Completeness-Preserving Dominance Techniques for Satisficing Planning

Submission #3

Abstract

Dominance pruning methods have recently been introduced for optimal planning. They compare states based on their goal distance to prune those that can be proven to be worse than others. In this paper, we introduce dominance techniques for satisficing planning. We extend the definition of dominance, showing that being closer to the goal is not a prerequisite for dominance in the satisficing setting. We develop a new method to automatically find dominance relations in which a state dominates another if it has achieved more serializable sub-goals. We take advantage of dominance relations in different ways; while in optimal planning their usage focused on dominance pruning and action selection, we also use it to guide enforced hill-climbing search, resulting in a complete algorithm.

Introduction

Satisficing planning is the problem of, given an input planning task, finding a sequence of actions that go from the initial state to a state that satisfies the goal condition. Most satisficing planners use search algorithms like Greedy Best-First Search (GBFS) or Enforced-Hill Climbing (EHC) guided with heuristics such as the delete-relaxation heuristic and extensions thereof (Hoffmann and Nebel 2001; Domshlak *et al.* 2015) plus certain diversification techniques (Richter *et al.* 2011; Röger and Helmert 2010) and/or sub-goal selection strategies (Chen *et al.* 2004; Porteous *et al.* 2001; Hoffmann *et al.* 2004). Both GBFS and EHC use heuristics, but they use them in different ways. In GBFS, heuristics determine the order in which states are expanded. EHC, on the other hand, uses heuristics to compare newly generated states against the initial state, restarting the search when it finds a state with lower heuristic value than the initial state. The success of EHC highly depends on the accuracy of the heuristics. When the heuristic is accurate EHC finds solutions very quickly, but it is incomplete in tasks with unrecognized dead-end states, i.e., states that the heuristic finds promising but have no solution (Hoffmann 2005). The interest on EHC has been recently renewed by heuristic refinement approaches that make it a complete algorithm (Fickert and Hoffmann 2017).

Dominance pruning techniques have recently been introduced for optimal planning (Hall *et al.* 2013; Torralba and Hoffmann 2015). They reduce the search space by pruning states that are dominated by others. The definition of dominance is based on goal distance: a state dominates another state if it can be proven to be at least as close to the goal.

In this paper we explore the use of dominance methods to compare states in satisficing search. We redefine the notion of dominance for satisficing planning, substituting the optimality guarantee by a completeness guarantee that ensures that at least one plan (not necessarily optimal) will be preserved. We also consider how dominance relations can be used to reduce the size of the search space. Like in optimal planning, one can prune states that are dominated by others, but lifting any considerations with respect to the cost of reaching such states. Also, a state s can be replaced by any of its successors s' if s' strictly dominates s. Based on this, we define a variant of EHC that is complete.

Our work builds on previous methods to automatically find dominance relations for a given planning task. We strengthen their reasoning and specialize them for satisficing search. To do this, we define a new dominance relation that serializes the planning task, inspired by sub-goal serialization approaches (Barrett and Weld 1993). Our experiments show that these serialized dominance relations are able to identify dominance in a number of domains to guide a dominance-based EHC.

Background

A *labeled transition system* (LTS) is a tuple $\Theta = \langle S, L, T, s^I, \mathcal{S}^G \rangle$ where S is a finite set of *states*, L is a finite set of *labels*, $T \subseteq S \times L \times S$ is a set of *transitions*, $s^I \in S$ is the *start state*, and $\mathcal{S}^G \subseteq S$ is the set of *goal states*. A *plan* for a state s is a path from s to any $s_G \in \mathcal{S}^G$. By $h^*(s)$ ($g^*(s)$) we denote the length of a shortest plan for s (path from s^I to s). A plan for s is *optimal* iff its cost equals $h^*(s)$. Since our goal is to find solutions fast, regardless of their cost, we assume unit-cost domains.[1]

Following previous work on dominance pruning, we consider a planning task as a set of LTSs on a common set of labels, $\{\Theta_1, \dots, \Theta_n\}$. Given a planning task in the more common SAS+ formalism (Bäckström and Nebel 1995), the atomic transition systems representation with one LTS for each SAS+ variable can be easily obtained (Helmert *et al.*

[1] We also simplify the explanation of previous work for optimal planning based on this assumption.

2014). The state space of the task is the synchronized product of all the LTSs: $\Theta = \Theta_1 \otimes \cdots \otimes \Theta_n$. The synchronized product of two LTSs $\Theta_1 \otimes \Theta_2$ is another LTS with states $S = \{(s_1, s_2) \mid s_1 \in \Theta_1 \wedge s_2 \in \Theta_2\}$, transitions $T = \{(s_1, s_2) \xrightarrow{l} (s'_1 s'_2) \mid s_1 \xrightarrow{l} s'_1 \wedge s_2 \xrightarrow{l} s'_2\}$, s.t. $(s_1, s_2) \in \mathcal{S}^G$ iff $s_1 \in \mathcal{S}_1^G$ and $s_2 \in \mathcal{S}_2^G$. We use subscripts to differentiate states in the state space Θ (s, s', t) and their projection into an Θ_i (s_i, s'_i, t_i). We say that a transition $s \rightarrow s'$ in Θ affects Θ_i if it modifies its value, $s_i \neq s'_i$.

