Beyond Pruning: Leveraging Dominance Relations for Heuristic Propagation

María Fernanda Salerno Garmendia, Daniel Fišer, Álvaro Torralba

Aalborg University, Denmark
{mfsga, dafi, alto}@cs.aau.dk

Abstract

In optimal heuristic search, A* is known to be optimally efficient in terms of node expansions when the only source of information available to guide the search is a heuristic function. However, when a dominance relation is available, dominance pruning can further reduce the number of node expansions while preserving optimality. In this work, we show that pruning nodes does not fully leverage the information dominance relations provide. Instead, one can use dominance to propagate heuristic information between states. This results in a search algorithm that shares information among different regions of the search space. This is always as informative as dominance pruning, and can potentially reduce search effort.

1 Introduction

Optimal search algorithms deal with the problem of finding a minimum-cost path between a source state and a set of possible goal states. A* (Hart, Nilsson, and Raphael 1968) is widely used, as it is optimally efficient (Dechter and Pearl 1985), i.e., it expands the minimal number of nodes (up to tie-breaking) among all optimal search algorithms that use a heuristic function as a sole source of information. However, even almost perfect heuristics may incur an exponential search space (Helmert and Röger 2008), motivating search guidance methods beyond heuristics (Domshlak, Katz, and Shleyfman 2012; Torralba 2017; Wehrle et al. 2013).

We focus on *dominance relations*, which compare pairs of states to determine whether one is as close to the goal as the other (Torralba and Hoffmann 2015). So far, dominance has been used for pruning, discarding any state s that is dominated by another t (denoted as $s \preceq t$) if t was reached with a cost at most equal to the cost of reaching s. This pruning is safe because if s is part of an optimal solution then another one passes through t instead. Dominance pruning can significantly reduce the number of nodes explored by A* retaining optimality guarantees. Indeed, this is optimally efficient (under assumptions of consistency) among all search algorithms using dominance *only* for pruning (Torralba 2021).

However, this leaves an open question: Are there other ways of using dominance relations beyond pruning that can further reduce the number of node expansions This is important regardless of whether it requires additional per-node computational overhead. First, there are situations where the operation of expanding a state may be very expensive so that saving node expansions is essential and could compensate the overhead (Narayanan and Likhachev 2017; Weiss, Felner, and Kaminka 2023). Second, this paves the way for meta-reasoning algorithms to balance the per-node overhead with savings in node expansions (Barley, Franco, and Riddle 2014; Lelis et al. 2016; Karpas et al. 2018).

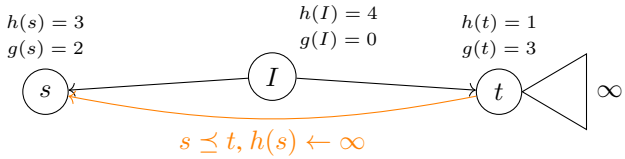

Figure 1: Propagation of dominance information.

We show that, indeed, A* with dominance pruning makes unnecessary expansions. To understand why, consider the example of Fig. 1. After expanding I, we have two open states t and s s.t. $s \preceq t$, so we know that t is as close to the goal as s. However, dominance pruning is not applicable because the cost of reaching s from I is lower than that of t. Indeed, if both states had the same cost to goal, the only optimal plan would go through s so pruning it would lose optimality. Therefore, all previous algorithms, which do not re-evaluate s after expanding t, need to expand s. Yet, there are situations where we can avoid expanding s. If no solution is found after exploring the whole subtree of t, then we can propagate this information from t to s, avoiding the need to explore s and the whole subtree underneath.

We show how to automate this reasoning during the search by propagating information across the search space, using dominance relations and Pathmax rules (Mérő 1984). The result is a dynamic heuristic that improves as the search progresses (Franco and Torralba 2019; Seipp 2021; Christen et al. 2025). We characterize the best heuristic $\mathcal{H}^{\text{fix}}$ achievable through this propagation, show it is admissible, and prove it can be used in A* without losing optimality guarantees. We also show how to compute $\mathcal{H}^{\text{fix}}$ by dealing with cases where the propagation can enter into (infinite) loops. Furthermore, we prove that using this heuristic is always as good as A* with dominance pruning in terms of expansions,

and that there are cases where it expands fewer nodes.

2 Background

A transition system is a tuple $\Theta = \langle S, L, T, s_I, S_G \rangle$ where S is a finite set of states, L is a finite set of labels each associated with a non-zero positive cost $c(l) \in \mathbb{N}^+$, $T \subseteq S \times L \times S$ is a set of transitions, $s_I \in S$ is the start state, and $S_G \subseteq S$ is the set of goal states. We write $s \xrightarrow{l} t$ as a shorthand for $(s, l, t) \in T$. A path in a transition system is a sequence of transitions $s_0 \xrightarrow{l_1} s_1 \xrightarrow{l_2} \dots \xrightarrow{l_n} s_n$ such that $s_i \xrightarrow{l_{i+1}} s_{i+1}$ for all $0 \leq i < n$. A plan for a state s is a path from s to any $s_G \in S_G$. We use $h^*(s)$ ($g^*(s)$) to denote the cost of a cheapest plan for s (path from s_I to s). A plan for s is optimal iff its cost equals $h^*(s)$. The sum $f^*(s) = g^*(s) + h^*(s)$ is the cost of the cheapest plan for s_I passing through s. F^* denotes the cost of the optimal plan for s_I, i.e., $F^* = f^*(s_I) = h^*(s_I)$.

A heuristic $h : S \to \mathbb{N}_0 \cup \{\infty\}$ is admissible if it never overestimates the goal distance, i.e., $h(s) \leq h^*(s)$ for all $s \in S$, and consistent if for any $s \xrightarrow{l} t$ it holds that $h(s) \leq h(t) + c(l)$. $\mathbb{H}$ denotes the set of all possible heuristic functions. Given $h, h' \in \mathbb{H}$, $h \leq h'$ denotes that h' is pointwise greater than h, i.e., $h(s) \leq h'(s)$ for all $s \in S$.

