Homomorphisms of Lifted Planning Tasks: The Case for Delete-free Relaxation Heuristics

Rostislav Horčík[1], **Daniel Fišer**[1,2], **Álvaro Torralba**[3]

[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Prague, Czech Republic
[2] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[3] Aalborg University, Denmark
xhorcik@fel.cvut.cz, danfis@danfis.cz, alto@cs.aau.dk

Abstract

Classical planning tasks are modelled in PDDL which is a schematic language based on first-order logic. Most of the current planners turn this lifted representation into a propositional one via a grounding process. However, grounding may cause an exponential blowup. Therefore it is important to investigate methods for searching for plans on the lifted level. To build a lifted state-based planner, it is necessary to invent lifted heuristics. We introduce maps between PDDL tasks preserving plans allowing to transform a PDDL task into a smaller one. We propose a novel method for computing lifted (admissible) delete-free relaxed heuristics via grounding of the smaller task and computing the (admissible) delete-free relaxed heuristics there. This allows us to transfer the knowledge about relaxed heuristics from the grounded level to the lifted level.

Introduction

Classical planning tasks are usually modelled in the standard PDDL language based on first-order logic (McDermott 2000). Nevertheless, a vast majority of planners do not work with this first-order (lifted) representation but with a simpler one based on propositional logic (i.e., grounded representation); usually either STRIPS (Fikes and Nilsson 1971) or SAS$^+$ (Bäckström and Nebel 1995). In order to translate the PDDL representation into a propositional one, one has to go through the so-called grounding process where the first-order action schemata are translated into propositional ones by substituting all possible combinations of objects for the variables. This typically results in an exponentially larger representation than the original PDDL representation.

There are several approaches to address this problem. The most common ones try to prune the resulting grounded representation, e.g., by a relaxed reachability analysis (Hoffmann and Nebel 2001; Helmert 2009), or by detection of unreachable and dead-end actions (Fišer 2020). Another approach is to avoid the grounding process altogether and work directly with the PDDL representation. To do this one needs two ingredients: a sufficiently fast algorithm computing successor states in the search (Corrêa et al. 2020), and (admissible) lifted heuristics that help to navigate in the search space. Heuristics for lifted planning can be obtained by lifting existing algorithms computing heuristics on the grounded representation, as was done for h^{add} and h^{max} by Corrêa et al. (2021). An approximation of the Fast Forward (FF) and causal graph heuristics was done by Ridder and Fox; Ridder (2014; 2014). Second, we might try to transform a given PDDL task $\mathcal{P}$ into a smaller one $\mathcal{P}'$ which can be grounded and compute any heuristic in the grounded version of $\mathcal{P}'$. It is clear that such a transformation must preserve the existence of a plan and, for optimal planning, the cost of optimal plans in $\mathcal{P}'$ must be the same or smaller. The second approach was applied by Lauer et al. (2021), over-approximating n-ary predicates by conjunctions of unary predicates. In this paper, we investigate an alternative way of transforming the task.

We introduce PDDL homomorphisms as maps between PDDL tasks preserving plans and their costs. PDDL homomorphisms generalize PDDL endomorphisms introduced by Horčík and Fišer (2021). We employ PDDL homomorphisms similarly as abstractions are used in pattern databases (Edelkamp 2001) or merge-and-shrink heuristics (Helmert et al. 2014). On the propositional level, the large underlying labelled transition system of a planning task is succinctly represented via facts (i.e., ground atomic formulas). Consequently, abstractions on the propositional level are defined in terms of facts. Our approach goes even lower to the level of objects. PDDL tasks are defined over a set of objects $\mathcal{B}$ serving as values to be substituted for the variables during the grounding process. So we formulate our abstractions in terms of objects instead of facts.

We look for PDDL homomorphisms mapping the original PDDL task $\mathcal{P}$ over a set of objects $\mathcal{B}$ into a PDDL task $\mathcal{P}'$ over a sufficiently small set of objects $\mathcal{B}' \subseteq \mathcal{B}$. Consequently, we use the standard grounding process on $\mathcal{P}'$ and compute an (admissible) heuristic. To construct $\mathcal{P}'$, we consider maps $\sigma\colon \mathcal{B} \to \mathcal{B}$ with a small image $\sigma(\mathcal{B})$ serving as the set of objects $\mathcal{B}'$ of $\mathcal{P}'$. Depending on the definition of $\mathcal{P}'$, the map $\sigma\colon \mathcal{B} \to \mathcal{B}$ has to satisfy several conditions which we formulate in the definition of PDDL homomorphisms. Since this is an introductory paper on PDDL homomorphisms, we investigate rather simple and naive methods defining the abstract PDDL task $\mathcal{P}'$ and σ, and show that they can be used to guide lifted search effectively. Since the most complicated condition in the definition of PDDL homomorphism is easily satisfied when $\mathcal{P}'$ has no delete effects, we focus in this paper only on delete-relaxation heuristics. This allows defining the abstract PDDL task $\mathcal{P}'$ for almost any map $\sigma\colon \mathcal{B} \to \mathcal{B}$. Nevertheless, it is in principle possible to keep some delete effects provided we restrict the maps σ. We postpone this investigation for future work.

First-order Logic

We first recall a few definitions from first-order logic (for details see e.g. (Hodges 1997, Chapter 1)) and we introduce our notation and conventions. Given a set S, we denote a tuple $\langle s_1, \dots, s_n\rangle$ of elements from S shortly by $\vec{s}$. The i-th component of $\vec{s}$ is denoted $s_i \in S$. The Cartesian product of k-many copies of a set S is denoted S^k.

Given two sets B, C and a map $\sigma\colon B \to C$, we will extend σ element-wise to tuples, i.e., if $\vec{b} = \langle b_1, \dots, b_n\rangle \in B^n$ then $\sigma(\vec{b}) = \langle \sigma(b_1), \dots, \sigma(b_n)\rangle \in C^n$. In order to decrease the amount of parentheses in mathematical expressions, we adopt the common convention of removing parentheses in $\sigma(\vec{b})$, i.e., writing $\sigma\vec{b}$ instead. Further, we extend σ on subsets of B. For $B' \subseteq B$ we define $\sigma(B') = \{\sigma(b) \mid b \in B'\}$.

A **relational first-order language** $\mathcal{L}$ consists of a set of **variables** $\mathcal{V} = \{v_1, v_2, \dots\}$ and a set of **predicate symbols** $\mathcal{P} = \{p_1, p_2, \dots\}$, each predicate symbol p_i has its arity $\mathrm{ar}(p_i)$. Even though constants are allowed in PDDL, we do not consider constants as a part of the first-order language $\mathcal{L}$ in order to simplify our formalisms. Nevertheless, all our results can be straigthforwardly reformulated for languages with constants.

