What Makes You Special? Contrastive Heuristics Based on Qualified Dominance

Rasmus G. Tollund , **Kim G. Larsen** , **Álvaro Torralba**
Aalborg University, Aalborg, Denmark
rgt@cs.aau.dk, kgl@cs.aau.dk, alto@cs.aau.dk

Abstract

In cost-optimal planning, dominance pruning methods discard states during the search that are dominated by others. However, the binary nature of pruning fails to exploit information when we cannot prove that a state is fully dominated. To this end, we introduce *qualified dominance*, an automatic method that given a pair of states s, t synthesizes a finite state automaton that represents a language of plans from s that are dominated by t. This not only explains why s cannot be pruned, but can also be used to improve the heuristic function to guide the search. This results in a new type of heuristic, which we call *contrastive heuristics*, that are dependent on the search performed so far. We provide the theoretical foundation for showing that such contrastive heuristics can be used to find optimal plans even when their more informative estimates are not admissible.

1 Introduction

The most common approach for cost-optimal planning is to use A* [Hart *et al.*, 1968] with a heuristic function that estimates the goal distance from each state. To ensure optimality the heuristic should be admissible, meaning that the real distance is never overestimated. However, the search effort can grow exponentially with almost perfect heuristics, even for problems solvable in polynomial time [Pearl, 1984; Helmert and Röger, 2008]. This naturally raises the question of how to extend the notion of heuristics to gain more expressive power.

We introduce *contrastive heuristics*, which estimate goal distance by considering previously explored nodes. Similar ideas appear in novelty heuristics [Katz *et al.*, 2017; Lipovetzky and Geffner, 2017; Groß *et al.*, 2020; Singh *et al.*, 2021; Rosa and Lipovetzky, 2024], which use state statistics to diversify search but lack formalization of the search and lose optimality guarantees. In [Karpas and Domshlak, 2012] they propose an inadmissible heuristic that retains optimality by considering alternative paths. Contrastive heuristics provide a formal framework for search-state-dependent heuristics that retain optimality while relaxing admissibility. We prove that A* with contrastive heuristics remains optimal if for each inadmissible node there is a better node that is admissible. Thus, goal distance can be evaluated *in contrast* to other nodes, i.e. through paths only available to this node.

We also show how to automatically construct contrastively admissible heuristics for any planning task. To this end, we extend dominance pruning, which prunes states s shown to be at least as far from the goal as some t [Torralba and Hoffmann, 2015]. But full dominance is often hard to guarantee. Thus, our second contribution is *qualified dominance*. When t does not dominate s, we ask: "Why? What can s do that t cannot?" We characterize this as a set of plans from s for which no equivalent from t could be proven. We show how to automatically synthesize a finite-state automaton whose language captures plans from s that are dominated by t.

Finally, we show that qualified dominance information can be encoded in the description of the planning task, so any domain-independent admissible heuristic that can estimate the goal-distance for any planning task can be used to compute contrastive heuristic estimates. We provide a first implementation on top of the operator-counting framework [Pommerening *et al.*, 2014]. Our experiments show that our qualified dominance techniques are able to find information across many tasks, even though this is not very complementary with highly informative heuristics.

2 Background

A labelled transition system (LTS) is a tuple $\Theta = (S, L, T, S^I, S^G)$, where S is a set of states, L is a set of labels with cost $c(\ell) \in \mathbb{R}^+$, [1], $T \subseteq S \times L \times S$ 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 a set of initial states simplifies the notation [Büchner *et al.*, 2024]. We write $s \xrightarrow{\ell}_\Theta s'$ whenever $(s, \ell, s') \in T$, or simply $s \xrightarrow{\ell} s'$ if the Θ is clear from context.

A path from s_0 to s_n in Θ is a sequence of transitions $s_0 \xrightarrow{\ell_1} s_1 \dots \xrightarrow{\ell_n} s_n$. L^* is the set of finite sequences of labels and ε is the empty sequence. A plan $\pi \in L^*$ for a state $s \in S$ is a sequence of labels s.t. there exists a path from s to $s_n \in S^G$. 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, or by $P_\Theta(s)$

[1] We assume non-zero cost labels to simplify the notation. But we can support 0-cost labels by treating them as having ϵ-cost.

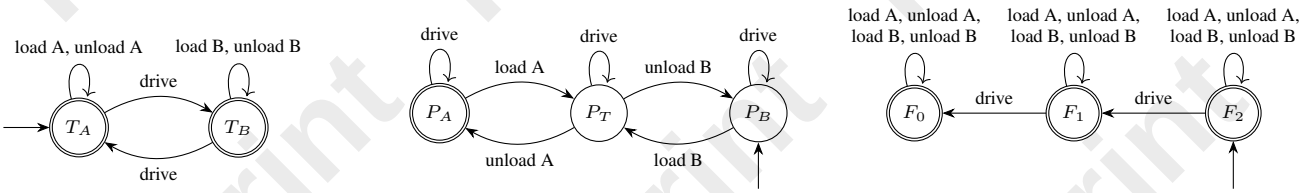

Figure 1: Planning task of our running example with three factors, representing the position of the truck, the package, and the amount of fuel.

for a particular LTS 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 $s_0 \in S^I$ with cost $h^*(S^I) = \min_{s_0 \in S^I} h^*(s_0)$.

We consider planning tasks in factored transition system (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, instead of a set, to refer to the individual factors by their index. We denote by $\circ$ tuple concatenation. States in the individual factors $s_i \in S_i$ are called *facts*. A state in $\mathcal{T}$ is a tuple of facts $s = (s_1, \dots, s_n)$, 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 $S_\mathcal{T} = S_1 \times \dots \times S_n$ is the Cartesian product of each factor's states, $T_\mathcal{T} = \{((s_1, \dots, s_n), \ell, (t_1, \dots, t_n) \mid (s_i, \ell, t_i) \in T_i \text{ for } 1 \leq i \leq n\}$, $S_\mathcal{T}^I = S_1^I \times \dots \times S_n^I$, 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 facts from each factor Θ_i (e.g., s_i, s_i', t_i) so that $s_i = s[i]$.

An LTS is *deterministic* if $(s, \ell, s'), (s, \ell, s'') \in T$ implies $s' = s''$, and $|S^I| = 1$. An FTS task $\mathcal{T}$ is deterministic if all of its factors are deterministic, which implies that the state space $\Theta_\mathcal{T}$ is also deterministic. Tasks in other common planning formalisms such as STRIPS [Fikes and Nilsson, 1971] and SAS$^+$ [Bäckström and Nebel, 1995] can be compiled into a deterministic FTS task [Sievers and Helmert, 2021].

A heuristic is a function $h_\mathcal{T} : S \to \mathbb{R}_0^+$ that estimates the cost of the cheapest path from a state to a goal state in a planning task $\mathcal{T}$. The planning task is omitted when it is clear from context. A heuristic is *admissible* if it never overestimates the cost, i.e. $h(s) \leq h^*(s)$ for all $s \in S$.

