Make Yourself Special: Qualified Dominance Task Reformulation For Optimal Planning

Rasmus G. Tollund, Álvaro Torralba

Aalborg University, Aalborg, Denmark
rgt@cs.aau.dk, alto@cs.aau.dk

Abstract

In cost-optimal planning, task reformulation methods aim to transform a planning task into a simpler one, while preserving that an optimal solution for the new task is an optimal solution for the original problem. They do so by exposing some hidden structure within the original planning task and making it explicit in the transformed task. In this paper, we show how to reformulate planning tasks based on qualified dominance. Qualified dominance is a novel technique that automatically analyzes a planning task to characterize the differences among states in terms of what non-dominated plans each state has. We extend this reasoning and instead use it to remove unnecessary redundancy from the planning task. We show that this reformulation can further reduce the search space and/or enhance heuristic information. In our experimental evaluation, we see that in specific cases this reduction can be very helpful; however, in general, these patterns are uncommon in our benchmarks.

Introduction

Domain-independent planning algorithms aim to efficiently solve planning problems. The key is to leverage the description of the planning task in order to automatically discover properties that make the task easier to solve than naively using exhaustive search. Often, this knowledge is encoded in terms of a heuristic function to guide the search. Reformulation methods, by contrast, aim to transform the planning task to reduce accidental complexity and make some of these properties explicit (Haslum 2007). Some examples of reformulation techniques include the merge-and-shrink framework (Helmert et al. 2014; Torralba and Sievers 2019; Sievers and Helmert 2021), reductions on SAS$^+$ tasks (Tožička et al. 2016), and irrelevance pruning (Haslum, Helmert, and Jonssson 2013).

Dominance analysis compares states s and t such that, whenever t dominates s, t is at least as close to the goal as s. This is achieved by identifying state features that are always at least as good as others. This is primarily used for discarding states during search using dominance pruning (Torralba and Hoffmann 2015; Torralba 2021). However, it can also be used for task reformulation. Subsumed transition pruning (Torralba and Kissmann 2015) simplifies the planning task by eliminating transitions $s \xrightarrow{\ell} t$ that are dominated by another transition $s \xrightarrow{\ell'} t'$ of the same state.

However, in many cases, it is not possible to find state features that are *always* at least as good as others. To tackle this, qualified dominance is a recent technique for comparing states during the search (Tollund, Larsen, and Torralba 2025), where each state pair (s, t) is assigned a transition system that represents a set of paths from s that are dominated by some path from t. The idea is that those paths can be ignored because, if they are optimal plans for s, then t has at least as good a plan as s. Tollund, Larsen, and Torralba (2025) used this to derive contrastive heuristics that estimate the distance from s to the goal, only taking into consideration non-dominated plans. However, this introduces prohibitive overhead for heuristic evaluation, calling for other methods to leverage qualified dominance.

In this paper, we consider how to use qualified dominance information to reformulate the planning task before the search starts. To that end, we revisit and extend previous methods for qualified dominance. Rather than identifying some paths from s that are dominated by t, our method considers a set of states T, which are reachable at a lower cost than s, and identifies plans from s that are dominated either by any $t \in T$ or by another path from s. These plans are redundant. We then show how this extension of qualified dominance can be used to reformulate planning tasks by directly removing parts of the task that can be proven to always result in dominated plans.

Just as qualified dominance is a generalization of dominance, the resulting reformulation is a generalization of subsumed transition pruning. Our transformation also gets rid of transitions that are fully dominated, and can additionally identify cases where a transition is not fully dominated, but taking it must have some consequence. For example, in a package delivery domain, if we load a package at some location, the truck must drive to a different location and unload it. Our reformulation is the first to make these commitments explicit, reducing the number of valid plans while guaranteeing that at least one optimal plan remains. By reformulating the task, this allows heuristics to utilize this information to improve the heuristic estimate.

In our experimental evaluation, we analyze the effects of the reformulation, compare different variants, and benchmark against state-of-the-art planning techniques. Our results show that the reformulation can significantly reduce search effort.

Background

A labelled transition system (LTS) is a tuple $\Theta = (S, L, \to, S^I, S^G)$, where S is a set of states, L is a set of labels with cost[1] $c(\ell) \in \mathbb{R}^+$, $\to \subseteq S \times L \times S$ is a transition relation, and $S^I, S^G \subseteq S$ are the set of initial and goal states, respectively. Typically, there is a single initial state; however, considering multiple initial states simplifies the notation (Büchner et al. 2024). We write $s \xrightarrow{\ell}_\Theta s'$ whenever $(s, \ell, s') \in \to$, or simply $s \xrightarrow{\ell} s'$ if Θ is clear from context.

A path from s_0 to s_n in Θ is a sequence of transitions $\rho = s_0 \xrightarrow{\ell_1} s_1 \dots \xrightarrow{\ell_n} s_n$, and we denote it as $s_0 \overset{\rho}{\rightsquigarrow} s_n$. L^* is the set of finite sequences of labels and ε is the empty sequence. $L(\rho) = \ell_1 \dots \ell_n$ is the sequence of labels of the path. A plan $\pi \in L^*$ for a state $s \in S$ is a sequence of labels s.t. there exists a path $s \overset{\rho}{\rightsquigarrow} s_g$ with $s_g \in S^G$ and $L(\rho) = \pi$. The cost of π is $c(\pi) = \sum_{\ell \in \pi} c(\ell)$. We denote by $P(s)$ the set of all plans from s to a goal state, by $P_\Theta(s)$ for a particular LTS Θ, or by $P(\Theta)$ for the union of plans over the initial states of Θ. We denote $h^*(s) = \min_{\pi \in P(s)} c(\pi)$. A plan for s is optimal if $c(\pi) = h^*(s)$. We look for a plan from some $s_0 \in S^I$ with cost $h^*(S^I) = \min_{s \in S^I} h^*(s)$.

We consider planning tasks in factored transition systems (FTS) representation (Sievers and Helmert 2021; Torralba and Sievers 2019; Büchner et al. 2024). An FTS task $\mathcal{T} = (\Theta_1, \dots, \Theta_n)$ is a tuple of LTSs, called factors, with a common set of labels, such that $\Theta_i = (S_i, L, T_i, S_i^I, S_i^G)$. We use a tuple to refer to the individual factors by their index. A state in $\mathcal{T}$ is a tuple of states $s = (s_1, \dots, s_n)$, one for each factor, $s_i \in \Theta_i$. The FTS task compactly describes the *state space*. This is another LTS $\Theta_\mathcal{T} = \Theta_i \otimes \dots \otimes \Theta_n = (S_\mathcal{T}, L, T_\mathcal{T}, S_\mathcal{T}^I, S_\mathcal{T}^G)$, where $\otimes$ is the *synchronized product* meaning $S_\mathcal{T} = S_1 \times \dots \times S_n$ is the Cartesian product of each factor's states, $T_\mathcal{T} = \{((s_1, \dots, s_n), \ell, (t_1, \dots, t_n)) \mid (s_i, \ell, t_i) \in T_i \text{ for } 1 \le i \le n\}$, $S_\mathcal{T}^I = S_1^I \times \dots \times S_n^I$, and $S_\mathcal{T}^G = S_1^G \times \dots \times S_n^G$. Note that the order of the factors is fixed, but arbitrary, as the resulting products for different orderings are isomorphic. Throughout the paper, we use subscripts to differentiate states in $S_\mathcal{T}$ (e.g., s, s', t) and states in factors Θ_i (e.g., s_i, s_i', t_i).

Figure 1 shows the planning task of our running example, where a Mars rover must navigate past a blockage of R rocks to reach its destination, optionally using D sticks of dynamite. The task consists of three types of factors: a $Rock_i$ factor for each rock i, a $Dynamite_j$ factor for each dynamite j, and a single $Rover$ factor. Each $Rock_i$ factor has states blocked (b_i) and cleared (c_i); both are goal states since the goal only constrains the rover's location. Each $Dynamite_j$ factor has states at-base (b_j, initial and goal), held (h_j), and exploded (e_j). The $Rover$ factor has states base (b, initial), blockage (m), and destination (d, goal).

The rover has two strategies to pass the blockage: (1) clear each rock individually using $clear_i$ at cost i, or (2) grab a dynamite ($grab_j$), move to the blockage, and detonate it with $explode_j$ to clear all rocks at once. After explosion the dynamite is in state e_j, which is not a goal state; the pay_j action

[1]We assume non-zero cost labels to simplify the notation, but we can support 0-cost labels by treating them as infinitesimal cost.

