (Bonet and Geffner 2001; Corrêa et al. 2021) (h^{add}), and FF (Hoffmann and Nebel 2001; Corrêa et al. 2022) (h^{ff}) heuristics. We consider their extensions under the Generalised Quantified Both framework described in Equation (5) with the feature sets consisting of mapping states to atoms (`at`), to WL features described in Algorithm 1 with $L = 2$ and the ILG as the underlying graph representation (`wl`), and the union of both feature sets (`at;wl`). All heuristics are used in single-queue greedy best first search (GBFS) in their respective planning engines with a 4GB memory limit

[3] https://github.com/aibasel/downward; release 24.06

[4] https://github.com/abcorrea/powerlifted; commit 736b0c

	h^{gc}(least informative)				h^{add}(informative)				h^{ff}(most informative)			
Σ Coverage	Base	`at`	`wl`	`at;wl`	Base	`at`	`wl`	`at;wl`	Base	`at`	`wl`	`at;wl`
Unnormalised	**1959**	1892	1858	1824	1868	1992	2035	**2096**	1800	1921	1977	**2094**
Normalised	**38.73**	37.07	37.78	35.87	40.57	43.00	42.90	**45.91**	40.25	42.51	40.61	**45.97**

Table 1: Total unnormalised and normalised coverage (↑) of heuristics for single-queue GBFS across different domains. Green/red cells indicate that a novelty heuristic (`at`, `wl`, `at;wl`) performs better/worse than their base heuristic (h^{gc}, h^{add}, h^{ff}). Blue cells indicate the best value per row. Bold font indicates the best score in each local heuristic group.

and 300 second timeout. We refer to Table 1 for total unnormalised and normalised[5] coverage scores, and Appendix A for coverage scores per domain.

When does novelty help improve the base heuristic? From Table 1, we notice an interesting trend: novelty heuristics degrade the performance of h^{gc}, but improve the h^{add} and h^{ff} heuristics. Both the normalised and unnormalised coverage of h^{gc} is higher than its novelty counterparts, and the performance degrades with novelty across domains more often than it improves. This can be explained in the exploration vs. exploitation viewpoint of search: novelty heuristics boost exploration, while the informativeness of a heuristic corresponds to exploitation. The h^{gc} heuristic is not very informative and thus adding novelty on top of it leads to more exploration than exploitation. Conversely, the h^{add} and h^{ff} heuristics are more informative heuristics in ascending order and result in greater coverage gains when used with novelty heuristics. Interestingly, we observe from Appendix B that novelty heuristics `at` and `wl` generally expand *more* nodes than their base heuristic on problems that are solved by both heuristics. This is reasonable as exploration from novelty heuristics is forcing more expansions that may not be necessary when the base heuristic is strong enough.

Are WL features useful for generating novelty heuristics? From the previous answer, we note that WL features generally improve upon the base heuristic if the base heuristic is sufficiently informative. Thus, we analyse how they compare to the original quantified both novelty heuristics. We observe from Table 1 that `wl` novelty heuristics have a lower normalised coverage than `at` novelty heuristics, but higher unnormalised coverage. This suggests that they are generally incomparable. However, when looking at select domains, as from the theoretical intuition, `wl` novelty heuristics perform better on domains where a heuristic gets stuck on symmetric states such as in Childsnack. Conversely, the WL algorithm is an incomplete graph isomorphism problem which means that it may mistakingly mark some asymmetric states as symmetric. This can have an adverse affect on exploration on domains where WL cannot distinguish asymmetric states which require to be expanded.

Is combining feature sets helpful for novelty heuristics? From the aforementioned tables, we notice that combining both `at` and `wl` features into a *single* novelty heuristic (`at;wl`) achieves the best scores overall for h^{add} and h^{ff}, and by a non-trivial margin (over 12% and 16% increase, respectively). This may be attributed to `at` and `wl` having diverse and contrasting features that are complementary to each other that can further guide exploration. Conversely, `at;wl` performs worse when using the least informative h^{gc} heuristic as explained in the first question.

[5]This refers to the coverage divided by the total number of problems in a domain solved by at least one configuration. Normalised coverage accounts for heavily skewed problem distributions, e.g. IPC Miconic and HTG Genome each have over 300 problems each.

5 Related Work, Discussion, and Conclusion

Symmetries are an extensively studied topic in planning. Most closely related to our work, Shleyfman et al. (2015) study whether existing domain-independent heuristics are symmetry invariant under a notion of symmetry that considers actions and action costs. On the other hand, novelty heuristics are transition agnostic which simplifies our analysis of symmetries. Orthogonally, symmetries have been used for state-space pruning in various planning settings and search algorithms (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012, 2013; Wehrle et al. 2015; Gnad et al. 2017; Shleyfman, Kuroiwa, and Beck 2023; Bai, Thiébaux, and Trevizan 2025). Drexler et al. (2024) leverage symmetries to reduce redundant computation in generating training data for generalised planners.

