Computing Potential Heuristics Without Grounding PDDL Actions

Pascal Lauer[1,2], **Álvaro Torralba**[3], **Daniel Fišer**[3]

[1]School of Computing, The Australian National University, Canberra, Australia
[2]Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
[3]Aalborg University, Denmark
pascal.lauer@anu.edu.au, alto@cs.aau.dk, danfis@danfis.cz

Abstract

A lot of research in automated planning aims to develop planners capable of solving any task expressed in a predefined input format, such as PDDL. As grounding PDDL to its propositional form can be computationally intractable, lifted planners do not ground the whole PDDL task. A major research direction is developing lifted heuristics that guide a search without encountering this grounding bottleneck. We focus on potential heuristics that associate each (ground) fact with a numerical value (weight). The heuristic value of a state, represented as set of facts, is computed as the sum of weights of its facts. We show that it is possible to compute potential heuristics without grounding the entire task. On certain planning tasks, where grounding all actions causes an exponential blow-up, this allows to compute potential heuristics in polynomial time.

1 Introduction

A prominent challenge in automated planning is the development of domain-independent planners. Such planners aim to solve any task whose solution is a sequence of actions. To allow inputting tasks of any domain to a planner, they usually handle tasks described in the Planning Domain Definition Language (PDDL) (McDermott 2000). PDDL's syntax is based on first-order logic to allow schematic (lifted) descriptions. Many planners transform a PDDL input into its propositional (ground) form to perform a heuristic search (Hoffmann and Nebel 2001; Helmert and Domshlak 2009; Richter and Westphal 2010; Seipp and Helmert 2018; Fišer, Torralba, and Hoffmann 2022). However, ground tasks can be exponentially larger than their lifted counterparts. This problem becomes apparent in an increasing number of applications (Areces et al. 2014; Masoumi, Antoniazzi, and Soutchanski 2015; Wichlacz, Torralba, and Hoffmann 2019). Our work falls within the line of research that aims to avoid this grounding bottleneck (Corrêa et al. 2021, 2022; Lauer et al. 2021, 2025; Horčík and Fišer 2021) by performing a heuristic search without generating the ground task. This necessitates to compute the heuristics directly on the lifted PDDL input.

In this work, we focus on potential heuristics (Pommerening et al. 2015) that have not been studied in the lifted setting so far. Atomic potential heuristics map a numerical value (weight or potential) to each (ground) fact. The heuristic value of a state is the sum of weights of its facts. In the ground setting, atomic potential heuristics enabled to compute potential heuristics with higher-dimensional features (Pommerening, Helmert, and Bonet 2017; Fišer and Steinmetz 2024; Francès et al. 2019). They also enabled the use of operator potential heuristics for symbolic search (Fiser, Torralba, and Hoffmann 2024).

On top of that, potential heuristics proved to be useful for analyzing the hardness of planning tasks via correlation complexity (Seipp et al. 2016; Corrêa and Pommerening 2019; Dold and Helmert 2024a,b), or the complexity of synthesising heuristics that allow to find a plan without backtracking (Helmert et al. 2022).

We present a novel technique for computing (atomic) potential heuristics without grounding the actions of a PDDL task. Following prior works (Pommerening et al. 2015; Fišer, Horčík, and Komenda 2020), we formulate the computation of potential heuristics as a linear program ensuring consistency and goal-awareness of the heuristic, thus guaranteeing its admissibility. However, prior work constructs one linear constraint per ground action, which effectively requires grounding the task. To avoid this bottleneck, we construct a *lifted* linear program directly from the PDDL input. We show that the lifted linear program can be rewritten using rewriting techniques for Datalog programs (Morak and Woltran 2012; Bichler, Morak, and Woltran 2016). This rewriting often significantly reduces the number of constraints generated after grounding. It can allow to compute potential heuristics in polynomial time, even if the number of ground actions is exponential. In particular, in cases where the set of facts is tractable to represent and rewriting brings the number of variables per constraint close to the predicate arity. Prior work highlighted that is common when rewriting Datalog programs (Corrêa et al. 2021; Corrêa et al. 2023). We benefit from a similar effect.

Experiments confirm that our computation is feasible, even if its ground counterpart (Pommerening et al. 2015) is not. Although the ground setting allows to represent more information about the mutual exclusions of facts, heuristic values for initial states closely match our computation. The impact in terms of coverage is limited. But, our computation solves tasks that heuristic search with other lifted heuristics cannot. This gives hope that our computation is a solid foundation for extensions, similar to the ground setting.

2 Background

Ground Planning Tasks and Heuristics We define ground STRIPS planning tasks (Fikes and Nilsson 1971) similar to Hoffmann and Nebel (2001).

Definition 1 (Ground planning task)**.** A *ground planning task* is a tuple $\Pi = \langle F, A, I, G \rangle$. F is a finite set of *facts*. An *action* $a = \langle pre(a), add(a), del(a), cost(a) \rangle$ consists of the *precondition* $pre(a) \subseteq F$, *add list* $add(a) \subseteq F$, *delete list* $del(a) \subseteq F$ and the *action cost* $cost(a) \in \mathbb{Q}_{\geq 0}$. A is the finite set of actions in the task. $S = 2^F$ are the *states* of Π. $I \subseteq F$ is the *initial state*. $G \subseteq F$ is the *goal condition*. There is a *transition* $s \xrightarrow{a} s'$ between states $s \in S$ and $s' = (s \setminus del(a)) \cup add(a)$ over action $a \in A$ if $pre(a) \subseteq s$. A sequence $\pi = a_1, ..., a_n \in A$ is a *path* from $s_0 \in S$ to $s_n \in S$ iff $s_0 \xrightarrow{a_1} ... \xrightarrow{a_n} s_n$. The cost π is $\sum_{i=1}^{n} cost(a_i)$ for $s \in S$. A state $s \in S$ is *reachable* if there is a path from I to s. S_R denotes the set of all reachable states. A path from $s_0 \in S$ to $s_n \in S$ so that $G \subseteq s_n$ is a plan for s_0. A plan for I is a plan for Π.

To guide the search for a plan, we utilize heuristics that estimate the distance from each state to the goal.

Definition 2 (Heuristics)**.** Let $\Pi = \langle F, A, I, G \rangle$ be a ground planning task with states S. A heuristic (for Π) is a mapping $h : S \to \mathbb{Q} \cup \{\infty\}$. The perfect heuristic $h^* : S \to \mathbb{Q} \cup \{\infty\}$ computes the minimal cost of a plan for a given state, or ∞ if none exists. A heuristic $h : S \to \mathbb{Q}$ is:

- *admissible* if $h(s) \leq h^*(s)$ for all $s \in S_R$
- *consistent* if $h(s) - h(s') \leq cost(a)$ for all $s, s' \in S_R$ with $s \xrightarrow{a} s'$
- *goal-aware* if $h(s) \leq 0$ for all $s \in S_R$ with $G \subseteq s$

These definitions correspond to forward-consistency and forward-admissibility, since they are restricted to the set of (forward) reachable states S_R. It is well known that heuristics that are goal-aware and consistent are also admissible.

We consider atomic potential heuristics, which are defined through a weight function assigning a numerical value to each fact of the planning task (Pommerening et al. 2015).

Definition 3 (Potential heuristic)**.** Let $w : F \to \mathbb{Q}$ be a *weight function* for ground planning task $\Pi = \langle F, A, I, G \rangle$. The induced *potential heuristic* is $h^P_w(s) = \sum_{f \in s} w(f)$.

We use weight functions w that make h^P_w admissible by computing the solution to linear constraints that ensure consistency and goal-awareness. We formulate the constraints in Section 3.

Definition 4 (Mutex)**.** Two facts f_1, f_2 are *mutex* iff there is no reachable state $s \in S_R$ so that $\{f_1, f_2\} \not\subseteq s$.