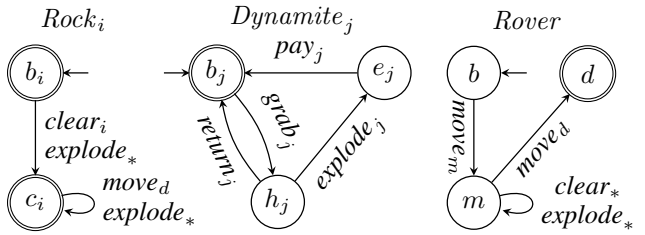

Figure 1: Mars Rover Dynamite example, parameterized with R and D for number of rocks and dynamites, thus $0 \le i < R$ and $R \le j < R + D$. Initial and goal states are marked with arrows and double circles, respectively. The $*$ denotes that a transition exists for all valid indices of the label. The cost of all labels are unit cost, except $c(clear_i) = i$ and $c(pay_j) = R^2$.

restores it to its goal state b_j at cost R^2, representing the high price of using dynamite. The $move_d$ label moves the rover from the blockage to the destination, but requires all rocks to be cleared (it only has transitions in cleared states c_i). All other labels have unit cost.

We write : after logical quantifiers to avoid ambiguity. We write $\forall x \in X : \exists y \in Y : \phi(x) \wedge \psi(x, y)$ to mean for all $x \in X$ there exists a $y \in Y$ such that $\phi(x) \wedge \psi(x, y)$. Additionally, $\forall s \xrightarrow{\ell} s' : \exists t \xrightarrow{\ell'} t' : \phi(s', t')$.

A task *reformulation* is a modification of the input task that preserves some desired properties. We consider reformulations that preserve an optimal plan, meaning at least one optimal plan in the original task is also an optimal plan in the reformulated task. The aim of a reformulation is to make the task easier to solve, by reducing the size of the state space or by exposing information to the heuristics. We call a reformulation a *simplification* if it only removes states and transitions from the state space.

Definition 1. *A dominance relation is a relation $\sqsubseteq \subseteq S \times S$, s.t. $s \sqsubseteq t \implies h^*(s) \ge h^*(t)$ (Torralba and Hoffmann 2015).*

Intuitively, $s \sqsubseteq t$ means that t is at least as close to the goal as s. Dominance pruning is usually used to prune states that are provably further from the goal than another state. To find dominance relations, we draw on the concept of simulation relations (Milner 1971).

Definition 2. *A relation $\sqsubseteq$ is a $\preceq$-*simulation *relation for Θ if $s \sqsubseteq t$ implies that (i) $t \in S^G \vee s \notin S^G$ and (ii) $\forall s \xrightarrow{\ell} s' : \exists t \xrightarrow{\ell'} t' : \ell \preceq \ell' \wedge s' \sqsubseteq t'$.*

A $\preceq_c$-simulation (where $\ell \preceq_c \ell'$ iff $c(\ell') \le c(\ell)$) is a cost-simulation. A cost-simulation is also a dominance relation. A cost-simulation can be computed in polynomial time in the size of $\Theta_\mathcal{T}$, but this is still exponential in the size of the planning task $\mathcal{T}$. To address this, one can compute a state-dominance relation for each factor. For each Θ_i, we define a relation $\sqsubseteq_i \subseteq S_i \times S_i$ that allows us to express that a state is as good as another. Similarly, we can define when a label is as good as another in a factor.

Definition 3. *A label ℓ' dominates ℓ w.r.t. $\sqsubseteq$ if $\forall s \xrightarrow{\ell} s' : \exists s \xrightarrow{\ell'} s'' : s' \sqsubseteq s''$. (Torralba and Hoffmann 2015)*

This means that in this factor any time that ℓ can be applied, one can apply ℓ' instead and reach an at least as good state. According to this definition, we construct label-relations $\preceq_{\mathrm{i}} = \{(\ell, \ell') \mid \ell' \text{ dominates } \ell \text{ in } \Theta_i \wedge c(\ell') \leq c(\ell)\}$. Based on this, we can reason which labels can be applied without negative side effects on any other factor, $\preceq_{\bar{\mathrm{i}}} = \bigcap_{j \neq i} \preceq_j$. Finally, the label-dominance simulation technique computes a $\preceq_{\bar{i}}$-simulation over each factor, which is a tuple $(\sqsubseteq_1, \ldots, \sqsubseteq_n)$ of relations such that whenever $s_i \sqsubseteq_i t_i$ then $t_i \in S_i^G \vee s_i \notin S_i^G$ and $\forall s_i \xrightarrow{\ell} s_i' : \exists t_i \xrightarrow{\ell'} t_i' : s_i' \sqsubseteq_i t_i' \wedge \ell \preceq_{\bar{i}} \ell'$. Despite being a recursive definition, the coarsest label-dominance simulation can be computed in polynomial time in the size of the planning task (Torralba and Hoffmann 2015). Furthermore, one can compute a dominance relation for $\Theta_{\mathcal{T}}$ as $s \sqsubseteq t$ iff $s_i \sqsubseteq_i t_i$ for all $i \in \{1, \ldots, n\}$.

We always consider a *noop* transformation, adding a 0-cost label *noop* with a transition $s \xrightarrow{noop} s$ for all $s \in S$. This allows us to reply to any transition $s \xrightarrow{\ell} s'$ with $t \xrightarrow{noop} t$ whenever $s' \sqsubseteq t$.

The concept of qualified dominance has previously been studied by Tollund, Larsen, and Torralba (2025). We generalize their definition to allow for comparing against multiple states. Intuitively, a qualified dominance function maps a state s and a set of states T to a set of paths from s for which a state in T is guaranteed to have a better plan. A *qualified dominance function* is a function $\mathcal{D} : S \times 2^S \to 2^{L^*}$ from a state and set of states to a language over the labels s.t. $\forall \pi \in \mathcal{D}(s, T) \cap P(s) : \exists t \in T : h^*(t) \leq c(\pi)$. In previous work, this was computed by constructing an LTS with states (s, t) for every $s, t \in S$ encoding the paths from s that are dominated by t. This LTS is then determinized and complemented such that we have an LTS describing paths of s that are *not* dominated by t. Their technique requires the task to be deterministic. In this paper, we introduce a new, stronger technique which constructs the negated LTS directly. In addition, while the previous method required the input task to be deterministic, our method does not.

Task Reformulation

Task reformulation methods transform a planning task into another, such that a solution to the reformulated task can be transformed back to a solution to the original task. In this paper, we are interested in reformulations that preserve the labels and their cost, and such that any plan for the reformulated task is a plan for the original task (the reformulation must not introduce any new plans). Furthermore, at least one optimal plan should be preserved. Thus, if we find an (optimal) solution to the reformulated task, this is also an (optimal) solution for the original one.

Definition 4. *A* task reformulation λ *is a procedure that reformulates a planning task $\mathcal{T}$ into $\mathcal{T}^\lambda = \lambda(\mathcal{T})$ in such a way that $P(\mathcal{T}^\lambda) \subseteq P(\mathcal{T})$ and $P^*(\mathcal{T}) \cap P^*(\mathcal{T}^\lambda) \neq \emptyset$.*

Algorithm 1 shows an overview of the procedure to compute the task reformulation. The algorithm transforms one factor at a time in each iteration:

$$\lambda_i(\mathcal{T}) = (\Theta_1, \ldots, \Theta_{i-1}, \Theta_i^\lambda, \Theta_{i+1}, \ldots, \Theta_n).$$

Our task reformulation will be based on qualified dominance, which requires computing dominance on the task. First, we compute a label-dominance simulation relation on the original task. Then, we reformulate the factors one by one, replacing the original factors by the reformulated ones and recomputing dominance for the new task. The DOMINANCE procedure computes dominance as described in Torralba and Hoffmann (2015). In the following sections, we will explain the QD-REFORMULATE procedure, which is based on qualified dominance.