A heuristic is a function $h : S \rightarrow \mathbb{N}$ that estimates the distance from every state to the goal. A state is *alive* iff it is solvable, reachable, and not a goal state. A heuristic h is *descending* if all alive states have a successor with lower heuristic value. A heuristic is dead-end aware if $h(s) = \infty$ for all dead-end states s. Most common search algorithms in satisficing planning (e.g., hill-climbing or GBFS) will solve the planning task with at most $h(s^I)$ expansions if h is a descending and dead-end aware heuristic (Seipp *et al.* 2016).[2]

A relation $\preceq$ is a set of pairs of states. A relation $\preceq$ is a preorder iff it is reflexive and transitive. We write $s \prec t$ as a shorthand for $s \preceq t$ and $t \not\preceq s$ (i.e., $\prec$ is a strict partial-order). Let h be a heuristic, $\preceq$ approximates h iff $s \preceq t$ implies $h(t) \leq h(s)$. A relation $\preceq$ is a dominance relation if $s \preceq t$ (t dominates s) then it must be at least as close to the goal ($h^*(t) \leq h^*(s)$). Torralba and Hoffmann (2015) introduced label-dominance simulation, a method to compute a set of relations $\{\preceq_1, \dots, \preceq_n\}$ that can be combined to derive a dominance relation $\preceq$ for Θ where $s \preceq t$ iff $s_i \preceq_i t_i$ for all i.

Quantitative dominance extends the previous method by considering numeric functions instead of relations (Torralba 2017). A function $\mathcal{D} : S \times S \rightarrow \mathbb{Q} + \{-\infty\}$ is a quantitative dominance function (QDF) if $\mathcal{D}(s, t) \leq h^*(s) - h^*(t)$. QDFs are computed as a set of functions $\{\mathcal{D}_1, \dots, \mathcal{D}_n\}$ such that $\mathcal{D}(s, t) = \sum_i \mathcal{D}_i(s_i, t_i)$. To guarantee that the sum of all $\mathcal{D}_i$ is a QDF, they must fulfill the equation: $\mathcal{D}_i(s_i, t_i) = \min_{s_i \xrightarrow{l} s'_i} \max_{u_i \xrightarrow{l'} u'_i} \mathcal{D}_i(s'_i, u'_i) - h^\tau(t_i, u_i) + \sum_{j \neq i} \mathcal{D}_j^L(l, l')$.

The equation above requires defining h^τ and $\mathcal{D}_j^L$. h^τ accounts for transitions that only affect a single LTS Θ_i. A label is a τ-label for Θ_i iff $s_j \xrightarrow{l} s_j \in \Theta_j$ $\forall \Theta_j \neq \Theta_i, s_j \in \Theta_j$. The τ-distance from s_i to t_i, written $h^\tau(s_i, t_i)$, is the cost of a minimum-cost path from s_i to t_i in Θ_i using only transitions with τ labels or ∞ if no such path exists. $\mathcal{D}_j^L(l, l')$ measures how good is to apply l' instead of l in Θ_j. If $\mathcal{D}_j^L(l, l') \geq 0$, it means that any time we can apply l to reach some s_j, we can also apply l' to reach t_j s.t. $\mathcal{D}(s_j, t_j) \geq 0$.

QDFs can be used, apart for dominance pruning, to perform action selection, a type of pruning where a state s is replaced by its immediate successor t if $\mathcal{D}(s, t) \geq c(s, t)$ where $c(s, t)$ is the cost of the transition from s to t.

[2] Seipp *et al.* (2016) consider the more general case of dead-end avoiding heuristics instead of dead-end aware heuristics.

Satisficing Dominance

In optimal planning, a dominance relation is one where for any $s \preceq t$, t should be as close to the goal as s. However, this is sometimes too restrictive for satisficing search. For example consider a problem where there are two paths to the goal, one requires solving a hard combinatorial problem and the other follows a straightforward, but potentially longer, path. Assuming that providing any guarantees about the cost of solving the combinatorial problem is hard, no dominance can be proven for optimal planning. However, it is simple to manually design a dominance relation where the states in the simpler path dominate those related to solving the combinatorial problem, directly guiding the search towards the goal. With this aim, we define a satisficing dominance relation as one that preserves solutions, no matter their cost or length.

Definition 1 (Satisficing Dominance Relation) *A preorder $\preceq$ is a satisficing dominance relation if there exists a descending and dead-end aware heuristic $h^\preceq$ such that $s \preceq t \implies h^\preceq(t) \leq h^\preceq(s)$.*

Intuitively, $h^\preceq$ should be dead-end aware so that unsolvable states do not dominate solvable states, and descending to avoid the case where a state dominates all its successors, hence rendering the search incomplete. However, simply requiring each state to not dominate one of its solvable successors is not enough to guarantee that a plan is preserved.[3]

This is a generalization of dominance relations used in optimal planning, since the perfect heuristic h^* is descending and dead-end aware. Note that any descending and dead-end aware heuristic can be defined via computing h^* after changing the cost of the transitions in Θ. Therefore, Definition 1 can also be interpreted as a dominance relation for an instance with a different cost function.