Given a ground planning task Π, multiple methods can find a set of mutexes M_Π (Gerevini and Schubert 1998; Alcázar and Torralba 2015; Fišer and Komenda 2018). For any $F' \subseteq F$ we denote $M(F') := \{f_2 \mid f_1 \in F', (f_1, f_2) \in M_\Pi\}$ to the set of facts mutex with any fact in F'.

Lifted Planning Tasks To avoid enumerating all ground actions, one can use a lifted action, i.e., a schema describing multiple ground actions. Lifted actions are built from atoms, i.e., schematic facts defined through a relational language.

Definition 5 (Relational language)**.** A *Relational Language* is a tuple $Rl = (P, X, O)$ of a set of *predicates* P, where each $p \in P$ has arity $|p|$, a set of *objects* O, and a set of *variables* X.

An *atom* $p(\vec{x})$ is the combination of a predicate $p \in P$ and a sequence of $|p|$ language variables or objects. P^X is the set of all atoms. An atom $p(\vec{x})$ is called *ground* iff all elements of $\vec{x}$ are objects, we may emphasize this by writing $p(\vec{o})$. The set of all grounded atoms is P^O. We may use the symbol φ for atoms if the concrete attribution of p and $\vec{x}$ is irrelevant.

We now define lifted planning tasks, following Lauer, Lin, and Bercher (2025) to reflect STRIPS fragment of PDDL.

Definition 6 (Lifted action)**.** Let $Rl = (P, X, O)$ be a relational language. A lifted action a over Rl is a tuple $(pre(a), add(a), del(a), cost(a))$, where $pre(a), add(a), del(a) \subseteq P^X$ are sets of lifted atoms and $cost(a) \in \mathbb{N}_0$ is the action cost.

A lifted planning task is a tuple $\Pi = (Rl, \mathcal{A}, I, G)$ where $Rl = (P, X, O)$ is a relational language. $\mathcal{A}$ are lifted actions over Rl. The initial state $I \subseteq P^O$ and goal condition $G \subseteq P^O$ are sets of ground atoms.

The preconditions of actions are also called queries.

Definition 7 (Conjunctive query)**.** Given $Rl = (P, X, O)$, a conjunctive query is a set of lifted atoms $\mathcal{L} \subseteq P^X$. A query $\mathcal{L}$ is satisfied under a set of facts $F \subseteq P^O$ iff there exists a substitution θ such that $\theta(\mathcal{L}) \subseteq F$.

Each lifted task induces a ground planning task, which can be obtained by enumerating all possible substitutions of variables with objects. In reality one often considers a subset $F \subseteq P^O$ for grounding.

Definition 8 (Lifted planning task)**.** Let $\Pi = (Rl, \mathcal{A}, I, G)$ be a lifted planning task with $Rl = (P, X, O)$. Applying a substitution $\theta : X \to O$ to some atom $\varphi = p(x_1, ..., x_n)$ results in a grounded atom $\theta(\varphi) = p(\theta(x_1), ..., \theta(x_n))$.

An action schema $a = (pre(a), add(a), del(a), cost(a))$ is grounded by substitution $\theta : X \to O$ as $\theta(a) = (\theta(pre(a)), \theta(add(a)), \theta(del(a)), cost(a))$. For each $a \in \mathcal{A}$, the atoms occurring in it are $\mathcal{L}_a = pre(a) \cup add(a) \cup del(a)$. Let $F \subseteq P^O$ be a set of facts so that $I \subseteq F$ and $G \subseteq F$. The groundings for $\mathcal{A}$ under F are:

$$\mathcal{A}^F := \{\theta(a) \mid a \in \mathcal{A}, \theta : X \to O, \theta(\mathcal{L}_a) \subseteq F\}$$

The grounding of Π under F is $\Pi^F := (F, \mathcal{A}^F, I, G)$.

A common choice for F is the set of forward reachable facts (Helmert 2009). This is the smallest set $F \supseteq I$ s.t.:

$$\forall a \in \mathcal{A}, \theta : X \to O : \theta(pre(a)) \subseteq F \implies \theta(add(a)) \subseteq F$$

F can be obtained by repeatedly adding all add effects of actions that are applicable in the current set, starting from I.

In our setting, we extend this set to $F' = F \cup \bigcup_{a \in \mathcal{A}, \theta : X \to O, \theta(pre(a)) \subseteq F} \theta(del(a))$. F' also includes the delete effects of all applicable actions in F because we require that all atoms occurring in $del(a)$ to be mapped to elements in F when grounding. This will be important to match the definition of lifted linear constraints.

Finally, we extend the definition of mutex to lifted atoms.

Definition 9 (Lifted mutex)**.** Two lifted atoms $p(\vec{x})_1$ and $p(\vec{x})_2$ are mutex on planning task Π under facts F if for all $\theta : X \to O$ with $\theta(p(\vec{x})_1) \neq \theta(p(\vec{x})_2)$: $\theta(p(\vec{x})_1)$ is mutex with $\theta(p(\vec{x})_2)$ in Π^F.

Similar to the ground setting, we can identify identify a set of lifted mutexes based on lifted mutex groups (Fišer 2020).

3 Ground STRIPS Potentials

Pommerening et al. (2015) show that the conditions to enforce potential heuristics to be admissible can be encoded by linear constraints over the assigned weights that ensure consistency and goal-awareness. Thus, admissible potential heuristics can be automatically derived by using a linear program whose optimization function strives to make the heuristic informative, e.g., maximizing the heuristic value of the initial state.

However, the formulation is given for planning tasks in the SAS$^+$ representation (Bäckström and Nebel 1995) and assumes transition normal form (Pommerening and Helmert 2015). SAS$^+$ mainly differs from STRIPS by representing states as assignments of state variables over finite domains, rather than as sets of propositional facts. The state variables and their domains are generated by a mutex analysis and compilation steps applied to a ground STRIPS task (Helmert 2009). As a first step towards our lifted encoding, in this section we rederive the formulation by Pommerening et al. (2015) for STRIPS using linear constraints.

Definition 10 (Ground linear constraints)**.** $\sum_{i=1}^{n} c_i \cdot v_i \leq c_0$ is a ground linear constraint over constraint variables $V = \{v_1, ..., v_n\}$ and constants $c_0, ..., c_n \in \mathbb{Q}$. A solution to the constraint is a weight function $w : V \to \mathbb{Q}$ so that $\sum_{i=1}^{n} c_i \cdot w(v_i) \leq c$. Multiple ground linear constraints $C_1, ..., C_m$ over the same variables V may be combined a as conjunction $C_1 \wedge ... \wedge C_m$.

Note that this matches the standard linear constraint formulations used in the literature. We use this format here to make the later introduction of lifted constraints easier. For readability, we allow syntactic sugar when writing linear terms. For example, the same constraint variable may occur multiple times in a term summation, or not at all. Such expressions are assumed to be transformed into the canonical form of Definition 10 by standard algebraic rewriting.

By using transition normal form, Pommerening et al. (2015) avoid to explicitly distinguishing whether an operator is guaranteed to add or delete a fact. However, in STRIPS this is not possible. The distinction is important for computing the heuristic value. E.g., the dual formulation (Pommerening et al. 2014) of the operator consistency constraints in Pommerening et al. (2015) makes this explicit by distinguishing "always consumed" facts (guaranteed deletes) and "always produced" facts (guaranteed adds).

Definition 11 (Guaranteed add/delete)**.** Let $\Pi = \langle F, A, I, G \rangle$ be a ground planning task. A fact $f \in del(a)$ is called *guaranteed delete* of action $a \in A$ iff $f \in s \setminus s'$ for all transitions $s \xrightarrow{a} s'$ in Π. Similarly, fact $f \in add(a)$ is called *guaranteed add* of action $a \in A$ iff $f \in s' \setminus s$ for all transitions $s \xrightarrow{a} s'$ in Π.

The sets of $gdel(a)$, $gadd(a)$ can be derived via mutexes M_Π by setting $gadd(a) = \{f \mid f \in add(a) \cap M(pre(a))\}$ and $gdel(a) = \{f \mid f \in del(a) \cap pre(a)\}$.