As we have no functional symbols in our first-order language $\mathcal{L}$, our **atomic formulas** (shortly **atoms**) are just expressions of the form $p(\vec{v})$ where $p \in \mathcal{P}$ is a predicate symbol and $\vec{v} = \langle v_1, \dots, v_n\rangle$ is an n-tuple of variables for $n = \mathrm{ar}(p)$. The set of all atoms is denoted $\Phi(\mathcal{V})$.

If we have a set of objects $\mathcal{B}$, we can define ground atoms. Let $\sigma\colon \mathcal{V} \to \mathcal{B}$ be a map assigning objects to variables. For each atom $p(\vec{v})$ such a map defines its corresponding **ground atom** $p(\sigma\vec{v})$. The set of all ground atoms over the set of objects $\mathcal{B}$ is denoted by $\Phi(\mathcal{B}) = \{p(\sigma\vec{v}) \mid p(\vec{v}) \in \Phi(\mathcal{V}),\ \sigma\colon \mathcal{V} \to \mathcal{B}\}$.

An $\mathcal{L}$-structure $\langle \mathcal{B}, \psi\rangle$ is a set of objects $\mathcal{B}$ together with a set of ground atoms $\psi \subseteq \Phi(\mathcal{B})$. The set ψ can be understood as interpretations for predicate symbols.[1]

Let $\langle \mathcal{B}, \psi\rangle$ and $\langle \mathcal{B}', \psi'\rangle$ be two $\mathcal{L}$-structures. A map $\sigma\colon \mathcal{B} \to \mathcal{B}'$ is called a **homomorphism** if it preserves ground atoms. More precisely, if $p(\vec{b}) \in \psi$ then $p(\sigma\vec{b}) \in \psi'$. Equivalently, σ is a homomorphism if $\sigma(\psi) \subseteq \psi'$ where $\sigma(\psi) = \{p(\sigma\vec{b}) \in \Phi(\mathcal{B}') \mid p(\vec{b}) \in \psi\}$. We will denote the fact that σ is a homomorphism by $\sigma\colon \langle \mathcal{B}, \psi\rangle \to \langle \mathcal{B}', \psi'\rangle$. We will also need the following trivial lemma.

[1]The $\mathcal{L}$-structures are usually defined in logic as sets of objects endowed with relations interpreting predicate symbols from $\mathcal{L}$. Here we identify these interpretations with the corresponding set of ground atoms to be closer to the notation used in planning, i.e., understanding a state as a set of ground atoms rather than an $\mathcal{L}$-structure.

Lemma 1. *Let $\langle \mathcal{B}, \psi\rangle$ be an $\mathcal{L}$-structure and $\sigma\colon \mathcal{B} \to \mathcal{B}'$ a map into a set $\mathcal{B}'$ then σ is a homomorphism from $\langle \mathcal{B}, \psi\rangle$ to $\langle \mathcal{B}', \sigma(\psi)\rangle$.*

PDDL Planning Tasks

We consider the normalized non-numeric, non-temporal PDDL tasks without conditional effects, axioms, and negative preconditions, and with all formulas being conjunctions of atoms (represented as sets of atoms). The types are modelled as unary predicates. So for each type (i.e., a set of objects) there is a corresponding unary predicate interpreted by that set of objects. We also split the effects of PDDL actions into add effects (positive literals) and delete effects (negative literals) directly in the definition below to simplify the presentation.

Similarly, as we fixed a relational first-order language $\mathcal{L}$ and then defined its $\mathcal{L}$-structures, we first define a domain language $\mathcal{D}$ and then its PDDL tasks.

Definition 2. A **domain language** $\mathcal{D} = \langle \mathcal{V}, \mathcal{P}, \mathcal{A}_S\rangle$ is a relational first-order language $\langle \mathcal{V}, \mathcal{P}\rangle$ extended with a set of action symbols $\mathcal{A}_S$. Each action symbol a has its arity $\mathrm{ar}(a)$, i.e., a number of variables it depends on.

Let $a \in \mathcal{A}_S$ and $\vec{v}$ denote a tuple of pair-wise distinct variables of length $\mathrm{ar}(a)$. An **action schema** $a(\vec{v})$ is a triple $a(\vec{v}) = \langle a_{\mathrm{pre}}(\vec{v}), a_{\mathrm{add}}(\vec{v}), a_{\mathrm{del}}(\vec{v})\rangle$ where $a_{\mathrm{pre}}(\vec{v})$, $a_{\mathrm{add}}(\vec{v})$ and $a_{\mathrm{del}}(\vec{v})$ are sets of atoms built of variables $\vec{v}$, called **preconditions**, **add effects**, and **delete effects**, respectively.

Analogously to ground atoms, we define ground actions. Given a set of objects $\mathcal{B}$, a map $\sigma\colon \mathcal{V} \to \mathcal{B}$ and an action schema $a(\vec{v})$, the corresponding **ground action** $a(\sigma\vec{v})$ is created by substituting objects $\sigma\vec{v}$ for variables $\vec{v}$. Given a set of action schemas $\mathcal{A} = \{a(\vec{v}) \mid a \in \mathcal{A}_S\}$ and a set of objects $\mathcal{B}$, the set of all ground actions is denoted $\mathcal{A}(\mathcal{B}) = \{a(\sigma\vec{v}) \mid a(\vec{v}) \in \mathcal{A}, \sigma\colon \mathcal{V} \to \mathcal{B}\}$.

Definition 3. Let $\mathcal{D}$ be a domain language. A **normalized PDDL task** over $\mathcal{D}$ is a tuple $\mathcal{P} = \langle \mathcal{B}, \mathcal{A}, \psi_I, \psi_G\rangle$ where $\mathcal{B}$ is a non-empty set of objects, $\mathcal{A} = \{a(\vec{v}) \mid a \in \mathcal{A}_S\}$ a set of action schemas, one action schema for each action symbol, $\psi_I \subseteq \Phi(\mathcal{B})$ and $\psi_G \subseteq \Phi(\mathcal{B})$ are sets of ground atoms, called **initial state** and **goal**, respectively.

A **state** in $\mathcal{P}$ is a set of ground atoms $s \subseteq \Phi(\mathcal{B})$. A ground action $a(\vec{b})$ is **applicable** in a state s if $a_{\mathrm{pre}}(\vec{b}) \subseteq s$. In other words, $a(\vec{b})$ is applicable if all its preconditions hold in the structure $\langle \mathcal{B}, s\rangle$. The **resulting state** of applying an applicable action $a(\vec{b})$ in a state s is the state $a(\vec{b})[\![s]\!] = (s \setminus a_{\mathrm{del}}(\vec{b})) \cup a_{\mathrm{add}}(\vec{b})$.