We consider forward search algorithms that are unidirectional, deterministic, expansion-based, and black box (UDXBB) (Dechter and Pearl 1985; Eckerle et al. 2017), so they have access to the state space only via the following small set of primitives. $Start()$ returns the initial state s_I; $IsGoal(s)$ returns true iff s is a goal state; $Succ(s)$ returns the set of successor states $\{t \mid s \xrightarrow{l} t\}$; and $c(s,t)$ returns the cost of reaching t from s, i.e., $c(s,t) = \min_{s \xrightarrow{l} t} c(l)$.

Best-first search (BFS) algorithms maintain an open and a closed list with all nodes that have been seen so far. A search node n_s characterizes a path from s_I to s of cost $g(n_s)$. We write $n_s \xrightarrow{l} n_t$ as a shorthand for $s \xrightarrow{l} t$ and $g(n_t) = g(n_s) + c(l)$. The open list is initialized with the initial state that has the g-value of 0. At each step, a node is selected from the open list for expansion. When a node is expanded, it is removed from open. Each successor of the expanded node is inserted into the open list if it has not been encountered before, or if a cheaper path to it has been found, i.e., if the newly computed g-value is lower than any previously recorded g-value for that state. The closed list keeps all nodes that have been expanded to avoid duplicates so that a node is not expanded if another node with the same state and a lower or equal g-value is already expanded.

A* is a best-first search algorithm that always selects for expansion a node with minimum f-value where $f(n_s) = g(n_s) + h(s)$. Algorithm 1 shows the pseudocode of A* with a consistent heuristic. This serves as the basis for modifications that we will introduce later. The lines in orange correspond to the propagation of heuristic values we propose and describe later. The lines in blue correspond to dominance pruning (Torralba and Hoffmann 2015) explained below.

A state s can be reached by multiple paths, producing multiple nodes n_s. A* uses duplicate detection to avoid unnecessarily re-expanding states. When a newly generated node n_t is about to be inserted into the open list, A* checks whether a node with the same state t has already been expanded or is already in open. If a node with state t is found in the closed list with a g-value no greater than that of n_t, then n_t is discarded; otherwise, if the closed node has a higher g-value, the state is reopened and n_t is inserted into the open list. If a node with state t is in the open list with a higher g-value, it is replaced by n_t, since n_t provides a cheaper path to the same state. This mechanism allows A* to effectively handle multiple parents of the same state by always preserving the lowest-cost path discovered so far.

Dominance pruning methods reduce search effort by skipping a node if another can be proven to lead to an at least as good solution. To that end, these methods leverage a dominance relation $\preceq \subseteq S \times S$ such that if $s \preceq t$ then $h^*(t) \leq h^*(s)$. As with heuristics, there are methods to derive dominance relations automatically (Torralba and Hoffmann 2015). However, we do not make any assumptions on how the dominance relation is obtained, and it could also be defined manually with expert knowledge. Using dominance relations for pruning in A* is straightforward (see blue lines in Algorithm 1). Anytime a node n_t is generated, we check for another node n_u in open or closed such that $t \preceq u$ and $g(n_u) \leq g(n_t)$. If this is the case, n_t is pruned, i.e., it is removed from open without being expanded or closed.

Algorithm 1: A* with pruning and propagation

Require: UDXBB task
Require: Admissible heuristic h
Require: Dominance relation $\preceq$ ▷ for pruning & propagation

$s^I = Start()$
$open, closed \leftarrow \emptyset$
Insert s^I in $open$ with $g(s^I) = 0$ and $f(s^I) = h(s^I)$
$\mathcal{S}_{\text{gen}} \leftarrow \{s^I\}, \mathcal{S}_{\text{exp}} \leftarrow \emptyset$, $parents \leftarrow \{s^I \mapsto \{\}\}$
$\mathcal{H}^{\text{fix}} \leftarrow \{s^I \mapsto h(s^I)\}$
while $open \neq \emptyset$ **do**
  $n_s \leftarrow \arg\min_{n_s \in open} f(n_s)$
  $open \leftarrow open \setminus \{n_s\}$
  if $f(n_s) - g(n_s) < \mathcal{H}^{\text{fix}}(s)$ **then**
    Reinsert n_s in $open$ with $f(n_s) = g(n_s) + \mathcal{H}^{\text{fix}}(s)$
    continue
  $closed \leftarrow closed \cup \{n_s\}$
  if $IsGoal(s)$ **then**
    return n_s
  $\mathcal{S}_{\text{exp}} \leftarrow \mathcal{S}_{\text{exp}} \cup \{s\}$
  $\mathcal{S}_{\text{gen}} \leftarrow \mathcal{S}_{\text{gen}} \cup Succ(s)$
  $parents(t) \leftarrow parents(t) \cup \{s\}$ for $t \in Succ(s)$
  $\mathcal{H}^{\text{fix}} \leftarrow \texttt{Propagation}(s, \mathcal{S}_{\text{gen}}, \mathcal{S}_{\text{exp}}, parents, \mathcal{H}^{\text{fix}}, h)$
  for $t \in Succ(s)$ **do**
    $n_t \leftarrow$ new node with parent n_s and state t
    $g(n_t) \leftarrow g(n_s) + c(s,t)$
    if $\exists n_{t'} \in open \cup closed$ s.t. $t = t'$ **then**
      if $g(n_{t'}) > g(n_t)$ **then**
        Replace n'_t with n_t and re-insert in $open$
    else
      if $\exists_{n_u \in open \cup closed}\ t \preceq u$ and $g(n_u) \leq g(n_t)$ **then**
        continue
      $f(n_t) \leftarrow g(n_t) + h(t)$ / $f(n_t) \leftarrow g(n_t) + \mathcal{H}^{\text{fix}}(t)$
      Insert n_t in $open$ with $f(n_t)$ and $g(n_t)$
return NO SOLUTION

3 Heuristic Values Propagation

In this section, we show a dominance relation $\preceq$ can be used to improve the accuracy of admissible heuristics h. The idea is quite straightforward. Since $s \preceq t$ implies $h^*(t) \leq h^*(s)$, it follows that whenever $s \preceq t$ and $h(s) \leq h(t)$, we can assign $h(t)$ to s. This will improve the heuristic value of s without compromising the admissibility of the heuristic. This can be applied for any pair of states in the state space. However, in UDXBB algorithms, we make no assumptions on the structure of the state space, so states are only accessible once they are generated (using $Start$ and $Succ$). We formalize this reasoning in terms of propagation rules that dynamically improve the heuristic during the search.