Existing novelty heuristics (Lipovetzky and Geffner 2017; Katz et al. 2017; Ramírez et al. 2018; Teichteil-Königsbuch, Ramírez, and Lipovetzky 2020; Chen and Thiébaux 2024b) all derive their features from atoms and numeric variable assignments. Our work differs in generalising the novelty heuristic framework that takes as input a feature generator on top of a base heuristic, as described in Equation (5). This simple insight allows us to make use of other existing work on generating features for planning problems such as description logic features (Martín and Geffner 2004) and embeddings from pretrained, domain-independent neural networks (Shen, Trevizan, and Thiébaux 2020; Chen, Thiébaux, and Trevizan 2024) as examples.

Our approach is simple yet opens up many new avenues of research. On the empirical side, we can further boost planning performance. This involves experimenting with additional WL parameters (Chen 2025) and other feature generators, as well as orthogonal search techniques such as using lazy heuristic evaluation, preferred operators, and multiple queues. On the theoretical side, one can extend and generalise the complexity theory of width, serialisation and subgoaling (Lipovetzky and Geffner 2012; Dold and Helmert 2024; Drexler, Seipp, and Geffner 2024) for planning.

References

Bai, Y.; Thiébaux, S.; and Trevizan, F. 2025. Learning Efficiency Meets Symmetry Breaking. In *ICAPS*.

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

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

Chen, D. Z.; Hofmann, T.; Klassen, T. Q.; and McIlraith, S. A. 2025. MOOSE: Satisficing and Optimal Generalised Planning via Goal Regression. In *Proceedings of the First International Workshop on Trends in Knowledge Representation and Reasoning (TKR)*.

Chen, D. Z.; and Thiébaux, S. 2024a. Graph Learning for Numeric Planning. In *NeurIPS*.

Chen, D. Z.; and Thiébaux, S. 2024b. Novelty Heuristics, Multi-Queue Search, and Portfolios for Numeric Planning. In *SOCS*.

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

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

Corrêa, A. B.; Francès, G.; Pommerening, F.; and Helmert, M. 2021. Delete-Relaxation Heuristics for Lifted Classical Planning. In *ICAPS*.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2020. Lifted Successor Generation Using Query Optimization Techniques. In *ICAPS*.

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

Corrêa, A. B.; and Seipp, J. 2022. Best-First Width Search for Lifted Classical Planning. In *ICAPS*.

Corrêa, A. B.; and Seipp, J. 2024. Consolidating LAMA with Best-First Width Search. *CoRR*, abs/2404.17648.

Dold, S.; and Helmert, M. 2024. Novelty vs. Potential Heuristics: A Comparison of Hardness Measures for Satisficing Planning. In *AAAI*.

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

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry Breaking: Satisficing Planning and Landmark Heuristics. In *ICAPS*.

Drexler, D.; Seipp, J.; and Geffner, H. 2024. Expressing and Exploiting Subgoal Structure in Classical Planning Using Sketches. *J. Artif. Intell. Res.*, 80.

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

Francès, G.; Ramírez, M.; Lipovetzky, N.; and Geffner, H. 2017. Purely Declarative Action Descriptions are Overrated: Classical Planning with Simulators. In *IJCAI*.

Fuentetaja, R.; and de la Rosa, T. 2016. Compiling irrelevant objects to counters. Special case of creation planning. *AI Commun.*, 29(3): 435–467.

Geffner, H.; and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers.

Ghallab, M.; Knoblock, C.; Wilkins, D.; Barrett, A.; Christianson, D.; Friedman, M.; Kwok, C.; Golden, K.; Penberthy, S.; Smith, D.; Sun, Y.; and Weld, D. 1998. PDDL – The Planning Domain Definition Language. Technical report.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry Breaking in Star-Topology Decoupled Search. In *ICAPS*.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool Publishers.

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

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting Novelty to Classical Planning as Heuristic Search. In *ICAPS*.

Lauer, P.; Torralba, Á.; Fiser, D.; Höller, D.; Wichlacz, J.; and Hoffmann, J. 2021. Polynomial-Time in PDDL Input Size: Making the Delete Relaxation Feasible for Lifted Planning. In *IJCAI*.

Lei, C.; Lipovetzky, N.; and Ehinger, K. A. 2023. Novelty and Lifted Helpful Actions in Generalized Planning. In *SOCS*.

Lipovetzky, N.; and Geffner, H. 2012. Width and Serialization of Classical Planning Problems. In *ECAI*.

Lipovetzky, N.; and Geffner, H. 2017. Best-First Width Search: Exploration and Exploitation in Classical Planning. In *AAAI*.

Martín, M.; and Geffner, H. 2004. Learning Generalized Policies from Planning Examples Using Concept Languages. *Appl. Intell.*, 20(1): 9–19.

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

Ramírez, M.; Papasimeon, M.; Lipovetzky, N.; Benke, L.; Miller, T.; Pearce, A. R.; Scala, E.; and Zamani, M. 2018. Integrated Hybrid Planning and Programmed Control for Real Time UAV Maneuvering. In *AAMAS*.

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

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and Symmetries in Classical Planning. In *AAAI*.