Algorithm 1: Qualified dominance task reformulation overview

Input: Planning task $\mathcal{T} = (\Theta_1, \ldots, \Theta_n)$ and $k \in \mathbb{N}_0$
Output: Transformed planning task $\mathcal{T}^\lambda$

```
⊑_1, ..., ⊑_n, ≼_1, ..., ≼_n ← DOMINANCE(T)
for i ← 1 to n do
    ≼_ī ← ⋂_{j≠i} ≼_j
    Θ_i^λ ← QD-REFORMULATE(Θ_i, ≼_c ∩ ≼_ī, k)
    T^λ ← (Θ_1^λ, ..., Θ_i^λ, Θ_{i+1}, ..., Θ_n)
    ⊑_1, ..., ⊑_n, ≼_1, ..., ≼_n ← DOMINANCE(T^λ)
```

Qualified Dominance Revisited

The core idea of qualified dominance is, given an LTS Θ, to construct a new LTS Θ_{QD} whose states correspond to pairs (s, T), where $s \in S$ is an original state and $T \subseteq S$ is a set of *comparison states*. It answers the question: what can s do that T cannot? A plan of (s, T) is a plan of s for which no state $t \in T$ has a plan that dominates it. Intuitively, (s, T) filters out the plans of s that are provably suboptimal because some $t \in T$ can do better. We will elaborate on how these sets of states T are chosen later.

We simplify the definition of the qualified dominance transition system by defining a way to construct it directly. This improves over the definition of Tollund, Larsen, and Torralba (2025) by not needing determinization and negation, and by ensuring that the plans of the qualified dominance LTS are a subset of the original LTS. The definition is relative to a *label relation* $\preceq \subseteq L \times L$, specifying which label substitutions are allowed (e.g., replacing a label with a cheaper one). Thus, $t \in T$ does not need to have the same plan as s; rather, it needs a plan where each label is at least as good according to $\preceq$.

For a single LTS, we can use the cost-label relation $\preceq_c$ (i.e., $\ell \preceq_c \ell'$ iff $c(\ell') \leq c(\ell)$), ensuring plans of comparison states no more expensive. In a multi-factor planning task, when reformulating factor Θ_i, we need to also consider the effects of ℓ and ℓ' in the other factors. Therefore, we use the label dominance in all other factors, $\preceq_{\bar{i}}$. Recall that when ℓ' dominates ℓ in factor j ($\ell \preceq_j \ell'$) it means that in Θ_j whenever you can apply ℓ you can also apply ℓ' and get to an at least as good state. Intuitively, this means if $\ell \preceq_{\bar{i}} \ell'$ and $\ell \preceq_c \ell'$ we can allow substituting ℓ' with ℓ in Θ_i.

Definition 5. *Let $\Theta = (S, L, \to, S^I, S^G)$ be an LTS and $\preceq \subseteq L \times L$ a label relation. The* qualified dominance LTS *is $\Theta_{QD} = (S \times 2^S, L, \to_{\Theta_{QD}}, \{(s_i, \emptyset) \mid s_i \in S^I\}, S^G_{QD})$, where $(s,T) \in S^G_{QD}$ iff $s \in S^G \wedge \forall t \in T : t \notin S^G$, and for every transition $s \xrightarrow{\ell}_\Theta s'$ and $T \subseteq 2^S$ we have a transition $(s,T) \xrightarrow{\ell}_{\Theta_{QD}} (s', T')$ with $T' = \{\, t' \in S \mid \exists t \in T : \exists t \xrightarrow{\ell'}_\Theta t' : \ell \preceq \ell' \,\}$.*

Each transition $(s,T) \xrightarrow{\ell}_{\Theta_{QD}} (s',T')$ encodes that $s \xrightarrow{\ell}_\Theta s'$ in the original LTS and that for every $t' \in T'$ there was some $t \in T$ that has a transition to t' with a label that is as good as ℓ according to $\preceq$. A state (s,T) is only a goal state if s is a goal state and no state in T is a goal state. This is because if s is a goal state but there is also a goal state $t \in T$, then we can safely discard the (empty) plan of s, as it is not better than the (empty) plan of t.

In the previous work, they utilized this to infer constraints for the non-dominated plans of a state s, given a set of states T where $\forall t \in T : g^*(t) \leq g^*(s)$. Since the states in T are reachable with no greater cost than s, we can discard plans of s dominated by any $t \in T$. However, this is prohibitively expensive because for every search node expansion we need to find a set T among all expanded search nodes to compare to and construct (or have precomputed) the part of the qualified dominance LTS reachable from (s,T).

Qualified Dominance As Reformulation

To overcome the expensive computation during search, we do the qualified dominance analysis up-front as a task reformulation. To achieve this, we need to somehow know which states are possible to compare to. We address this by extending Definition 5 with a new rule for T' when considering the transition $(s,T) \xrightarrow{\ell}_{\Theta_{QD}} (s',T')$. We now also compare against *sibling transitions*: other transitions from s. Thus, for a transition $s \xrightarrow{\ell} s'$, we compare it with siblings like $s \xrightarrow{\ell'} s''$ where $\ell \preceq \ell'$, resulting in $(s,\emptyset) \xrightarrow{\ell} (s', \{s''\})$. We also compare to the *parent* state s using the sibling $s \xrightarrow{noop} s$. Thus we can start with the initial states $(s_i, \emptyset)$ for $s_i \in S^I$ and construct the reachable state-space from those. We can also prune unsolvable states. A sufficient (but not necessary) condition for an state (s,T) to be unsolvable is if there is $t \in T$ s.t. $s \sqsubseteq t$. Then the dominance relation guarantees that t has an at least as good plan for *every* plan of s.

Comparing sibling transitions raises a problem when two of them are equally good. For example, if $\ell \preceq \ell', \ell' \preceq \ell, s \xrightarrow{\ell} s'$ and $s \xrightarrow{\ell'} s'$ we cannot construct both $(s,\emptyset) \xrightarrow{\ell} (s', \{s'\})$ and $(s,\emptyset) \xrightarrow{\ell'} (s', \{s'\})$. These states are both dead-ends and we essentially prune ℓ because of ℓ' but then also ℓ' because of ℓ. Therefore, we need some tie-breaking strategy to ensure that one of the equally good transitions is always preserved. We encode this using a transition order $\lhd$. The transition order should only break ties between equally good labels, so $(s \xrightarrow{\ell} s') \lhd (s \xrightarrow{\ell'} s')$ if $\ell \preceq \ell'$ and $\ell' \not\preceq \ell$. For equally good labels we pick an arbitrary order.

The new definition is parameterized with a *label relation*; a *transition order* $\lhd$, a strict partial order over transitions used to break ties between equally good transitions so they do not prune each other; and a *decision strategy* α, which selects a subset of comparison states (T') to track (used later for polynomial-time approximations). For now, we can assume this is the identity, so $\alpha(s \xrightarrow{\ell}_\Theta s', T') = T'$.

Definition 6. *Let $\Theta = (S, L, \to, S^I, S^G)$ be an LTS, $\preceq \subseteq L \times L$ a label relation, $\lhd$ a strict partial order over transitions, and $\alpha : (\to \times 2^S) \to 2^S$ a decision strategy. The* qualified dominance reformulation *is $\lambda_{\preceq,\lhd,\alpha}(\Theta) = (S \times 2^S, L, \to_{\Theta^\lambda}, \{(s_i,\emptyset) \mid s_i \in S^I\}, S^G_\lambda) = \Theta^\lambda$, where $(s,T) \in S^G_\lambda$ iff $s \in S^G \wedge \forall t \in T : t \notin S^G$, and for every transition $s \xrightarrow{\ell}_\Theta s'$ and $T \subseteq 2^S$ we have a transition $(s,T) \xrightarrow{\ell}_{\Theta_\lambda} (s', T'')$ with $T'' = \alpha(s \xrightarrow{\ell}_\Theta s', T')$ and $T' = \{\, t' \in S \mid \mathbf{previous}(t') \vee \mathbf{sibling}(t') \,\}$, where*

$$\mathbf{previous}(t') = \exists t \in T : \exists t \xrightarrow{\ell'}_\Theta t' : \ell \preceq \ell'$$
$$\mathbf{sibling}(t') = \exists s \xrightarrow{\ell'} t' : \ell \preceq \ell' \wedge (s \xrightarrow{\ell} t) \lhd (s \xrightarrow{\ell'} t')$$

Importantly, we only need to construct the reachable part of this LTS and we can prune dead-end states. Still, the size of Θ_λ may be exponentially larger than Θ. However, we apply the construction *per factor*, so the run-time is polynomial in the number of factors. In many domains, the maximum number of states per factor is bounded and thus the entire construction runs in polynomial time. When this bound is small, there is no issue. For the general case, we can under-approximate with a decision strategy $\alpha(s \xrightarrow{\ell} s', T) = T'$ with $|T'| \leq k$ for a fixed constant k and $T' \subseteq T$. We select arbitrary states to keep in T'. This limits the number of states to $\mathcal{O}(|S|^{k+1})$, which is polynomial for fixed k.