The most common algorithm for optimal planning is A* [Hart *et al.*, 1968]. A search node n_s represents a path from the initial state to the state s, and $g(n_s)$ denotes the cost of the path. We denote by $\mathcal{N}$ the set of all search nodes for a state space. A* maintains a closed and an open list of all the search nodes seen so far. The open list is initialized with a node for each $s_0 \in S^I$, representing the empty path from s_0. A node is expanded by removing it from the open list, adding it to the closed list, and adding all of its successors to the open list. A* iteratively expands a node with minimal f-value until a goal state is reached, where $f(n_s) = g(n_s) + h(s)$. When h is admissible, A* is guaranteed to find the optimal plan. We also define $f^*(n_s) = g(n_s) + h^*(s)$.

A dominance relation is a relation $\sqsubseteq \subseteq S \times S$, s.t. $s \sqsubseteq t \implies h^*(t) \leq h^*(s)$ [Torralba and Hoffmann, 2015]. Intuitively, $s \sqsubseteq t$ means that t is at least as close to the goal as s. In dominance pruning, a node n_s can be pruned whenever there exists a node n_t in the open or closed list s.t. $s \sqsubseteq t$ and $g(n_t) \leq g(n_s)$, thus guaranteeing that there is a plan through n_t that is no more expensive than any plan through n_s. A* with dominance pruning (A$^*_{pr}$) always finds an optimal plan.

To find dominance relations, we draw on the concept of simulation relations [Milner, 1971]. Specifically, a relation $\sqsubseteq$ is a *cost-simulation* for Θ if $s \sqsubseteq t$ implies that (i) $t \in S^G \vee s \notin S^G$ and (ii) $\forall s \xrightarrow{\ell} s' \,.\, \exists t \xrightarrow{\ell'} t' \,.\, c(\ell') \leq c(\ell) \wedge s' \sqsubseteq t'$. 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 to reply to any transition $s \xrightarrow{\ell} s'$ with $t \xrightarrow{noop} t$ whenever $s' \sqsubseteq t$. Any cost-simulation on $\Theta_\mathcal{T}$ is a dominance relation. This 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 fact-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 fact is as good as another. We also construct a label-relation $\preceq_{\mathrm{i}} = \{(\ell, \ell') \mid \forall s_i \xrightarrow{\ell}_{\Theta_i} s_i' \,.\, \exists s_i \xrightarrow{\ell'}_{\Theta_i} t_i' \,.\, s_i' \sqsubseteq_i t_i' \wedge c(\ell') \leq c(\ell)\}$ expressing that, on factor Θ_i, any time that ℓ can be applied, one can apply ℓ' instead. 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 tuple $(\sqsubseteq_1, \dots, \sqsubseteq_n)$ of relations for each factor such that whenever $s_i \sqsubseteq_i t_i$ then $t_i \in S_i^G \vee s_i \neg \in S_i^G$ 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 very recursive definition, the coarsest label-dominance simulation $(\sqsubseteq_1^{\mathrm{LD}}, \dots, \sqsubseteq_n^{\mathrm{LD}})$ 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 $s \sqsubseteq^{\mathrm{LD}} t$ iff $s_i \sqsubseteq_i^{\mathrm{LD}} t_i$ for all $i \in \{1, \dots, n\}$.

Figure 1 shows the planning task of our running example, where a truck with limited amount of fuel delivers a package. It consists of 3 factors, representing the location of the truck, the package, and the amount of fuel. The initial state is (T_A, P_B, F_2) and the goal states are of the form $(\cdot, P_A, \cdot)$. Here, label-dominance simulation discovers that having more fuel is always better ($F_0 \sqsubseteq F_1 \sqsubseteq F_2$). We also consider scaled versions of the task, e.g., with more fuel available.

3 Contrastive Heuristics

We generalize the notion of heuristics to contrastive heuristics, which evaluate states in the context of the search tree explored so far. Unlike traditional heuristics that estimate the cost to a goal independently for each state, a contrastive

heuristic considers the relationship between states. In our running example (with many more than three fuel levels), suppose we have two states, $s = (T_A, P_B, F_{70})$ and $t = (T_A, P_B, F_{50})$, which are identical except that s has more fuel than t. If t has an optimal plan with cost 4, then this plan is also optimal for s, so the optimal heuristic value for both, when evaluated independently, is 4. However, when evaluating s in contrast to t, the heuristic value for s should be larger. This is because any plan for s that uses no more fuel than the plan for t could also be applied to t. If a drive action that consumes 1 fuel has cost 1, then the heuristic value of s in contrast to t should be at least 51. Any plan for s that uses less than 51 drive actions would also be valid for t. Thus, in this sense, the contrastive heuristic for s relative to t exceeds the independent heuristic value. Importantly, this heuristic is inadmissible since t has a plan, s has the same plan and thus the same goal distance. However, as we will show, this inadmissibility does not compromise optimality.

In this section, we formally define contrastive heuristics and demonstrate how they can be used in A* search to obtain optimal solutions.

Definition 1. *A* contrastive heuristic *is a function* $\eta : \mathcal{N} \times 2^{\mathcal{N}} \to \mathbb{R}_0^+$*, s.t.* $\eta(n, N)$ *defines the heuristic value of node* n *in contrast to the nodes in* N*.*

In contrast to a normal heuristic, a contrastive heuristic is defined for search nodes, and not states. This allows for even more information to be considered for heuristic evaluation, as the path to the node can also be taken into account, i.e. when considering $N = \emptyset$ it reduces to path-dependent heuristics [Bagchi and Mahanti, 1983; Richter *et al.*, 2008; Karpas and Domshlak, 2009]. We call N the contrastive set.

Usually, admissibility of a heuristic is used to guarantee optimality of the search algorithm. However, in order to ensure the optimality of A*, it suffices that the heuristic always assigns an admissible value to at least one optimal node in the *open* list. We will consider a stricter requirement that is easier to reason about.

Definition 2. *A contrastive heuristic* η *is* contrastively admissible *if for all* n_s*,* $\eta(n_s, N) \leq h^*(s)$ *or there exists* $n_t \in N$ *s.t.* $\eta(n_t, N) \leq h^*(t) \wedge f^*(n_t) \leq f^*(n_s) \wedge (f^*(n_t) < f^*(n_s) \vee h^*(t) \leq h^*(s))$*.*

Intuitively, a contrastively admissible heuristic guarantees that a node is assigned an inadmissible value only if there is a better node with an admissible value estimate. The contrapositive is that if there is no better node, then the node has an admissible value. Here, *better* means it has a lower f^*-value, or it has the same f^*-value but it is closer to the goal. The latter condition ensures for an optimal node in the *open* list, any better node must be in the *open* list as well.