In optimal planning, dominance relations have been used in two different ways: for dominance pruning (eliminating states that are dominated by others) and action selection pruning (automatically applying an action if this action is guaranteed to start an optimal plan). Next, we adapt these types of pruning to satisficing planning. Dominance pruning can be applied in a similar way as in optimal planning, but slightly stronger since the cost of reaching each state does not matter.

Theorem 1 *Let $\preceq$ be a satisficing dominance relation. Then, greedy best-first search in which s may be pruned if there exists $t \in open \cup closed$ such that $s \preceq t$ is complete.*

Proof Sketch: We show that whenever a solvable state s is pruned, there exists another solvable state $u \in open$, with a path to the goal $\pi(u)$ s.t. for any $u' \in \pi(u)$, $u' \notin closed$ and $u' \neq s$. Let $h^\preceq$ be the dead-end aware and descending heuristic approximated by $\preceq$. Then, it suffices to show that there exists $u \in open$ such that $h^\preceq(u) \leq h^\preceq(s)$. s was pruned so there must exist $t \in open \cup closed$ such that $h^\preceq(t) \leq h^\preceq(s)$. If $t \in open$ then $u = t$. Otherwise, if $t \in closed$, we can prove the existance of u by induction on

[3] Consider a case where there are two paths to the goal $I \rightarrow s \rightarrow s' \rightarrow G$ and $I \rightarrow t \rightarrow t' \rightarrow G$. The relation $t' \preceq s$ and $s' \preceq t$ could cause all solutions to be pruned.

the value of $h^{\preceq}(s)$. Since $h^{\preceq}(u) \leq h^{\preceq}(s) < \infty$, u is solvable. As $h^{\preceq}$ is descending, the path from u does not contain any state dominated by s. □

Action selection can also be adapted for the satisficing case. In this case, we do not care about the solution cost so quantitative dominance is not required anymore. Instead, we consider strict dominance to avoid loops in which two states that dominate each other are constantly replaced by one another. We can also generalize action selection to consider not only immediate successors, but also any successor that is reached by a sequence of actions. This is far more useful in satisficing than in optimal planning because the cost of the action sequence can be ignored.

Theorem 2 *Let $\prec$ be a strict satisficing dominance relation. Let s be a state and t be the result of executing any sequence of actions in s. Then, if $s \prec t$, s can be replaced by t (skipping any other successors of s).*

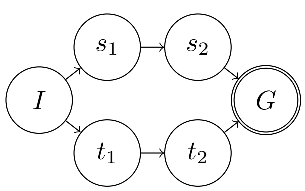
Proof Sketch: As $h^{\prec}(t) \leq h^{\prec}(s)$, and $h^{\prec}$ is descending, t must have a solution that does not traverse s, since all states t_i in the solution have $h^{\prec}(t_i) < h^{\prec}(s)$ so $t_i \neq s$ and $t_i \not\prec s$. By transitivity neither t nor any state in its solution can be substituted by s or any state s' such that $s' \prec s$. □

It should be noted that both types of pruning can be applied at the same time, but only if they use the same relation.

Theorem 3 *Let $\preceq$ be a satisficing dominance relation. Then:*

1. *Performing dominance pruning with $\preceq$ and action selection with $\prec$ is always safe.*
2. *Let $\prec'$ be a different strict satisficing dominance relation. Then, there exist cases where performing dominance pruning with $\preceq$ and action selection with $\prec'$ is not safe.*

Proof Sketch: To show (2.) consider the following example:



Let $\preceq$ be a relation such that $s_1 \preceq s_2 \preceq I \preceq t_1 \preceq t_2 \preceq G$ and $\prec'$ be a relation where $t_1 \preceq t_2 \preceq I \preceq s_1 \preceq s_2 \preceq G$. Then, by Theorem 2, I can be replaced by t_1, and then later t_2 could be pruned according to Theorem 1 because it is dominated by a previously expanded state (e.g. I).

To show (1.), if both relations $\preceq$ and $\prec$ approximate the same heuristic $h^{\preceq}$, a loop like the one in the example above cannot happen because the value of $h^{\prec}$ can only decrease along the path to the goal. Since $h^{\prec}(t_2) < h^{\prec}(t_1)$, t_2 cannot be pruned by any state that has a larger $h^{\prec}$ value (e.g any state that is replaced by t_1). □

Dominance-Based Enforced Hill-Climbing

Enforced Hill-Climbing (EHC) is a well-known search algorithm for planning (Hoffmann and Nebel 2001). EHC performs a breadth first search from the initial state s^I until finding a state s such that $h(s) < h(s^I)$. At that point, if s is a goal state a plan has been found. Otherwise, the initial state is replaced by s and the search is restarted.

According to Theorem 2, any time we find a state strictly better than s according to a satisficing dominance relation $\preceq$, we can remove s and all other of its successors from consideration. This may be expensive to do for all states in the search, but can be easily done for the special case where s is the initial state s^I. In that case, the search is restarted from the newly found state that dominates s^I. This is a form of EHC, where the search is restarted whenever a state better than s^I is found, substituting the heuristic function by a dominance relation to determine which states are better than s^I.