Example. Figure 2 shows the qualified dominance reformulation for our running example (Figure 1). We walk through the $Dynamite_j$ factor step by step. The label-relation relevant for transforming this fator is $\preceq^r_j$, with $return_j \preceq^r_j noop$ and $grab_j \preceq^r_j noop$, meaning that $return_j$ and $grab_j$ do not have positive effects in the other factors, so $noop$ is as good as them in all factors except $Dynamite_j$. The superscript r indicates that this is a reduced version (i.e. a subset) of the label-dominance relation. We will explain the derivation of $\preceq^r$ later. *Initial state $(b_j, \emptyset)$:* The only transition is $b_j \xrightarrow{\overrightarrow{grab}_j} h_j$. Condition **previous** does not apply since $T = \emptyset$. However, **sibling** applies: b_j also has the implicit sibling self-loop $b_j \xrightarrow{noop} b_j$, and $grab_j \preceq^r_j noop$. So b_j is added to the comparison set, yielding $(b_j, \emptyset) \xrightarrow{grab_j} (h_j, \{b_j\})$. *State $(h_j, \{b_j\})$:* We are in h_j, comparing against b_j. Two transitions are possible: (1) $explode_j$, there is no better sibling transition, so $(h_j, \{b_j\}) \xrightarrow{explode_j} (e_j, \emptyset)$. (2) $return_j$: Since $return_j \preceq^r_j noop$ and $b_j \xrightarrow{noop} b_j$, **previous** adds b_j to T', giving $(h_j, \{b_j\}) \xrightarrow{return_j} (b_j, \{b_j\})$. But $b_j \in T$ and $b_j \sqsubseteq b_j$, making it a *dead end*. Intuitively, grabbing the dynamite and returning it is no better than doing nothing—both $grab_j$ and $return_j$ could be replaced by $noop$. *State $(e_j, \emptyset)$:* Applying pay_j leads to $(b_j, \emptyset)$, closing the cycle. The transformation has discovered that any plan using $grab_j$ must also use $explode_j$ and pay_j—grabbing and returning the dynamite is never useful.

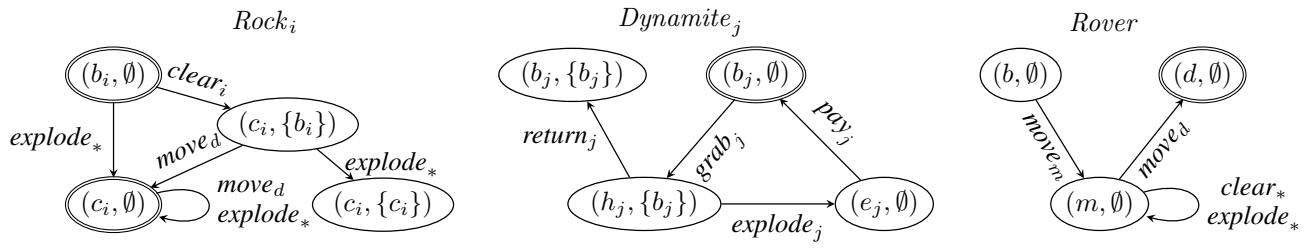

Figure 2: Qualified Dominance task reformulation of Mars Rover Dynamite example. The reformulation is achieved with $noop$ transformation and by preserving the reduced label relation $(\preceq_i^r)_{i=0}^n$ where $clear_i \preceq_i^r noop$ for all $0 \leq i < R$; $return_j \preceq_j^r noop$ and $grab_j \preceq_j^r noop$ for all $R \leq j < D$.

Proof of Correctness

In this section, we prove that our reformulation satisfies Definition 4, meaning it creates no new plans and preserves at least one optimal plan. For some proofs, we have provided proof sketches. The full proofs will be made available in an appendix. We first prove properties about the relation between the original and reformulated factor, ignoring the rest of the task. Then, we extend these properties to the case of transforming a factor as part of a task with several factors.

First, we prove that the properties of individual transitions from Definition 6 extend to paths. We extend label relations $\preceq$ and transitions orders $\lhd$ to sequences of labels and transitions, respectively, as $\ell_1\ell_2\cdots\ell_n \preceq_* \ell'_1\ell'_2\cdots\ell'_m$ iff $n = m$ and $\forall 1 \leq i \leq n : \ell_i \preceq \ell'_i$, and respectively for $\lhd_*$.

Lemma 1. *Let Θ be an LTS and $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$. For all paths $(s,T) \overset{\rho}{\rightsquigarrow}_{\Theta^\lambda} (s',T')$, there exists $s \overset{\ell_1}{\rightsquigarrow}_\Theta s'$, s.t. $L(\rho) = L(\rho_1)$ and for all $t' \in T'$ there is a path ρ_2 s.t. $L(\rho_1) \preceq_* L(\rho_2)$ and either (I) $\exists t \in T : t \overset{\rho_2}{\rightsquigarrow}_\Theta t'$, or (II) $s \overset{\rho_2}{\rightsquigarrow}_\Theta t' \wedge \rho_1 \lhd_* \rho_2$.*

Proof sketch. Full proof is in the appendix. The proof applies induction to show that the properties of individual transition (i) and (ii) from Definition 6 extend to paths in the reformulated LTS. □

We now prove that the transformation does not create any new plans, and afterwards that it preserves an optimal plan. With these two properties, the transformation satisfies the Definition 4 about task reformulations.

Lemma 2. *For an LTS Θ and $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$, $P(\Theta^\lambda) \subseteq P(\Theta)$.*

Proof. Let $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$. We show that $P_{\Theta^\lambda}((s,T)) \subseteq P_\Theta(s)$ for all $s \in S$ and $T \subseteq S$. We prove by induction on the length of the plans n with the hypothesis $\forall s \in S, T \subseteq S : \forall \pi \in P_{\Theta^\lambda}((s,T)) : |\pi| \leq n \implies \pi \in P_\Theta(s)$. *Base case ($n = 0$)*: By Definition 6 $(s,T) \in S^{G\prime}$ only if $s \in S^G$. *Inductive case*: Let $\pi \in P_{\Theta^\lambda}(s)$, $|\pi| = n+1$ and $\pi = \ell\pi'$, then $|\pi'| = n$ and there must exist a transition $(s,T) \xrightarrow{\ell}_{\Theta^\lambda} (s',T')$ s.t. $\pi' \in P_{\Theta^\lambda}((s',T'))$. Consequently, there is $s \xrightarrow{\ell}_\Theta s'$ and by the induction hypothesis $\pi' \in P_\Theta(s')$. Thus, $\pi \in P_\Theta(s)$. □

Lemma 3. *If $\preceq \subseteq \preceq_c$ then $\lambda_{\preceq,\lhd,\alpha}$ preserves an optimal plan π which has a maximal path ρ w.r.t. $\lhd_*$, s.t. $s_0 \overset{\rho}{\rightsquigarrow} s_g$ and $s_g \in S^G$.*

Proof. Let $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$. We prove by contradiction. Assume $\pi \notin P(\Theta^\lambda)$. Let $(s_0,\emptyset) \overset{\rho'}{\rightsquigarrow} (s_g,T)$ be a path with $L(\rho') = \pi$. Since π is not a plan of Θ^λ then $(s_g,T) \notin S^{G\prime}$. Therefore, there exists $t \in T$ s.t. $t \in S^G$. Now, by Lemma 1, there is a path $s_0 \overset{\rho''}{\rightsquigarrow} t$ s.t. $\rho \lhd_* \rho''$ and since $\pi \preceq_* L(\rho'')$ and $\preceq \subseteq \preceq_c$ then $c(\rho'') \leq c(\rho)$. This contradicts that π is an optimal plan with a maximal path. □

We have proved that the qualified dominance transformation is a task reformulation when considering a task consisting of a single factor. Specifically, after transforming one factor of the planning task, we get a new planning task which has no new plans and preserves one optimal plan. The core idea is to now use the label-dominance to reason about how the labels affect the other factors. By allowing labels to only be substituted with labels that are guaranteed to always be at least as good in all the other factors, this ensures that the path we compare against is at least as good in all other factors. In the following, we prove that we preserve a shortest optimal plan (i.e., an optimal plan that has the minimum number of actions among all optimal plans). This is relevant, as it ensure the preserved plan does not have any $noop$, and thus they can safely be removed again. Recall that we add $noop$ in order to improve the dominance reasoning.