Shleyfman, A.; Kuroiwa, R.; and Beck, J. C. 2023. Symmetry Detection and Breaking in Linear Cost-Optimal Numeric Planning. In *ICAPS*.

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

Teichteil-Königsbuch, F.; Ramírez, M.; and Lipovetzky, N. 2020. Boundary Extension Features for Width-Based Planning with Simulators on Continuous-State Domains. In *IJCAI*.

Wang, R.; and Trevizan, F. 2025. Leveraging Action Relational Structures for Integrated Learning and Planning. In *ICAPS*.

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

Weisfeiler, B.; and Leman, A. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9).

Zhang, M. J. 2024. *Learning for Planning Under Uncertainty: Predicting Features of SSPs with Novel Graph Representation*. Bachelor's thesis, The Australian National University.

A Coverage Results

	h^{gc} (least informative)				h^{add} (informative)				h^{ff} (most informative)			
Domain	Base	at	wl	at;wl	Base	at	wl	at;wl	Base	at	wl	at;wl
agricola	0	**3**	0	0	**11**	4	0	3	**6**	5	0	3
assembly	0	0	**10**	7	11	**30**	27	23	**30**	**30**	25	27
barman	4	0	**5**	0	0	0	**1**	0	2	**11**	4	4
blocks	**35**	33	34	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
caldera	**16**	13	14	10	**18**	**18**	17	16	12	16	12	**17**
cavediving	**7**	**7**	**7**	**7**	**7**	6	**7**	**7**	**7**	**7**	**7**	**7**
childsnack	0	0	0	0	2	1	1	**4**	0	0	1	**3**
citycar	0	0	5	**8**	**20**	15	**20**	**20**	0	8	**19**	**19**
data	1	1	**4**	3	1	**3**	**3**	**3**	**4**	3	2	**4**
depot	14	16	13	**17**	13	19	**21**	**21**	15	17	19	**21**
driverlog	**19**	**19**	18	**19**	18	**19**	17	**19**	17	19	18	**20**
elevators	33	**39**	24	29	15	**30**	**30**	**30**	11	8	16	**19**
floortile	0	0	**4**	3	**10**	9	8	8	**10**	6	**10**	**10**
folding	8	**9**	6	6	0	0	**3**	**3**	0	2	**3**	**3**
freecell	46	**57**	43	53	**80**	77	75	79	**79**	**79**	76	77
ged	**20**	**20**	19	19	0	**11**	8	9	0	0	**3**	1
grid	**3**	**3**	**3**	**3**	3	**5**	2	**5**	4	**5**	2	**5**
gripper	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
hiking	**2**	**2**	**2**	**2**	19	18	**20**	19	**20**	**20**	**20**	**20**
labyrinth	**4**	2	3	2	0	0	0	0	0	0	0	0
logistics	35	**36**	34	**36**	**53**	48	51	52	**55**	50	53	52
maintenance	14	**16**	7	9	16	**17**	10	13	**11**	7	6	6
miconic	**371**	**371**	364	363	**439**	436	427	**439**	437	**439**	**439**	436
movie	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
mprime	21	**22**	21	21	31	**35**	**35**	**35**	30	**35**	**35**	**35**
mystery	15	15	14	**16**	18	**19**	**19**	**19**	17	**19**	**19**	**19**
nomystery	6	**10**	9	8	6	**12**	6	11	9	**17**	8	15
nurikabe	**13**	**13**	12	12	9	7	8	**10**	7	**11**	7	9
openstacks	**40**	34	33	28	29	28	**34**	**34**	36	36	**38**	**38**
optical	5	4	**18**	**18**	6	**21**	16	15	4	**13**	10	7
parking	0	0	0	0	10	16	21	**23**	23	5	29	**31**
pegsol	**50**	46	44	44	**50**	44	46	48	**50**	46	48	48
philosophers	20	**26**	11	12	**47**	28	24	22	**48**	**48**	17	24
pipesworld	61	**65**	52	60	43	80	76	**84**	50	81	74	**84**
psr	**92**	79	69	69	**82**	78	69	69	58	56	**73**	**73**
recharging	**12**	11	8	7	9	**10**	8	7	8	**11**	8	7
ricochet	**4**	0	2	0	**4**	3	0	1	**12**	5	2	1
rovers	**21**	**21**	17	19	26	**30**	26	28	24	**31**	30	29
rubiks	4	**6**	4	4	4	**8**	5	5	**20**	9	6	6
satellite	14	**18**	13	14	**30**	26	28	27	**27**	22	26	25
scanalyzer	**50**	48	36	48	**50**	48	48	46	**46**	**46**	44	**46**
schedule	78	63	**106**	98	19	77	**128**	123	31	77	**143**	140
settlers	0	0	0	0	**5**	3	3	4	0	0	1	**2**
slitherlink	0	0	0	0	0	0	0	0	0	0	0	0
snake	4	**6**	3	2	3	4	**11**	9	5	6	9	**10**
sokoban	**36**	28	15	15	**46**	45	23	38	**46**	42	22	34
spider	11	**12**	10	9	7	2	**12**	11	12	14	14	**18**
storage	18	**20**	16	**20**	16	26	18	**27**	18	**26**	18	24
termes	**9**	1	0	0	**4**	0	1	1	**12**	1	0	1
tetris	**19**	16	16	13	**16**	14	10	12	4	13	4	**15**
thoughtful	**5**	**5**	**5**	**5**	13	**19**	15	16	8	17	13	**19**
tidybot	19	**20**	19	**20**	17	17	**18**	17	16	17	**18**	**18**
tpp	13	**15**	**15**	**15**	21	**28**	20	21	19	**28**	20	19
transport	**41**	27	27	26	36	25	**40**	36	12	27	21	**34**
trucks	9	12	**16**	13	15	14	20	**21**	14	18	20	**23**
visitall	**40**	21	37	12	3	7	5	**10**	3	**16**	1	14
woodworking	15	**25**	19	17	35	48	48	**49**	41	46	**50**	**50**
zenotravel	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
Σ IPC	**1447**	1406	1356	1346	1551	1693	1694	**1757**	1535	1676	1668	**1777**
blocksworld	9	2	**15**	13	0	1	**3**	**3**	2	1	**3**	2
childsnack	**19**	2	17	14	23	23	**64**	62	23	23	62	**65**
genome	**312**	252	260	261	100	**106**	89	81	45	43	**63**	57
labyrinth	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**	**40**
logistics	3	1	**31**	3	**1**	0	0	0	0	0	0	**2**
organic	44	**47**	45	45	31	32	30	**34**	33	**34**	32	32
pipesworld	23	31	23	**34**	19	**28**	27	26	17	**30**	22	29
rovers	0	0	**2**	1	**7**	0	3	3	**11**	0	3	3
visitall	62	**111**	69	67	**96**	69	85	90	**94**	74	84	87
Σ HTG	**512**	486	502	478	317	299	**341**	339	265	245	309	**317**
Σ	**1959**	1892	1858	1824	1868	1992	2035	**2096**	1800	1921	1977	**2094**