A sequence of ground actions $\pi = \langle a_1(\vec{b}_1), \dots, a_n(\vec{b}_n)\rangle$ is applicable in a state s_0 if there are states $s_1, \dots, s_n$ such that $a_i(\vec{b}_i)$ is applicable in s_{i-1} and $s_i = a_i(\vec{b}_i)[\![s_{i-1}]\!]$ for $i \in \{1, \dots, n\}$. The resulting state of this application is $\pi[\![s_0]\!] = s_n$. Let s be a state. The sequence π is called an s-**plan** if π is applicable in s and $\pi[\![s]\!] \supseteq \psi_G$. In particular, if $s = \psi_I$ then π is called simply a **plan**. A state s is **reachable** if there exists an action sequence π such that $\pi[\![\psi_I]\!] = s$. In case of optimal planning, we assume that for

$$\begin{tikzcd}
s \arrow[r, "a(\vec{b})"] \arrow[d, "\sigma"] & t \arrow[d, "\sigma"] \\
s' \arrow[r, "a(\sigma\vec{b})"] & t'
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
s \arrow[r, "a_1(\vec{b}_1)"] \arrow[d, "\sigma"] & s_1 \arrow[r, "a_2(\vec{b}_2)"] \arrow[d, "\sigma"] & \dots \arrow[r, "a_n(\vec{b}_n)"] & s_n \arrow[d, "\sigma"] \\
s' \arrow[r, "a_1(\sigma\vec{b}_1)"] & s'_1 \arrow[r, "a_2(\sigma\vec{b}_2)"] & \dots \arrow[r, "a_n(\sigma\vec{b}_n)"] & s'_n
\end{tikzcd}$$

(b)

Figure 1: (a) the condition (P3), (b) the plan preservation

each action schema $a(\vec{v})$ there is a cost function c_a assigning a cost $\mathrm{c}_a(\vec{b})$ to the ground action $a(\vec{b})$. It allows defining a cost of the plan π as $\mathrm{c}_\pi = \sum_{i=1}^{n} \mathrm{c}_{a_i}(\vec{b}_i)$. A plan with a minimum cost is called **optimal**.

PDDL Homomorphisms

In this section, we introduce the notion of PDDL homomorphism as a map between two sets of objects. Later, we show that given a PDDL task $\mathcal{P}$, we can use a PDDL homomorphism σ to construct another *smaller* task $\mathcal{P}'$ preserving plans in a sense that if π is a plan in $\mathcal{P}$, then its homomorphic image $\sigma(\pi)$ is a plan in $\mathcal{P}'$ and the cost of $\sigma(\pi)$ is at most the cost of π.

Definition 4. Let $\mathcal{D}$ be a domain language and $\mathcal{P} = \langle \mathcal{B}, \mathcal{A}, \psi_I, \psi_G \rangle$, $\mathcal{P}' = \langle \mathcal{B}', \mathcal{A}', \psi'_I, \psi'_G \rangle$ PDDL tasks over $\mathcal{D}$. A map $\sigma\colon \mathcal{B} \to \mathcal{B}'$ is called a **PDDL homomorphism**, denoted by $\sigma\colon \mathcal{P} \to \mathcal{P}'$, if the following conditions are satisfied:

(P1) σ is a homomorphism from $\langle \mathcal{B}, \psi_I \rangle$ to $\langle \mathcal{B}', \psi'_I \rangle$,

(P2) $\sigma(\psi_G) \supseteq \psi'_G$,

(P3) for each reachable state s in $\mathcal{P}$, each state s' in $\mathcal{P}'$ and each ground action $a(\vec{b})$ applicable in s if $\sigma\colon \langle \mathcal{B}, s \rangle \to \langle \mathcal{B}', s' \rangle$ and $t = a(\vec{b})[\![s]\!]$ then $a(\sigma\vec{b})$ is applicable in s' and $\sigma\colon \langle \mathcal{B}, t \rangle \to \langle \mathcal{B}', t' \rangle$ for $t' = a(\sigma\vec{b})[\![s']\!]$.

(P4) for optimal planning, we further require that $\mathrm{c}_a(\sigma\vec{b}) \leq \mathrm{c}_a(\vec{b})$ for all ground actions $a(\vec{b})$.

The condition (P1) states that the ground atoms in the initial state ψ_I are preserved. (P2) states that the goal ψ'_G is weaker than the σ-image of ψ_G. The condition (P3) states that whenever we have a reachable state s in $\mathcal{P}$ and a state s' in $\mathcal{P}'$ such that σ is a homomorphism from $\langle \mathcal{B}, s \rangle$ to $\langle \mathcal{B}', s' \rangle$ and a ground action $a(\vec{b})$ is applicable in s, then the ground action $a(\sigma\vec{b})$ is applicable in s' and σ remains a homomorphism after applications of $a(\vec{b})$ and $a(\sigma\vec{b})$ as depicted in Figure 1 (a). Finally, (P4) ensures that ground actions in $\mathcal{P}$ are mapped by σ to cheaper actions in $\mathcal{P}'$.

PDDL homomorphisms generalize the notion of PDDL endomorphisms. A PDDL homomorphism $\sigma\colon \mathcal{P} \to \mathcal{P}'$ is a PDDL endomorphism if $\mathcal{P} = \mathcal{P}'$; see (Horčík and Fišer 2021). Analogously to PDDL endomorphisms, we can show that PDDL homomorphisms preserve plans.

Theorem 5. *Let $\sigma\colon \mathcal{P} \to \mathcal{P}'$ be a PDDL homomorphism, s a reachable state in $\mathcal{P}$ and s' a state in $\mathcal{P}'$ such that $\sigma\colon \langle \mathcal{B}, s \rangle \to \langle \mathcal{B}', s' \rangle$. If $\pi = \langle a_1(\vec{b}_1), \dots, a_n(\vec{b}_n) \rangle$ is an s-plan in $\mathcal{P}$, then $\pi' = \langle a_1(\sigma\vec{b}_1), \dots, a_n(\sigma\vec{b}_n) \rangle$ is a s'-plan in $\mathcal{P}'$. In particular, if π is a ψ_I-plan, then π' is a ψ'_I-plan. Moreover, $\mathrm{c}_{\pi'} \leq \mathrm{c}_\pi$.*