Theorem 4. *Let $\mathcal{T} = (\Theta_1,\ldots,\Theta_n)$ be a planning task, $\mathcal{T}' = (\Theta_1,\ldots,\Theta_{i-1},\lambda_{\preceq,\lhd,\alpha}(\Theta_i),\Theta_{i+1},\ldots,\Theta_n)$, and $\preceq = \preceq_{\bar{i}} \cap \preceq_c$. Then, $P(\mathcal{T}') \subseteq P(\mathcal{T})$ and $\mathcal{T}'$ preserves a shortest optimal plan of $\mathcal{T}$.*

Proof sketch. $P(\mathcal{T}') \subseteq P(\mathcal{T})$ follows from Lemma 2, since the plans of $\mathcal{T}'$ is the intersection of the plans of each factor.

The core idea of the proof is to show that every plan π with a maximal (w.r.t. $\lhd_*$) path ρ_i in Θ_i is preserved. For simplicity we assume that the task is deterministic, thus there is a one-to-one correspondence between plans and paths in the LTSs. We prove by contradiction: assume $\pi \notin P(\mathcal{T}')$. This means applying π in Θ_i^λ leads to a non-goal state (g_i,T_i) where g_i is a goal state of Θ_i and there is a goal state $t_i \in T_i$. By Lemma 1 there is a path ρ'_i s.t. $\rho_i \lhd_* \rho'_i$. In the full proof, we show that this path in Θ_i corresponds

to a plan of $\mathcal{T}$, contradicting that π is a plan with a maximal path in Θ_i. The intuition is that since $L(\rho_i) \preceq_* L(\rho_i')$, and $\preceq = \preceq_{\bar{i}} \cap \preceq_c$, every label applied from ρ_i' is guaranteed to result in an at least as good state in all other factors, thus $L(\rho_i')$ is also an at least as good plan in all other factors. □

We have now shown that the reformulation works for a single factor of the planning task. Since the result is just another planning task, we can simply repeat the process on the other factors. However, when constructing the next factor, we need a label-dominance relation for the new task $\mathcal{T}'$. Transforming the first factor may change the label-dominance in that factor, which may then change the dominance in other factors and so on. Recall that in Algorithm 1, we need to recompute dominance on the new task after modifying each factor. In practice, we do not need to completely recompute dominance, but only to reset the dominance in the new factor and rerun the dominance computation.

Preserving Label-Dominance

When transforming one factor the dominance may change in other factors, and fewer labels dominate each other, and therefore we might not be able to do as effective of a transformation. Thus, the overall reformulation procedure is dependent on the order in which the factors are transformed. Since it is not clear in which factors the transformation will benefit the most, it may be detrimental to the performance if a wrong order is chosen. In this section we provide two approximate approaches to preserve the label-dominance.

In order to preserve label-dominance in the factor, it means that whenever $\ell \preceq_i \ell'$ we need that $\forall (s,T) \xrightarrow{\ell} (s',T') : \exists (s,T) \xrightarrow{\ell'} (s'',T'') : (s',T') \sqsubseteq (s'',T'')$. Therefore, we need to know the state-dominance in the factor. However, this requires us to first do the transformation $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$, then compute dominance on Θ^λ, and finally do some transformation that ensures label-dominance is kept and that it is still an optimal plan preserving transformation. We simplify this process by under-approximating the state-dominance of Θ^λ by using the state-dominance of Θ. The intuition is that when deciding if $(s,T) \sqsubseteq (s',T')$, the more states there are in T the fewer plans s has, so if $s \sqsubseteq s'$ and T is more limiting than T', then (s',T') must dominate. We know that it is more limiting if all the states of T' are dominated by a state of T.

Lemma 5. *Let $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$ and $\sqsubseteq$ be a $\preceq$-simulation relation on Θ, with $\alpha(\cdot,T) = T$. Let $\sqsubseteq'$ be a relation where $(s,T) \sqsubseteq' (s',T')$ iff $s \sqsubseteq s'$ and $\forall t' \in T' : \exists t \in T : t' \sqsubseteq t$. Then, $\sqsubseteq'$ is a $\preceq$-simulation relation on Θ^λ.*

Proof. We show that $\sqsubseteq'$ satisfies Definition 2. Firstly, for (i) we show that $(s,T) \sqsubseteq (s',T')$ implies $(s',T') \in S^{G\prime} \vee (s,T) \notin S^{G\prime}$. If $s \notin S^G$, then $(s,T) \notin S^{G\prime}$ which satisfies the condition. If $s \in S^G$, then $s' \in S^G$ because $s \sqsubseteq s'$. If $\exists t \in T : t \in S^G$ then also $(s,T) \notin S^{G\prime}$. Otherwise, we must have $\forall t \in T : t \notin S^G$ and since we have $t' \sqsubseteq t$ for all $t' \in T'$, then also $t' \notin S^G$ and thus $(s',T') \in S^{G\prime}$. Secondly, for (ii) we show it also implies that $\forall (s,T) \xrightarrow{\ell} (s'',T'') : \exists (s',T') \xrightarrow{\ell'} (s''',T''') : \ell \preceq \ell' \wedge (s'',T'') \sqsubseteq (s''',T''')$. We know that there is a transition $(s',T') \xrightarrow{\ell'} (s''',T''')$ since there is $s' \xrightarrow{\ell'} s'''$ that $s'' \sqsubseteq s'''$, since $\sqsubseteq$ is a $\preceq$-simulation relation. Similarly, we also know that $\forall t' \in T : \exists t \in T : \forall t' \xrightarrow{\ell'''} t''' : \exists t \xrightarrow{\ell''} t'' : t'' \sqsubseteq t'''$. Thus, $(s'',T'') \sqsubseteq (s''',T''')$. □

When constructing the transformation in Definition 6, we can under-approximate the T sets and still get an optimal plan preserving transformation, by creating a decision strategy for which elements to include in T based on label-dominance. Our strategy to preserve label-dominance is to consider a state (s,T) and whenever $\ell \preceq_i \ell'$ then for all $(s,T) \xrightarrow{\ell} (s',T')$ we know there is a transition $(s,T) \xrightarrow{\ell'} (s'',T'')$ s.t. $s' \sqsubseteq s''$. We then change this transition to $(s,T) \xrightarrow{\ell'} (s'',T_r'')$ where $T_r'' \subseteq T''$ s.t. $\forall t' \in T_r'' : \exists t \in T' : t \sqsubseteq t'$. In other words, when we violate label-dominance because we don't know if $(s',T') \sqsubseteq (s'',T'')$, we remove elements from T'' until it dominates. Importantly, by Lemma 5, we can always choose $T_r'' = \emptyset$, if no larger set permits it.

For each state (s_i,T_i) of the factor Θ_i we are transforming, we compute a mapping $a_{(s_i,T_i)}(s_i \xrightarrow{\ell}_{\Theta_i} s_i') = T_i$, from transitions to some T_i sets, for all transitions of s, s.t. $\forall \ell, \ell' \in L : \ell \preceq_i \ell' \implies \forall s_i \xrightarrow{\ell} s_i' : \exists s_i \xrightarrow{\ell'} s_i'' : \forall t' \in a_{(s_i,T_i)}(s_i \xrightarrow{\ell'} s_i'') : \exists t \in a_{(s_i,T_i)}(s_i \xrightarrow{\ell} s_i') : t' \sqsubseteq t$. Fulfilling this condition means, by Lemma 5, that the target state of ℓ' will dominate that of ℓ. We compute such a mapping by initializing each entry to the T' generated by Definition 6, and then we remove states of T' that do not satisfy the condition until all do. Then we use the decision strategy $\alpha(s \xrightarrow{\ell} s', T) = a_{(s,T)}(s \xrightarrow{\ell} s')$. In practice, we compute a lazily when needed.

An important caveat of this technique is that Lemma 5 only applies when we do not modify T with α. Since this changes the behaviour and we can no longer be sure which states are in T or not. For example, after altering the state (s,T) there could be another state (s',T') which depended on the original behaviour of (s,T), and now the dominance $(s',T') \sqsubseteq (s'',T'')$ doesn't hold. Therefore, this is only an approximate method to preserve label-dominance, and we then check if it is actually preserved. If it is not preserved, we must fall back to recomputing dominance for all factors. However, in practice, we have never observed that this method does not manage to preserve the label-dominance.