Also, while the original EHC algorithm used breadth-first search to escape the current plateau, there is no reason to not consider other best-first search algorithms with different priority functions as well. We define the dominance-based EHC DEHC$_{\prec}(X)$ algorithm relative to any best-first search algorithm X and strict preorder $\prec$. DEHC$_{\prec}(X)$ runs algorithm X until finding a goal state or any state s such that $s^I \prec s$. In the latter case, it restarts from s.

Theorem 4 *Let X be a sound and complete best-first search algorithm, and let $\prec$ be a strict preorder such that for any pair of reachable states s, t, if $s \prec t$ and s is solvable then t is solvable. Then DEHC$_{\prec}(X)$ is sound and complete.*

Proof: Soundness follows from soundness of X. Completeness: If the instance is solvable, each run of X can finish either finding a goal, or finding a state t such that $s^I \prec t$ and another instance of X is restarted. Then, as t must be solvable, X can never be restarted on an unsolvable state. The algorithm always terminates because $\prec$ is a strict preorder so the number of times X may be called is bounded by the number of reachable states which is always finite. □

The conditions required for $\prec$ are weaker than what is required for a satisficing dominance relation. If $\prec$ is based on a satisficing dominance relation $\preceq$, dominance pruning can be used in X. In this case, any state dominated by the initial state in each call of X can be pruned, thereby ensuring that the search does not re-expand any previous initial state. However, DEHC can also be used with relations defined from heuristic functions in the following way.

Definition 2 (Heuristic-based Relation for DEHC) *Let $\mathcal{D}$ be a quantitative dominance function and h be any heuristic. We define $\prec^h$ as a relation such that $s \prec^h t$ if and only if $\mathcal{D}(s,t) > -\infty$ and $h(t) < h(s)$.*

As $\mathcal{D}(s,t) > -\infty$ implies that $h^*(s) - h^*(t) > -\infty$ this means that if s is solvable, t must be solvable as well. This results in a complete variant of EHC with any heuristic function that uses dominance only to avoid dead-ends. The role of $\prec$ in this context is to select when to be more or less greedy following the heuristic advice, interpolating between GBFS (when no dominance is found) and EHC.

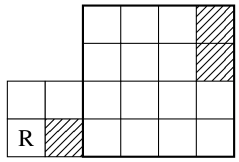


Figure 1: Example where a robot must visit all tiles in a square grid. The robot has two units of fuel which are consumed when moving into striped cells so the robot must not enter the square grid via the shortest path.

Practical Methods for Computing Satisficing Dominance Relations

In this section, we introduce a new method to compute dominance relations for satisficing planning.

Serialized Dominance Relations

Consider a planning task represented as a set of LTSs $\{\Theta_1, \ldots, \Theta_n\}$ like the example of Figure 1 with one LTS representing the position of the robot, one LTS describing the available fuel, and one LTS for each cell in the square grid that represents if the cell has been visited or not. We can obtain a QDF $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ where each $\mathcal{D}_i$ is comparing states only according to their value in Θ_i. For the position of the robot we obtain $\mathcal{D}(x, y) = -d(x, y)$ where $d(x, y)$ is the distance from cell x to cell y using only movements that do not consume fuel. For the fuel, we obtain $\mathcal{D}(s, t) = 0$ if t has at least as much fuel than s or $-\infty$ otherwise. For each cell, we have a value of $-\infty$ if the cell has been visited in s and not in t and 0 otherwise. In optimal planning, t dominates s if $\sum_i \mathcal{D}_i(s_i, t_i) \geq 0$. In our example this means that a state is better if it has visited more cells, it has at least as much fuel and the position of the robot is the same. The latter condition is an important limitation because, in order to find a state that dominates the initial state, the robot must go back to its initial position every time that it visits more cells. This is undesirable since it will be harder to find a state that dominates s^I and it will result in longer plans.

Intuitively, we prefer states where more cells have been visited, regardless of the position of the robot. This is possible because these sub-goals are serializable, i.e., no sub-goal must be undone in order to achieve the rest. To achieve this, we serialize the LTSs so that a state dominates another if it is as good for the first $j-1$ LTSs (has not un-visited any position), it is strictly better in Θ_j (has visited a new position), and there exists a solution for all other LTSs without using any label that is "dangerous" for the previous LTSs. We define a label as dangerous for an LTS Θ_i if applying it on some state s_i requires to go to a potentially worse state s'_i s.t. $s_i \not\preceq s'_i$.

Definition 3 (Dangerous label) *Let $\preceq_i$ be a relation for Θ_i. We say that a label l is dangerous for $\preceq_i$ if there exists a state $s_i \in \Theta_i$ such that there exists $s_i \xrightarrow{l} s'_i \in \Theta_i$ and there does not exist $s_i \xrightarrow{l} t_i \in \Theta_i$ s.t. $s_i \preceq_i t_i$.*

For example, labels associated with movements that consume fuel are dangerous for the LTS that represents the amount of available fuel. However, movements of the robot are not dangerous for the LTSs that correspond to whether a cell has been visited or not.