3.1 A* with Dynamic Heuristics

We denote the heuristic resulting from the propagation process as $\mathcal{H}^{\text{fix}}$. This corresponds to the heuristic computed by the propagation algorithm explained later in Section 3.5. $\mathcal{H}^{\text{fix}}$ is a function of the set of expanded states $\mathcal{S}_{\text{exp}}$, and generated states $\mathcal{S}_{\text{gen}}$, which are uniquely defined as $\mathcal{S}_{\text{gen}} = \{s_I\} \cup \bigcup_{s \in \mathcal{S}_{\text{exp}}} Succ(s)$ given $\mathcal{S}_{\text{exp}}$. We write $\mathcal{H}^{\text{fix}}(s, \mathcal{S}_{\text{exp}})$ to denote the value of $\mathcal{H}^{\text{fix}}$ for state $s \in \mathcal{S}_{\text{gen}}$ at the time when the set of expanded states is $\mathcal{S}_{\text{exp}}$. Whenever it is clear from context, we use $\mathcal{H}^{\text{fix}}(s)$ and omit $\mathcal{S}_{\text{exp}}$. Note that defining $\mathcal{H}^{\text{fix}}(s, \mathcal{S}_{\text{exp}})$ only for states $s \in \mathcal{S}_{\text{gen}}$ is sufficient as A* (and UDXBB algorithms in general) use heuristics only on states generated during the search.

Computing and using $\mathcal{H}^{\text{fix}}$ in A* requires several modifications of the A* algorithm (see orange lines in Algorithm 1). First, when ordering nodes in the open list (line 28), we use $\mathcal{H}^{\text{fix}}$ instead of h, i.e., we replace h with a more accurate dynamically calculated estimates. Second, we initialize $\mathcal{H}^{\text{fix}}$ as h (line 5) at start, and then we update $\mathcal{H}^{\text{fix}}$ whenever we expand the next node (lines 15-18). In other words, we update $\mathcal{H}^{\text{fix}}$ whenever we obtain additional information about the state space. Updating $\mathcal{H}^{\text{fix}}$ is done by recursively propagating the information from the expanded node n_s to its parents and to states dominated by s according to the dominance relation $\preceq$. To this end, we need to keep track of the set of generated states and all paths that lead to them.

We store in $\mathcal{S}_{\text{gen}}$ the set of states generated during the search, and in $\mathcal{S}_{\text{exp}}$ the set of states expanded during the search. Then we build the mapping $parents : \mathcal{S}_{\text{gen}} \to 2^{\mathcal{S}_{\text{gen}}}$ associating each state $s \in \mathcal{S}_{\text{gen}}$ with the set of its parent states that are already expanded. Specifically, $parents(s)$ is the set of all $p \in \mathcal{S}_{\text{exp}}$ such that p is a parent of s (i.e., $s \in Succ(p)$). Note that, for simplicity, the pseudocode makes use of several calls to $Succ(s)$. However, storing the results in a cache, a single call is sufficient.

Finally, as $\mathcal{H}^{\text{fix}}(s)$ may improve after n_s is inserted in $open$, we ensure that nodes are expanded in the ascending order of $g(n_s) + \mathcal{H}^{\text{fix}}(s)$ as follows. When extracting a node from $open$, we check if its value improved since insertion. If so, we delay its expansion by re-inserting it in $open$ with the current $\mathcal{H}^{\text{fix}}(s)$ (lines 9-11). This simple change is sufficient when heuristics only monotonically increase, as in our case.

This modification of A* corresponds with using a dynamic heuristic whose values depend on the set of expanded states. Thus, this is an instance of the framework for search with dynamic heuristics by Christen et al. (2025), when using DYN-A* with re-evaluation. After expanding the nodes, they update the heuristic values of the states with the new information obtained, which is a general case of calling the propagation function. This variant of A* still guarantees optimal solutions, as long as the dynamic-heuristic monotonically increases while remaining admissible.

3.2 Propagation Rules

We consider three rules to update the heuristic values of the generated states. We use $\mathcal{H}$ to denote the changing heuristic during the application of the rules. First, we propose the *dominance propagation rule*, which states that whenever $s \preceq t$, we can propagate the heuristic value of t to s.

Dominance rule: Let $s, t \in \mathcal{S}_{\text{gen}}$ such that $s \preceq t$.

$$\mathcal{H}(s) \leftarrow \max(\mathcal{H}(s), \mathcal{H}(t))$$

The *pathmax rules* were first introduced by Mérő (1984) in the B' algorithm (a variant of A*). They aim to improve the heuristic estimate of a state by considering the heuristic values of its predecessors and successors. These rules adjust the heuristic values of states based on the portion of the search space that has been explored, propagating the value among each expanded state and its successors.

Pathmax rule 1: Let $p \in \mathcal{S}_{\text{exp}}, s \in Succ(p)$.

$$\mathcal{H}(s) \leftarrow \max(\mathcal{H}(s), \mathcal{H}(p) - c(p, s))$$

Pathmax rule 2: Let $s \in \mathcal{S}_{\text{exp}}$.

$$\mathcal{H}(s) \leftarrow \max(\mathcal{H}(s), \min_{t \in Succ(s)} (\mathcal{H}(t) + c(s, t)))$$

Theorem 1. *Let $\mathcal{H} \in \mathbb{H}$ be an admissible heuristic, then the heuristic after applying dominance rule, pathmax rule 1, or pathmax rule 2 is still admissible.*

Proof. Since $\mathcal{H}(s) \leq h^*(s)$ for every $s \in \mathcal{S}_{\text{gen}}$, it suffices to show that none of the rules causes $\mathcal{H}(s) > h^*(s)$:

(Dominance) Since $s \preceq t$, and $\mathcal{H}(t)$ was admissible we have that $\mathcal{H}(t) \leq h^*(t) \leq h^*(s)$.

(Pathmax 1) From consistency of h^* we have that $h^*(p) - c(p, s) \leq h^*(s)$, and from admissibility of $\mathcal{H}$ we have that $\mathcal{H}(p) - c(p, s) \leq h^*(p) - c(p, s) \leq h^*(s)$.