Preserving Reduced Label-Dominance

Preserving label-dominance reduces the information of the reformulation, weakening the technique. When preserving nothing, dominance needs to be recomputed for the entire task and it risks making transformations that changes label-dominance s.t. in other factors the transformation is weaker. Therefore, we take a middle way: we only preserve enough label-dominance to ensure dominance is preserved in other factors. We select subsets $\preceq_i^r \subseteq \preceq_i$ of the label-dominance relations in each factor i s.t. each initially computed $\preceq_i$-simulation relation $\sqsubseteq_i$ is also a $\preceq_i^r$-simulation relation. In other words, $\preceq_i^r$ is a subset of the label-dominance in all other factors than i that is sufficient to prove the relation $\sqsubseteq_i$.

Computing the reduced label-dominance relation can be done by considering all state pairs where $s_i \sqsubseteq t_i$

and constructing for each $s_i \xrightarrow{\ell} s'_i$ a set of label-pairs $\{ (\ell, \ell') \mid t_i \xrightarrow{\ell'} t'_i \wedge s'_i \sqsubseteq t'_i \wedge \ell \preceq \ell' \}$. This set represents a disjunctive condition for the label-dominance that must be present in order for the $s_i \xrightarrow{\ell} s'_i$ transition to be dominated. We then compute a hitting set (a set that intersects all the sets) over all the disjunctive label-pair sets. Since the hitting set problem is NP-complete (Karp 1972), we use a polynomial-time approximation algorithm. This gives us $\preceq^r_i$, and then for a factor j the label-dominance we need to preserve in that factor is $\preceq^r_j = \bigcup_{k \neq j} \preceq^r_k$.

In Algorithm 1, if we preserved label-dominance, line 5 skipped entirely. If we preserved reduced label-dominance we only need to recompute the dominance of the transformed factor.

Example Revisiting Figure 2, we can now understand how the label-relation was computed. In the rock factor, we have $b_i \sqsubseteq c_i$. To preserve this, we need that $\mathit{clear}_i \preceq \mathit{noop}$ because for the transition $b_i \xrightarrow{\mathit{clear}_i} c_i$ only have the transition $c_i \xrightarrow{\mathit{noop}} c_i$. This allows us to have the transition $(b_i, \emptyset) \xrightarrow{\mathit{clear}_i} (c_i, \{b_i\})$. Observe now that, in this state, we can no longer explode the dynamite without being dominated. In the dynamite factor, we need to preserve that $e_j \sqsubseteq b_j$, which means $\mathit{pay}_j \preceq \mathit{noop}$. However, this does not affect the transformation because pay_j is only relevant in this factor. The reformulation has captured something quite powerful, namely that after clearing a rock it doesn't make sense to explode the dynamite, because we could have just not cleared the rock in the first place. Similarly, after grabbing the dynamite it doesn't make sense to return it without exploding it. This simplifies the problem a lot and allows the heuristic to focus on the relevant parts of the search space.

Task Shrinking

Since the reformulation can increase the task size, and thus potentially increase the apparent complexity of solving the task, we also consider a method for shrinking the task after the transformation. The simplest approach is to apply an abstraction that shrinks all states originating from the same source state together. For a given state in the original factor s, we thus shrink $(s, T), (s, T'), (s, T''), \ldots$ to one state. When performing this step, the reformulation is guaranteed to be a simplification, (i.e., there are no more states or transitions in the task). Formally, we create an *abstraction* $\alpha : S^{\lambda}_{\mathit{pruned}} \to S$, which maps non-dead states from the pruned qualified dominance reformulation back to original states, s.t. $\alpha((s, T)) = s$. Pruned meaning we have removed unreachable or dead-end state. In our running example, the $(b_j, \{b_j\})$ state is pruned because it is a dead-end. The abstraction induces the LTS $\Theta^{\lambda,\alpha} = (S, L, \to_{\lambda,\alpha}, S^I, S^G_{\lambda,\alpha})$, where $s \xrightarrow{\ell}_{\lambda,\alpha} s'$ iff $\exists T, T' : (s, T) \xrightarrow{\ell}_{\Theta^{\lambda}} (s', T')$, and $s \in S^G_{\lambda,\alpha}$ iff $\exists T : (s, T) \in S^G_{\lambda}$.

Theorem 6. *$P(\Theta^{\lambda,\alpha}) \subseteq P(\Theta)$ and $P^*(\Theta^{\lambda,\alpha}) \cap P^*(\Theta) \neq \emptyset$*

Proof. There are no new plans because a transition can only be in $\Theta^{\lambda,\alpha}$ if it was in Θ. An optimal plan is preserved because it is an abstraction and thus it's plans are a superset of those of Θ^{λ}. □

Experiments

We perform A^* heuristic search on the reformulated task, so the heuristic is computed on a factored task. Abstraction heuristics for factored planning tasks have been studied by Büchner et al. (2024). We use the Merge-and-Shrink heuristic (Helmert et al. 2014), as it is easy to adapt to our setting, specifically the DFP-B-50K configuration described in Sievers, Wehrle, and Helmert (2014).

We implemented our approach on top of Fast Downward (Helmert 2006). We ran experiments with Lab (Seipp et al. 2017) on AMD EPYC 7551 CPUs with memory/time cutoffs of 10 GB and 30 minutes. We use the Autoscale benchmark set (Torralba, Seipp, and Sievers 2021), which consists of 42 domains with 30 tasks each. Additionally, we include the *mars-rover-dynamite* domain from our running example, with 30 tasks that scale the number of rocks and dynamites. The *mars-rover-dynamite* domain is designed to have disjoint solution strategies (clear rocks vs. use dynamite), which is exactly the kind of structure our method exploits. All tasks are automatically transformed into FTS tasks (Helmert 2009; Sievers and Helmert 2021). Code and experiment data will be made available upon publication.

We structure this section into research questions: *RQ0:* What kinds of reformulations do we achieve? *RQ1:* How does the limit k on the size of the T sets affect performance? *RQ2:* How do the different approaches to preserving label-dominance affect performance? *RQ3:* How does shrinking the reformulated factors affect performance? *RQ4:* Does the qualified dominance reformulation improve state-of-the-art techniques?

RQ0: To understand the reformulations, we show three metrics of the reformulated factors in Figure 3: number of goal states, states, and irrelevant labels. The number of transitions is almost identical to the number of states. Irrelevant labels are labels that have only self-loops in all states, i.e., they have no precondition or effect on this factor. For goal states, in the benchmark used in each factor, either all states are goal states or exactly one state is a goal state. Therefore, if everything is a goal state, the number of goal states can only decrease. This often happens, which is good because it allows the heuristic to reason about these new goals. For states, we see that the size of the factors sometimes explodes, which is evidence that limiting k is desirable. Additionally, the number of irrelevant labels decreases, meaning the algorithm discovers previously unknown relevance of these labels. This is interesting, as it discovers hidden interactions between factors.

RQ1: In Figure 4 (left), we see the number of expansions when varying the maximum size of the T sets. There is only a small difference between limiting the size. We choose $k = 3$ as our default setting.

RQ2: In Table 1, we compare the different approaches to preserving label-dominance. All are worse than the baseline, but preserving reduced label-dominance and preserving nothing have the most cases of beating the baseline. Since preserving reduced label-dominance has better runtime performance, we choose this as our default approach. There are many instances where preserving full label-dominance is better; however, we found these are instances where the

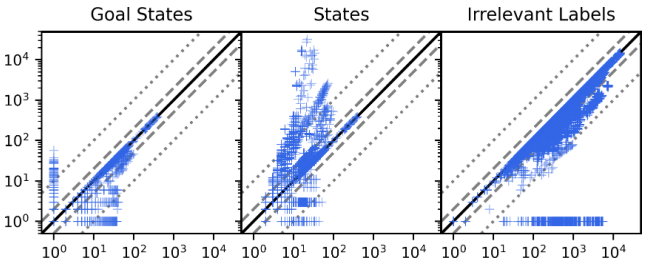

Figure 3: Scatter plots showing, on the x-axis, the number before reformulation and, on the y-axis, the number after reformulation for goal states, states, and irrelevant labels for all factors in all instances, when not limiting the size of T.