We now reproduce the linear constraints that ensure consistency and goal-awareness for ground STRIPS planning.

Definition 12. Let $\Pi = (F, A, I, G)$ be a ground planning task. For every fact $f \in F$ we define the ground linear constraints $maxminc(f)$ as:

$$max\_f \geq f \wedge max\_f \geq 0 \ \wedge \ min\_f \leq f \wedge min\_f \leq 0$$

For $a \in A$, $gdel(a) \subseteq del(a)$ are guaranteed deletes and $gadd(a) \subseteq add(a)$ are guaranteed adds of a. The action consistency constraint $aconsist(a)$ for a is the ground linear constraint:

$$\sum_{f \in gdel(a)} f + \sum_{f \in del(a) \setminus gdel(a)} max\_f - \sum_{f \in gadd(a)} f - \sum_{f \in add(a) \setminus gadd(a)} min\_f \leq cost(a)$$

The goal-awareness constraint $gaware$ is the ground linear constraint:

$$\sum_{f \in G} f + \sum_{f \in F \setminus (G \cup M(G))} max\_f \leq 0$$

The set $pot(\Pi)$ contains the solutions to

$$gaware \ \wedge \ \bigwedge_{f \in F} maxminc(f) \ \wedge \ \bigwedge_{a \in \mathcal{A}} aconsist(a)$$

Consistency is enforced by ensuring that for every state transition over an action a:

"Sum over weights of facts deleted by a"
$-$"Sum over weights of facts added by a" $\leq cost(a)$

To capture the "Sum over weights of facts deleted by a" we sum up all weights $w(f)$ for the elements $f \in del(a)$ that are guaranteed deletes. For the remaining delete effects that are not guaranteed, we need to use $\max(w(f), 0)$ which reflects cases, that the fact is present in s (f), or not (0). To capture this maximum we use the auxiliary variables max_f. For the "Sum over weights of facts added by a" we do the same with $\min$ instead of $\max$.

Goal-awareness is enforced by summing up the value of all facts that can appear in goal states. I.e., we sum the potential $w(f)$ of goal facts $f \in G$, and $\max(w(f), 0)$ for any non-goal fact f that is not mutex with the goal.

Note that we can always find a potential function as assigning 0 everywhere fulfills all constraints. We now prove that this formulation results in admissible potential heuristics. In practice, we select a the solution that maximizes the heuristic value of the initial state.

Proposition 13. *Let $w \in pot(\Pi)$ for a ground planning task Π, then h^P_w is consistent, goal-aware, and admissible for Π.*

Proof. We prove admissibility by proving consistency and goal-awareness. We start with consistency, this means: $h^P_w(s) - h^P_w(s') \leq cost(a) \Leftrightarrow \sum_{f \in s} w(f) - \sum_{f \in s'} w(f) \leq cost(a)$

$$\Leftrightarrow \sum_{f \in s \setminus s'} w(f) - \sum_{f \in s' \setminus s} w(f) \leq cost(a)$$

for all $s \in S_R, s \xrightarrow{a} s'$. We prove this holds by bounding:

$$\sum_{f \in s \setminus s'} w(f) \quad - \quad \sum_{s' \setminus s} w(f)$$
$$\leq \sum_{f \in gdel(a)} w(f) + \sum_{f \in del(a) \setminus gdel(a)} w(max\_f) - \sum_{f \in gdel(a)} w(f) - \sum_{f \in add(a) \setminus gadd(a)} w(min\_f)$$

for all $a \in \mathcal{A}$. This ensures consistency as the left-hand side of the inequality is the reformulation of $h^P_w(s) - h^P_w(s')$ and the right-hand side is $\leq cost(a)$ by our consistency constraint from Def. 12. We bound:

$$\sum_{f \in (s \setminus s')} w(f) = \sum_{f \in del(a)} \begin{cases} w(f) & \text{, if } f \in s \setminus s' \\ 0 & \text{, if } f \notin s \setminus s' \end{cases}$$
$$\leq \sum_{f \in del(a)} \begin{cases} w(f) & \text{, if } f \in s \setminus s' \\ \max(w(f), 0) & \text{, if } f \notin s \setminus s' \end{cases}$$
$$\leq \sum_{f \in del(a)} \begin{cases} w(f) & \text{, if } f \in gdel(a) \\ \max(w(f), 0) & \text{, if } f \in del(a) \setminus gdel(a) \end{cases}$$
$$\leq \sum_{f \in del(a)} \begin{cases} w(f) & \text{, if } f \in gdel(a) \\ w(max\_f) & \text{, if } f \in del(a) \setminus gdel(a) \end{cases}$$
$$\leq \sum_{f \in gdel(a)} w(f) + \sum_{f \in del(a) \setminus gdel(a)} w(max\_f)$$

Analogously we can prove the bound

$$-\sum_{f \in (s' \setminus s)} w(f) \leq -\sum_{f \in gadd(a)} w(f) - \sum_{f \in add(a) \setminus gadd(a)} w(min\_f)$$

For goal-awareness exploit that $\max(f, 0)$ overestimates the values for $f \in F \setminus (G \cup M(G))$ not enforced by the goal. That is, we prove that $h^P_w(s) \leq 0$ for all $s \in S$ with $G \subseteq s$ as follows:

$$h^P_w(s) \overset{\text{[Def. 3]}}{=} \sum_{f \in s} w(f) \overset{[G \subseteq s]}{\leq} \sum_{f \in G} w(f) + \sum_{f \in s \setminus G} \max(w(f), 0)$$
$$\overset{\left[\substack{s \subseteq G \cup F \setminus M(G); \\ \max(\cdot, 0) \geq 0}\right]}{\leq} \sum_{f \in G} w(f) + \sum_{f \in F \setminus (G \cup M(G))} \max(w(f), 0)$$
$$\overset{\text{[Def. 12]}}{\leq} \sum_{f \in G} w(f) + \sum_{f \in F \setminus (G \cup M(G))} w(max\_f) \overset{\text{[Def.12]}}{\leq} 0$$

□

4 Lifted STRIPS Potentials

When planning tasks are hard to ground, it is usually infeasible to generate all ground actions $\mathcal{A}^F$ for the set of facts F. But, it is often feasible to generate and store F (Corrêa et al. 2021, 2022). Thus, it is also feasible to store the weights $w : F \to \mathbb{Q}$. But, to compute w we would generate one constraint per ground action in $\mathcal{A}^F$, making the computation infeasible. In this section we show how to lift up the constraints. This is the first step in making the computation feasible. The next section will show how to rewrite the lifted constraints, to bring down the number of ground constraints.

Here, we first introduce lifted linear constraints and use them to reproduce the potential constraints. To see how constraints are lifted up, consider an example constraint enforcing that you can't have your cake and eat it too:

$$have(your\_cake) + eat(your\_cake) \leq 1$$

Here, $have(your\_cake)$ and $eat(your\_cake)$ are constraint variables capturing how much of your cake you have and have eaten. They are atoms combined from object $your\_cake$ and predicates eat and $have$. To generalize the constraint from $your\_cake$ to any object o in $O = \{your\_cake, my\_cake, your\_bread, ...\}$ we use a lifted constraint $have(x) + eat(x) \leq 1$. Lifted constraints contain variables, here $X = \{x\}$, that are replaced by objects, to enumerate all possible ground constraints. In our example:

$$\bigwedge_{\theta : X \to O} eat(\theta(x)) + have(\theta(x)) \leq 1$$

Which includes the original constraint. For constraints with predicates of arity at most n, the lifted formulation is exponentially smaller in n than the ground one.