Lemma 1. *Let* η *be a contrastively admissible heuristic, and* $N \subseteq \mathcal{N}$ *a non-empty set of nodes,* $N_{f^*} = \arg\min_{n \in N} f^*(n)$ *and* $N^* = \arg\min_{n_s \in N_{f^*}} h^*(s)$*. There exists a node* $n_s \in N^*$ *s.t.* $\eta(n_s, N) \leq h^*(s)$*.*

Proof. Let $n_s \in N^*$. From Definition 2, if $\eta(n_s, N) > h^*(s)$, then there exists another node n_t in N, that by construction is in N^*, such that $\eta(n_t, N) \leq h^*(t)$. □

In order to utilize contrastive heuristics in A* search, we make a small modification to the search algorithm. Instead of expanding a node with minimum f-value, we expand the node with minimal $\hat{f}$-value, given by $\hat{f}_N(n) = g(n) + \eta(n, N)$. We denote by $\hat{f}_{oc}(n) = g(n) + \eta(n, \textit{open} \cup \textit{closed})$ the $\hat{f}$-value that uses the union of the *open* and *closed* list as the contrastive set. We denote this version of A* as A$^*_{oc}$.

Theorem 2. *Let* η *be a contrastively admissible contrastive heuristic.* A$^*_{oc}$ *guarantees an optimal solution.*

Proof. The proof is a modification of the proof of the optimality of A* from [Pearl, 1984, p. 78]. The optimal solution cost is $f^*(n_{s_0})$. We first show that there is always a node $n_t \in$ *open* s.t. $\hat{f}_{oc}(n_t) \leq f^*(n_t) = f^*(n_{s_0})$. By Lemma 1, there is a node n_t with minimum f^*-value and minimum h^*-value that is admissible. This node must be in *open*, otherwise its children would be in *open* and have smaller h^*-value. Now, suppose A$^*_{oc}$ expands a goal node n_s. We know that it's $\hat{f}_{oc}$-value is minimal among all nodes in *open*. Therefore, $\hat{f}_{oc}(n_s) \leq f^*(n_t)$, and n_s is optimal. □

The theorem above even holds when using a subset of *open*∪*closed*, because for a contrastive heuristic, if there is no better node in the contrastive set, then it must be admissible.

In practice, requiring the $\hat{f}$-value of each node in the *open* list to be dependent on the current *open* and *closed* list is infeasible, as it requires recomputing the heuristic value for all nodes in the open list whenever a node is expanded. Instead, we evaluate each node upon generation, and keep the value fixed afterwards, resulting in different contrastive sets for each node. The algorithm is still optimal, as long as the heuristic cannot decrease when evaluated in a larger context.

Definition 3. *A contrastive heuristic* η *is* monotonic *if for all* $n \in \mathcal{N}$ *and* $N, N' \subseteq \mathcal{N}$ $N \subseteq N' \implies \eta(n, N) \leq \eta(n, N')$*.*

The monotonicity property of the heuristic ensures that the $\hat{f}$-value of a node will not decrease when increasing the contrastive set. This restriction is quite sensible, as it means that the heuristic value does not decrease when considering more information.

Theorem 3. *Let* η *be a contrastively admissible monotonic heuristic and* $\nu : \mathcal{N} \times 2^{\mathcal{N}} \to 2^{\mathcal{N}}$ *be any mapping s.t.* $\nu(n, N) \subseteq N$*.* A* *with* $\hat{f}_\nu(n) = g(n) + \eta(n, \nu(n, \textit{open} \cup \textit{closed}))$ *guarantees an optimal solution.*

Proof. From the monotonicity property, $\eta(n, \nu(n, \textit{open} \cup \textit{closed})) \leq \eta(n, \textit{open} \cup \textit{closed})$ and thus $\hat{f}_\nu(n) \leq \hat{f}_{oc}(n)$. Thus, we can consider $\hat{f}_\nu$ to be an under-approximation of $\hat{f}_{oc}$. With this, the same proof as of Theorem 2 is applicable. □

4 Qualified Dominance

A dominance relation provides information about whether one state is at least as good as another. However, previous work only considers when a state is completely dominated and can be entirely removed from the search. Let us revisit the running example. The reason we could assign the state

$s = (T_A, P_B, F_{70})$ a higher heuristic value in contrast to $t = (T_A, P_B, F_{50})$, is that we knew that if s does not use the additional fuel, t is just as good as s. This is a form of partial-dominance relation, where t dominates all plans of s that do not use more fuel than t has. In this section, we extend the notion of dominance to include some middle ground: the label sequences under which a state is dominated by another state. In this sense, we cannot completely prune the state, but we can disconsider these label sequences that can be simulated by the other state when computing heuristics.

Definition 4. *A* qualified dominance function *is a function* $\mathcal{D} : S \times S \to 2^{L^*}$ *from a pair of states to a language over the labels, s.t.* $\forall \pi \in \mathcal{D}(s,t) \,.\, \pi \in P(s) \implies h^*(t) \leq c(\pi)$.

A qualified dominance function describes a subset of s-plans where t is guaranteed to have a no more expensive plan. Additionally, any label sequence that is not a plan of s is trivially dominated by t. Observe that a qualified dominance function does not need to be maximal, so the set of dominated plans can be arbitrarily underapproximated.

Qualified dominance generalizes the previous notion of a dominance relation, where t dominates s only if $h^*(t) \leq h^*(s)$. For any dominance relation $\sqsubseteq$, we can construct a qualified dominance function $\mathcal{D}_{\sqsubseteq}(s,t) = L^*$ if $s \sqsubseteq t$ and $\mathcal{D}_{\sqsubseteq}(s,t) = \emptyset$ otherwise. $\mathcal{D}_{\sqsubseteq}$ and $\sqsubseteq$ are equivalent as $\mathcal{D}(s,t) = L^*$ means that all s-plans are dominated by t. But qualified dominance can also express that certain paths are dominated even when a state is not fully dominated. In our running example, we can assign $\mathcal{D}((T_A, P_B, F_{70}), (T_A, P_B, F_{50})) = \{\, \pi \in L^* \mid \pi \text{ has less than 51 drive actions} \,\}$.

4.1 Inference of Qualified Dominance

Next, we consider how to represent and compute a qualified dominance function. We begin by describing how it can be defined over the entire state space of the planning task—though this offers little practical value, as it is as hard as solving the task itself. We then show how it can be approximated by computing it over the factored task instead.

Definition 5. *A function* $\mathcal{D} : S \times S \to 2^{L^*}$ *is a* qualified $\preceq$-simulation *for* Θ *if for all* $s, t \in S$ *(i)* $\varepsilon \in \mathcal{D}(s,t)$ *only if* $t \in S^G \vee s \notin S^G$ *and (ii)* $\ell_1 \ell_2 \dots \ell_n \in \mathcal{D}(s,t)$ *only if*

$$\forall s \xrightarrow{\ell_1} s' \,.\, \exists t \xrightarrow{\ell_1'} t' \,.\, \ell_1 \preceq \ell_1' \wedge \ell_2 \dots \ell_n \in \mathcal{D}(s',t')$$

Intuitively, a $\preceq$-simulation allows only to simulate a label with another label according to the label relation $\preceq$. An important special case is $\ell \preceq_c \ell'$ iff $c(\ell') \leq c(\ell)$ which we call a *qualified cost-simulation*.