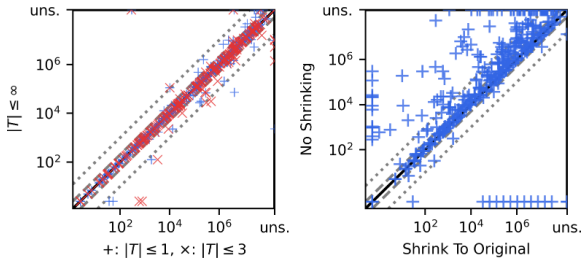

Figure 4: Number of **expansions** during search with MAS-DFP. On the left, we compare limiting the size of the T sets: on the y-axis, no limit; on the x-axis, $+$ indicates limiting to 1, and $\times$ limiting to 3. On the right, we compare shrinking the reformulated factors back to the original states.

Heuristic	Blind				MAS-DFP			
Preserve	BL	None	LD	RLD	BL	None	LD	RLD
BL	0	**84**	**62**	**69**	0	**74**	**67**	**70**
None	22	0	5	0	27	0	11	0
LD	20	**25**	0	**10**	23	**14**	0	10
RLD	22	**15**	5	0	27	4	**11**	0

Table 1: Number of instances that *row* solved and *column* did not. Entry (row,col) is bold if it is larger than (col,row). BL is the baseline, without qualified dominance. None preserves nothing, LD preserves full label-dominance, and RLD preserves reduced label-dominance.

Heuristic	Blind				MAS-DFP			
	BL	QD	QD-SO	STP	BL	QD	QD-SO	STP
BL	0	**69**	**32**	27	0	**70**	21	10
QD	22	0	0	0	27	0	8	19
QD-SO	30	**45**	0	3	**30**	**60**	0	13
STP	**30**	**50**	**8**	0	**23**	**75**	**17**	0

Table 2: Number of instances that *row* solved and *column* did not. Entry (row,col) is bold if it is larger than (col,row). BL is the baseline, i.e., without reformulation. QD is qualified dominance with $k = 3$; QD-SO is the same but also shrinks to original states; STP is our version of subsumed transition pruning.

baseline is already best, and since the transformation is weaker, it performs better.

RQ3: In *RQ0*, we saw that the reformulation often creates many new states. This can increase the size of the state space, meaning a smaller fraction of the states will be duplicates. This weakens the duplicate detection mechanism during search, leading to more expansions. To mitigate this, we try shrinking the reformulated factors back to the original states by shrinking all states (s, T) and (s, T') into one state s, ensuring that the transformation is a simplification, i.e., it has no more states or transitions. In Figure 4 (right), we see the number of expansions when shrinking to original states compared to not shrinking. We see that shrinking to original states is usually best, but similarly to before, this is often because the transformation is undesirable in these problems. We also tried shrinking bisimilar states, but there was only very minor improvement over no shrinking, indicating that the transformation does not create many bisimilar states.

RQ4: In Table 2, we compare our approach to no transformation and to subsumed transition pruning (Torralba and Kissmann 2015). We reimplement the subsumed transition pruning method in our framework by setting $k = 0$ and pruning transitions $(s, \emptyset) \xrightarrow{\ell} (s', T')$ whenever there exists $t \in T'$ such that $s' \sqsubseteq t$. This differs slightly from the original method because they handle maintaining valid dominance relations differently. We observe that, when using the MAS-DFP heuristic, qualified dominance without shrinking beats the baseline in 27 instances and STP in 19 instances. However, the advantage is largely due to our own domain, *mars-rover-dynamite*, in 18 instances for both. For a more detailed table, refer to Table 3 in the appendix. This shows that there are cases where qualified dominance is beneficial beyond subsumed transition pruning, but overall STP is best.

Overall, we find that qualified dominance reformulations can be very beneficial, but these cases are rare in standard benchmarks. A key problem is that it increases the number of states, sometimes by a lot, and this weakens the duplicate detection mechanism during search, wherefore we see more expansions compared to the baseline during blind search.

Our technique works well in *mars-rover-dynamite* because of the two different choices for solving the problem: either collect all rocks without using dynamite or use dynamite to blow up rocks. Since they are disjoint, label-dominance is effective in realizing that they should not be mixed. In other domains, this is less clear, and the label-dominance technique is not able to perform this reasoning.

Conclusion

In this paper, we introduced a novel task reformulation technique based on qualified dominance. By extending the notion of qualified dominance, we enabled the systematic removal of parts of the task that can be proven to always yield dominated plans. Our experimental evaluation shows that this reformulation can significantly reduce the search effort, even if such cases are rare in standard benchmarks. We believe that further improvements to the inference technique will broaden the applicability of these reformulations.

References

Büchner, C.; Ferber, P.; Seipp, J.; and Helmert, M. 2024. Abstraction Heuristics for Factored Tasks. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 40–49. AAAI Press.

Haslum, P. 2007. Reducing Accidental Complexity in Planning Problems. In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 1898–1903.

Haslum, P.; Helmert, M.; and Jonssson, A. 2013. Safe, Strong and Tractable Relevance Analysis for Planning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 317–321. 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.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM*, 61(3): 16:1–63.

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E.; and Thatcher, J. W., eds., *Complexity of Computer Computations*, 85–103. Plenum Press.

Milner, R. 1971. An Algebraic Definition of Simulation Between Programs. In Cooper, D. C., ed., *Proceedings of the 2nd International Joint Conference on Artificial Intelligence (IJCAI 1971)*, 481–489. William Kaufmann.

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

Sievers, S.; and Helmert, M. 2021. Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems. *Journal of Artificial Intelligence Research*, 71: 781–883.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Tollund, R. G.; Larsen, K. G.; and Torralba, Á. 2025. What Makes You Special? Contrastive Heuristics Based on Qualified Dominance. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, 8652–8660. ijcai.org.

Torralba, Á. 2021. On the Optimal Efficiency of A* with Dominance Pruning. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, 12007–12014. AAAI Press.

Torralba, Á.; and Hoffmann, J. 2015. Simulation-Based Admissible Dominance Pruning. In Yang, Q.; and Wooldridge, M., eds., *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 1689–1695. AAAI Press.

Torralba, Á.; and Kissmann, P. 2015. Focusing on What Really Matters: Irrelevance Pruning in Merge-and-Shrink. In Lelis, L.; and Stern, R., eds., *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, 122–130. AAAI Press.

Torralba, Á.; Seipp, J.; and Sievers, S. 2021. Automatic Instance Generation for Classical Planning. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 376–384. AAAI Press.

Torralba, Á.; and Sievers, S. 2019. Merge-and-Shrink Task Reformulation for Classical Planning. In Kraus, S., ed., *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5644–5652. IJCAI.

Tožička, J.; Jakubův, J.; Svatoš, M.; and Komenda, A. 2016. Recursive Polynomial Reductions for Classical Planning. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 317–325. AAAI Press.

Proofs

Lemma 1. *Let Θ be an LTS and $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$. For all paths $(s,T) \overset{\rho}{\rightsquigarrow}_{\Theta^\lambda} (s',T')$, there exists $s \overset{\rho_1}{\rightsquigarrow}_\Theta s'$, s.t. $L(\rho) = L(\rho_1)$ and for all $t' \in T'$ there is a path ρ_2 s.t. $L(\rho_1) \preceq_* L(\rho_2)$ and either (I) $\exists t \in T : t \overset{\rho_2}{\rightsquigarrow}_\Theta t'$, or (II) $s \overset{\rho_2}{\rightsquigarrow}_\Theta t' \wedge \rho_1 \lhd_* \rho_2$.*

Proof. We prove this by induction of the length of the path $|\rho| = n$. *Base case ($n = 0$)*: $\rho = \rho_1 = \rho_2 = \epsilon$; trivially satisfying (II) for all $t' \in T'$. *Inductive case*: Let $|\rho| = n+1$ and $\rho = \rho'(s_n, T_n) \xrightarrow{\ell_n}_{\Theta^\lambda} (s',T')$. By Definition 6, we know that there is $s_n \xrightarrow{\ell_n}_\Theta s'$, and thus $\rho_1 = \rho_1' s_\eta \xrightarrow{\ell_n} s'$. By the induction hypothesis there exists ρ_1' s.t. $s \overset{\rho_1'}{\rightsquigarrow} s_n$ and for all $t_n \in T_n$ exists ρ_2' s.t. $L(\rho_1') \preceq L(\rho_2')$ and satisfying either (I) or (II). By Definition 6, $t' \in T'$ implies either **(previous)** $\exists t_n \in T_n : \exists t_n \xrightarrow{\ell'_n} t' : \ell_n \preceq \ell'_n$ or **(sibling)** $\exists s_n \xrightarrow{\ell'_n} t' : \ell_n \preceq \ell'_n \wedge (s_n \xrightarrow{\ell_n} t') \lhd (s_n \xrightarrow{\ell'_n} t')$. If **(previous)** holds, then $\rho_2 = \rho_2' t_n \xrightarrow{\ell'_n} t'$ and since $\ell_n \preceq \ell'_n$ we have $L(\rho_1) \preceq L(\rho_2)$. If **(sibling)** holds, then $\rho_2 = \rho_1' s_n \xrightarrow{\ell'_n} t'$ and since $\ell_n \preceq \ell'_n$ and $(s_n \xrightarrow{\ell_n} s') \lhd (s_n \xrightarrow{\ell'_n} t')$ then $L(\rho_1) \preceq L(\rho_2)$ and $\rho_1 \lhd_* \rho_2$. □