Definition 14. Let $Rl = (P, X, O)$ be a relational language and $\mathcal{L} \subseteq P^X$ a set of lifted atoms with $\mathcal{L} = \{\varphi_1, ..., \varphi_n\}$. $\sum_{i=1}^n c_i \cdot \varphi_i \leq c_0$ is a lifted linear constraint over $\mathcal{L}$ and constants $c_0, ..., c_n \in \mathbb{Q}$. Multiple lifted linear constraints $C_1, ..., C_m$ may be combined as conjunction $C_1 \wedge ... \wedge C_m$. A grounding $\theta(C)$ of C with substitution $\theta : X \to O$ is $\sum_{i=1}^n c_i \cdot \theta(\varphi_i) \leq c_0$. C^F denotes the conjunction $\bigwedge_{\theta : X \to O, \theta(\mathcal{L}) \subseteq F} \theta(C)$ of all groundings under $F \subseteq P^O$. This extends to conjunctions of lifted constraints $C_\wedge = C_1 \wedge ... \wedge C_m$ as $C^F_\wedge = C^F_1 \wedge ... \wedge C^F_m$.

As in the ground case, we allow syntactic sugar for lifted linear constraints, including atom repetition in sums and atoms on both sides of the inequality. All constraints are rewritten into the canonical form of Definition 14.

The lifted linear constraints for potential heuristics, should mirror the same constraints as in the ground formulation: We replace facts with lifted atoms and ground actions with lifted actions. E.g. consider the lifted action $a = paint\_blue\_walls$ with

$$add(a) = \{blue(x_{w1}),\ blue(x_{w2})\},\ \ del(a) = \{b\_paint()\}$$

and $cost(a) = 1$ designed to paint two walls blue, if there is $b\_paint()$. Assuming that adds and deletes are guaranteed, we would create the lifted consistency constraint C as:

$$b\_paint() - blue(x_{w1}) - blue(x_{w2}) \leq 1$$

in the same fashion as the ground constraint.

Collapsing Atoms

There is a problem: If $blue(x_{wall1}), blue(x_{wall2})$ are grounded to the same fact (collapse) with $\theta = \{x_{w1} \mapsto wall,\ x_{w2} \mapsto wall\}$,. Then $add(\theta(a)) = \{blue(wall)\}$ becomes a set of size one, instead two. But, when grounding the consistency constraint with θ to

$$b\_paint() - blue(wall) - blue(wall) \leq 1$$

contains $blue(wall)$ two times. And thus would allow to create an inconsistent potential function w with $w(b\_paint()) = 3, w(blue(wall)) = 1$. In reality, θ should probably never be used to ground a, as its designed to paint two walls, not one. Thus, we could require the walls to be unequal in a precondition. To factor in such conditions, we formalize the collapsing property w.r.t. an extra condition. We distinguish if atoms always collapse or never collapse.

Definition 15. Let $Rl = (P, X, O)$ be a relational language, $\mathcal{L}_1, \mathcal{L}_2 \subseteq P^X$ sets of lifted atoms and $F \subseteq P^O$ a set of facts.

$\mathcal{L}_1$ restricted to $\mathcal{L}_2$ *never collapses* under F iff there is no $\theta : X \to O$ so that $\theta(\mathcal{L}_1 \cup \mathcal{L}_2) \subseteq F$ and $|\mathcal{L}_1| > |\theta(\mathcal{L}_1)|$.

$\mathcal{L}_1$ restricted to $\mathcal{L}_2$ *always collapses* under F iff for all $\theta : X \to O$ with $\theta(\mathcal{L}_1 \cup \mathcal{L}_2) \subseteq F$ it holds $|\mathcal{L}_1| > |\theta(\mathcal{L}_1)|$.

We now explain how to check if atoms collapse using a single query. This is often feasible when full groundings is not.

Proposition 16. *Let $(P \cup \{eq\}, X, O)$ be a relational language with $eq \notin P$. Let $\varphi_1, \varphi_2 \in P^X$ have predicate $p \in P$, i.e., $\varphi_1 = p(x_1, ..., x_n)$ and $\varphi_2 = p(y_1, ..., y_n)$. And let $\mathcal{L} \subseteq P^X$ a set of lifted atoms, $F \subseteq P^O$ a set of facts.*

If the query $\mathcal{L} \cup \{eq(x_i, y_i) \mid i \in \{1, ..., n\}\} \cup \{\varphi_1, \varphi_2\}$ is unsatisfied under $F \cup \{eq(o, o) \mid o \in O\}$, then $\{\varphi_1, \varphi_2\}$ restricted to $\mathcal{L}$ never collapses under F.

Proof. Proof by contradiction. If $\{\varphi_1, \varphi_2\}$ restricted to $\mathcal{L}$ does collapse under F, then there is a $\theta : X \to O$ with $\theta(\mathcal{L} \cup \{\varphi_1, \varphi_2\}) \subseteq F$ and $\theta(\varphi_1) = \theta(\varphi_2)$. As $\theta(\varphi_1) = \theta(\varphi_2)$, for all $i \in \{1, .., n\}$ it holds $eq(\theta(x_i), \theta(y_i)) \in \{eq(o, o) \mid o \in O\}$ as $\theta(x_i) = \theta(y_i)$. Thus the query is satisfied. □

We check all φ_1, φ_2 in $add(a), del(a)$ restricted to $pre(a)$ under F for $a \in \mathcal{A}$ before the actual computation. In the experiments there was no single task where atoms collapse.

Proposition 17. *Let $(P \cup \{neq\}, X, O)$ be a relational language with $neq \notin P$. Let $\varphi_1, \varphi_2 \in P^X$ have predicate $p \in P$, i.e., $\varphi_1 = p(x_1, ..., x_n)$ and $\varphi_2 = p(y_1, ..., y_n)$. And let $\mathcal{L} \subseteq P^X$ a set of lifted atoms, $F \subseteq P^O$ a set of facts.*

If no query $\mathcal{L} \cup \{neq(x_i, y_i)\} \cup \{\varphi_1, \varphi_2\}$ for $i \in \{1, ..., n\}$ is satisfied under $F \cup \{neq(o_1, o_2) \mid o_1, o_2 \in O, o_1 \neq o_2\}$, then $\{\varphi_1, \varphi_2\}$ restricted to $\mathcal{L}$ always collapses under F.

Proof. Proof by contradiction. If $\{\varphi_1, \varphi_2\}$ restricted to $\mathcal{L}$ does not always collapses under F, then there is a $\theta : X \to O$ with $\theta(\mathcal{L} \cup \{\varphi_1, \varphi_2\}) \subseteq F$ and $\theta(\varphi_1) \neq \theta(\varphi_2)$. As $\theta(\varphi_1) \neq \theta(\varphi_2)$, there must be an $i \in \{1, .., n\}$ where $\theta(x_i) \neq \theta(y_i)$ and so $neq(\theta(x_i), \theta(y_i)) \in \{neq(o_1, o_2) \mid o_1, o_2 \in O, o_1 \neq o_2\}$ Thus query i is satisfied. A contradiction. □

Lifted Guaranteed Adds and Deletes

Recall that our main goal is to reformulate the ground constraints as lifted constraints. We observed that this should be possible as long as atoms do not collapse. We will prove this formally in next section. However, to actually write down the lifted constraints, we still need a way identify guaranteed adds and deletes at the lifted level.

Definition 18. Let $\Pi = (Rl, \mathcal{A}, I, G)$ be a lifted planning task with $Rl = (P, X, O)$ and $F \subseteq P^O$ a set of facts. Let $a \in \mathcal{A}$ be a lifted action. A lifted atom $\varphi \in del(a)$ is a *guaranteed delete* of a under F iff for all $\theta : X \to O$ with $\theta(a) \in \mathcal{A}^F$ $\theta(\varphi)$ is a guaranteed delete of $\theta(a)$ in Π^F.

Similarly, fact $f \in add(a)$ is a *guaranteed add* of a under F iff for all $\theta : X \to O$ with $\theta(a) \in \mathcal{A}^F$ $\theta(\varphi)$ is a guaranteed add of $\theta(a) \in \mathcal{A}^F$ in Π^F.