Theorem 4. *A qualified cost-simulation function is a qualified dominance function.*

Proof. Let $\mathcal{D}$ be a cost-simulation function and $\pi \in \mathcal{D}(s,t)$. We show that either $\pi \notin P(s)$ or $h^*(t) \leq c(\pi)$ by induction on the length of π. *Base case* ($|\pi| = 0$)*:* from (i) of Definition 5, if s is not a goal, then $\pi = \varepsilon \notin P(s)$. If s and t are goals, then $h^*(t) = 0 = c(\pi)$. *Induction step:* Assume that the induction hypothesis holds for all $|\pi'| = n$. Let $\pi = \ell_1 \ell_2 \dots \ell_{n+1}$. By (ii) of Definition 5, if $s \xrightarrow{\ell_1} s'$ then there is $t \xrightarrow{\ell_1'} t'$ s.t. $c(\ell_1') \leq c(\ell_1)$ and $\ell_2 \dots \ell_{n+1} \in \mathcal{D}(s',t')$. By the induction hypothesis, $h^*(t') \leq c(\ell_2 \dots \ell_{n+1})$, and thus $h^*(t) \leq c(\ell_1') + h^*(t') \leq c(\ell_1 \ell_2 \dots \ell_{n+1})$. □

A qualified cost-simulation is a type of qualified dominance function. It enforces a simpler version of qualified dominance, where the plan of t must step-wise simulate the plan of s, possibly using noop. Note that, whenever ℓ_1 is not applicable on s, the for all expression is trivially true. Otherwise, t must have a transition with some label ℓ_1' that is as good as ℓ_1 in terms of cost. We now show how to encapsulate a qualified simulation for an LTS Θ in a domination LTS Γ.

Definition 6 (Domination LTS)**.** *Let* $\Theta = (S, L, T, S^I, S^G)$ *be a deterministic LTS, and* $\preceq \subseteq L \times L$ *be a label relation. The domination LTS w.r.t.* Θ *and* $\preceq$ *is* $\Gamma = \big((S \times S) \cup \{\top\}, L, T_\Gamma, \emptyset, S^G_\Gamma\big)$*, where* $S^G_\Gamma = \{\top\} \cup \{\, (s,t) \in S \times S \mid t \in S^G \vee s \notin S^G \,\}$*, and* $(s,t) \xrightarrow{\ell}_\Gamma (s',t')$ *iff* $s \xrightarrow{\ell}_\Theta s' \wedge t \xrightarrow{\ell'}_\Theta t' \wedge \ell' \preceq \ell$*;* $(s,t) \xrightarrow{\ell}_\Gamma \top$ *iff* $s \not\xrightarrow{\ell}_\Theta$*; and* $\top \xrightarrow{\ell}_\Gamma \top$ *for all* $\ell \in L$.

Each state in Γ will represent a pair of states in the original LTS s.t. $P_\Gamma((s,t)) = \mathcal{D}(s,t)$. Additionally, we introduce the universal goal state, $\top$, which is used to represent that t dominates s if s performs an inapplicable action. We do not define initial states as we will only be concerned with goal distances. The transition relation simply encodes the properties of Definition 5, adding non-determinism to handle the existential quantification of t. We avoid having to support non-determinism for the universal quantification of s by assuming the input LTS to be deterministic, so the choices of s can be encoded in the labels of Γ. Finally, the goal states are those states where $\mathcal{D}(s,t)$ contains the empty sequence, i.e. $\top$ and when goal status of t is as good as s.

Γ can be interpreted as a finite-state automata that recognizes the language $\mathcal{D}(s,t)$. Therefore, it is clear that any dominance function represented as a domination LTS assigns a regular language to each pair of states. It is worth noting that, when minimizing these automata, $\top$ and all states of the form (s,s) will be reduced to a single state that represents L^*.

Theorem 5. *Let* Γ *be a domination LTS w.r.t. a deterministic LTS* Θ *and* $\preceq_c$*, where* $\ell \preceq_c \ell'$ *iff* $c(\ell') \leq c(\ell)$*. The function* $\mathcal{D}_\Gamma(s,t) = P_\Gamma((s,t))$ *is a qualified cost-simulation for* Θ.

Proof. Let $\pi \in P_\Gamma((s,t))$. We prove this by induction on the length of π. *Base case* ($|\pi| = 0$)*:* This implies that $(s,t) \in S^G_\Gamma$, which by definition of the goal states, means that $t \in S^G_\Theta \vee s \notin S^G_\Theta$. *Induction step:* Let $\ell_1 \ell_2 \dots \ell_{n+1} \in P_\Gamma((s,t))$. Then, there is some transition from (s,t) with label ℓ_1 either to $\top$ or to some (s',t') s.t. $\ell_2 \dots \ell_{n+1} \in P_\Gamma((s',t'))$. In the former case, this implies $s \xrightarrow{\ell_y}_\Theta$, and thus fulfils condition (ii) in Definition 5 vacuously. In the latter case, by construction of Γ there are transitions $s \xrightarrow{\ell_1}_\Theta s'$ and $t \xrightarrow{\ell_1'}_\Theta t'$ where $c(\ell_1') \leq c(\ell_1)$. By the assumption that Θ is deterministic, we know that there is at most one transition of s with label ℓ_1, thus there is a response from t for all transitions of s. □

4.2 Qualified Fact-Dominance

Constructing the domination LTS for the entire state space is senseless. However, we can utilize the label-dominance simulation [Torralba and Hoffmann, 2015] concept and compute a qualified dominance function by constructing it per factor.

We first use label-dominance simulation to compute a fact-dominance relation for each factor ($\sqsubseteq_1^{\text{LD}}, \dots, \sqsubseteq_n^{\text{LD}}$). Recall that the fact-dominance relation is defined as $s_i \sqsubseteq_i^{\text{LD}} t_i$ only if $\forall s_i \xrightarrow{\ell} s_i' \,.\, \exists t_i \xrightarrow{\ell'} t_i' \,.\, s_i' \sqsubseteq^{\text{LD}} t_i' \wedge \forall j \neq i \,.\, \ell \preceq_j^{\text{LD}} \ell'$ and the label dominance relation as $\ell \preceq_i^{\text{LD}} \ell'$ only if $c(\ell') \leq c(\ell)$ and for all $s_i \xrightarrow{\ell}_{\Theta_i} s_i'$ there exists $s_i \xrightarrow{\ell'}_{\Theta_i} s_i''$ s.t. $s_i' \sqsubseteq_i^{\text{LD}} s_i''$. We then construct for each factor Θ_i the domination LTS Γ_i w.r.t. Θ_i and the label relation $\preceq_{\bar{i}} = \bigcap_{j \neq i} \preceq_j^{\text{LD}}$. Intuitively, we do the same construction as previously, but with the restriction that we only allow substituting a label ℓ for ℓ' if it is at least as cheap and whenever ℓ is applicable in all other factors, ℓ' is also applicable and leads to a state that is at least as good.