Definition 4 (Serialized Dominance) *Let $\{\preceq_1, \ldots, \preceq_n\}$ be a label-dominance simulation for $\{\Theta_1, \ldots, \Theta_n\}$ and $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ a set of functions that satisfy the equations of a QDF where $\mathcal{D}^L_j(l, l') = -\infty$ for any $l' \in L$ and $l \in L$ that is dangerous for $\preceq_i$ for any $i < j$. We define the serialized dominance relation as $s \preceq^S t$ iff $s_j \preceq_j t_j$ for all $j \in [1, n]$ or exists i such that $s_j \preceq_j t_j$ for all $j \in [1, i)$, $s_i \prec_i t_i$ and $\mathcal{D}(s_j, t_j) > -\infty$ for all $j \in (i, n]$.*

Theorem 5 *A serialized dominance $\preceq^S$ is a satisficing dominance relation.*

Proof Sketch: We show that $\preceq^S$ approximates a descending and dead-end aware heuristic function $h^{\preceq}$ ($s \preceq^S t \implies h^{\preceq}(t) \leq h^{\preceq}(s)$). As $h^{\preceq}$ must be dead-end aware, if s is a dead-end then $h^{\preceq}(s) = \infty$ so the condition holds. If s is not a dead-end then t cannot be a dead end because $s \preceq^S t$ implies $\sum \mathcal{D}_i(s_i, t_i) > -\infty$. Therefore, it suffices to consider the case where s and t are both solvable.

We define $h^{\preceq}$ as the perfect goal distance under a cost function constructed from $\preceq$ such that (i) costs of all transitions affecting Θ_i cost more than those that only affect Θ_j for $i < j$ and (ii) if $s_i \prec_i t_i$ transitions from s_i cost more than transitions from t_i. In both cases, the cost difference must be large enough so that the most expensive transition dominates the cost of the entire path.

To prove that $s \preceq^S t \implies h^{\preceq}(t) \leq h^{\preceq}(s)$, we assume WLOG that $s_1 \prec_1 t_1$.[4] Then, for any path from s_1, $\pi^s = s_1 \xrightarrow{l_1} s^1_1 \ldots \xrightarrow{l_k} s^k_1$, there exists a path from t_1, $\pi^t = t_1 \xrightarrow{l'_1} t^1_1 \ldots \xrightarrow{l'_k} t^k_1$ such that $s^i_1 \preceq t^i_1$ for all $i \in [1, k]$. Then, the cost of π^t is lower than that of π^s because the first transition is more expensive from s_1 ($s_1 \prec_1 t_1$) and the rest are not.

Since $\mathcal{D}_i(s_i, t_i) > -\infty$ for all $i \in [2, n]$, by the properties of a QDF, the path π^t can always be extended into a plan for t by inserting additional actions. As $\mathcal{D}^L_i(l, l') = -\infty$ these additional actions are not dangerous for $\preceq_1$. Since in our cost function the cost of the most expensive transition dominates the overall cost, the complete path for t is still cheaper than the one for s under this cost function so $h^{\preceq}(t) < h^{\preceq}(s)$. □

The resulting dominance relation is heavily influenced by the ordering chosen for the LTSs. To preserve completeness, this order must be the same thorough the entire search. However, one does not need to decide the order a priori, but rather it can be dynamically chosen during the search. Initially, we keep a set with all $\{\Theta_1, \ldots, \Theta_n\}$ unsorted and the list of serialized LTSs is initialized empty. When comparing a state s against s^I, we check whether $s_i \prec_i s^I_i$ for some i and insert Θ_i in the list of serialized LTSs if and only if thanks to this we get that $s^I \prec^S s$. Using this policy in our running example, the order in which the cells are serialized

[4] Let j be the smallest index for which $s_j \prec_j t_j$. If $j > 1$, we can consider instead the synchronized product $\Theta_1 \otimes \cdots \otimes \Theta_j$. By the properties of label-dominance simulation, $(s_1, \ldots, s_j) \prec_{1,\ldots,j} (t_1, \ldots, t_j)$.

in the dominance relation is exactly the order in which they are found during the search.

Recursive and Positive τ-Labels

The method above is most interesting in situations where dominance relations in optimal planning cannot prove t to be closer to the goal than s, but where it can show that t is not a dead-end, i.e., $-\infty < \mathcal{D}(s,t) < 0$. For this, τ-labels are of great importance. Having more τ-labels can only decrease the tau distance between states (h^τ), which may in turn increase the value of $\mathcal{D}$. Previous work considered l a τ-label for Θ_i if it has self-loop transitions for any state in all other Θ_j. In other words, transitions labelled with l may change the value of Θ_i in any state without affecting the value of other LTSs. Hence, all τ-labels need to fulfill two properties:

1. It does not have preconditions on other LTSs so it is always applicable, and
2. It does not have side effects on other LTSs.

Here, we extend the notion of τ-labels in two different ways, relaxing each of these assumptions in order to find coarser dominance relations.

Recursive τ-labels Some labels are not τ-labels becase they have preconditions on other LTSs. For example, in a typical logistics transportation task, loading a package at some location is not a τ-label for the position of the package because it is not applicable in all states (the truck must be at the same location). However, the truck can always be driven from any given location to the position of the package, load it, and then drive back to the original position, reaching a state where the package is in the truck without affecting any other variable. Hence, we could introduce new transitions that correspond to those macro-actions. We use this in order to redefine the set of τ-labels for each LTS.