Table 2: Coverage (↑) of heuristics for single-queue GBFS across different domains. Green/red cells indicate that a novelty heuristic (`at`, `wl`, `at;wl`) performs better/worse than their base heuristic (h^{gc}, h^{add}, h^{ff}). Blue cells indicate the best value per row. Bold font indicates the best score in each local heuristic group.

	h^{gc} (least informative)				h^{add} (informative)				h^{ff} (most informative)			
Domain	Base	`at`	`wl`	`at;wl`	Base	`at`	`wl`	`at;wl`	Base	`at`	`wl`	`at;wl`
agricola	0.00	**0.23**	0.00	0.00	**0.85**	0.31	0.00	0.23	**0.46**	0.38	0.00	0.23
assembly	0.00	0.00	**0.33**	0.23	0.37	**1.00**	0.90	0.77	**1.00**	**1.00**	0.83	0.90
barman	0.25	0.00	**0.31**	0.00	0.00	0.00	**0.06**	0.00	0.12	**0.69**	0.25	0.25
blocks	**1.00**	0.94	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
caldera	**0.70**	0.57	0.61	0.43	**0.78**	**0.78**	0.74	0.70	0.52	0.70	0.52	**0.74**
cavediving	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.86	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
childsnack	0.00	0.00	0.00	0.00	0.50	0.25	0.25	**1.00**	0.00	0.00	0.25	**0.75**
citycar	0.00	0.00	0.25	**0.40**	**1.00**	0.75	**1.00**	**1.00**	0.00	0.40	**0.95**	**0.95**
data	0.17	0.17	**0.67**	0.50	0.17	**0.50**	**0.50**	**0.50**	**0.67**	0.50	0.33	**0.67**
depot	0.67	0.76	0.62	**0.81**	0.62	0.90	**1.00**	**1.00**	0.71	0.81	0.90	**1.00**
driverlog	**0.95**	**0.95**	0.90	**0.95**	0.90	**0.95**	0.85	**0.95**	0.85	0.95	0.90	**1.00**
elevators	0.85	**1.00**	0.62	0.74	0.38	**0.77**	**0.77**	**0.77**	0.28	0.21	0.41	**0.49**
floortile	0.00	0.00	**0.27**	0.20	**0.67**	0.60	0.53	0.53	**0.67**	0.40	**0.67**	**0.67**
folding	0.73	**0.82**	0.55	0.55	0.00	0.00	**0.27**	**0.27**	0.00	0.18	**0.27**	**0.27**
freecell	0.58	**0.71**	0.54	0.66	**1.00**	0.96	0.94	0.99	**0.99**	**0.99**	0.95	0.96
ged	**1.00**	**1.00**	0.95	0.95	0.00	**0.55**	0.40	0.45	0.00	0.00	**0.15**	0.05
grid	**0.60**	**0.60**	**0.60**	**0.60**	0.60	**1.00**	0.40	**1.00**	0.80	**1.00**	0.40	**1.00**
gripper	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
hiking	**0.10**	**0.10**	**0.10**	**0.10**	0.95	0.90	**1.00**	0.95	**1.00**	**1.00**	**1.00**	**1.00**
labyrinth	**1.00**	0.50	0.75	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
logistics	0.61	**0.63**	0.60	**0.63**	**0.93**	0.84	0.89	0.91	**0.96**	0.88	0.93	0.91
maintenance	0.82	**0.94**	0.41	0.53	0.94	**1.00**	0.59	0.76	**0.65**	0.41	0.35	0.35
miconic	**0.85**	**0.85**	0.83	0.83	**1.00**	0.99	0.97	**1.00**	**1.00**	**1.00**	**1.00**	0.99
movie	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
mprime	0.60	**0.63**	0.60	0.60	0.89	**1.00**	**1.00**	**1.00**	0.86	**1.00**	**1.00**	**1.00**
mystery	0.79	0.79	0.74	**0.84**	0.95	**1.00**	**1.00**	**1.00**	0.89	**1.00**	**1.00**	**1.00**
nomystery	0.33	**0.56**	0.50	0.44	0.33	**0.67**	0.33	0.61	0.50	**0.94**	0.44	0.83
nurikabe	**0.93**	**0.93**	0.86	0.86	0.64	0.50	0.57	**0.71**	0.50	**0.79**	0.50	0.64
openstacks	**0.70**	0.60	0.58	0.49	0.51	0.49	**0.60**	**0.60**	0.63	0.63	**0.67**	**0.67**
optical	0.24	0.19	**0.86**	**0.86**	0.29	**1.00**	0.76	0.71	0.19	**0.62**	0.48	0.33
parking	0.00	0.00	0.00	0.00	0.28	0.44	0.58	**0.64**	0.64	0.14	0.81	**0.86**