The intersection of the qualified cost-simulations in each factor is not necessarily a qualified cost-simulation for the state space. This is because the plans from t_i that dominate s_i in each factor may be disjoint, i.e. there are plans for each factor but no shared plan of all factors. However, we can still extract information when t dominates s in all but one factor.

Lemma 6. *Let $\mathcal{T} = (\Theta_1, \dots, \Theta_n)$ be a planning task and $\mathcal{D}_i$ be qualified $\preceq_{\bar{i}}$-simulations for Θ_i, respectively. Then,*

$$\mathcal{D}_{1,2}(s,t) = \begin{cases} \mathcal{D}_1(s[1],t[1]) & \text{if } \mathcal{D}_2(s[2],t[2]) = L^* \\ \mathcal{D}_2(s[2],t[2]) & \text{if } \mathcal{D}_1(s[1],t[1]) = L^* \\ \emptyset & \text{otherwise} \end{cases}$$

is a qualified $\preceq_{\overline{1,2}}$-simulation for $\Theta_{1,2} = \Theta_1 \otimes \Theta_2$ where $\preceq_{\overline{1,2}} = \preceq_{\bar{1}} \cup \preceq_{\bar{2}}$.

Proof. When $\mathcal{D}_{1,2}(s,t) = \emptyset$, there is nothing to prove. Assume W.L.O.G. that $\mathcal{D}_2(s,t) = L^*$. Let $\pi = \ell_1 \ell_2 \dots \ell_n \in \mathcal{D}_{1,2}(s,t)$. We prove this by induction on the length of π. *Base case ($|\pi| = 0$):* By (i) of Def. 5, $t[1] \in S_1^G \vee s[1] \notin S_1^G$ and $t[2] \in S_2^G \vee s[2] \notin S_2^G$, and this implies that $t \in S^G \vee s \notin S^G$. *Inductive step:* We show that $\pi \in \mathcal{D}_1(s[1],t[1]))$ implies that π is in a qualified cost-simulation of s,t. If $s[1] \not\xrightarrow{\ell_1}$ or $s[2] \not\xrightarrow{\ell_1}$ then $s \not\xrightarrow{\ell_1}$ and thus (ii) of Def. 5 holds vacuously. Otherwise, there exists $s[1] \xrightarrow{\ell_1} s[1]', t[1] \xrightarrow{\ell_1'} t[1]'$, and since $\mathcal{D}_1$ satisfies $\preceq_{\bar{1}}$, there also exists $s[2] \xrightarrow{\ell_1} s[2]', s[2] \xrightarrow{\ell_1'} s[2]''$ such that $\mathcal{D}_2(s[2]', s[2]'') = L^*$. Since $\mathcal{D}_2(s[2],t[2]) = L^*$, there also exists $t[2] \xrightarrow{\ell_1''} t[2]'$ s.t. $\mathcal{D}_2(s[2]', t[2]') = L^*$. From label dominance, we again have that $t[1] \xrightarrow{\ell_1''} t[1]''$ s.t. $\mathcal{D}_1(t[1]', t[1]'') = L*$. Now observe that if $\mathcal{D}_{1,2}(b,c) = L^*$ then $\mathcal{D}_{1,2}(a,b) \subseteq \mathcal{D}_{1,2}(a,c)$ is valid. Therefore, for $s \xrightarrow{\ell_1} (s[1]', s[2'])$ then there exists $t \xrightarrow{\ell_1''} (t[1]'', t[2]')$ s.t. $\ell_2 \dots \ell_n \in \mathcal{D}_{1,2}((s[1]', s[2]'), (t[1]'', t[2]'))$. Thus, this is a $\preceq_{\overline{1,2}}$-simulation because $\ell_1 \preceq_{\bar{2}} \ell_1''$. □

Theorem 7. *Let $\mathcal{T} = (\Theta_1, \dots, \Theta_n)$ be a deterministic planning task. Let $\mathcal{D}_i$ be a qualified $\preceq_{\bar{i}}$-simulation for each factor Θ_i. Then, the function $\mathcal{D}(s,t) = \{ \pi \in \bigcap_i \mathcal{D}_i(s_i,t_i) \mid \exists 1 \leq i \leq n \,.\, \forall j \neq i \,.\, \mathcal{D}_j(s_j,t_j) = L^* \}$ is a qualified cost-simulation for $\Theta_{\mathcal{T}}$.*

Proof. We prove by induction on the planning task size. *Base case ($n = 1$):* Follows from $\preceq_{\bar{1}} \subseteq \preceq_c$. *Inductive step:* By Lemma 6 we construct $\mathcal{T}' = (\Theta_{1,2}, \Theta_3, \dots, \Theta_n)$ and $\mathcal{D}_{1,2}$. Since $\mathcal{D}_{1,2}$ is a $\preceq_{\overline{1,2}}$-simulation for $\Theta_{1,2}$, $\preceq_{\overline{1,2}} \subseteq \bigcap_{j \notin \{1,2\}} \preceq_j^{LD}$, and $|\mathcal{T}'| = n - 1$ we can apply the induction hypothesis. □

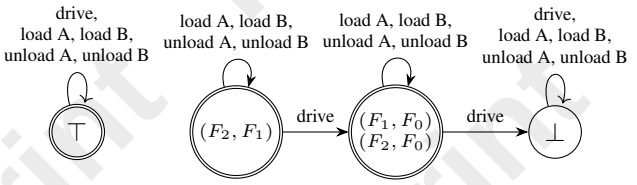

Figure 2: Minimized domination LTSs for the fuel factor in our running example. $\top$ also represents all states (F_i, F_j) where $i \leq j$.

Figure 2 shows the minimized domination LTSs for the fuel factor. The language accepted by starting in a state (s_i, t_i) corresponds to the plans in the domination LTS of that factor. Many cases are fully dominated (e.g. $s_i \sqsubseteq_i^{\text{LD}} t_i$), and thus are associated with the $\top$ state. The rest are named as the state pairs it is associated with. In the figure, we can see the pattern that was explained before, any state is dominated by a state with less fuel, unless it drives more times than the amount of fuel. For instance, F_2 is dominated by F_1, unless it does two drive actions. This shows us that the technique discovers the information of the motivating example, as this generalises to more fuel. In particular, the problem might be solvable with one fuel, but the contrastive heuristic will discover that if you have the same state but with one fuel, then the heuristic can be raised to the cost of two drive actions.

5 Qualified Dominance Heuristics

We now consider how to leverage qualified dominance to construct contrastively admissible heuristics. First, we construct a contrastive heuristic based on a node ordering and the perfect heuristic, h^*. Then, we provide a framework for using any domain-independent heuristic for planning tasks.