We determine guaranteed adds and deletes in a similar spirit to the ground case. For guaranteed deletes, we check that the delete effect occurs in the preconditions, i.e., the delete effect and a precondition collapse. For guaranteed adds, we check for a precondition in the same mutex group. As, the precondition and add effect could be the same, we additionally require that the add effect and precondition never collapse.

Proposition 19. *Let $\Pi = (Rl, \mathcal{A}, I, G)$ be a lifted planning task with $Rl = (P, X, O)$ and $F \subseteq P^O$ a set of facts.*

(1) $\varphi_a \in add(a)$ is a guaranteed add of $a \in \mathcal{A}$ under F if there is a $\varphi_p \in pre(a)$ so that φ_a, φ_p are mutex under F and $\{\varphi_a, \varphi_p\}$ restricted to $pre(a)$ never collapses under F.

(2) $\varphi_d \in del(a)$ is a guaranteed delete of $a \in \mathcal{A}$ under F if there is a $\varphi_p \in pre(a)$ so that $\{\varphi_d, \varphi_p\}$ is always collapsing.

Proof. (1) Let s be a reachable state, a a lifted action with $s \xrightarrow{\theta(a)} s'$ in Π^F for some $\theta : X \to O$. As $s \xrightarrow{\theta(a)} s'$, we know that $pre(\theta(a)) \subseteq s$ and so $\theta(\varphi_p) \in s$. It also means that as $\{\varphi_a, \varphi_p\}$ is a lifted mutex group that restricted to $pre(a)$ never collapses under F, and so $\theta(\varphi_a)$, $\theta(\varphi_p)$ are mutex in Π^F. Thus $\theta(\varphi_a) \notin s$. As this holds independent of the choice for s and θ, φ_a is a guaranteed add.

(2) Let s be a reachable state, a a lifted action with $s \xrightarrow{\theta(a)} s'$ in Π^F for some $\theta : X \to O$. As $\{\varphi_a, \varphi_p\}$ restricted to $pre(a)$ always collapses under F, this means $\theta(\varphi_d) = \theta(\varphi_p)$. As $s \xrightarrow{\theta(a)} s'$, we know that $pre(\theta(a)) \subseteq s$ and so $\theta(\varphi_d) = \theta(\varphi_p) \in s$. Thus $\theta(\varphi_d) \notin s$. As this holds independent of the choice for s and θ, φ_a is a guaranteed delete. □

We perform the checks for all φ_1, φ_2 in $add(a), del(a)$ restricted to $pre(a)$ under F for $a \in \mathcal{A}$ before computing w. In each check, we determine two lifted atoms to be mutex by checking if they are part of a lifted mutex group (Fišer 2020), up to variable renaming.

Note that the identified guaranteed adds and deletes at a lifted level are inherently weaker than the ones inferred in the ground setting. In the worst case, not all fact groundings of a lifted atom are guaranteed adds/deletes. Then, one could identify the individual facts as adds/deletes, but not the lifted atom. As a result, the heuristic estimates of the lifted formulation, may be weaker.

Admissible for Non-Collapsing Actions

We now restate the previous potentials formulation with lifted constraints.

Definition 20. Let $\Pi = (Rl, \mathcal{A}, I, G)$ be a lifted planning task and F a set of facts. For every predicate $p \in P$ we define the lifted linear constraints $lmaxminc(p)$ as:

$$
\begin{aligned}
& max\_p(x_1, ..., x_{|p|}) \geq p(x_1, ..., x_{|p|}) \\
\wedge\ & max\_p(x_1, ..., x_{|p|}) \geq 0 \\
\wedge\ & min\_p(x_1, ..., x_{|p|}) \leq p(x_1, ..., x_{|p|}) \\
\wedge\ & min\_p(x_1, ..., x_{|p|}) \leq 0
\end{aligned}
$$

For every action $a \in \mathcal{A}$ we define the lifted linear constraints $laconsist(a)$ as:

$$\sum_{p(\vec{x}) \in pre(a)} 0 \cdot p(\vec{x}) + \sum_{p(\vec{x}) \in gdel(a)} p(\vec{x}) + \sum_{p(\vec{x}) \in del(a) \setminus gdel(a)} max\_p(\vec{x}) - \sum_{p(\vec{x}) \in gadd(a)} p(\vec{x}) - \sum_{p(\vec{x}) \in add(a) \setminus gadd(a)} min\_p(\vec{x}) \leq cost(a)$$

Where $gdel(a) \subseteq del(a)$ contains guaranteed deletes and $gadd(a) \subseteq add(a)$ contains of a under F.

The goal-awareness constraint $gaware$ is the ground linear constraint:

$$\sum_{p(\vec{o}) \in G} p(\vec{o}) + \sum_{p(\vec{o}) \in F \setminus (G \cup M(G))} max\_p(\vec{o}) \leq 0$$

The set $lpot(\Pi, F)$ contains the solutions to

$$gaware \wedge \bigwedge_{p \in P} lmaxminc(p)^F \wedge \bigwedge_{a \in \mathcal{A}} laconsist(a)^{F_m}$$

where $F_m := F \cup \{max\_p(o_1, ..., o_{|p|}) \mid p(o_1, ..., o_{|p|}) \in F\} \cup \{min\_p(o_1, ..., o_{|p|}) \mid p(o_1, ..., o_{|p|}) \in F\}$.

We can determine the set $M(G)$ by grounding lifted mutex groups (Fišer 2020) in time (and space) $O(|F|)$. We add the additional sum of zeroes over the precondition to restrict the groundings of the constraints. The construction would still generate consistent constraints without including the zero sum, but this way it yields the same groundings one would generate for Π^F according to the ground constraints as introduced in Section 3.

Theorem 21. *Let $Rl = (P, X, O)$ be a relational language and $F \subseteq P^O$. Let $\Pi = (Rl, \mathcal{A}, I, G)$ be a lifted planning task so that for all actions $a \in \mathcal{A}$ $add(a)$ restricted to $pre(a)$ never collapses under F and $del(a)$ restricted to $pre(a)$ never collapses under F. And, let $w \in lpot(\Pi, F)$ for some facts $F \subseteq P^O$. Then, the heuristic $h^P_w(w)$ is consistent, goal-aware, and admissible for Π^F.*

Proof. We will prove that, as action effects do not collapse, the lifted constraints from Def. 20, after grounding, could be generated according to Def. 12 for Π^F. This implies that h^P_w(w) is admissible by Prop. 13.

$gaware$ is exactly the same constraint, as the goal is not modified when grounding to Π^F. $lmaxminc(p)^F$ introduces one constraint for each $p \in P$ grounded to F, making it one constraint fact per $f \in F$ matching exactly $maxminc(f)$.

$laconsist(p)^{F_m}$ introduces one constraint for each $a \in A$ grounded to F_m, making it one constraint per ground action $\theta(a) \in A^F$, matching $aconsist(\theta(a))$ (up to the zero-sum, which can be disregarded). assuming that $\theta(gadd(a))$ are the guaranteed adds for Π^F and $\theta(gdel(a))$ are guaranteed deletes for Π^F. This is possible as Def. 18 ensures that $\theta(gadd(a))$ are the guaranteed adds for Π^F and $\theta(gdel(a))$ are guaranteed deletes for Π^F. Therefore $h^P_w(w)$ is admissible for Π^F. □

5 Reducing the Number of Ground Linear Constraints via Lifted Decomposition

In this section we show how to rewrite lifted linear constraints so that they are grounded to less ground linear constraints. We start with an example. Consider the lifted linear constraint $p(x) + q(y) \leq 3$ with variables $X = \{x, y\}$. We can rewrite it to equivalent constraints:

$$p(x) \leq tmp_1 \wedge q(y) \leq tmp_2 \wedge tmp_1 + tmp_2 \leq 3$$

which uses one, instead of two, variables per lifted linear constraint. Thus the conjunction is grounded to a linear number of constraints in $|O|$, rather than quadratic. One way to justify why this transformation is possible, is by considering the maximum value $\max_{\theta: X \to O} w(\theta(p(x))) + w(\theta(q(y)))$ of a solution w. The constraint essentially constraints this maximum to ≤ 3. Instead of maximizing over all variables groundings, we can split up the maximization per variable resulting in $\max_{\theta: \{x\} \to O} w(\theta(p(x))) + \max_{\theta: \{y\} \to O} w(\theta(q(y)))$. Each tmp_i is assigned to one such maximum (or more).