(Pathmax 2) Let $t_{\text{opt}} \in Succ(s)$ be the first state on an optimal plan for s (if there is no plan, then $h^*(s) = \infty$ and any value is admissible). From admissibility of $\mathcal{H}$ we have that $\mathcal{H}(t_{\text{opt}}) \leq h^*(t_{\text{opt}})$ so $\min_{t \in Succ(s)}(\mathcal{H}(t) + c(s, t)) \leq h^*(t_{\text{opt}}) + c(s, t_{\text{opt}}) = h^*(s)$. □

On their own, the pathmax rules do not alter the expansion order if the heuristic was already consistent. However, when combined with the dominance rule, positive synergies arise. For example, the changes produced by pathmax 2 can cause $\mathcal{H}(t)$ to be momentarily greater than $\mathcal{H}(s)$ for some t that dominates s. When this happens, the dominance rule will be applied, improving the heuristic value of s and making the dominance relation consistent with the heuristic again.

In Fig. 2, we illustrate the behavior of the three propagation rules: We assume r dominates p. Initially, $h(p) =$

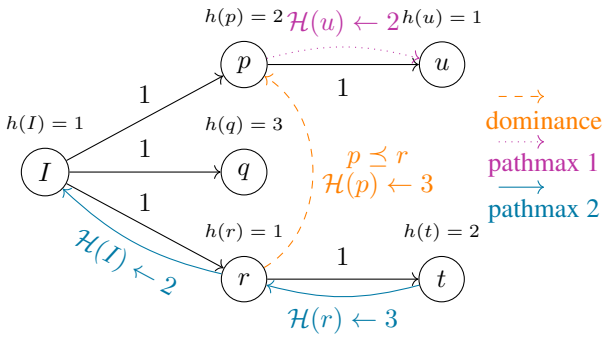

Figure 2: Illustration of propagation rules behavior.

$2 \geq h(r) = 1$, meaning that the dominance rule is not applied. However, the pathmax rules can be applied. The first rule that is triggered is the pathmax rule 2, propagating $\mathcal{H}(r) + c(I, r) = 2$ from r to I. Then, the pathmax rule 2 propagates $\mathcal{H}(t) + c(r, t) = 2 + 1 = 3$ from t to r. At this moment, the consistency between the heuristic and the dominance relation is broken, since $p \preceq r$ but $\mathcal{H}(r) > \mathcal{H}(p)$. Consequently, the dominance propagation rule propagates the value 3 from r to p. This breaks the consistency between p and u, since $\mathcal{H}(p) \not\leq \mathcal{H}(u) + c(p, u)$, triggering pathmax rule 1, which propagates $\mathcal{H}(p) - c(p, u) = 3 - 1 = 2$ to u.

3.3 Propagation Heuristic

Theorem 1 shows that if we initialize $\mathcal{H}$ with an admissible heuristic, then repeatedly updating $\mathcal{H}$ using the propagation rules always retains admissibility (cf. line 15 of Algorithm 1). Now, we explore what is the heuristic that can be obtained from repeatedly applying the propagation rules.

The first observation is that all rules are monotone: a function f is monotone if $x \leq y$ implies $f(x) \leq f(y)$. So, as long as all rules are applied infinitely often, the order in which they are applied is irrelevant to determine the final outcome. To facilitate the analysis of the resulting heuristic after repeatedly applying the rules until convergence, we introduce the function δ, which simultaneously applies all rules. This simplifies reasoning about the fixed point of the propagation process and abstracts away the specifics of the order in which rules are applied.

Given the current state of the search algorithm $(\mathcal{S}_{\text{exp}}, \mathcal{S}_{\text{gen}}, \mathit{parents})$, we define $\delta \colon \mathbb{H} \mapsto \mathbb{H}$ so that for every $\mathcal{H} \in \mathbb{H}$ and $s \in \mathcal{S}_{\text{gen}}$:

$$(\delta(\mathcal{H}))(s) = \max \begin{cases} \mathcal{H}(s) \\ \max_{d \in Dom(s, \mathcal{S}_{\text{gen}})} \mathcal{H}(d) \\ \max_{p \in parents(s)} (\mathcal{H}(p) - c(p, s)) \\ [\![s \in \mathcal{S}_{\text{exp}}]\!] \min_{t \in Succ(s)} (\mathcal{H}(t) + c(s, t)) \end{cases} \tag{1}$$

where $Dom(s, \mathcal{S}_{\text{gen}}) = \{t \in \mathcal{S}_{\text{gen}} \mid s \preceq t\}$ is the set of generated states that dominate s, and $[\![s \in \mathcal{S}_{\text{exp}}]\!]$ is an indicator function that is equal to 1 if $s \in \mathcal{S}_{\text{exp}}$ and 0 otherwise, meaning that the last case is only considered for expanded states. Note that δ depends on the current set of expanded states, and the set of generated states $\mathcal{S}_{\text{gen}}$ and *parents* is uniquely determined by $\mathcal{S}_{\text{exp}}$.

Second, we assume that the propagation process starts from the original heuristic h. This ignores that, in previous iterations of A*, we already applied propagation, and this is not restarted. However, this does not change the final outcome, as with each iteration $\mathcal{S}_{\text{exp}}$, $\mathcal{S}_{\text{gen}}$ and *parents* can only grow. As the order in which rules are applied is irrelevant, it does not matter whether we have previously performed propagation with a subset of them.