Proof. Let $\pi = \langle a_1(\vec{b}_1), \dots, a_n(\vec{b}_n) \rangle$ be a plan and σ a PDDL homomorphism. Consider the diagram depicted at Figure 1 (b). In the upper row there is the s-plan π leading to a goal state $s_n \supseteq \psi_G$. By assumption $\sigma\colon \langle \mathcal{B}, s \rangle \to \langle \mathcal{B}', s' \rangle$ is a homomorphism. Applying (P3), we infer that $\sigma\colon \langle \mathcal{B}, s_1 \rangle \to \langle \mathcal{B}', s'_1 \rangle$ is a homomorphism, where $s'_1 = a(\sigma\vec{b}_1)[\![s']\!]$. Applying (P3) repeatedly, we finally get that $\sigma\colon \langle \mathcal{B}, s_n \rangle \to \langle \mathcal{B}', s'_n \rangle$. Thus $\sigma(s_n) \subseteq s'_n$. By (P2) we have $\psi'_G \subseteq \sigma(\psi_G) \subseteq \sigma(s_n) \subseteq s'_n$. Consequently, π' is a s'-plan. The particular case for the initial state holds by (P1). The condition (P4) ensures that $\mathrm{c}_{\pi'} \leq \mathrm{c}_\pi$. $\square$

Corollary 6. *Let $\sigma\colon \mathcal{P} \to \mathcal{P}'$ be a PDDL homomorphism, and let s denote a reachable state in $\mathcal{P}$. The cost of an optimal $\sigma(s)$-plan in $\mathcal{P}'$ is an admissible heuristic value for s in $\mathcal{P}$.*

Proof. If there is no s-plan in $\mathcal{P}$, then any heuristic value is admissible. Given an optimal s-plan $\pi = \langle a_1(\vec{b}_1), \dots, a_n(\vec{b}_n) \rangle$, the image $\sigma(s) \subseteq \Phi(\mathcal{B}')$ of s is a state in $\mathcal{P}'$. By Lemma 1, σ is a homomorphism from $\langle \mathcal{B}, s \rangle$ to $\langle \mathcal{B}', \sigma(s) \rangle$. Thus $\pi' = \langle a_1(\sigma\vec{b}_1), \dots, a_n(\sigma\vec{b}_n) \rangle$ is a $\sigma(s)$-plan in $\mathcal{P}'$ whose cost is at most c_π by Theorem 5. $\square$

PDDL Homomorphisms and Delete Relaxation

So far, we have introduced a general notion of PDDL homomorphisms. Now, the question is how to find and use PDDL homomorphisms in practice. Especially finding sufficient conditions to satisfy the condition (P3) may be nontrivial in the general case. Here, we focus on the case of delete relaxed planning tasks $\mathcal{P}^+$. It turns out that *any* map $\sigma\colon \mathcal{B} \to \mathcal{B}$ can be used to generate a homomorphic image of $\mathcal{P}^+$.

Definition 7. Let $\mathcal{P} = \langle \mathcal{B}, \mathcal{A}, \psi_I, \psi_G \rangle$ be a PDDL task over a domain language $\mathcal{D}$. We define its **delete relaxation** as $\mathcal{P}^+ = \langle \mathcal{B}, \mathcal{A}^+, \psi_I, \psi_G \rangle$ where $\mathcal{A}^+ = \{\langle a_{\mathrm{pre}}(\vec{v}), a_{\mathrm{add}}(\vec{v}), \emptyset \rangle \mid \langle a_{\mathrm{pre}}(\vec{v}), a_{\mathrm{add}}(\vec{v}), a_{\mathrm{del}}(\vec{v}) \rangle \in \mathcal{A}\}$.

Starting from $\mathcal{P}^+$ and a $\sigma\colon \mathcal{B} \to \mathcal{B}$, we can construct a new PDDL task over the same domain language $\mathcal{D}$ as $\sigma(\mathcal{P}^+) = \langle \sigma(\mathcal{B}), \mathcal{A}^+, \sigma(\psi_I), \sigma(\psi_G) \rangle$.

In case of optimal planning, we must modify the costs of actions in $\mathcal{A}^+$ as follows: $\mathrm{c}_a(\sigma\vec{b}) = \min\{\mathrm{c}_a(\vec{c}) \mid \vec{c} \in \mathcal{B}^{\mathrm{ar}(a)},\ \sigma\vec{c} = \sigma\vec{b}\}$. In other words, the cost of $a(\sigma\vec{b})$ is the minimum from the costs of all actions $a(\vec{c})$ which are mapped to $a(\sigma\vec{b})$ by σ.

Now we get to the main result of this section showing that any $\sigma\colon \mathcal{B} \to \mathcal{B}$ is a PDDL homomorphism from $\mathcal{P}$ to $\sigma(\mathcal{P}^+)$.

Theorem 8. *Let $\mathcal{P}$ be a PDDL task with a set of objects $\mathcal{B}$ and $\sigma\colon \mathcal{B} \to \mathcal{B}$. Then σ is a PDDL homomorphism from $\mathcal{P}$ to $\sigma(\mathcal{P}^+)$.*

Proof. (P1) follows from Lemma 1. (P2) and (P4) hold by the construction. We check (P3). Suppose that s is a state in $\mathcal{P}$ such that $\sigma\colon \langle \mathcal{B}, s\rangle \rightarrow \langle \sigma(\mathcal{B}), s'\rangle$ and $a(\vec{b})$ is applicable in s. First, we have to prove that $a(\sigma\vec{b})$ is applicable in s'. Note that $a_{\mathrm{pre}}(\sigma\vec{b}) = \sigma(a_{\mathrm{pre}}(\vec{b})) \subseteq \sigma(s) \subseteq s'$. Thus $a(\sigma\vec{b})$ is applicable in s'. It remains to prove that $\sigma\colon \langle \mathcal{B}, t\rangle \rightarrow \langle \sigma(\mathcal{B}), t'\rangle$ for $t = a(\vec{b})[\![s]\!]$ and $t' = a(\sigma\vec{b})[\![s']\!]$. Consider $p(\vec{c}) \in t$. If $p(\vec{c}) \in a_{\mathrm{add}}(\vec{b})$, then $p(\sigma\vec{c}) \in a_{\mathrm{add}}(\sigma\vec{b})$. Thus $p(\sigma\vec{c}) \in t'$ in this case. If $p(\vec{c}) \notin a_{\mathrm{add}}(\vec{b})$, then $p(\vec{c}) \in s$ and consequently $p(\sigma\vec{c}) \in s'$ since σ is a homomorphism. As the action schemas in $\sigma(\mathcal{P}^+)$ has no delete effects, we must have $p(\sigma\vec{c}) \in t'$. □

Moving from theory to practice, Theorem 8 and Corollary 6 can be used to compute (admissible) heuristics in the following way. First, we find a map $\sigma\colon \mathcal{B} \rightarrow \mathcal{B}$ with sufficiently small image $\sigma(\mathcal{B})$. Then we construct the reduced PDDL task $\sigma(\mathcal{P}^+)$. And finally, $\sigma(\mathcal{P}^+)$ is grounded into a STRIPS task Π. Now, we can use the ground task Π to compute a heuristic value for the states in $\mathcal{P}$: For each state s from $\mathcal{P}$, we apply the map σ, ground the state $\sigma(s)$ to the corresponding ground state t in Π, and compute the heuristic value for t which is used as the heuristic value for the PDDL state s. Moreover, if we use an admissible heuristic for STRIPS, the heuristic estimate is also admissible for our original planning task in PDDL (Corollary 6).