Lemma 7. *Let Θ be an LTS and $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta)$. For all paths $\rho = s_1 \xrightarrow{\ell_1} s_2 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_{n-1}} s_n$ of Θ, there exists a path $\rho' = (s_1,T_1) \xrightarrow{\ell_1} (s_2,T_2) \cdots \xrightarrow{\ell_{n-1}} (s_n,T_n)$ for all $T_1 \subseteq 2^S$.*

Proof. This follows directly from Definition 6, since we have for every transition $s \xrightarrow{\ell}_\Theta s'$ a transition $(s,T) \xrightarrow{\ell}_{\Theta^\lambda} (s',T')$. □

Lemma 8. *Let $\mathcal{T} = (\Theta_1,\ldots,\Theta_n)$, $1 \le i \le n$, $\preceq = \preceq_{\bar{i}} \cap \preceq_c$, $s \overset{\rho}{\rightsquigarrow}_{\Theta_\mathcal{T}} g$, and $s_i \overset{\rho'}{\rightsquigarrow}_{\Theta_i} t_i$ where $L(\rho) \preceq_* L(\rho')$. There exists $s \overset{\rho''}{\rightsquigarrow}_{\Theta_\mathcal{T}} u$ s.t. $t_i \sqsubseteq u_i$ and $\forall j \neq i : g_j \sqsubseteq u_j$.*

Proof. To clarify, $s,g,t,u \in S$ and $s = (s_1,\ldots,s_n)$, $t = (t_1,\ldots,t_n)$ and so on. We consider the product of the factors other than i, $\Theta_j = \bigotimes_{h \neq i} \Theta_h$. It has been shown in (Torralba and Hoffmann 2015) that the dominance relation also composes into $\sqsubseteq_j$. We prove by induction on $k = |\rho|$, i.e. the length of ρ. *Base case ($k = 0$):* The only option is $\rho = \rho' = \rho'' = \varepsilon$ and thus $g = t = u$ and since $\sqsubseteq$ is reflexive, this satisfies the statement. *Inductive case:* Let $\rho = s \overset{\rho_k}{\rightsquigarrow} g' \xrightarrow{\ell_{k+1}} g$ and $\rho' = s_i \overset{\rho'_k}{\rightsquigarrow} t'_i \xrightarrow{\ell'_{k+1}} t_i$, since $|\rho_k| = k$ by the induction hypothesis we have $s \overset{\rho''_k}{\rightsquigarrow} u'$ s.t. $g'_i \sqsubseteq t'_i$ and $g'_j \sqsubseteq u'_j$. Now, since $\ell_{k+1} \preceq_{\bar{i}} \ell'_{k+1}$ we know that $g'_j \xrightarrow{\ell'_{k+1}} g''_j$ and $g_j \sqsubseteq g''_j$. Since $g'_j \sqsubseteq u'_j$ there must exist $u'_j \xrightarrow{\ell''_{k+1}} u_j$ s.t. $g_j \sqsubseteq g''_j \sqsubseteq u_j$ and $\ell'_{k+1} \preceq_{\bar{j}} \ell''_{k+1}$, which means ℓ''_{k+1} dominates ℓ'_{k+1} in Θ_i. Thus, we have a transition $t'_i \xrightarrow{\ell''_{k+1}} u_i$ s.t. $t_i \sqsubseteq u_i$. Therefore, $\rho'' = s \overset{\rho''_k}{\rightsquigarrow} u' \xrightarrow{\ell''_{k+1}} u$ satisfies the statement. □

Theorem 4. *Let $\mathcal{T} = (\Theta_1,\ldots,\Theta_n)$ be a planning task, $\mathcal{T}' = (\Theta_1,\ldots,\Theta_{i-1},\lambda_{\preceq,\lhd,\alpha}(\Theta_i),\Theta_{i+1},\ldots,\Theta_n)$, and $\preceq = \preceq_{\bar{i}} \cap \preceq_c$. Then, $P(\mathcal{T}') \subseteq P(\mathcal{T})$ and $\mathcal{T}'$ preserves a shortest optimal plan of $\mathcal{T}$.*

Proof. $P(\mathcal{T}') \subseteq P(\mathcal{T})$ follows from Lemma 2, since the plans of $\mathcal{T}'$ is the intersection of the plans of each factor.

Let $\Theta^\lambda = \lambda_{\preceq,\lhd,\alpha}(\Theta_i)$. Let π be the plan among the strong optimal plans of $\mathcal{T}$ which has the maximum path ρ_i in Θ_i w.r.t. $\lhd_*$ s.t. $s_i \overset{\rho_i}{\rightsquigarrow}_{\Theta_i} g_i$ with $s_i \in S_i^I$ and $s_g \in S_i^G$. We prove by contradiction: assume $\pi \notin P(\mathcal{T}')$. By Lemma 7 there is a path $(s_i,\emptyset) \overset{\rho'_i}{\rightsquigarrow}_{\Theta_i^\lambda} (g_i,T_i)$ and, since π is not a plan of $\mathcal{T}'$ and the other factors are unaltered, (g_i,T_i) is not a goal state. Therefore, there exists $t_i \in T_i$ s.t. $t_i \in S_i^G$. Now, by Lemma 1, there is a path $s_i \overset{\rho''_i}{\rightsquigarrow}_{\Theta_i} t_i$ s.t. $\rho_i \lhd_* \rho''_i$ and $\pi \preceq_{\bar{i}} L(\rho''_i)$. Let $\pi' = L(\rho''_i)$. From $t \in S_i^G$ we know $\pi' \in P(\Theta_i)$ and since $\preceq \subseteq \preceq_c$ we have $c(L(\rho'_i)) \le c(\pi)$. Thus, π' is just as good as π and is a plan of Θ_i.

We now show that there exists π'' that is a plan of $\mathcal{T}'$ and $\pi' \preceq \pi''$. Since $\pi \in P(\mathcal{T})$, there is a path $s \overset{\rho}{\rightsquigarrow} g$, with $s \in S^I, g \in S^G$. By Lemma 8, there exists $s \overset{\rho'}{\rightsquigarrow} u$ s.t. $g_i \sqsubseteq t_i$ and $\forall j \neq i : g_j \sqsubseteq u_j$. Because $t_i \in S_i^G$ and $\forall j \neq i : g_j \in S_j^G$, the dominance of $t_i \sqsubseteq u_i$ and $\forall j \neq i : g_j \sqsubseteq u_j$ ensures that $u \in S^G$. Thus, $\pi'' = L(\rho')$ is a plan of $\mathcal{T}$.

Finally, we show that we can construct ρ' s.t. it has a maximum path in Θ_i. ρ_i and ρ''_i have a (possibly empty) common prefix. Consider the first index k s.t. $\rho_i[k] \neq \rho''_i[k]$ (by $\rho[k]$ we denote the k'th transition of ρ). Let $\rho_i[k] = q_k \xrightarrow{\ell_k} r_k$ and $\rho''_i[k] = q_k \xrightarrow{\ell'_k} r'_k$. They must have the same source. We know that $\ell_k \preceq \ell'_k$, thus we can choose π'' s.t. $\pi''[k] = \ell'_k$ and that there is a path $s_i \overset{\rho'''_i}{\rightsquigarrow} u_i$ and $L(\rho'''_i) = \pi''$. By Definition 6, $(q_k \xrightarrow{\ell_k} r_k) \lhd (q_k \xrightarrow{\ell'_k} r'_k)$ and thus $\rho_i \lhd_* \rho'''_i$, which contradicts that π was the strong optimal plan with the maximum path ρ_i. □