The function δ corresponds to the application of multiple propagation rules, but only once per state. So, we need to apply it multiple times. We denote δ^i to the repeated application of δ i times, e.g., $\delta^2(h) = \delta(\delta(h))$ and $\delta^0(h) = h$. Observe that δ is monotonically non-decreasing: for any $\mathcal{H} \in \mathbb{H}$ we have that $\mathcal{H} \leq \delta(\mathcal{H})$, because $(\delta(\mathcal{H}))(s)$ is defined as $\max(\mathcal{H}(s), \ldots)$. Moreover, as a corollary of Theorem 1, whenever $\mathcal{H} \in \mathbb{H}$ is admissible we have that $\delta(\mathcal{H}) \leq h^*$. Therefore, it follows that repeated application of propagation rules has a well-defined limit: $\mathcal{H}^{\text{fix}} = \lim_{n \to \infty} \delta^n(h)$. To be precise, we say that the limit converges for a state $s \in \mathcal{S}_{\text{gen}}$ if there exists some $k \in \mathbb{N}_0$ such that $\delta^k(h)(s) = \delta^i(h)(s)$ for all $i \geq k$. In that case, $it(s)$ the last iteration in which the value of s changed and after which the heuristic value of s is fixed, i.e., $it(s) = \min\{k \geq 0 \mid \forall i \geq k, \delta^k(h)(s) = \delta^i(h)(s)\}$. If the limit does not converge to a finite value for some s, we write $it(s) = \infty$. With this, we can finally define the result of the propagation as follows.

Definition 1 (Propagation Heuristic)**.** *Let Θ be a transition system, h a heuristic function, and $(\mathcal{S}_{\text{exp}}, \mathcal{S}_{\text{gen}}, \mathit{parents})$ the current state of the search. The propagation heuristic is defined for $s \in \mathcal{S}_{\text{gen}}$ as $\mathcal{H}^{\text{fix}}(s, \mathcal{S}_{\text{exp}}) = (\delta^{it(s)}(h))(s)$ if $it(s) < \infty$ and $\mathcal{H}^{\text{fix}}(s) = \infty$ otherwise.*

The resulting heuristic $\mathcal{H}^{\text{fix}}$ remains admissible.

Theorem 2. *Let Θ be a transition system, h a heuristic function, and $\mathcal{S}_{\text{exp}}$, $\mathcal{S}_{\text{gen}}$ sets of expanded and generated states, respectively. If h is admissible, the corresponding propagation heuristic $\mathcal{H}^{\text{fix}}$ is also admissible for all $s \in \mathcal{S}_{\text{gen}}$.*

Proof. Each application of δ corresponds to applying propagation rules. Therefore, if h is admissible by Theorem 1, then $\delta^n(h)(s) \leq h^*(s)$ for all $n \in \mathbb{N}_0$ and $s \in \mathcal{S}_{\text{gen}}$.

For any $s \in \mathcal{S}_{\text{gen}}$ such that $\mathcal{H}^{\text{fix}}(s) < \infty$, $\mathcal{H}^{\text{fix}}(s) = \delta^n(h)(s)$ for some $n \in \mathbb{N}_0$, so $\mathcal{H}^{\text{fix}}(s)$ is admissible.

For $s \in \mathcal{S}_{\text{gen}}$ such that $\mathcal{H}^{\text{fix}}(s) = \infty$, we show by contradiction that $h^*(s) = \infty$ too. Assume that $h^*(s) < \infty$. Then, the value of $\delta^i(h)(s)$ can increase at most $h^*(s) - h(s)$ times because $\delta^i(h)(s) \in \mathbb{N}_0$. Therefore, the value of $\delta^n(h)(s)$ converges, contradicting that $\mathcal{H}^{\text{fix}}(s) = \infty$. □

3.4 Bounding $\mathcal{H}^{\text{fix}}$

Unfortunately, we cannot always reach the limit $\mathcal{H}^{\text{fix}}$ by repeatedly applying δ (or the propagation rules) a *finite* number of steps. In problems with dead-ends, there can be states with $\mathcal{H}^{\text{fix}}(s) = h^*(s) = \infty$. In such cases, repeated application of δ may continually increase the heuristic values of those states. In the example in Fig. 3, the pathmax rule 2 propagates $\mathcal{H}(s) + c(t, s)$ to t. Since $s \preceq t$ and the new value of $\mathcal{H}(t)$ exceeds $\mathcal{H}(s)$, the dominance rule is triggered, causing t to propagate its updated value back to s. This, in turn,

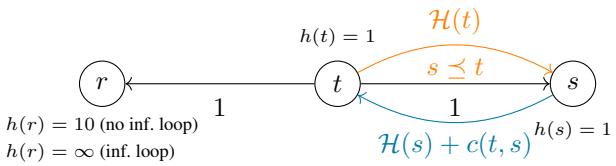

Figure 3: Infinite loop example.

causes the pathmax rule 2 to apply again, leading to a loop between s and t. If $h(r) = \infty$, this loop will not terminate in a finite number of steps.

Addressing this by detecting loops is not trivial, as a loop does not necessarily imply an infinite loop. Since the minimizer node for the pathmax rule 2 (r in the example) can change, the loop could be broken. In Fig. 3, if $h(r)=\infty$ the loop will continue infinitely. Otherwise, e.g. if $h(r) = 10$, the loop will terminate when $\mathcal{H}(s)$ has increased enough to make r the new minimizer, which will happen when $\mathcal{H}(s) = 11$. Although this is a simple case involving only two states, cycles can arise across larger sets of states.

To address this, we analyze the properties of $\mathcal{H}^{\text{fix}}$ and show that we can compute a finite bound $\mathcal{C}^* \in \mathbb{N}_0$ such that $\mathcal{H}^{\text{fix}}(s) \leq \mathcal{C}^*$ for all $s \in \mathcal{S}_{\text{gen}}$ with $\mathcal{H}^{\text{fix}}(s) < \infty$. Therefore, whenever propagation rules increase the heuristic for some state s above $\mathcal{C}^*$, we can safely assign $\mathcal{H}^{\text{fix}}(s) = \infty$.

The existence of such a bound follows from Theorem 2, since $\mathcal{H}^{\text{fix}}(s) \leq h^*(s)$ and it is possible to bound the maximum goal distance (e.g. by the number of states in the state space times the maximum action cost). However, such a bound is impractical as it depends on the entire state space and not only the current search tree. Fortunately, we can also find a tighter bound on the maximum finite heuristic value that can be achieved via propagation by analyzing how values are propagated among states. The intuition is that if $\mathcal{H}^{\text{fix}}(s) < \infty$ the value $\mathcal{H}^{\text{fix}}(s)$ can always be "justified" by an acyclic sequence of states that we call *propagation chain*.