The grounding of $\sigma(\mathcal{P}^+)$ can also safely utilize some pruning techniques. Applying reachability analysis, such as simple relaxed reachability or h^2 heuristic (Bonet and Geffner 2001), is always safe, because this type of pruning removes only facts that cannot be part of $\sigma(s)$. Moreover, a standard part of the grounding process is removing static facts, i.e., facts that are part of all reachable states. Static facts can be part of $\sigma(s)$ and they do not have their counterparts in the STRIPS task Π, but they can be safely ignored.

In the process described above, we decrease the number of objects in order to decrease the number of ground actions in the resulting PDDL task $\sigma(\mathcal{P}^+)$ in comparison with the original PDDL task $\mathcal{P}$. This is usually the case. Nevertheless, in some cases, the number of ground actions can increase in $\sigma(\mathcal{P}^+)$ due to pruning techniques. More precisely, collapsing objects can cause that some unreachable states become reachable. Consider for instance the transportation domain and a task where we have a road map of locations (i.e., a graph) with two disconnected components C_1, C_2 so that C_1 is unreachable from C_2. It is clear that if $\sigma(c_1) = \sigma(c_2)$ for $c_1 \in C_1$ and $c_2 \in C_2$, the components become connected in $\sigma(\mathcal{P}^+)$. Consequently, the relaxed reachability analysis generates more ground actions.

Experimental Evaluation

The accuracy of the heuristic function highly depends on σ, which ideally should preserve the structure of the original planning task as much as possible while making $\sigma(\mathcal{P}^+)$ small enough to be grounded and used for efficiently computing heuristic values during the search. Here, we focus on analyzing the behavior of randomly generated PDDL homomorphism maps. More precisely, we generate a sequence of mappings $\sigma_1, \dots, \sigma_n$ in the following way. For each σ_i, two random objects o and o' from the same minimal type (i.e., a type that has no sub-types) are selected, and σ_i is set to be identity except for $\sigma_i(o) = o'$. This process is repeated until $\sigma_1(\cdots\sigma_n(\mathcal{P}^+)\cdots)$ has only r objects, or no such pair of objects exists. The resulting homomorphism map is then the composition $\sigma = \sigma_1 \circ \ldots \circ \sigma_n$. We call this strategy `rnd-t`, as the result is a random mapping that preserves object types.

Moreover, we also try a variant, where we enforce σ to be identity on all objects from the goal of $\mathcal{P}$. This strategy, denoted `rnd-g`, preserves both object types and the goal, in order to preserve at least some information about goal-distance. However, this can prevent us from significantly reducing the task size if the goal depends on many objects.

We implemented a lifted planner with the aforementioned strategies in C.[2] Our planner uses a successor generator based on the SQLite database system (Hipp 2020) implementing ideas laid out by Corrêa et al. (2020). We compared our implementation of blind search with the one of Corrêa et al. and the planners seem to be competitive. Moreover, whenever we compare to the grounded planner, we use the same basic code based on SQLite also for grounding to get more comparable results.

We used a cluster of computing nodes with Intel Xeon Scalable Gold 6146 processors. The time and memory limits were set to 30 minutes and 8 GB, respectively, for each task. As the main purpose of using homomorphisms is to compute heuristics at a lifted level, we focus our evaluation on hard-to-ground instances. We use the same benchmark set used in previous work for lifted classical planning (Lauer et al. 2021). When reporting on the results, we merged different variants of childsnack and visitall into one whenever our method showed similar behavior, leaving 9 domains.

We evaluate our heuristics on satisficing and optimal planning. For satisficing planning we use Greedy Best-First Search (GBFS) with the FF heuristic (Hoffmann and Nebel 2001) computed on the grounded $\sigma(\mathcal{P}^+)$ task. For optimal planing, we use A^* search (Hart, Nilsson, and Raphael 1968) with the LM-cut heuristic (Helmert and Domshlak 2009) computed on the grounded $\sigma(\mathcal{P}^+)$ task. We remark that, in principle, one can use any heuristic. However, FF and LM-cut are particularly suitable for our purposes since they are informative heuristics that ignore the delete-effects therefore they are not affected by the fact that we apply the mapping σ to the delete-relaxed task, $\mathcal{P}^+$.

Evaluation of PDDL Homomorphisms

First, we evaluate the impact that homomorphisms have on the heuristic computation, both in terms of heuristic accuracy and in how much they reduce the computational effort of computing the heuristics. Of course, this greatly depends on the selection of the map σ. For each of the strategies, `rnd-t` and `rnd-g`, we generate 50 random mappings on each instance and measure the reduction in task size (in terms of the number of operators of the grounded task with/without reduction) and heuristic accuracy (by comparing the heuristic value of the initial state with/without reduc-

[2]https://gitlab.com/danfis/cpddl, branch aaai22-lifted-hmorph

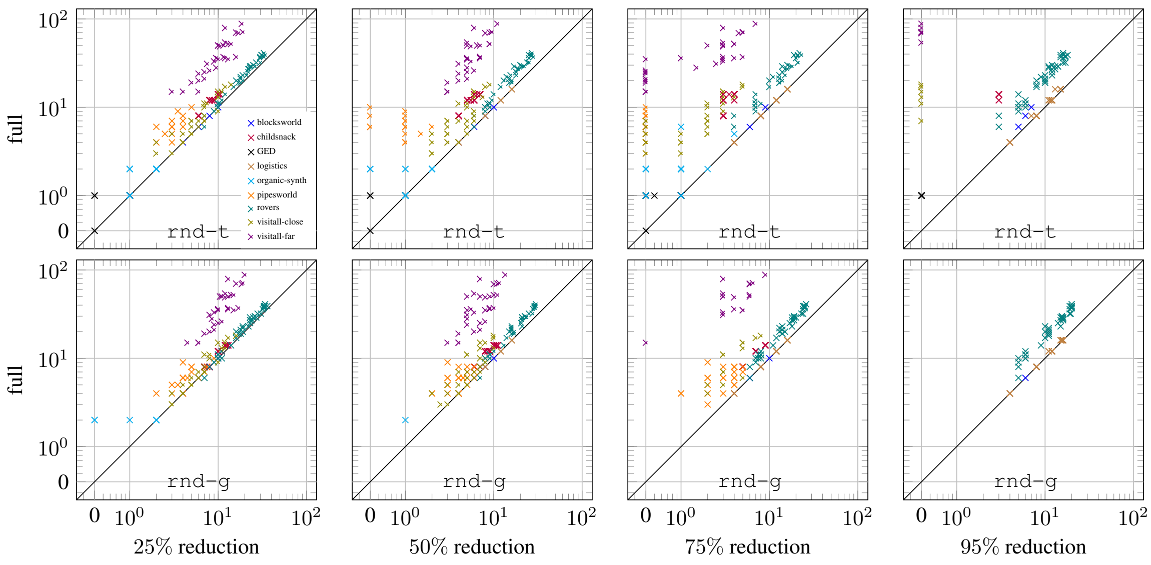

Figure 2: LM-cut heuristic value for the initial state: median value from 50 runs of `rnd-t` (above) and `rnd-g` (below).