pegsol	**1.00**	0.92	0.88	0.88	**1.00**	0.88	0.92	0.96	**1.00**	0.92	0.96	0.96
philosophers	0.42	**0.54**	0.23	0.25	**0.98**	0.58	0.50	0.46	**1.00**	**1.00**	0.35	0.50
pipesworld	0.71	**0.76**	0.60	0.70	0.50	0.93	0.88	**0.98**	0.58	0.94	0.86	**0.98**
psr	**1.00**	0.86	0.75	0.75	**0.89**	0.85	0.75	0.75	0.63	0.61	**0.79**	**0.79**
recharging	**0.86**	0.79	0.57	0.50	0.64	**0.71**	0.57	0.50	0.57	**0.79**	0.57	0.50
ricochet	**0.29**	0.00	0.14	0.00	**0.29**	0.21	0.00	0.07	**0.86**	0.36	0.14	0.07
rovers	**0.66**	**0.66**	0.53	0.59	0.81	**0.94**	0.81	0.88	0.75	**0.97**	0.94	0.91
rubiks	0.20	**0.30**	0.20	0.20	0.20	**0.40**	0.25	0.25	**1.00**	0.45	0.30	0.30
satellite	0.47	**0.60**	0.43	0.47	**1.00**	0.87	0.93	0.90	**0.90**	0.73	0.87	0.83
scanalyzer	**1.00**	0.96	0.72	0.96	**1.00**	0.96	0.96	0.92	**0.92**	**0.92**	0.88	**0.92**
schedule	0.52	0.42	**0.71**	0.66	0.13	0.52	**0.86**	0.83	0.21	0.52	**0.96**	0.94
settlers	0.00	0.00	0.00	0.00	**0.71**	0.43	0.43	0.57	0.00	0.00	0.14	**0.29**
slitherlink	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
snake	0.36	**0.55**	0.27	0.18	0.27	0.36	**1.00**	0.82	0.45	0.55	0.82	**0.91**
sokoban	**0.75**	0.58	0.31	0.31	**0.96**	0.94	0.48	0.79	**0.96**	0.88	0.46	0.71
spider	0.58	**0.63**	0.53	0.47	0.37	0.11	**0.63**	0.58	0.63	0.74	0.74	**0.95**
storage	0.64	**0.71**	0.57	**0.71**	0.57	0.93	0.64	**0.96**	0.64	**0.93**	0.64	0.86
termes	**0.75**	0.08	0.00	0.00	**0.33**	0.00	0.08	0.08	**1.00**	0.08	0.00	0.08
tetris	**0.95**	0.80	0.80	0.65	**0.80**	0.70	0.50	0.60	0.20	0.65	0.20	**0.75**
thoughtful	**0.25**	**0.25**	**0.25**	**0.25**	0.65	**0.95**	0.75	0.80	0.40	0.85	0.65	**0.95**
tidybot	0.95	**1.00**	0.95	**1.00**	0.85	0.85	**0.90**	0.85	0.80	0.85	**0.90**	**0.90**
tpp	0.46	**0.54**	**0.54**	**0.54**	0.75	**1.00**	0.71	0.75	0.68	**1.00**	0.71	0.68
transport	**0.87**	0.57	0.57	0.55	0.77	0.53	**0.85**	0.77	0.26	0.57	0.45	**0.72**
trucks	0.36	0.48	**0.64**	0.52	0.60	0.56	0.80	**0.84**	0.56	0.72	0.80	**0.92**
visitall	**1.00**	0.53	0.93	0.30	0.08	0.18	0.12	**0.25**	0.08	**0.40**	0.03	0.35
woodworking	0.30	**0.50**	0.38	0.34	0.70	0.96	0.96	**0.98**	0.82	0.92	**1.00**	**1.00**
zenotravel	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Σ IPC	**33.80**	32.48	31.51	30.50	36.38	39.36	38.22	**41.19**	35.79	38.95	36.13	**41.29**
blocksworld	0.60	0.13	**1.00**	0.87	0.00	0.07	**0.20**	**0.20**	0.13	0.07	**0.20**	0.13
childsnack	**0.29**	0.03	0.26	0.22	0.35	0.35	**0.98**	0.95	0.35	0.35	0.95	**1.00**
genome	**1.00**	0.81	0.83	0.84	0.32	**0.34**	0.29	0.26	0.14	0.14	**0.20**	0.18
labyrinth	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
logistics	0.09	0.03	**0.97**	0.09	**0.03**	0.00	0.00	0.00	0.00	0.00	0.00	**0.06**
organic	0.92	**0.98**	0.94	0.94	0.65	0.67	0.62	**0.71**	0.69	**0.71**	0.67	0.67
pipesworld	0.55	0.74	0.55	**0.81**	0.45	**0.67**	0.64	0.62	0.40	**0.71**	0.52	0.69
rovers	0.00	0.00	**0.18**	0.09	**0.64**	0.00	0.27	0.27	**1.00**	0.00	0.27	0.27
visitall	0.48	**0.87**	0.54	0.52	**0.75**	0.54	0.66	0.70	**0.73**	0.58	0.66	0.68
Σ HTG	4.93	4.59	**6.27**	5.37	4.19	3.63	4.67	**4.72**	4.46	3.56	4.48	**4.69**
Σ	**38.73**	37.07	37.78	35.87	40.57	43.00	42.90	**45.91**	40.25	42.51	40.61	**45.97**