Definition 2 (Propagation chain)**.** *Let $\mathcal{H}^{\text{fix}}$ be the propagation heuristic for Θ, h, and $\mathcal{S}_{\text{exp}}$. A sequence of states $s_1, s_2, \ldots, s_n$ is a propagation chain for s if $s = s_1$, $\mathcal{H}^{\text{fix}}(s_n) = h(s_n)$, and for all $i \in 1, \ldots, n-1$ the value of s_i was propagated from s_{i+1}, denoted $s_i \leftarrow s_{i+1}$. Thus $it(s_i) > it(s_{i+1})$ and in the last iteration where s_i was updated ($it(s_i)$), $\mathcal{H}^{\text{fix}}(s_{i+1})$ maximizes the right-hand side of Eq. 1. We distinguish three types of propagation:*

- $s_i \leftarrow_{\text{p2}} s_{i+1}$ *(pathmax rule 2): In this case, $s_{i+1} \in Succ(s_i)$ and $\mathcal{H}^{\text{fix}}(s_i) = \mathcal{H}^{\text{fix}}(s_{i+1}) + c(s_i, s_{i+1})$.*
- $s_i \leftarrow_{\text{p1}} s_{i+1}$ *(pathmax rule 1): In this case, s_{i+1} is a parent of s_i and $\mathcal{H}^{\text{fix}}(s_i) = \mathcal{H}^{\text{fix}}(s_{i+1}) - c(s_{i+1}, s_i)$.*
- $s_i \leftarrow_{\preceq} s_{i+1}$ *(dominance rule): In this case, s_{i+1} dominates s_i, i.e., $s_i \preceq s_{i+1}$, and $\mathcal{H}^{\text{fix}}(s_i) = \mathcal{H}^{\text{fix}}(s_{i+1})$.*

Note that propagation chains only refer to the last time that the value of $\mathcal{H}(s)$ changed, and therefore involve only states where $\mathcal{H}^{\text{fix}}(s) < \infty$. Also, propagation chains are always acyclic, due to the strict inequality $it(s_i) > it(s_{i+1})$.

To illustrate a propagation chain, consider state u in Fig. 2, whose propagation chain is $u \leftarrow_{\text{p1}} p \leftarrow_{\preceq} r \leftarrow_{\text{p2}} t$. We can also construct a chain for state t in Fig. 3 when $h(r) = 10$ which is $t \leftarrow_{\text{p2}} r$. Note that, even though the value of t was updated many times due to the loop increasing the values of s and t, the propagation chain only cares about r, which is some state with $it(r) = 0 < it(t)$ that maximizes the value in the last iteration where t was updated. An important property is that such an acyclic propagation chain always exists for any state x such that $\mathcal{H}^{\text{fix}}(x) < \infty$.

Lemma 1. *For every state s s.t. $\mathcal{H}^{\text{fix}}(s) < \infty$, there is an acyclic propagation chain $s = s_1, s_2, \ldots, s_n$ where for all $i \in 1, \ldots, n-1$ it holds that $it(s_i) > it(s_{i+1})$, $\mathcal{H}^{\text{fix}}(s_n) = h(s_n)$, and $s_i \leftarrow_{\text{p2}} s_{i+1}$, $s_i \leftarrow_{\text{p1}} s_{i+1}$, or $s_i \leftarrow_{\preceq} s_{i+1}$.*

Proof. We show that this is the case by induction on $it(s)$. If $it(s) = 0$ then $\mathcal{H}^{\text{fix}}(s) = h(s)$, so the propagation chain consists only of s.

Otherwise, $it(s) > 0$ and there exists some s' such that $it(s') < it(s)$, i.e. there exists a state s' that has reached its fixed value before s. To see why such an s' must exist when $it(s) > 0$, consider the right-hand side of Eq. 1:

1. If $\delta(\mathcal{H}(s)) = \max_{s' \in Dom(s, \mathcal{S}_{\text{gen}})} \mathcal{H}(s')$, then for every state s' that maximizes the expression, $it(s') < it(s)$. Otherwise, when any $\mathcal{H}(s')$ changes at $it(s')$, $\mathcal{H}(s)$ would change at $it(s') + 1$.
2. If $\delta(\mathcal{H}(s)) = \max_{s' \in parents(s)}(\mathcal{H}(s') - c(s', s))$, then for every state s' that maximizes the expression, $it(s') < it(s)$. Otherwise, when $\mathcal{H}(s')$ changes at $it(s')$, $\mathcal{H}(s)$ would change at $it(s') + 1$.
3. If $\delta(\mathcal{H}(s)) = [\![s' \in \mathcal{S}_{\text{exp}}]\!] \min_{s' \in Succ(s)}(\mathcal{H}(s') + c(s, s'))$, then $s \in \mathcal{S}_{\text{exp}}$ and there must exist some $s' \in Succ(s)$ minimizing the expression such that $it(s') < it(s)$. Otherwise, after updating $\mathcal{H}(s')$ at $it(s')$ for all s' that minimize the expression, $\mathcal{H}(s)$ would change.

Due to the inequalities with it, there an acyclic sequence $s_0, s_1, ..., s_n$ s.t. $s = s_0$, $\mathcal{H}(s_n) = h(s_n)$, and $s_i \leftarrow s_{i+1}$. □

The existence of propagation chains means that the value of $\mathcal{H}^{\text{fix}}(s)$ can always be traced back to propagation from some value of the original heuristic $h(s_n)$. This results in the bound

$$\mathcal{C}^* = (\max_{s \in \mathcal{S}_{\text{gen}}, h(s) \neq \infty} h(s)) + \sum_{p \in \mathcal{S}_{\text{exp}}} \max_{s \in Succ(p)} c(p, s), \quad (2)$$

where $\max_{s \in \mathcal{S}_{\text{gen}}, h(s) \neq \infty} h(s)$ is the maximum finite value of h for any generated state, and $\max_{s \in Succ(p)} c(p, s)$ is the cost of the most costly transition from p.