For every s_i with a path $\pi_l^\tau(s_i, s_j) = (s_i, s_j) \xrightarrow{\tau^*} (s_i, s_j') \xrightarrow{l} (s_i', s_j') \xrightarrow{\tau^*} (s_i', s_j)$ we may introduce a new transition $s_j \xrightarrow{l} s_j$. The cost of this new transition is equal to the cost of the τ actions in $\pi_l^\tau(s_i, s_j)$. Thanks to these self-loops, l may become a τ label with a cost equal to the maximum $\pi_l^\tau(s_i, s_j)$ for any (s_i, s_j). After introducing new τ-labels, the process can be repeated.

Positive τ-labels Some labels are not τ-labels because they have side effects. In our running example movements are not τ-labels for the LTSs representing the position of the robot because they have the side-effect of marking a cell as visited. However, these side-effects are always positive according to our dominance relation so they can be ignored for the computation of τ labels. A label is a positive τ-label for Θ_i iff $\forall \Theta_j \neq \Theta_i, s_j \in \Theta_j$, there exists $s_j \xrightarrow{l} t_j \in \Theta_j$ s.t. $\mathcal{D}_j(s_j, t_j) \geq 0$.

When using this definition one must be careful, due to the circular dependency between the values of $\mathcal{D}$ and the set of τ-labels. $\mathcal{D}$ is typically computed by assuming a very coarse dominance relation and then iteratively refining it until a fixpoint is reached. However, the values of $\mathcal{D}$ during this computation have not been proven correct until it ends, so positive τ-labels cannot be defined in terms of the $\mathcal{D}$ that is being computed. Hence, to avoid such circular dependencies we first compute $\mathcal{D}$ based on the previous notion of τ-labels, and then compute a new set of τ-labels and re-compute $\mathcal{D}$. This process can be repeated until no more labels are added to the set of τ labels.

Experiments

We run experiments on all satisficing-track STRIPS planning instances from the international planning competitions (IPC'98 – IPC'14). All experiments were conducted on a cluster of Intel Xeon E5-2650v3 machines with time (memory) cut-offs of 30 minutes (4 GB). Our goal is to evaluate the potential of current dominance techniques to enhance search in satisficing planning. As a simple baseline, we use lazy GBFS in Fast Downward (Helmert 2006) with the h^{FF} heuristic, and compare the results against dominance-based EHC guided with blind search (h^B) and the h^{FF} heuristic (Hoffmann and Nebel 2001). We also include the performance of LAMA as representative of more modern planners (Richter and Westphal 2010).

We run several configurations comparing our new serialized dominance ($\preceq^S$) against quantitative dominance relations ($\preceq^{\mathcal{D}}$) used in previous work on optimal planning (Torralba 2017). However, satisficing planning benchmarks are much larger than those for optimal planning so we change the dominance pruning setting in two important aspects. On the one hand, previous work considered using merge and shrink (Helmert *et al.* 2014) to reduce the number of LTSs. But, as the overhead is too large for these benchmarks, we consider instead only the atomic transition systems. This reduces the number of domains in which state of the art methods are effective to find dominance. On the other hand, previous work considered pruning states that are dominated by any previously expanded state. This check is too expensive in satisificing planning so instead we only compare each state against its parent and the initial state.

Table 1 shows coverage results on domains where dominance has a non-negligible effect either by restarting the search from states that dominate the initial state (top part of the table) or by pruning states (bottom part). The results show that our dominance techniques are able to find useful dominance relations in a number of IPC domains, even when only considering atomic transition systems. Compared to the baseline, the results are quite good on most domains. Results could be even better but our implementation is not able to finish the preprocessing in the largest tasks of some domains (e.g. Logistics, Rovers, Satellite, and Visitall). This explains why not all instances of Visitall are solved by DEHC(h^B) and the difference wrt. the baseline in domains where no dominance pruning occurs.

The general trend is that serialized dominance relations ($\preceq^S$) are most useful to find states better than the initial state in the context of DEHC (domains in the upper part of the table), while the dominance relation based on the distance to the goal $\preceq^{\mathcal{D}}$ is more effective for dominance pruning and

Domain	Baseline GBFS (h^{FF})	Baseline LAMA	Pruning DEHC(h^B) $\preceq^{\mathcal{D}}$	DEHC(h^B) $\preceq^S$	DEHC(h^{FF}) $\preceq^{\mathcal{D}}$	DEHC(h^{FF}) $\preceq^S$	GBFS(h^{FF}) $\preceq^{\mathcal{D}}$	GBFS(h^{FF}) $\preceq^S$
Logistics (63)	53	**63**	42	42	**57**	56	54	53
Miconic (150)	150	150	150	150	150	150	150	150
Rovers (40)	23	**40**	14	11	22	**25**	23	23
Satellite (36)	**29**	**36**	9	10	24	23	26	26
Scanalyzer (50)	44	**50**	**48**	**48**	44	44	44	44
Visitall (40)	5	**40**	3	**27**	6	**27**	4	4
Zenotravel (20)	20	20	13	13	20	20	20	20
$\sum$	324	**399**	279	301	323	**345**	321	320
Barman (40)	**31**	**39**	0	0	30	29	29	30
Floortile (40)	8	7	8	8	**13**	**13**	**13**	**13**
Mainten (20)	5	2	0	0	**6**	**6**	**6**	**6**
Nomystery (20)	9	14	**20**	9	**20**	15	**20**	15
Parking (40)	24	**40**	0	0	26	25	26	26
Pathways (30)	11	**24**	4	4	**12**	**12**	**12**	**12**
Pipes-NT (50)	**30**	**43**	14	14	29	29	29	29
Pipes-T (50)	22	**43**	10	10	22	22	**23**	**23**
$\sum$	140	**212**	56	45	**158**	151	**158**	154
Total (1636)	1216	**1495**	616	627	1212	**1226**	1208	1204