Table 3: Normalised coverage (↑) of heuristics for single-queue GBFS across different domains. Green/red cells indicate that a novelty heuristic (`at`, `wl`, `at;wl`) performs better/worse than their base heuristic (h^{gc}, h^{add}, h^{ff}). Blue cells indicate the best value per row. Bold font indicates the best score in each local heuristic group.

B Expansion Results

Domain	Base	at	Base	wl	Base	at;wl	at	wl
agricola	2	**5**	**6**	0	**3**	**3**	**5**	0
assembly	**30**	0	**29**	1	**29**	1	**25**	5
barman	1	**11**	2	**4**	2	**4**	**9**	4
blocks	15	**20**	12	**23**	11	**23**	9	**24**
caldera	8	**9**	**7**	**7**	6	**10**	6	**10**
cavediving	0	**7**	0	**7**	0	**7**	0	**7**
childsnack	0	0	0	**1**	0	**3**	0	**1**
citycar	0	**8**	0	**19**	0	**19**	1	**18**
data	1	**3**	**2**	**2**	1	**4**	**3**	0
depot	8	**10**	5	**15**	3	**18**	3	**17**
driverlog	**16**	3	**17**	3	**16**	4	5	**15**
elevators	5	**6**	0	**16**	1	**18**	2	**14**
floortile	**10**	0	4	**7**	**5**	**5**	0	**10**
folding	0	**2**	0	**3**	0	**3**	**2**	**2**
freecell	**61**	17	**62**	17	**57**	22	**48**	32
ged	0	0	0	**3**	0	**1**	0	**3**
grid	2	**3**	**4**	0	2	**3**	**4**	1
gripper	**20**	0	0	**20**	0	**20**	0	**20**
hiking	**10**	**10**	2	**18**	2	**18**	4	**16**
labyrinth	0	0	0	0	0	0	0	0
logistics	**53**	1	**54**	2	**53**	2	1	**51**
maintenance	**11**	3	**11**	2	**11**	2	1	**6**
miconic	**404**	17	**397**	20	**402**	19	138	**280**
movie	**30**	0	**30**	0	**30**	0	0	0
mprime	6	**15**	9	**13**	9	**13**	**11**	5
mystery	**4**	**4**	**5**	3	**4**	3	**4**	1
nomystery	4	**13**	**6**	4	5	**10**	**14**	3
nurikabe	2	**9**	**5**	3	3	**6**	**8**	3
openstacks	2	**34**	0	**38**	0	**38**	8	**29**
optical	0	**13**	0	**10**	0	**7**	**10**	2
parking	**22**	1	9	**26**	5	**26**	1	**29**
pegsol	**45**	5	**33**	17	**36**	14	15	**32**
philosophers	18	**30**	**48**	0	**48**	0	**47**	1
pipesworld	18	**60**	14	**56**	15	**65**	29	**50**
psr	**38**	21	24	**47**	27	**46**	22	**51**
recharging	5	**6**	4	**6**	4	**5**	**7**	5
ricochet	**10**	2	**11**	2	**11**	1	**5**	1
rovers	**16**	14	**16**	12	13	**15**	8	**23**
rubiks	**14**	2	**19**	1	**15**	1	**9**	0
satellite	**25**	3	**19**	8	**22**	5	1	**25**
scanalyzer	**37**	10	**27**	23	23	**24**	19	**27**
schedule	8	**65**	3	**134**	3	**131**	24	**113**
settlers	0	0	0	**1**	0	**2**	0	**1**
snake	2	**5**	0	**9**	0	**10**	2	**7**
sokoban	**41**	7	**43**	1	**43**	5	**37**	5
spider	**9**	7	7	**10**	6	**13**	5	**12**
storage	8	**14**	**12**	3	7	**13**	**18**	4
termes	**11**	1	**12**	0	**11**	1	**1**	0
tetris	1	**13**	**4**	2	1	**14**	**13**	0
thoughtful	8	**9**	**8**	6	8	**11**	**14**	3
tidybot	**12**	5	**13**	5	**12**	6	6	**12**
tpp	3	**20**	5	**11**	5	**11**	**21**	3
transport	5	**21**	2	**18**	1	**32**	12	**14**
trucks	9	**10**	6	**14**	5	**18**	4	**18**
visitall	0	**16**	**3**	0	0	**14**	**16**	0
woodworking	**35**	12	**33**	15	**33**	15	6	**39**
zenotravel	**17**	0	**14**	5	**11**	6	3	**16**
Σ IPC	**1122**	582	**1058**	693	**1020**	790	666	**1070**
blocksworld	**2**	0	**1**	**1**	**1**	0	0	**3**
childsnack	9	**18**	1	**61**	1	**64**	0	**62**
genome	**28**	16	24	**50**	23	**45**	21	**52**
labyrinth	2	**7**	**31**	9	0	**9**	**31**	9
logistics	0	0	0	0	0	**2**	0	0
organic	**3**	1	**4**	1	**4**	1	**2**	1
pipesworld	10	**19**	10	**12**	9	**20**	13	**17**
rovers	**11**	0	**9**	0	**10**	0	0	**3**
visitall	**46**	10	**31**	28	24	**33**	11	**41**
Σ HTG	**111**	71	111	**162**	72	**174**	78	**188**
Σ	**1233**	653	**1169**	855	**1092**	964	744	**1258**