Reformulations in this spirit appear in the literature. E.g., Pommerening, Helmert, and Bonet (2017) use bucket elimination to reduce the amount of constraints for computing higher-dimensional potential heuristics. But, only through groupings over constraint variables, not over PDDL language variables. Corrêa et al. (2021) show how to decompose lifted actions into smaller ones to compute h^{max} (Bonet and Geffner 2001). The decomposition can be seen as a constraint reformulation over PDDL language variables, for the specific linear constraints that define h^{max}.

We now introduce a similar reformulation that makes use of the decomposition methods that have been successful for decomposing lifted actions (Corrêa et al. 2021, 2022; Corrêa et al. 2023). Our formulation is general and works for any lifted constraints. In particular, we capture variable dependencies using join trees [Beeri et al. 1983; Cond. 3.9].

Definition 22 (Join tree)**.** A directed graph is a tuple $G = (N, E)$ with $E \subseteq N^2$. The undirected extension of G is $(N, E \cup \{(m, n) \mid (n, m) \in E\})$. A sequence $n_1, ..., n_m$ is a path of length $m - 1$ in G if for $i \in \{2, ..., m\}$ all $(n_{i-1}, n_i) \in E$. A cycle is a path $n_1, ..., n_m$ with $n_1 = n_m$.

A graph $T = (N, E)$ is a rooted tree if its undirected extension has no cycle and there is exactly one node $root(T) \in N$ without incoming a edge $(n, root(T)) \in E$ for some $n \in N$. Let $n \in N$. The children of n are $children(n) = \{n_c \mid (n, n_c) \in E\}$. If $children(n) = \emptyset$, n is a leaf. The set of all leafs is $leafs(T) \subseteq N$.

Let $\mathcal{L}$ be a set of lifted atoms. A join tree $T = (\mathcal{L}, E)$ for $\mathcal{L}$ is a rooted tree so that for all $\varphi_1, \varphi_2 \in \mathcal{L}$, $x \in X(\varphi_1) \cap X(\varphi_2)$, x occurs in all nodes on in the unique path of shortest length from φ_1 to φ_2 in the undirected extension of T.

A join tree for $p(x), q(y)$ could consist of a single edge $(p(x), q(y))$, indicating that relating $p(x)$ and $q(y)$ requires only their shared variables, which are none. Therefore, auxiliary atoms tmp_1 and tmp_2 have no arguments.

However, a join tree does not always exist for a set of lifted atoms $\mathcal{L}$, and join trees do not introduce auxiliary

atoms like tmp_1 and tmp_2. Trees that introduce such auxiliary atoms, with shared variables when needed, are called decomposition trees (Bichler, Morak, and Woltran 2016).

Definition 23 (Decomposition tree)**.** Let $\mathcal{L}, \mathcal{L}'$ be sets of lifted atoms s.t. $\mathcal{L}' \supseteq \mathcal{L}$. $\mathcal{L}'$ extends $\mathcal{L}$ with auxiliary atoms if for all $p(\vec{x}), p(\vec{y}) \in \mathcal{L}'$: $\vec{x} \neq \vec{y}$ implies $p(\vec{x}), p(\vec{y}) \in \mathcal{L}$.

Let $\mathcal{L}'$ extend $\mathcal{L}$ with auxiliary atoms. A decomposition tree T of $\mathcal{L}$ is a join tree for $\mathcal{L}'$ so that $leafs(T) = \mathcal{L}$.

The bottom up groundings of T under facts F are $F \cup \bigcup_{\theta: X \to O, \theta(\mathcal{L}) \subseteq F} \theta(\mathcal{L}')$.

We now generalize the construction from our example based on a given decomposition tree.

Definition 24. Given a lifted linear constraint C of the form $\sum_{i=1}^{k} c_i \cdot \varphi_i \leq c_0$ over $\mathcal{L} \subseteq P^X$ and a decomposition tree $T = (N, E)$ for $\mathcal{L}$, we define the conjunction of lifted linear constraints $rw(C, T)$ as:

$$c_{root(T)} root(T) \leq c_0 \tag{1}$$

$$\wedge \bigwedge_{n \in N \setminus leafs(T)} \sum_{n' \in children(n)} c_{n'} n' \leq n \tag{2}$$

where for all $n \in N$: $c_n = c_i$ if there is an $i \in \{1, ..., k\}$ so that $\varphi_i = n$ and otherwise $c_n = 1$.

Note that the language variables occurring in each constraint are exactly those of a node in the decomposition tree and its children. Hence, each constraint contains at most twice the number of language variables of the largest decomposition tree node. Methods for constructing decomposition trees aim to minimize the number of variables per node. In the best case, the variables of each inner node are subset of the variables of one of its children. If this holds, the underlying query for which the decomposition tree is constructed is called acyclic. Acyclic queries were noticed to be very frequent as preconditions in the IPC benchmarks and exploited for lifted successor generation (Corrêa et al. 2020) and for a lifted computation of h^{add} (Bonet and Geffner 2001; Lauer et al. 2025).

This property implies that the number of variables per decomposition tree node never exceeds the maximum predicate arity. We now formally observe that in such cases, where the number of variables per node is bounded by a constant (e.g., because predicate arity is fixed and the queries are acyclic) grounding the linear constraints remains tractable even when grounding all actions is not. Note that even if a query is not acyclic, decomposition methods may still substantially reduce the number of variables per node, in some cases even to a constant (Corrêa et al. 2023).

Proposition 25. *Fix $c \in \mathbb{N}$. Let C be a lifted linear constraint over lifted atoms $\mathcal{L}$, T a decomposition tree of $\mathcal{L}$.*

If $\max_{p(\vec{x}) \in N} |p| \leq c$, then, the size of $rw(C, T)^{P^O}$ is polynomially upper-bounded w.r.t. the size of C, O.

Proof. Note that nodes with exactly one child can be removed from a decomposition tree by connecting its parent and children directly. So, w.l.o.g., assume that in the given decomposition tree $T = (N, E)$ it holds that $|children(n)| \neq 1$ for $n \in N$.

Given that $\max_{p(\vec{x}) \in N} |p| \leq c$, the amount of variables occurring in each lifted constraint of $rw(C, T)$ is bounded by $2c$. Thus the number of unique ground constraints for one lifted constraint in $rw(C, T)$ is bounded by $|O|^{2c}$. As the number of nodes $|N|$ is bounded by $2|\mathcal{L}|$, where all atoms of $\mathcal{L}$ occur in C, this proves the claim. □

We now conclude that our construction is correct. This means that even though the join tree rewriting extends the set of solutions by having additional weight assignments to the auxilary atoms, it preserves the set of corresponding potential functions.

Note that our decomposition assumes that all constants $c \in \mathbb{Q}$ are positive, because we upper-bound the maximum value a term $c \cdot p(\vec{x})$ would be replaced with. Negative constants $c \leq 0$ would instead require lower-bounding the minimum. This is no restriction: Any term $c \cdot p(\vec{x})$ with negative coefficient $c \in \mathbb{Q}_{\leq 0}$ can be equivalently replaced with $-c \cdot negp(\vec{x})$ by enforcing a constraint $-p(\vec{x}) = negp(\vec{x})$.

Theorem 26. *Let C be a lifted linear constraint over lifted atoms $\mathcal{L}$ of shape $\sum_{i=1}^{n} c_i \varphi_i \leq c_0$ so that $c_i \geq 0$ for $i \in \{1, ..., n\}$. Let T a decomposition tree of $\mathcal{L}$. Let F be a set of facts and F' the bottom up groundings of T under F.*

For any solution w to C^F, there is a solution w' to $rw(C, T)^{F'}$ so that for all $f \in F$: $w(f) = w'(f)$. And vice versa, for every solution w' to $rw(C, T)^{F'}$ there exists a solution w to C^F with the same property.