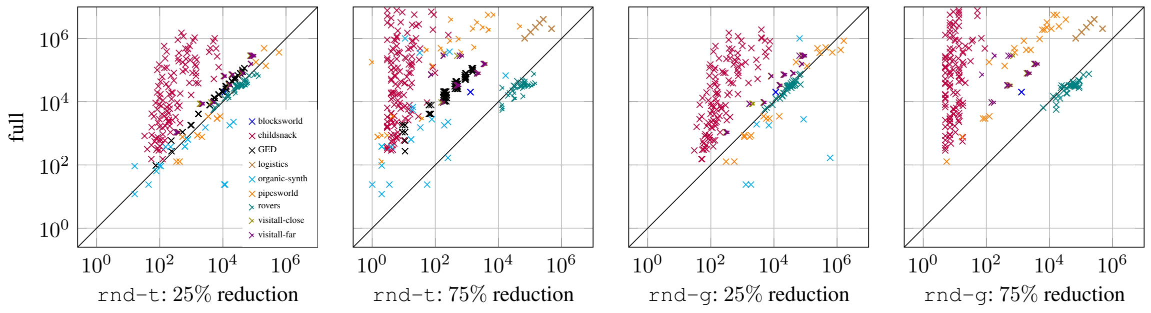

Figure 3: Number of operators of the grounded task with/without reduction. For each instance, we select the same homomorphism as the one used in Figure 2, i.e., the one with a median value for the heuristic value of the initial state.

tion). In most domains, the results have a large variance with respect to the choice of the random mapping both in terms of grounded task size and/or heuristic accuracy. However, there is often not a direct correlation, i.e., in most cases, there are mappings with the same heuristic accuracy and very different task size or vice-versa. To report representative results, we choose the mapping resulting in the median heuristic value for the initial state. This ensures that the task reductions shown correspond with the reported heuristic values.

Figure 2 shows the LM-cut heuristic value[3], compared to the baseline where the instances are fully grounded, for different mappings that reduce the number of objects by a given percentage. Note that in some instances, it is not possible to define mappings that reduce the given percentage of objects while preserving object types (for `rnd-t`) and goals (`rnd-g`). Therefore, the plots with higher percentage reduction have fewer points, as the cases where not enough objects were eliminated are excluded. We observe that, when the reduction is small, the heuristic values are close to the original value and, as the reduction increases, the heuristic accuracy decreases. However, in many cases, we can significantly reduce the number of objects while still obtaining an informative heuristic (greater than 0). Therefore, PDDL homomorphisms are a flexible way to trade off task size and heuristic accuracy. Compared to `rnd-t`, `rnd-g` obtains slightly better accuracy (e.g., childsnack, pipesworld), so mappings preserving the goal are mildly beneficial. For larger reductions, we observe that `rnd-g` is almost always somewhat informative, whereas `rnd-t` sometimes returns a heuristic value of 0. However, this is only possible on instances where the goal does not depend on more than 5% of the objects.

But, how much impact does it have to reduce the number of objects on the size of the grounded task? Figure 3 measures this by comparing the number of grounded operators of the fully grounded task against that of the reduced task, using the same homomorphisms as in Figure 2. As heuristic computation is often polynomial in the size of the grounded task, this is representative of the effort on computing the heuristic. On one hand, significant reductions of several orders of magnitude can be attained even when only reducing

[3]Results with the FF heuristic are very similar, suggesting that our conclusions generalize to other heuristics as well.

domain	optimal (A* + `lmc`)											satisficing (GBFS + `hff`)											
	`rnd-t`				`rnd-g`							`rnd-t`				`rnd-g`							
	25	50	75	95	25	50	75	95	`bl`	`pl`	`fd`	25	50	75	95	25	50	75	95	`fd`	`pl`	`plgb`	`ur`
blocksworld (40)	4	5	6	9	4	6	9	**11**	0	0	4	1	1	2	8	1	1	3	**9**	2	5	1	6
childsnack (144)	4	4	4	4	5	5	4	4	4	1	**8**	57	43	24	16	62	50	43	38	50	75	34	**94**
GED (156)	**18**	17	16	16	**18**	**18**	**18**	**18**	16	16	**18**	24	20	18	20	82	83	83	83	66	139	103	**156**
logistics (40)	9	12	16	19	9	13	20	**27**	5	3	12	8	6	5	3	10	9	9	5	29	**40**	3	0
organic-synth (56)	21	20	28	**44**	28	29	29	29	**44**	43	33	21	20	29	45	28	29	30	30	32	**47**	45	46
pipesworld (50)	8	9	12	11	7	11	**15**	**15**	11	8	8	11	13	11	11	11	12	13	13	17	**28**	21	13
rovers (40)	11	7	3	2	10	6	5	3	0	1	**12**	13	5	3	1	11	4	5	5	30	**33**	11	16
visitall-close (90)	29	39	46	35	29	42	**55**	**55**	34	49	30	35	50	60	36	36	51	72	71	30	**88**	78	81
visitall-far (90)	3	2	0	0	3	2	0	0	0	19	**25**	18	10	1	0	21	13	12	11	30	56	39	**67**
Σ (706)	107	115	131	140	113	132	155	**162**	114	140	150	188	168	153	140	262	252	270	265	286	**511**	335	479

Table 1: Coverage results for optimal and satisficing planning.

the number of objects by 25%. On the other hand, the size of the ground representation can also increase when reducing the number of object (as already discussed). Therefore, under the right mapping, PDDL homomorphisms can enable the computation of heuristics at the lifted level.

Optimal Planning Results

To analyze whether lifted homomorphisms can produce useful heuristics in practice, we analyze their performance on a lifted search, and compare them against other lifted and grounded planning approaches. In order to choose an informative homomorphism, we use our strategies, `rnd-t` and `rnd-g`, to generate 5 different homomorphisms and select the one with the highest heuristic value for the initial state. Note that in this case, whenever it is not possible to reduce the number of objects, the method uses the highest possible reduction it was able to achieve. For example, `rnd-g`-95% will often preserve all objects in the goal and map all other objects to a single non-goal object per type.