Table 4: Problems per domain where a column expands fewer nodes than its paired column. The higher value in each pair is indicated in bold. The (non-)novelty extensions are over the h^{ff} heuristic.

Domain	Base	at	Base	wl	Base	at;wl	at	wl
agricola	0.10	**0.25**	**0.30**	0.00	**0.15**	**0.15**	**0.25**	0.00
assembly	**1.00**	0.00	**0.97**	0.03	**0.97**	0.03	**0.83**	0.17
barman	0.03	**0.28**	0.05	**0.10**	0.05	**0.10**	**0.23**	0.10
blocks	0.43	**0.57**	0.34	**0.66**	0.31	**0.66**	0.26	**0.69**
caldera	0.20	**0.23**	**0.17**	**0.17**	0.15	**0.25**	0.15	**0.25**
cavediving	0.00	**0.35**	0.00	**0.35**	0.00	**0.35**	0.00	**0.35**
childsnack	0.00	0.00	0.00	**0.05**	0.00	**0.15**	0.00	**0.05**
citycar	0.00	**0.40**	0.00	**0.95**	0.00	**0.95**	0.05	**0.90**
data	0.05	**0.15**	**0.10**	**0.10**	0.05	**0.20**	**0.15**	0.00
depot	0.36	**0.45**	0.23	**0.68**	0.14	**0.82**	0.14	**0.77**
driverlog	**0.80**	0.15	**0.85**	0.15	**0.80**	0.20	0.25	**0.75**
elevators	0.10	**0.12**	0.00	**0.32**	0.02	**0.36**	0.04	**0.28**
floortile	**0.25**	0.00	0.10	**0.17**	**0.12**	**0.12**	0.00	**0.25**
folding	0.00	**0.10**	0.00	**0.15**	0.00	**0.15**	**0.10**	**0.10**
freecell	**0.76**	0.21	**0.78**	0.21	**0.71**	0.28	**0.60**	0.40
ged	0.00	0.00	0.00	**0.15**	0.00	**0.05**	0.00	**0.15**
grid	0.40	**0.60**	**0.80**	0.00	0.40	**0.60**	**0.80**	0.20
gripper	**1.00**	0.00	0.00	**1.00**	0.00	**1.00**	0.00	**1.00**
hiking	**0.50**	**0.50**	0.10	**0.90**	0.10	**0.90**	0.20	**0.80**
labyrinth	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
logistics	**0.84**	0.02	**0.86**	0.03	**0.84**	0.03	0.02	**0.81**
maintenance	**0.55**	0.15	**0.55**	0.10	**0.55**	0.10	0.05	**0.30**
miconic	**0.90**	0.04	**0.88**	0.04	**0.89**	0.04	0.31	**0.62**
movie	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00	0.00	0.00
mprime	0.17	**0.43**	0.26	**0.37**	0.26	**0.37**	**0.31**	0.14
mystery	**0.13**	**0.13**	**0.17**	0.10	**0.13**	0.10	**0.13**	0.03
nomystery	0.20	**0.65**	**0.30**	0.20	0.25	**0.50**	**0.70**	0.15
nurikabe	0.10	**0.45**	**0.25**	0.15	0.15	**0.30**	**0.40**	0.15
openstacks	0.03	**0.57**	0.00	**0.63**	0.00	**0.63**	0.13	**0.48**
optical	0.00	**0.27**	0.00	**0.21**	0.00	**0.15**	**0.21**	0.04
parking	**0.55**	0.03	0.23	**0.65**	0.12	**0.65**	0.03	**0.72**
pegsol	**0.90**	0.10	**0.66**	0.34	**0.72**	0.28	0.30	**0.64**
philosophers	0.38	**0.62**	**1.00**	0.00	**1.00**	0.00	**0.98**	0.02
pipesworld	0.18	**0.59**	0.14	**0.55**	0.15	**0.64**	0.29	**0.50**