Lemma 2. *For all states $s \in \mathcal{S}_{\text{gen}}$, if $\mathcal{H}^{\text{fix}}(s) < \infty$ then $\mathcal{H}^{\text{fix}}(s) \leq \mathcal{C}^*$*

Proof. By Lemma 1, every state s with $\mathcal{H}^{\text{fix}}(s) < \infty$ has a propagation chain $s = s_1, s_2, \ldots, s_n$ ending in a state s_n with $\mathcal{H}^{\text{fix}}(s_n) = h(s_n)$, where $s_i \leftarrow_{\text{p1}} s_{i+1}$, $s_i \leftarrow_{\text{p2}} s_{i+1}$, or $s_i \leftarrow_{\preceq} s_{i+1}$. Among all update rules, the only one that can increase $\mathcal{H}(s_i)$ above $\mathcal{H}(s_{i+1})$ is Pathmax 2, which only applies to states $s_i \in \mathcal{S}_{\text{exp}}$, and increases it at most by $c(s_i, s_{i+1}) \leq \max_{s' \in Succ(s_i)} c(s_i, s')$. Thus, for each such

s_i in the chain, $\mathcal{H}(s_i) \leq \mathcal{H}(s_{i+1}) + \max_{s' \in Succ(s_i)} c(s_i, s')$. Unrolling this along the chain yields:

$$\mathcal{H}(s_1) \leq \mathcal{H}(s_n) + \sum_{\substack{1 \leq i < n \\ s_i \in \mathcal{S}_{\text{exp}}}} \max_{s' \in Succ(s_i)} c(s_i, s').$$

Since $\mathcal{H}(s_n) = h(s_n) \leq \max_{h(s) \neq \infty} h(s)$, we obtain the desired bound. □

We use this bound to define a modified version of δ, called $\delta_{\mathcal{C}^*}$, that limits the heuristic values to this bound. Given the current state of the search algorithm $(\mathcal{S}_{\text{exp}}, \mathcal{S}_{\text{gen}}, parents)$ and the corresponding bound $\mathcal{C}^*$ from Eq. 2, we define the function $\delta_{\mathcal{C}^*} : \mathbb{H} \mapsto \mathbb{H}$ so that for every heuristic $\mathcal{H} \in \mathbb{H}$ and state $s \in \mathcal{S}_{\text{gen}}$, $(\delta_{\mathcal{C}^*}(\mathcal{H}))(s) = (\delta(\mathcal{H}))(s)$ if $(\delta(\mathcal{H}))(s) \leq \mathcal{C}^*$, and ∞ otherwise.

Theorem 3. *Let $h \in \mathbb{H}$ be an admissible heuristic, and let $\mathcal{C}^*$ be as in Eq. 2. Then there exists a finite $n \in \mathbb{N}_0$ such that $(\delta_{\mathcal{C}^*})^n(h) = (\delta_{\mathcal{C}^*})^{n-1}(h) = \mathcal{H}^{\text{fix}}$, i.e., repeated application of $\delta_{\mathcal{C}^*}$ reaches a fixed point equal to $\mathcal{H}^{\text{fix}}$.*

Proof. From Theorem 2, we have that $\mathcal{H}^{\text{fix}}$ is admissible, i.e., $\mathcal{H}^{\text{fix}} \leq h^*$. From Lemma 2, it follows that if $\mathcal{H}^{\text{fix}}(s) > \mathcal{C}^*$ for some $s \in \mathcal{S}_{\text{gen}}$, then $\mathcal{H}^{\text{fix}}(s) = \infty = h^*(s)$. Therefore, since, for any $\mathcal{H} \in \mathbb{H}$ and $s \in \mathcal{S}_{\text{gen}}$, $\delta_{\mathcal{C}^*}(\mathcal{H})(s) = \infty$ only if $\delta(\mathcal{H})(s) > \mathcal{C}^*$ and $\delta_{\mathcal{C}^*}(\mathcal{H})(s) = \delta(\mathcal{H})(s)$ otherwise, it follows that if $\mathcal{H}^{\text{fix}}(s) = \infty$, then also $(\delta^m(h))(s) > \mathcal{C}^*$ for some finite m. Therefore $(\delta_{\mathcal{C}^*})^n(h) = (\delta_{\mathcal{C}^*})^{n-1}(h) = \mathcal{H}^{\text{fix}}$ for some finite n. □

3.5 Computation of $\mathcal{H}^{\text{fix}}$

In practice, repeatedly applying $\delta_{\mathcal{C}^*}$ may converge slowly, as this requires to check whether the value increases for all states. Furthermore, this must be repeated after every expansion, where only a few new states have been generated. This means that one can just focus on propagating the new information after each expansion instead of recomputing the value for all states where nothing has changed.

Algorithm 2 updates $\mathcal{H}$ after each expansion, and returns $\mathcal{H} = \mathcal{H}^{\text{fix}}$ for the current state of the search. It maintains the set $\mathcal{U}$ of updated states, which contains every state u for which $\mathcal{H}(u)$ has been updated, and this update has not yet been propagated. The function is called after each expansion, with $\mathcal{U}$ initialized with the successors of the expanded node, as they are the only newly updated states. It then iteratively applies the propagation rules to each state in $\mathcal{U}$ when possible, until no more updates can be made, i.e., $\mathcal{U}$ is empty. The result is the same as applying $\delta_{\mathcal{C}^*}$, as only updated states inserted in $\mathcal{U}$ can lead to further changes.

4 Consistency of $\mathcal{H}^{\text{fix}}$

Next, we consider whether the propagation process can create inconsistencies or whether one may need to re-open states even if the original heuristic h was consistent. At first glance, this may look trivial to show because pathmax rules cause the heuristic to be consistent across the explored search tree. However, this is only true on the portion of the state space that has been explored, which is not enough to avoid re-expansions (Holte 2010). Specifically, if we have

Algorithm 2: Propagation($s_e, \mathcal{S}_{\text{gen}}, \mathcal{S}_{\text{exp}}, parents, \mathcal{H}, h$)