Table 1: Coverage on IPC instances. We highlight the best configuration apart from LAMA. At the top are domains containing an instance where DEHC restarts from a state that dominates the initial state at least 10 times in a single instance. At the bottom, domains where dominance pruning has a non-negligible effect on coverage.

action selection. The reason is that the serialization is global for the entire search, slightly reducing the ability of dominance pruning and action selection.

Focusing on domains where dominance is useful for DEHC, one can observe that there is no much synergy with heuristics and they can even be harmful, like in Scanalyzer and Visitall. The reason is that the heuristic is not aware of the dominance relation so it may guide the search in a direction where no states dominating the initial state can be found.

Even though the results of DEHC are quite good on these domains compared to the baseline, they are still far behind LAMA, which easily solves all instances in those domains. LAMA uses landmarks in order to achieve sub-goals very greedily so is extremely effective in domains where all sub-goals are serializable. Therefore, one may wonder whether there are cases where DEHC can beat LAMA. One example is Nomystery, where action selection pruning is specially effective. But our running example illustrates the strengths of dominance even better.

Figure 2 shows the comparison of DEHC$_{\preceq^S}$ against LAMA in our running example. The particular feature of our running example is that it combines sub-goals that are easily serializable (visiting normal cells), with others that are not (visiting stripped cells). Heuristic approaches that greedily try to maximize the number of achieved sub-goals fall into a dead-end trap and are unable to solve the task. However, $\preceq^S$ is able to identify which sub-goals are safe and which ones could potentially be dangerous. As the heuristics are not

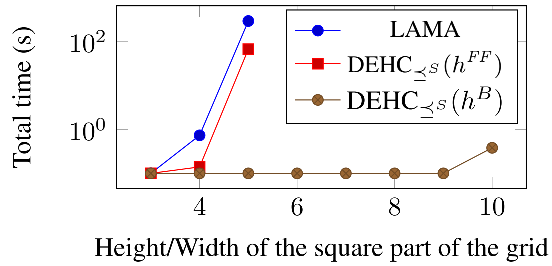


Figure 2: Total time of DEHC and LAMA in our running example.

aware of the dominance relation, this only works in combination with blind search. Otherwise the search is guided by the heuristic towards a part of the state space where no dominance can be found (correctly so, because it is a dead-end trap). DEHC$_{\preceq^D}(h^B)$ also beats LAMA in this domain, but it is still worse than DEHC$_{\preceq^S}(h^B)$ because using dominance purely based on goal distance the robot needs to go back to the initial state every time it visits a new cell, which is hard to do without any heuristic guidance towards there.

Related Work

The notion of dominance is related to approaches that characterize when a task can be solved in polynomial time. Our serialized dominance relation reminds of serializable sub-goals (Korf 1987; Barrett and Weld 1993). A set of sub-goals is serializable if they can always be achieved sequentially without undoing any of them. Our dominance relation also imposes a serialization on the LTSs that form the planning task. This is slightly different from sub-goals in that we may obtain dominance if progress has been made in an LTS (e.g. a package being in the truck is better than at the initial position) while sub-goals only consider its goal value (the package being at its destination). If a problem has several serializable sub-goals, we can always construct a dominance relation that represents this information. Up to the best of our knowledge, there are no automatic algorithms to prove that a set of sub-goals is serializable. Serialized dominance could be tailored for this purpose.

There is a long list of works that identify tractable fragments of the optimal and satisficing planning problem (Bäckström and Klein 1991; Jonsson and Bäckström 1998; Brafman and Domshlak 2003; Giménez and Jonsson 2008; Katz and Domshlak 2008; Chen and Giménez 2010). Our dominance techniques can capture some of the structure exploited by these tractable fragments, like acyclic causal graphs (using recursive τ-labels). But, at the same time, we are not limited by such features of the planning tasks (e.g. the causal graph of our running example is not acyclic). Moreover, dominance relations can be useful in tasks where planning is intractable but some part of the problem is easy to solve. In those cases, the use of dominance techniques can still dramatically reduce the search space.

There are several parametrized search algorithms that run in polynomial time in a width parameter w. These algorithms are based on a substantial amount of pruning either by only allowing to change the value of up to w variables (Chen and Giménez 2007) or pruning states with a novelty greater than w (Lipovetzky and Geffner 2012). In both cases, these algorithms do not solve the entire planning task, but are used to find a state "better" than the initial state in terms of the achieved sub-goals. Dominance relations offer an alternative way to compare states, which is more general than the criterion used by Chen and Giménez and offers completeness guarantees unlike simple sub-goal based criteria suggesting potential in combining these approaches.