psr	**0.38**	0.21	0.24	**0.47**	0.27	**0.46**	0.22	**0.51**
recharging	0.25	**0.30**	0.20	**0.30**	0.20	**0.25**	**0.35**	0.25
ricochet	**0.50**	0.10	**0.55**	0.10	**0.55**	0.05	**0.25**	0.05
rovers	**0.40**	0.35	**0.40**	0.30	0.33	**0.38**	0.20	**0.57**
rubiks	**0.70**	0.10	**0.95**	0.05	**0.75**	0.05	**0.45**	0.00
satellite	**0.69**	0.08	**0.53**	0.22	**0.61**	0.14	0.03	**0.69**
scanalyzer	**0.74**	0.20	**0.54**	0.46	0.46	**0.48**	0.38	**0.54**
schedule	0.05	**0.43**	0.02	**0.89**	0.02	**0.87**	0.16	**0.75**
settlers	0.00	0.00	0.00	**0.05**	0.00	**0.10**	0.00	**0.05**
snake	0.10	**0.25**	0.00	**0.45**	0.00	**0.50**	0.10	**0.35**
sokoban	**0.82**	0.14	**0.86**	0.02	**0.86**	0.10	**0.74**	0.10
spider	**0.45**	0.35	0.35	**0.50**	0.30	**0.65**	0.25	**0.60**
storage	0.27	**0.47**	**0.40**	0.10	0.23	**0.43**	**0.60**	0.13
termes	**0.55**	0.05	**0.60**	0.00	**0.55**	0.05	**0.05**	0.00
tetris	0.05	**0.65**	**0.20**	0.10	0.05	**0.70**	**0.65**	0.00
thoughtful	0.40	**0.45**	**0.40**	0.30	0.40	**0.55**	**0.70**	0.15
tidybot	**0.60**	0.25	**0.65**	0.25	**0.60**	0.30	0.30	**0.60**
tpp	0.10	**0.67**	0.17	**0.37**	0.17	**0.37**	**0.70**	0.10
transport	0.07	**0.30**	0.03	**0.26**	0.01	**0.46**	0.17	**0.20**
trucks	0.30	**0.33**	0.20	**0.47**	0.17	**0.60**	0.13	**0.60**
visitall	0.00	**0.40**	**0.07**	0.00	0.00	**0.35**	**0.40**	0.00
woodworking	**0.70**	0.24	**0.66**	0.30	**0.66**	0.30	0.12	**0.78**
zenotravel	**0.85**	0.00	**0.70**	0.25	**0.55**	0.30	0.15	**0.80**
Σ IPC	**20.88**	14.73	**19.79**	15.74	17.78	**19.55**	15.00	**19.65**
blocksworld	**0.05**	0.00	**0.03**	**0.03**	**0.03**	0.00	0.00	**0.07**
childsnack	0.06	**0.12**	0.01	**0.42**	0.01	**0.44**	0.00	**0.43**
genome	**0.06**	0.03	0.05	**0.11**	0.05	**0.10**	0.04	**0.11**
labyrinth	0.05	**0.17**	**0.78**	0.23	0.00	**0.23**	**0.78**	0.23
logistics	0.00	0.00	0.00	0.00	0.00	**0.05**	0.00	0.00
organic	**0.05**	0.02	**0.07**	0.02	**0.07**	0.02	**0.04**	0.02
pipesworld	0.20	**0.38**	0.20	**0.24**	0.18	**0.40**	0.26	**0.34**
rovers	**0.28**	0.00	**0.23**	0.00	**0.25**	0.00	0.00	**0.07**
visitall	**0.26**	0.06	**0.17**	0.16	0.13	**0.18**	0.06	**0.23**
Σ HTG	**1.01**	0.79	**1.53**	1.19	0.72	**1.42**	1.18	**1.50**
Σ	**21.89**	15.52	**21.32**	16.93	18.50	**20.97**	16.18	**21.15**

Table 5: Percentage of problems per domain where a column expands fewer nodes than its paired column. The higher value in each pair is indicated in bold. The (non-)novelty extensions are over the h^{ff} heuristic.