Proof. We convert w to a w' vice versa in two directions.

"⇒": Let $w : F \to \mathbb{Q}$ be a solution for C^F. Let $\Theta = \{\theta : X \to O \mid \theta(C) \in C^F\}$. Construct $w' : F' \to \mathbb{Q}$ with $w'(f) = w(f)$ for $f \in F$ and $w'(n) = \max_{\theta \in \Theta} \sum_{n_c \in children(n)} c_{n_c} w'(\theta(n_c))$ for $n_i \in F' \setminus F$.

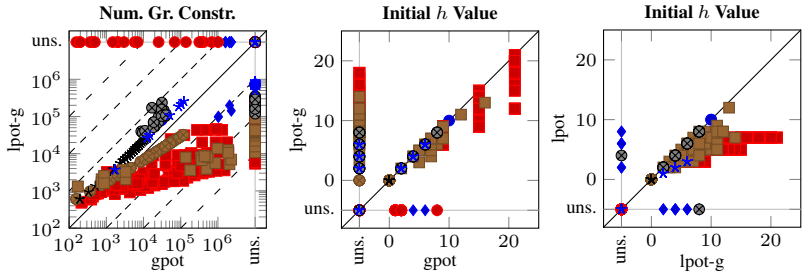
We prove that w' is a solution to all $\theta(rw(C, T)) \in rw(C, T)^{F'}$ with $\theta : X \to O$. Note that $\theta(C) \in C^F$ by standard join tree properties (Abiteboul, Hull, and Vianu 1995).

w' is constructed so that $\max_{\theta \in \Theta} \sum_{i=1}^{n} c_i w(\theta(\varphi_i)) = \max_{\theta \in \Theta} \sum_{i=1}^{n} c_i w'(\theta(\varphi_i)) = \max_{\theta \in \Theta} w'(\theta(root(T)))$. The first equality holds by $w'(f) = w(f)$ for $f \in F$ and the second by inductively applying the definition of w' too tree nodes starting from $\theta'(root(T))$ until only leafs are left.

This implies that $\theta(root(c)) \leq c_0$ for all $\theta : X \to O$ meaning w' is a solution to constraint (1) in Def. 24. Further by maximizing over all possible $\theta : X \to O, \theta(C) \in C^F$ it is a solution to all constraints of shape (2) in Def. 24. This means it is a solution for $\theta(rw(C, T)) \in rw(C, T)^{F'}$.

"⇐": Let $w' : F' \to \mathbb{Q}$ be a solution to $rw(C, T)^{F'}$. Construct $w : F \to \mathbb{Q}$ with $w(\varphi) = w'(\varphi)$ for all $\varphi \in F$. We prove that w is a solution for $\theta(C) \in C^F$ with $\theta : X \to O$. Note that $\theta(rw(C, T)) \in rw(C, T)^{F'}$ by standard join tree properties (Abiteboul, Hull, and Vianu 1995).

By a similar argument as before, we can inductively apply inequalities of shape (2) in Def. 24 to inequality (1) and obtain: $c_0 \geq \sum_{i=1}^{n} c_i w'(\theta(\varphi_i))$ where $\sum_{i=1}^{n} c_i w'(\theta(\varphi_i)) = \sum_{i=1}^{n} c_i w(\theta(\varphi_i))$ by Definition of w. This proves w is a solution to $\theta(C) \in C^F$ □



Domain	Potentials Comp. gpot	lpot	lpot-g	Coverage l-blind	l-hmax	gpot	lpot	lpot-g
Blocksw	10	**20**	**20**	0	0	0	**20**	17.5
Childsn	78.4	**100**	**100**	2.1	0	**4.2**	2.1	2.1
GED	88.8	**93.6**	**93.6**	**13.2**	7.2	9.6	7.2	7.2
⋆GED-spl	**93.6**	**93.6**	**93.6**	**13.2**	7.2	10.8	9.6	9.6
♦Logistics	20	**45**	**45**	**12.5**	10	5	7.5	2.5
OrgSyn	**55.8**	0	0	**79.2**	**79.2**	55.8	0	0
Pipesw	38	**100**	**100**	22	12	**32**	26	22
Rovers	50	**97.5**	**97.5**	0	2.5	**5**	0	0
⋆Visitall	25.2	**39.6**	**39.6**	21.6	**40.8**	10.2	11.4	15.6
Sum	460	**590**	**590**	**164**	159	133	84	77

Figure 1: Plots comparing the size of the LP in number of constraints and initial heuristic value between grounded (gpot) and our lifted potential formulations (lpot and lpot-g). Extreme points (uns.) indicate timeouts before the property is computed. The table shows the percentage of tasks where potential functions could be computed within the time limit, and solved using A* search with the resulting heuristics.

6 Experiments

We implemented the lifted LP construction from Section 4 combined with the transformation based on a join tree generated by lpopt (Bichler, Morak, and Woltran 2016) and grounded by GrinGo (Gebser, Schaub, and Thiele 2007) as available in the source code of Corrêa et al. (2023). The grounded LP was written out in Minizinc format (Nethercote et al. 2007) and solved with CPLEX v12.7.1. As optimization criteria, we maximize the heuristic value for the initial state $\sum_{f \in I} w(f)$. We compare this to the ground potentials formulation for SAS$^+$ of Pommerening et al. (2015) (gpot) with the same optimization objective as implemented in CPDDL (Fišer 2025) on tasks grounded with the technique by Corrêa et al. (2023). As an additional baseline we compare to blind search (l-blind) and h^{max} (Bonet and Geffner 2001; Corrêa et al. 2021) (l-hmax) integrated into the lifted successor generation of Corrêa et al. (2020).

For our lifted potentials, we run a basic version lpot without guaranteed adds and deletes and the advanced computation described in Sec. 4 (lpot-g). To evaluate planning tasks where enumerating actions is a bottleneck we consider the hard-to-ground benchmarks used by Lauer et al. (2021). To make up for the big discrepancy on the number of tasks per domain, we follow previous work (Höller and Behnke 2022), and report the percentage of tasks solved in each domain. The experiments were run on a cluster of machines with Intel Xeon E5-2660 CPUs with a clock speed of 2.20GHz using the Lab framework (Seipp et al. 2017). Time and memory limits were set to 30 minutes and 4GB respectively for all runs. The source code will be made available upon publication.

Our experimental data is summarized in Fig. 1. Our main research question was whether the new lifted potentials can be computed in hard to ground tasks. Our results in the table show that indeed, lpot and lpot-g significantly increase the number of instances where potential heuristics can be computed in most domains. The main reason can be observed in the leftmost plot, which shows that the number of grounded constraints produced by our approach (lpot-g) is dramatically reduced compared to grounding the task and computing potential heuristics like Pommerening et al. (2015) (gpot), sometimes up to three orders of magnitude. We observe that in some cases the number of constraints increase up to a factor of 10. The reason is that gpot simplifies the task after grounding (e.g. removing static predicates, and removing unreachable/irrelevant grounded actions) which causes our LPs to have additional variables and constraints.

All in all, these results demonstrate that encoding and decomposing LPs at the lifted level enables us to compute a significant number of solutions that would otherwise be intractable to compute. This advantage could extend to other use cases that combine LPs with automated planning, e.g., to constrain action repetitions (Lauer 2025; Lauer et al. 2026).

The next question is by how much the quality of the resulting heuristics is reduced due to missing mutex information at the lifted level. We compare this in terms of the initial heuristic value. This is an objective metric as LPs optimize for it, so it is not affected by variance on which solution happens to be selected by the LP solver. The plots in Fig. 1 show that, except for Childsnack, the heuristic values are quite close between our formulation lpot-g and the standard SAS$^+$ potential heuristics gpot. From the theoretical formulation, it is guaranteed that gpot dominates lpot-g, so being so close is actually a good empirical result. On the other hand, lpot-g dominates lpot, and using lifted mutexes to compute guaranteed adds and deletes definitely pays off.