As a baseline for optimal planning, we use lifted blind search (`bl`), the PowerLifted planner with A* and the lifted $h^{\max}$ heuristic (`pl`) (Corrêa et al. 2020, 2021), and we also compare against the grounded variant of our heuristics (`fd`) implemented in the Fast Downward planner (Helmert 2006).

The left side of Table 1 shows coverage results for optimal planning. As we can see, the results are quite competitive for all our variants, clearly outperforming the blind search. Setting the target reduction as high as possible often improves the results, though there are some exceptions like rovers. Preserving goal objects (`rnd-g`) performs better than `rnd-t`. The reason is that, these strategies reduce the task as much as possible, while keeping some information regarding goal distance. While our results show that preserving goal objects can attain more informative heuristic values, this only pays off whenever a good mapping is selected, which is not guaranteed with our random selection.

Figure 4 also compares our best configuration against the two baselines, `bl` and `fd`, in terms of expanded nodes and runtime. The plots confirm the trends shown in the previous section. In terms of heuristic accuracy and expanded nodes, the lifted homomorphism heuristics (LHH) are a middle ground between not using any heuristic and using the heuristic on the fully grounded task. However, there is a significant reduction in the grounded task size, which reflects on the time for computing the heuristic. The results in terms of runtime are quite diverse. LHH can pay off in several domains (e.g., blocksworld, logistics, pipesworld, and visitall-close), as shown by the coverage results and runtime plots. Despite expanding more nodes than the fully grounded version and taking longer for the successor generation (`fd`), our heuristics greatly reduce the grounding effort, achieving significant runtime advantages when grounding was a bottleneck. Compared to the lifted $h^{\max}$ heuristic (`pl`), we observe that LHH outperforms it except in organic-synth, visitall-far and some tasks from visitall-close, as LM-cut on the reduced task can be more informative than $h^{\max}$ on the full task, and our method is often orders of magnitude faster.

Satisficing Planning Results

For satisficing planning, we run GBFS with h^{FF}. As baselines, we consider the (fully grounded) Fast Downward planner with GBFS and h^{FF} (`fd`); the PowerLifted planner with GBFS and the lifted h^{add} (`plgb`); lazy evaluation with preferred operators and the same heuristic (`pl`) (both Corrêa et al. 2020, 2021); and GBFS with goal counting breaking ties with the unary-relaxation heuristic (`ur`) (Lauer et al. 2021). The overall performance is not close to the current state-of-the-art satisficing lifted planners `pl` and `ur`. However, note that we run LHH with a vanilla GBFS, without other search enhancements like lazy evaluation and preferred operators (`pl`), or combining multiple heuristics (`ur`). To more closely evaluate the performance of the heuristics, Figure 5 compares the expanded nodes and runtime against GBFS with the fully grounded h^{FF} and the lifted h^{add} heuristics. Here, we see a similar picture as in optimal planning, where our heuristic is less informative than the grounded h^{FF} or lifted h^{add}, but can be computed with less overhead, e.g., in GED and visitall-close (for grounded h^{FF}), or childsnack and pipesworld (for lifted h^{add}).

Related Work

The idea of aggregating similar objects has been used in the past in different contexts. Sievers et al. (2019) define structural symmetries as PDDL automorphisms, i.e., permutations of the planning task that preserve some properties. Röger, Sievers, and Katz (2018) use a similar notion to compute reachability on a reduced task while ensuring that relaxed reachability is preserved. Compared to them, we focus on obtaining heuristic functions. Our mappings are more

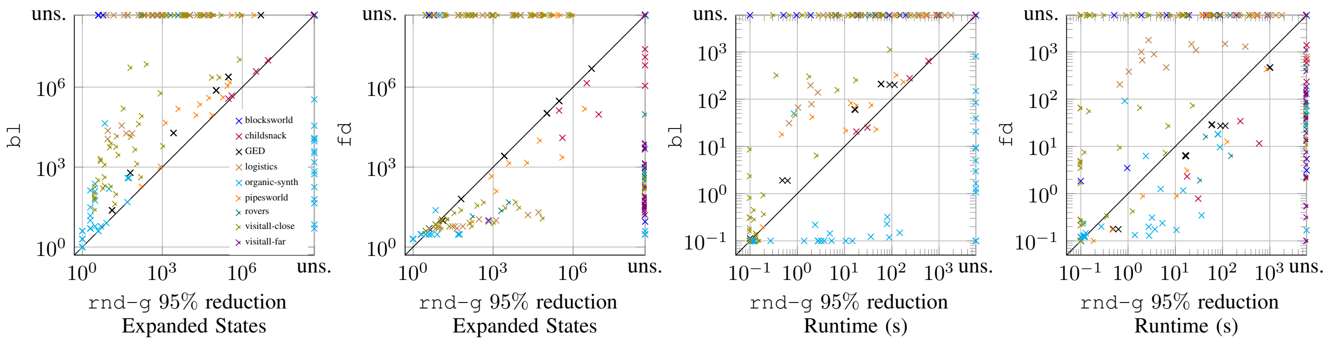

Figure 4: Expanded nodes and runtime for optimal planning, versus blind search (`bl`), and fully grounding the task (`fd`).

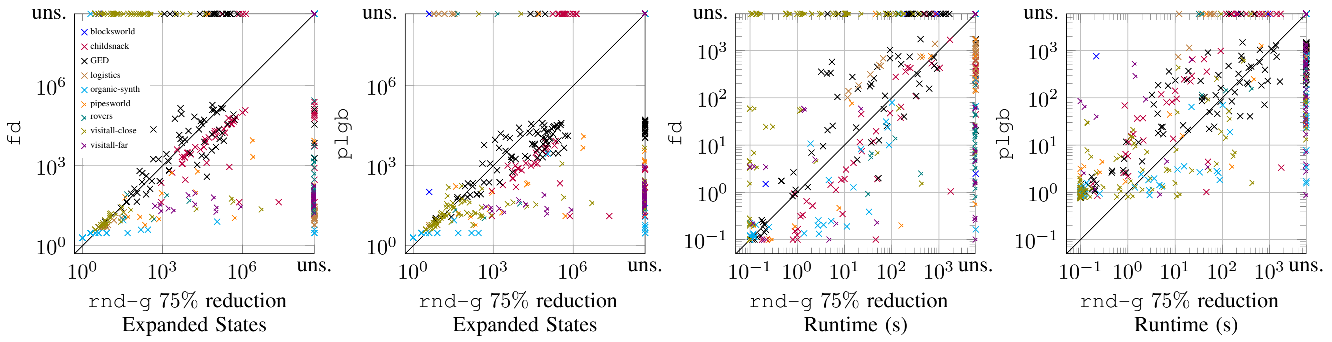

Figure 5: Expanded nodes and runtime for satisficing planning, versus fully grounding the task (`fd`) and GBFS with lifted h^{add}.

flexible, though possibly incurring an information loss as they do not guarantee preserving relaxed reachability. Ridder and Fox (2014) compute relaxed-plan heuristics, such as the FF heuristic, by aggregating some objects during the computation of the relaxed planning graph. As opposed to them, we focus on a general framework, where any given object mapping can be used to define a grounded task on which any existing heuristic function can be computed. An interesting open question is whether these methods can be tailored towards defining informative PDDL homomorphisms.