5.1 Combining Qualified Dominance Information

Intuitively, we use qualified dominance to exclude plans when estimating the goal-distance of a state. E.g., when computing the heuristic of n_s in contrast to $n_t \in N$, we want to estimate the goal distance of s excluding the paths in $\mathcal{D}(s,t)$.

The first consideration is which nodes from N we can compare to while remaining contrastively admissible. Consider search nodes n_s, n_t and a path $\pi \in \mathcal{D}(s,t)$. We can only disconsider π if we can guarantee that a plan of n_t produces a no worse overall plan, i.e. $g(n_s) + c(\pi) \geq g(n_t) + h^*(t)$. Since $c(\pi) \geq h^*(t)$ this simplifies to $g(n_s) \geq g(n_t)$. Taking cost difference into account requires additional analysis. Thus, we use qualified dominance information for a node when considering in contrast to nodes with lower or equal g-value.

We must also avoid circular reasoning. For example, consider search nodes n_s and n_t with equal g-value, let $\pi \in \mathcal{D}(s,t)$ and $\pi' \in \mathcal{D}(t,s)$. Crucially, we cannot disconsider both; it may be that π and π' are the only optimal plans of s and t, respectively. Therefore, we impose an ordering on the search nodes. Whenever, $g(n_s) \neq g(n_t)$ the ordering is easy because the node with larger g-value depends on the one with lower g-value. When $g(n_s) = g(n_t)$, we assume some arbitrary ordering. In practice, we use the order in which the states are evaluated.

Let $\prec$ be any strict total order on $\mathcal{N}$ satisfying $g(n) < g(n') \implies n \prec n'$ and $n \prec n' \implies g(n) \leq g(n')$, i.e. $\prec$ respects the g-value ordering and totally orders the nodes. The set of dominated plans for a search node n_s in contrast to N given $\mathcal{D}$ and $\prec$ is

$$D_{\prec,\mathcal{D}}(n_s, N) = \bigcup_{n_t \in N_{\prec n_s}} \mathcal{D}(s,t),$$

where $N_{\prec n} = \{\, n' \in N \mid n' \prec n \,\}$ is the lower set of N w.r.t. $\prec$ and n. The importance of $\prec$ is that it guarantees that some better plan is always preserved.

Lemma 8. *For all $\pi \in D_{\prec,\mathcal{D}}(n_s, N)$, there exists a node $n_t \in N_{\prec n_s}$ and a plan $\pi' \in P(t)$ s.t. $\pi' \notin D_{\prec,\mathcal{D}}(n_t, N)$ and $c(\pi') \leq c(\pi)$.*

Proof. Let n_t be the minimal node in $N_{\prec n_s}$ w.r.t. $\prec$ s.t. $h^*(t) \leq c(\pi)$. By Definition 4, such a node exists. Let $\pi' \in P(t)$ be a plan s.t. $c(\pi') = h^*(t)$. Assume that $\pi' \in D_{\prec,\mathcal{D}}(n_t, N)$. Then there exists $n_u \in N_{\prec n_t} \subseteq N_{\prec n_s}$ s.t. $\pi' \in \mathcal{D}(t,u)$. However, this means that $h^*(u) \leq c(\pi') = h^*(t)$ and $n_u \prec n_t$, which contradicts the minimality of n_t w.r.t. $\prec$. Thus, $\pi' \notin D_{\prec,\mathcal{D}}(n_t, N)$. □

We will now define the best possible heuristic for a dominance function and an ordering $\prec$.

Definition 7. *Let $\mathcal{D}$ be a qualified dominance function, $n_s \in \mathcal{N}$ and $N \subseteq \mathcal{N}$. The optimal contrastive heuristic w.r.t. $\mathcal{D}$ is*

$$\eta^*_{\prec,\mathcal{D}}(n_s, N) = \min\{c(\pi) \mid \pi \in P(s) \setminus D_{\prec,\mathcal{D}}(n_s, N)\},$$

if the minimum exists and $\eta^*_{\prec,\mathcal{D}}(n_s, N) = \infty$ otherwise.

Theorem 9. *$\eta^*_{\prec,\mathcal{D}}$ is contrastively admissible.*

Proof. Let n_u be a node in N with minimal $f^*(n_u)$ and among those minimal $h^*(u)$. Let n_s be a node s.t. n_s is better than n_u (see Def. 2). Since, $\eta^*_{\prec,\mathcal{D}}$ takes the minimum over a subset of all plans of s, $\eta^*_{\prec,\mathcal{D}}(n_s, N) \geq h^*(s)$. Assume $\eta^*_{\prec,\mathcal{D}}(n_s, N) > h^*(s)$, then there exists a plan $\pi \in D_{\prec,\mathcal{D}}(n_s, N)$ s.t. $c(\pi) = h^*(s)$. From Lemma 8, we know that there must then exist a node $n_t \prec n_s$ with a plan $\pi' \notin D_{\prec,\mathcal{D}}(n_t, N)$ with $c(\pi') \leq c(\pi)$ and thus $g(n_t) \leq g(n_s)$ and $h^*(t) \leq h^*(s)$. If $f^*(n_s) < f^*(n_u)$ then $f^*(n_t) < f^*(n_u)$, which contradicts the minimality of n_u. Otherwise, $h^*(s) < h^*(u)$ and thus $h^*(t) < h^*(u)$, contradicting the minimality of n_u. Thus, any node better than n_u is admissible, and $n_u \in N$ is admissible. Thus, $\eta^*_{\prec,\mathcal{D}}$ is contrastively admissible. □

Importantly, any under-approximation of $\eta^*_{\prec,\mathcal{D}}$ is also contrastively admissible. $\eta^*_{\prec,\mathcal{D}}$ is also monotonic, however, any under-approximation of $\eta^*_{\prec,\mathcal{D}}$ is not necessarily monotonic.

Algorithm 1: Qualified Dominance Heuristic

Input: A planning task $\mathcal{T} = (\Theta_1, \dots, \Theta_n)$ and heuristic h
Output: A contrastive heuristic function

`/* Precomputation */`
$(\sqsubseteq^{\text{LD}}_1, \dots, \sqsubseteq^{\text{LD}}_n) \leftarrow$ compute LD simulation on $\mathcal{T}$
$(\Gamma_1, \dots, \Gamma_n) \leftarrow$ compute domination LTSs w.r.t. $\sqsubseteq^{\text{LD}}$
$(\bar{\Gamma}_1, \dots, \bar{\Gamma}_n) \leftarrow$ determinize, then minimize, and then complement $\Gamma_1, \dots, \Gamma_n$
Func `Qualified-Dominance-Heuristic`(n_s, N)**:**
  $\hat{\mathcal{T}} \leftarrow (\Theta_1, \cdots, \Theta_n)$
  $\hat{s} \leftarrow s$
  forall $n_t \in N_{\prec n_s}$ ***where*** $\exists i \,.\, \forall j \neq i \,.\, s_j \sqsubseteq^{LD}_j t_j$ **do**
    $\hat{\mathcal{T}} \leftarrow \hat{\mathcal{T}} \circ \bar{\Gamma}_i$
    $\hat{s} \leftarrow \hat{s} \circ (\{(s_i, t_i)\})$
  return $h_{\hat{\mathcal{T}}}(\hat{s})$