However, the new lifted potential heuristics do not translate in an overall coverage gain compared to other lifted searches like l-blind and l-hmax. The exception is Blocksworld, where no other approach solves any task but lpot solves 20%. This shows that there are tasks where our heuristic can pay off and motivates future work to either improve the formulation or consider other optimization criteria (Fišer, Horčík, and Komenda 2020).

7 Conclusion

We introduced a lifted formulation for computing potential heuristics without full grounding. On certain planning tasks, where grounding all actions causes exponential blowup, this allows heuristics to be computed in polynomial time.

Experiments show that our approach can compute heuristics in tasks where grounding fails, producing values close to standard grounded potentials. While the impact for solving tasks is limited, our approach solves tasks that other lifted heuristics cannot. This motivates future work on improving the computed heuristic similar to the ground setting (Pommerening, Helmert, and Bonet 2017; Fišer and Steinmetz 2024; Lauer and Fišer 2025), potentially combined with advantages from other lifted constraint transformations (Kersting, Mladenov, and Tokmakov 2017; Capelli et al. 2024).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley. ISBN 0-201-53771-0.

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2–6. AAAI Press.

Areces, C.; Bustos, F.; Dominguez, M.; and Hoffmann, J. 2014. Optimizing Planning Domains by Automatic Action Schema Splitting. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press.

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

Beeri, C.; Fagin, R.; Maier, D.; and Yannakakis, M. 1983. On the Desirability of Acyclic Database Schemes. *Journal of the Association for Computing Machinery (JACM)*, 30: 479–513.

Bichler, M.; Morak, M.; and Woltran, S. 2016. lpopt: A Rule Optimization Tool for Answer Set Programming. In *Proceedings of the 26th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'16)*, 114–130.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial intelligence (AIJ)*, 129: 5–33.

Capelli, F.; Crosetti, N.; Niehren, J.; and Ramon, J. 2024. Linear Programs with Conjunctive Database Queries. *Logical Methods in Computer Science (LMCS)*, 20.

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.; Hecher, M.; Helmert, M.; Longo, D. M.; Pommerening, F.; and Woltran, S. 2023. Grounding Planning Tasks Using Tree Decompositions and Iterated Solving. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS'23)*, 100–108. AAAI Press.

Corrêa, A. B.; and Pommerening, F. 2019. An Empirical Study of Perfect Potential Heuristics. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, 114–118. AAAI Press.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Frances, 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.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Frances, G. 2022. The FF heuristic for lifted classical planning. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, 9716–9723. AAAI Press.

Dold, S.; and Helmert, M. 2024a. Higher-Dimensional Potential Heuristics: Lower Bound Criterion and Connection to Correlation Complexity. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*, 151–161. AAAI Press.

Dold, S.; and Helmert, M. 2024b. Novelty vs. potential heuristics: A comparison of hardness measures for satisficing planning. In *Proceedings of the 38th AAAI conference on artificial intelligence (AAAI'24)*, 20692–20699. AAAI Press.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence (AIJ)*, 2: 189–208.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proceedings of the 34th AAAI conference on artificial intelligence (AAAI'20)*, 9835–9842. AAAI Press.

Fišer, D.; and Komenda, A. 2018. Fact-Alternating Mutex Groups for Classical Planning. *Journal of Artificial Intelligence Research (JAIR)*, 61: 475–521.

Fišer, D.; and Steinmetz, M. 2024. Towards Feasible Higher-Dimensional Potential Heuristics. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS'24)*, 210–2020. AAAI Press.

Fišer, D.; Torralba, A.; and Hoffmann, J. 2022. Operator-Potential Heuristics for Symbolic Search. In *Proceedings of the 36th AAAI conference on artificial intelligence (AAAI'22)*, 9750–9757. AAAI Press.

Fiser, D.; Torralba, Á.; and Hoffmann, J. 2024. Boosting optimal symbolic planning: Operator-potential heuristics. *Artificial intelligence (AIJ)*, 334: 104174.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 124–133. AAAI Press.

Fišer, D. 2025. cpddl.

Francès, G.; Corrêa, A. B.; Geissmann, C.; and Pommerening, F. 2019. Generalized Potential Heuristics for Classical Planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, 5554–5561. IJCAI Organization.

Gebser, M.; Schaub, T.; and Thiele, S. 2007. GrinGo: A New Grounder for Answer Set Programming. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'7)*, 266–271. Springer.

Gerevini, A.; and Schubert, L. K. 1998. Inferring State-Constraints for Domain Independent Planning. In *Proceedings of the 15th AAAI conference on artificial intelligence (AAAI'98)*, 905–912. AAAI Press.

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

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'9)*, 162–169. AAAI Press.

Helmert, M.; Sievers, S.; Rovner, A.; and B. Corrêa, A. 2022. On the Complexity of Heuristic Synthesis for Satisficing Classical Planning: Potential Heuristics and Beyond. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, volume 32, 124–133. AAAI Press.

Hoffmann, J.; and Nebel, B. 2001. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)*, 14: 253–302.

Höller, D.; and Behnke, G. 2022. Encoding Lifted Classical Planning in Propositional Logic. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, 134–144. AAAI Press.

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.

Kersting, K.; Mladenov, M.; and Tokmakov, P. 2017. Relational linear programming. *Artificial intelligence (AIJ)*, 244: 188–216.

Lauer, P. 2025. Arguments in Favor of Allowing a Modeler to Constrain Action Repetitions. In *Proceedings of the Workshop on Knowledge Engineering for Planning and Scheduling (KEPS) at ICAPS'25*.

Lauer, P.; and Fišer, D. 2025. Potential Heuristics: Weakening Consistency Constraints. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS'25)*. AAAI Press.

Lauer, P.; Lin, S.; and Bercher, P. 2025. Tight Bounds for Lifted HTN Plan Verification and Bounded Plan Existence. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS'25)*. 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 *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, 4119–4126. IJCAI Organization.

Lauer, P.; Torralba, Á.; Höller, D.; and Hoffmann, J. 2025. Continuing the Quest for Polynomial Time Heuristics in PDDL Input Size: Tractable Cases for Lifted hAdd. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS'25)*. AAAI Press.

Lauer, P.; Zhang, Y.; Haslum, P.; and Bercher, P. 2026. I Always Told My Mom That Order Is Overrated: Unordered HTN Planning is in PSPACE and Models Problems Beyond STRIPS. In *Proceedings of the 36th International Conference on Automated Planning and Scheduling (ICAPS'26)*. AAAI Press.

Masoumi, A.; Antoniazzi, M.; and Soutchanski, M. 2015. Modeling Organic Chemistry and Planning Organic Synthesis. In *Proceedings of the 1st Global Conference on Artificial Intelligence (GCAI'15)*, 176–195.

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

Morak, M.; and Woltran, S. 2012. Preprocessing of Complex Non-Ground Rules in Answer Set Programming. In *Proceedings of the 29th International Conference on Logic Programming (ICLP'12)*, volume 17, 247–258. Dagstuhl.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP'7)*, 529–543. Springer.

Pommerening, F.; and Helmert, M. 2015. A Normal Form for Classical Planning Tasks. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 188–192. AAAI Press.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning. In *Proceedings of the 31st AAAI conference on artificial intelligence (AAAI'17)*, 3636–3643. AAAI Press.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proceedings of the 29th AAAI conference on artificial intelligence (AAAI'15)*, 3335–3341. AAAI Press.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based heuristics for cost-optimal planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 226–234. AAAI Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39: 127–177.

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

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016. Correlation Complexity of Classical Planning Domains. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, 3242–3250. IJCAI Organization.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Wichlacz, J.; Torralba, A.; and Hoffmann, J. 2019. Construction-planning models in minecraft. In *Proceedings of the 2nd Workshop on Hierarchical Planning (HPLAN) at ICAPS'19*.