Abstraction heuristics, such as Pattern Databases (Edelkamp 2001), Merge-and-Shrink (Helmert et al. 2014) and Cartesian abstractions (Seipp and Helmert 2018), also induce a homomorphism on the state space of the planning task. However, the abstract mappings rely on a grounded task so it is not clear how to use those methods at a lifted level. Note that the success of abstraction heuristics has been enabled by a significant research effort on strategies for finding good mappings (Edelkamp 2006; Haslum et al. 2007; Franco et al. 2017; Fan, Müller, and Holte 2014; Sievers, Wehrle, and Helmert 2016)), as well as on combining multiple estimates (e.g., additively (Felner, Korf, and Hanan 2004), or by using cost-partitioning (Seipp, Keller, and Helmert 2020)). Even though those methods are not directly applicable to lifted object mappings, this is a promising direction for future research.

An alternative line of research for hard-to-ground instances is to ground only a subset of the task, by identifying which objects and/or actions are relevant and ignoring the rest (Gnad et al. 2019; Silver et al. 2021). These approaches rely on learning from previously solved instances in the same domain to determine which objects or actions are relevant. Moverover, as excluded objects are entirely ignored, the resulting task does not preserve solvability so it cannot be used to derive (admissible) heuristic functions.

Conclusion

We introduced a general framework to investigate a wide class of lifted heuristics based on collapsing objects. In particular, we defined the notion of a PDDL homomorphism between two PDDL tasks, which is a map on objects preserving (optimal) plans. To test this framework, we focused on delete-relaxation heuristics as they are conceptually the simplest ones. We proved that each map σ on objects induces admissible heuristics by grounding the σ-image of the delete-relaxation of the original problem and then computing an arbitrary heuristic.

In the experiments, we investigated how the choice of σ influences the quality of the resulting heuristics, in particular, its informativeness and the size of the grounded σ-image. It turned out that our lifted heuristics can reduce the computational resources needed to navigate a lifted planner in the hard-to-ground domains while preserving a reasonable amount of informativeness. However, there is still a margin to improve the overall performance of our lifted planner to the level of state-of-the-art planners, e.g., by inventing better selection strategies and/or combining multiple mappings.

Acknowledgements

We thank Augusto B. Corrêa for helping us with the experimental evaluation, in particular the implementation of A* algorithm in the PowerLifted planner. Rostislav Horčík was supported by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". Daniel Fišer was supported by the Czech Science Foundation (GAČR) under the project No. 19-22555Y.

References

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

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

Corrêa, A. B.; Francès, G.; Pommerening, F.; and Helmert, M. 2021. Delete-Relaxation Heuristics for Lifted Classical Planning. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS'21)*, 94–102. AAAI Press.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2020. Lifted Successor Generation Using Query Optimization Techniques. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 80–89. AAAI Press.

Edelkamp, S. 2001. Planning with Pattern Databases. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the 6th European Conference on Planning (ECP'01)*, 13–24. Springer-Verlag.

Edelkamp, S. 2006. Automated Creation of Pattern Database Search Heuristics. In *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*, 35–50.

Fan, G.; Müller, M.; and Holte, R. 2014. Non-Linear Merging Strategies for Merge-and-Shrink Based on Variable Interactions. In Edelkamp, S.; and Bartak, R., eds., *Proceedings of the 7th Annual Symposium on Combinatorial Search (SOCS'14)*. AAAI Press.

Felner, A.; Korf, R.; and Hanan, S. 2004. Additive Pattern Database Heuristics. *Journal of Artificial Intelligence Research*, 22: 279–318.

Fikes, R. E.; and Nilsson, N. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2: 189–208.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In Conitzer, V.; and Sha, F., eds., *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, 9835–9842. AAAI Press.

Franco, S.; Torralba, A.; Lelis, L. H.; and Barley, M. 2017. On Creating Complementary Pattern Databases. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, 4302–4309. AAAI Press/IJCAI.

Gnad, D.; Torralba, Á.; Domínguez, M. A.; Areces, C.; and Bustos, F. 2019. Learning How to Ground a Plan - Partial Grounding in Classical Planning. In Hentenryck, P. V.; and Zhou, Z.-H., eds., *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, 7602–7609. AAAI Press.

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.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In Howe, A.; and Holte, R. C., eds., *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*, 1007–1012. Vancouver, BC, Canada: AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the Association for Computing Machinery*, 61(3): 16.1–16.63.

Hipp, R. D. 2020. SQLite.

Hodges, W. 1997. *A Shorter Model Theory*. Cambridge University Press. ISBN 978-0-521-58713-6.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Horčík, R.; and Fišer, D. 2021. Endomorphisms of Lifted Planning Problems. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS'21)*, 174–183. AAAI Press.

Lauer, P.; Torralba, Á.; Fišer, D.; Höller, D.; Wichlacz, J.; and Hoffmann, J. 2021. Polynomial-Time in PDDL Input Size: Making the Delete Relaxation Feasible for Lifted Planning. In Zhou, Z., ed., *Proceedings of the 30st International Joint Conference on Artificial Intelligence (IJCAI'21)*, 4119–4126.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *The AI Magazine*, 21(2): 35–55.

Ridder, B. 2014. *Lifted heuristics : towards more scalable planning systems*. Ph.D. thesis, King's College London, UK.

Ridder, B.; and Fox, M. 2014. Heuristic Evaluation based on Lifted Relaxed Planning Graphs. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press.

Röger, G.; Sievers, S.; and Katz, M. 2018. Symmetry-Based Task Reduction for Relaxed Reachability Analysis. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS'18)*, 208–217. AAAI Press.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *Journal of Artificial Intelligence Research*, 62: 535–577.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *Journal of Artificial Intelligence Research*, 67: 129–167.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2019. Theoretical Foundations for Structural Symmetries of Lifted PDDL Tasks. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, 446–454. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS'16)*, 294–298. AAAI Press.

Silver, T.; Chitnis, R.; Curtis, A.; Tenenbaum, J. B.; Lozano-Pérez, T.; and Kaelbling, L. P. 2021. Planning with Learned Object Importance in Large Problem Instances using Graph Neural Networks. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*, 11962–11971. AAAI Press.