5.2 Computation in Planning Tasks

We now show how to utilize qualified dominance LTSs to compute heuristics. With a planning task $\mathcal{T} = (\Theta_1, \dots, \Theta_n)$ and a domain-independent admissible heuristic function h, which can be applied to any planning task so that $h_{\mathcal{T}}(s)$ is an admissible estimate from $s = (s_1, \dots, s_n)$ to the goal in $\Theta_{\mathcal{T}}$. The idea is that, for each node n_s and set of nodes N, we will compute a new planning task $\hat{\mathcal{T}} = (\Theta_1, \dots, \Theta_n, \dots)$ and state $\hat{s} = (s_1, \dots, s_n, \dots)$ extending $\mathcal{T}$ and s with additional factors that encode information about the qualified dominance of n_s compared to N. Then, we can use the contrastive heuristic $\eta_{\prec,\bar{\Gamma},h}(n_s, N) = h_{\hat{\mathcal{T}}}(\hat{s})$.

Algorithm 1 shows the overall process. First, there is a preprocessing phase before the search starts, where we compute a label-dominance simulation, $(\sqsubseteq^{\text{LD}}_1, \dots, \sqsubseteq^{\text{LD}}_n)$, and a domination LTS for each factor $(\Gamma_1, \dots, \Gamma_n)$, as described in Section 4.2. In each Γ, the plans for the state (s_i, t_i) represent the plans of s_i which are dominated by plans of t_i. When computing a heuristic for n_s, we will compare it against states t where $s_j \sqsubseteq t_j$ for all $j \neq i$, i.e., states t that dominate s in all factors except Θ_i. There, we are interested in extracting information about the plans of s_i that are *not* dominated by t_i. Therefore, we need the complement of Γ_i.

Unfortunately, Γ_i may be non-deterministic. Therefore, we must do a determinization of the transition system first, and then complement the goal states. This is identical to the complementation of a non-deterministic finite automaton, which can be achieved through the power set construction and then swapping goal and non-goal states [Sipser, 2013, p. 55].

The determinization can cause an exponentially larger number of states in the LTS. This can be a bottleneck in some cases, because the heuristic is not computable in polynomial time. A solution is to determinize the LTS by underapproximating it, e.g. by only encoding one response of t. In this paper, we use the full determinization, which is still viable as this is only done once as precomputation. We leave it to future research to study practical methods of approximation.

We have shown how to construct $\bar{\Gamma}_i$ for each factor i. For a node n_s, let $n^{i_1}_{t_1} \dots n^{i_k}_{t_k}$ be a sequence of nodes where $n^{i_j}_{t_j} \in N_{\prec n_s}$ and $\forall i' \neq i_j . \mathcal{D}_{\Gamma_{i'}}(s_{i'}, t_{i'}) = L^*$. We can now take any

domain-independent heuristic h that works on factored tasks, and construct the state-contrastive heuristic

$$\eta_{\prec,\bar{\Gamma},h}(n_s, N) = h_{\hat{\mathcal{T}}}(\hat{s}),$$

where $\hat{\mathcal{T}} = \mathcal{T} \circ (\bar{\Gamma}_{i_1}, \ldots, \bar{\Gamma}_{i_k})$ and $\hat{s} = s \circ ((s[i_1], t_1[i_1]), \ldots, (s[i_k], t_k[i_k]))$. In other words, we create a copy of $\bar{\Gamma}_{i_j}$ for each node $n^{i_j}_{t_j}$, and we add it as a factor to the planning task. We then evaluate the heuristic for the state where s is concatenated with $(s[i_j], t_j[i_j])$ for each $n^{i_j}_{t_j}$.

Theorem 10. *Let $\mathcal{T}$ be a planning task, $\prec$ a g-value respecting total order of $\mathcal{N}$, and h an admissible heuristic. The heuristic $\eta_{\prec,\bar{\Gamma},h}$ is contrastively admissible.*

Proof. From Theorem 5, we know that $\mathcal{D}_\Gamma(s,t) = P_\Gamma((s,t))$ is a qualified dominance function. We prove that $\eta_{\prec,\bar{\Gamma},h}$ is contrastively admissible by showing that $\eta_{\prec,\bar{\Gamma},h}(n_s, N) \leq \eta^*_{\prec,\mathcal{D}_\Gamma}(n_s, N)$. Let $n_t \in N$ s.t. all factors other than i are dominated. An admissible heuristic means that $h_\mathcal{T}(s) \leq \min\{\, c(\pi) \mid \pi \in P_\mathcal{T}(s) \,\}$, if s has a plan. For a planning task $\mathcal{T}$ and LTS Θ, it holds that $P_{\mathcal{T}\circ\Theta}(s \circ s') = P_\mathcal{T}(s) \cap P_\Theta(s')$ for all states s of $\mathcal{T}$ and s' of Θ. Thus, $P_{\mathcal{T}\circ\bar{\Gamma}_i}(s \circ (s[i], t[i])) = P_\mathcal{T}(s) \setminus D_{\prec,\mathcal{D}_{\Gamma_i}}(s[i], t[i])$ and $h_{\mathcal{T}\circ\bar{\Gamma}_i}(s \circ (s[i], t[i])) \leq \min\{\, c(\pi) \mid \pi \in P_\mathcal{T}(s) \setminus D_{\prec,\mathcal{D}_{\Gamma_i}}(s[i], t[i]) \,\} \leq \eta^*_{\prec,\Gamma_i}(s, \{t\})$. This argument can be used inductively for each node in $N_{\prec n_s}$. □

Theorem 11. *Let h be a domain-independent heuristic and $\prec$ a g-value respecting total order of $\mathcal{N}$. If $h_\Theta(s) \leq h_{\Theta\circ\Theta'}(s \circ s')$ for all LTSs Θ, Θ' and states s, s', then $\eta_{\prec,\bar{\Gamma},h}$ is monotonic.*

Proof. Recall Def. 3 and the definition of $\eta_{\prec,\bar{\Gamma},h}$. It follows that if $h_\Theta(s) \leq h_{\Theta\circ\Theta'}(s \circ s')$, then $\eta_{\prec,\bar{\Gamma},h}$ is also monotonic, because adding more nodes to N causes more LTSs for h. □

Clearly, there are practical concerns to consider here as well. Extending the planning task with a number of factors on the order of the nodes expanded so far is too cumbersome in practice. It remains an open research question how to choose the most informative factors to add.

In principle, any domain-independent heuristic can be used with our method. However, heuristics with heavy preprocessing are undesirable, as the planning task changes for each search state. Operator-counting heuristics [Pommerening *et al.*, 2014] are well-suited for this purpose. They model operator usage as a linear program and solve for the minimum-cost assignment. Each transition system of the extended task can be encoded as flow constraints and combined with other constraints [Bonet, 2013; Pommerening *et al.*, 2013; Imai and Fukunaga, 2015] to integrate information from new factors with heuristics for the original task.