Seipp *et al.* (2016) introduced another notion of width based on how hard is to represent a dead-end aware and descending potential heuristic (Pommerening *et al.* 2015). Many typical domains have a width of 2, meaning that it is easy to represent a heuristic that solves them in polynomial time. No method to automatically find such heuristic is known yet but, satisficing dominance relations approximate these kind of heuristics so any such algorithm could potentially be used to obtain dominance relations as well.

A question that naturally comes up is what is the advantage of using dominance relations over heuristic functions. Dominance relations are more expressive than heuristics because they are partial preorders while heuristics are total preorders. For example, we may have relations where $s \preceq t$ and $s' \preceq t'$, but the relation between s, t and s', t' remains unknown (e.g. $s \not\preceq t'$ and $s' \not\preceq t$). However, no assignment of heuristic values can represent this relation. In practice, this matters most in cases where dominance is able to discover some local information that can be exploited independently of the rest of the problem. Consider the Nomystery domain, where a truck transports a set of packages using a limited amount of fuel. The use of dominance allows us to identify that having a package at its destination is always good so we can unload it directly without considering any other alternative. The problem with heuristics is that they aggregate all estimations into a single value, making it very difficult to identify in which parts of the state space the heuristic is wrong. In Nomystery, most heuristics will correctly estimate that all packages need to be loaded and unloaded exactly once, but underestimate the number of truck movements. However, the search will equally explore the possibility of loading/unloading packages in different locations due to the heuristic inaccuracies.

Discussion

In this paper, we have introduced the notion of dominance for satisficing planning. Dominance can be used for dominance pruning as well as to identify states that are strictly better than others. This allows the search algorithm to be extremely greedy, restarting the search from any state that dominates the initial state, but still preserving completeness. We have adapted the algorithms to automatically find dominance relations for these purposes. Dominance can be a very powerful instrument to compare states, specially in instances with a mixture of complex and simple sub-goals.

Our experiments show the ability of dominance to guide EHC search. However, there is still a gap when compared against state-of-the-art planners. Our results also point out some limitations of current dominance techniques that may be worth exploring in future work. Considering more than one variable at a time could help to find stronger dominance relations in many domains. Also, heuristics could use the information captured by the dominance relation, increasing the synergy between them.

References

Christer Bäckström and Inger Klein. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, 7(4), November 1991.

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

Anthony Barrett and Daniel S. Weld. Characterizing subgoal interactions for planning. pages 1388–1393, 1993.

Ronen Brafman and Carmel Domshlak. Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research*, 18:315–349, 2003.

Hubie Chen and Omer Giménez. Act local, think global: Width notions for tractable planning. In Mark Boddy, Maria Fox, and Sylvie Thiebaux, editors, *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pages 73–80, Providence, Rhode Island, USA, 2007. Morgan Kaufmann.

Hubie Chen and Omer Giménez. Causal graphs and structurally restricted planning. *Journal of Computer and System Sciences*, 76(7):579–592, 2010.

Y. Chen, C. Hsu, and B. Wah. SGPlan: Subgoal partitioning and resolution in planning. In Stefan Edelkamp, Jörg Hoffmann, Michael Littman, and Håkan Younes, editors, *Proceedings of the 4th International Planning Competition*, Whistler, BC, Canada, Jun 2004. JPL.

Carmel Domshlak, Jörg Hoffmann, and Michael Katz. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114, 2015.

Maximilian Fickert and Jörg Hoffmann. Complete local search: Boosting hill-climbing through online heuristic-function refinement. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*. AAAI Press, 2017.

Omer Giménez and Anders Jonsson. The complexity of planning problems with simple causal graphs. *Journal of Artificial Intelligence Research*, 31:319–351, 2008.

David Hall, Alon Cohen, David Burkett, and Dan Klein. Faster optimal planning with partial-order pruning. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy, 2013. AAAI Press.

Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery*, 61(3), 2014.

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

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.

Peter Jonsson and Christer Bäckström. State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence*, 100(1–2):125–176, 1998.

Michael Katz and Carmel Domshlak. New islands of tractability of cost-optimal planning. *Journal of Artificial Intelligence Research*, 32:203–288, 2008.

Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33(1):65–88, 1987.

Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In Luc De Raedt, editor, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, pages 540–545, Montpellier, France, August 2012. IOS Press.

Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 3335–3341. AAAI Press, January 2015.

Julie Porteous, Laura Sebastia, and Jörg Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 37–48. Springer-Verlag, 2001.

Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.

Silvia Richter, Matthias Westphal, and Malte Helmert. LAMA 2008 and 2011 (planner abstract). pages 50–54, 2011.

Gabriele Röger and Malte Helmert. The more, the merrier: Combining heuristic estimators for satisficing planning. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 246–249. AAAI Press, 2010.

Jendrik Seipp, Florian Pommerening, Gabriele Röger, and Malte Helmert. Correlation complexity of classical planning domains. In Subbarao Kambhampati, editor, *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, pages 3242–3250. AAAI Press/IJCAI, 2016.

Álvaro Torralba and Jörg Hoffmann. Simulation-based admissible dominance pruning. In Qiang Yang, editor, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 1689–1695. AAAI Press/IJCAI, 2015.

Álvaro Torralba. From qualitative to quantitative dominance pruning for optimal planning. In Carles Sierra, editor, *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 4426–4432. AAAI Press/IJCAI, 2017.