6 Experiments

We implemented our approach on top of Fast Downward [Helmert, 2006], as a constraint generation method for operator counting heuristics. We complement them with constraints from LM-cut [Helmert and Domshlak, 2009] on the original planning task. Computing the LM-cut constraints for the transformed tasks is not possible in the current implementation, so we use only flow constraints for the new factors. Our technique should be understood as a proof of concept. We do not yet have a practical way to exploit the qualified dominance information. Therefore, we disregard runtime, as that will require further research into practical exploitation methods. Our main objective is to determine whether qualified dominance can produce more useful information to guide search than dominance pruning, which can be understood as a special case of qualified dominance that prunes (assigns a value of infinity) to any state fully dominated in all factors.

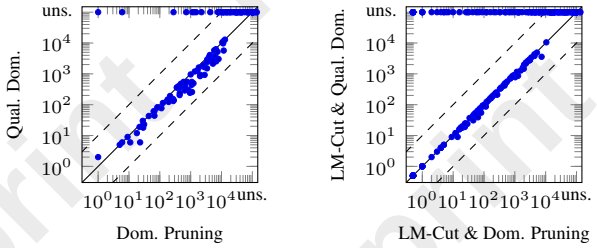

Figure 3: Expansions until last f-layer of A* w.o. heuristic (left) and LM-cut (right) using qualified dominance vs. dominance pruning.

We run experiments with Lab [Seipp *et al.*, 2017] on AMD EPYC 7551 CPUs with memory/time cut-offs of 4 GBs and 30 minutes. We use the Autoscale benchmark set [Torralba *et al.*, 2021], consisting of 42 domains with 30 tasks in each. All tasks are automatically transformed into deterministic FTS tasks [Helmert, 2009; Sievers and Helmert, 2021]. Code and experiment data will be made available upon publication.

The results show that qualified dominance is found on most tasks. The precomputation finishes in 81% of instances, and among those in 93% of cases, we find information over label-dominance simulation (i.e. having more than 3 nodes in the minimal domination LTS). In 79% of the instances where the precomputation finishes, the precomputation time is within a factor of 2 of dominance, and in >99% within a factor of 10. Fig. 3 compares the expansions compared to dominance pruning with the same heuristic. We see a reasonable decrease of expansions in many tasks (e.g., of a factor of 2). Therefore, this can potentially be used for improving heuristic estimates. The combination with LM-cut shows that, even though they are not entirely redundant, the additional constraints are not very complementary to this highly-informed heuristic. This points out the need for further research in how to extract constraints from the additional factors.

7 Conclusion

In this paper, we made two main contributions. First, we introduced contrastive heuristics, which consider the entire search to estimate goal distance from each node, in optimal planning. Second, we presented qualified dominance, a method to identify plans of a state that are dominated by another. We showed how this can be used to enhance heuristics into their contrastive counterpart. While contrastive heuristics based on qualified dominance are not yet practical, this invites further research on extracting heuristic information.

Acknowledgements

This research was partly supported by the Independent Research Fund Denmark through a Sapere Aude: DFF-Starting Grant under reference number 3120-00063B.

References

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Bagchi and Mahanti, 1983] Amitava Bagchi and Ambuj Mahanti. Search algorithms under different kinds of heuristics — a comparative study. *Journal of the ACM*, 30:1–21, 1983.

[Bonet, 2013] Blai Bonet. An admissible heuristic for SAS^+ planning obtained from the state equation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 2268–2274. AAAI Press, 2013.

[Büchner *et al.*, 2024] Clemens Büchner, Patrick Ferber, Jendrik Seipp, and Malte Helmert. Abstraction heuristics for factored tasks. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 40–49. AAAI Press, 2024.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Groß *et al.*, 2020] Joschka Groß, Álvaro Torralba, and Maximilian Fickert. Novel is not always better: On the relation between novelty and dominance pruning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*. AAAI Press, 2020.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pages 162–169. AAAI Press, 2009.

[Helmert and Röger, 2008] Malte Helmert and Gabriele Röger. How good is almost perfect? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 944–949. AAAI Press, 2008.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[Imai and Fukunaga, 2015] Tatsuya Imai and Alex Fukunaga. On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning. *Journal of Artificial Intelligence Research*, 54:631–677, 2015.

[Karpas and Domshlak, 2009] Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1728–1733. AAAI Press, 2009.

[Karpas and Domshlak, 2012] Erez Karpas and Carmel Domshlak. Optimal search with inadmissible heuristics. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 92–100. AAAI Press, 2012.

[Katz *et al.*, 2017] Michael Katz, Nir Lipovetzky, Dany Moshkovich, and Alexander Tuisov. Adapting novelty to classical planning as heuristic search. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, pages 172–180. AAAI Press, 2017.

[Lipovetzky and Geffner, 2017] Nir Lipovetzky and Hector Geffner. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3590–3596. AAAI Press, 2017.

[Milner, 1971] Robin Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence (IJCAI 1971)*, pages 481–489. William Kaufmann, 1971.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[Pommerening *et al.*, 2013] Florian Pommerening, Gabriele Röger, and Malte Helmert. Getting the most out of pattern databases for classical planning. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 2357–2364. AAAI Press, 2013.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 226–234. AAAI Press, 2014.

[Richter *et al.*, 2008] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 975–982. AAAI Press, 2008.

[Rosa and Lipovetzky, 2024] Giacomo Rosa and Nir Lipovetzky. Count-based Novelty Exploration in Classical Planning. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI 2024)*, pages 4181–4189. IOS Press, 2024.

[Seipp *et al.*, 2017] Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Malte Helmert. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

[Sievers and Helmert, 2021] Silvan Sievers and Malte Helmert. Merge-and-shrink: A compositional theory of transformations of factored transition systems. *Journal of Artificial Intelligence Research*, 71:781–883, 2021.

[Singh *et al.*, 2021] Anubhav Singh, Nir Lipovetzky, Miquel Ramirez, and Javier Segovia-Aguas. Approximate novelty search. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, pages 349–357. AAAI Press, 2021.

[Sipser, 2013] Michael. Sipser. *Introduction to the theory of computation*. Cengage Learning, 2013.

[Torralba and Hoffmann, 2015] Álvaro Torralba and Jörg Hoffmann. Simulation-based admissible dominance pruning. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1689–1695. AAAI Press, 2015.

[Torralba and Sievers, 2019] Álvaro Torralba and Silvan Sievers. Merge-and-shrink task reformulation for classical planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5644–5652. IJCAI, 2019.

[Torralba *et al.*, 2021] Álvaro Torralba, Jendrik Seipp, and Silvan Sievers. Automatic instance generation for classical planning. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, pages 376–384. AAAI Press, 2021.