Require: s_e is the state of the last expanded node
Require: $\mathcal{S}_{\text{gen}}$, $\mathcal{S}_{\text{exp}}$, and $parents$
Require: Dynamic and original heuristic functions $\mathcal{H}$ and h
1: $\mathcal{U} \leftarrow Succ(s_e)$
2: **for** $t \in Succ(s_e)$ s.t. $\mathcal{H}(t)$ undefined **do** ▷ rules on new states
3: $\quad \mathcal{H}(t) \leftarrow \max\left(h(t), \max_{d \in Dom(t, \mathcal{S}_{\text{gen}})} \mathcal{H}(d), \mathcal{H}(s_e) - c(s_e, t)\right)$
4: **while** $\mathcal{U} \neq \emptyset$ **do**
5: $\quad u \leftarrow \mathcal{U}.pop()$ ▷ pop any element of $\mathcal{U}$
6: $\quad$ **for** $p \in parents(u)$ **do**
7: $\quad\quad$ UpdateH(p, $\min_{t \in Succ(p)} (\mathcal{H}(t) + c(p,t))$) ▷ Pathmax 2
8: $\quad$ **if** $u \in \mathcal{S}_{\text{exp}}$ **then**
9: $\quad\quad$ **for** $t \in Succ(u)$ **do**
10: $\quad\quad\quad$ UpdateH($t, \mathcal{H}(t) - c(u,t)$) ▷ Pathmax 1
11: $\quad$ **for** $d \in Dom(u, \mathcal{S}_{\text{gen}})$ **do**
12: $\quad\quad$ UpdateH($d, \mathcal{H}(u)$) ▷ Dominance
13: **return** $\mathcal{H}$
14: **procedure** UPDATEH($s, value$)
15: $\quad$ **if** $\mathcal{H}(s) < value$ **then**
16: $\quad\quad \mathcal{H}(s) \leftarrow value$ if $value \leq \mathcal{C}^*$ else ∞
17: $\quad\quad \mathcal{U} \leftarrow \mathcal{U} \cup s$

two states in open s and s' where $s' \in Succ(s)$, but s' has been generated by another (possibly sub-optimal) path, then it could be the case that $\mathcal{H}^{\text{fix}}(s) > \mathcal{H}^{\text{fix}}(s') + c(s, s')$ and this is not detected by pathmax since s has not been expanded yet so we do not know that s is parent of s' yet. In turn, this could cause A* to expand s' first, so that it could be re-expanded later once we discover a better path to s' when expanding s.

However, it can be shown that this cannot happen if both the original heuristic and the dominance relation were consistent. Following Torralba (2021), we formalize an instance as a tuple $I = \langle \Theta, h, \preceq \rangle$, where Θ is a transition system, h is an admissible heuristic, and $\preceq$ is a dominance relation for Θ. An instance $I = \langle \Theta, h, \preceq \rangle$ is consistent if (1) h is consistent, (2) $\preceq$ is a transitive cost-simulation, and (3) $\preceq$ is consistent with h: $s \preceq t \Rightarrow h(t) \leq h(s)$. A dominance relation $\preceq$ is ***transitive*** if whenever $s \preceq t$ and $t \preceq u$, then $s \preceq u$, and a ***cost-simulation*** if whenever $s \preceq t$, for all $s \xrightarrow{l} s'$, either $s' \preceq t$ or there exists $t \xrightarrow{l'} t'$ s.t. $c(l') \leq c(l)$ and $s' \preceq t'$. If both properties hold, $\preceq$ is a ***transitive cost-simulation***.

Definition 3. *An instance $I = \langle \Theta, h, \preceq \rangle$ is **consistent** if (1) h is consistent, (2) $\preceq$ is a transitive cost-simulation, and (3) $\preceq$ is consistent with h: $s \preceq t \Rightarrow h(t) \leq h(s)$.*

Consistent instances are quite common when h is consistent, since dominance relations in the literature (Torralba and Hoffmann 2015) are transitive cost-simulations, and consistent with most heuristics (Torralba 2021).

As $\mathcal{H}^{\text{fix}}$ is defined over states in $\mathcal{S}_{\text{gen}}$, for it to be consistent it must hold that for any pair of states $s, t \in \mathcal{S}_{\text{gen}}, \mathcal{H}^{\text{fix}}(s) \leq \mathcal{H}^{\text{fix}}(t) + c(\pi_{(s,t)})$, where $c(\pi_{(s,t)})$ is the cost of an optimal path between s and t. This property is equivalent to consistency on heuristics over the entire state space as the heuristic resulting from assigning the value of non-generated states using pathmax 1 would be consistent.

Lemma 3. *On consistent instances, for all states s with*

$h(s) < \mathcal{H}^{\text{fix}}(s) < \infty$ *there exists a propagation chain* $s \leftarrow s_2 ... \leftarrow s_n$ *s.t. (1) ends with* $s_{n-1} \leftarrow_{\text{p2}} s_n$*, (2) does not contain* $s_i \leftarrow_{\text{p1}} s_{i+1}$*, and (3) does not contain* $s_i \leftarrow_{\preceq} s_{i+1} \leftarrow_{\preceq} s_{i+2}$*. Across this chain* $\mathcal{H}^{\text{fix}}(s_i) \geq \mathcal{H}^{\text{fix}}(s_{i+1})$.

We provide proof sketches below; full proofs are deferred to the appendix (Garmendia, Fišer, and Torralba 2026).

Proof Sketch. This follows from Lemma 1 and the following observations. We can show (1) holds because if h is consistent ending in $s_{n-1} \leftarrow_{\text{p1}} s_n$ cannot happen, because $\mathcal{H}^{\text{fix}}(s_n) = h(s_n)$. Similarly, if h and $\preceq$ are consistent with each other, ending in $s_{n-1} \leftarrow_{\preceq} s_n$ cannot happen. (3) holds because, for transitive $\preceq$, $s_i \leftarrow_{\preceq} s_{i+1} \leftarrow_{\preceq} s_{i+2}$ cannot happen as we would have $s_i \leftarrow_{\preceq} s_{i+2}$ directly. (2) holds because, as after applying some pathmax rule the parent and the children are consistent, chains with $s_i \leftarrow_{\text{p1}} s_{i+1} \leftarrow_{\text{p2}} s_{i+2}$ cannot happen. Also, if $\preceq$ is a cost-simulation, chains with $s_i \leftarrow_{\text{p1}} s_{i+1} \leftarrow_{\preceq} s_{i+2} \leftarrow_{\text{p2}} s_{i+3}$ imply that s_{i+2} has some successor t that dominates s_i so an alternative chain $s_i \leftarrow_{\preceq} t$ exists that does not rely on pathmax one. □

This characterizes the structure of the propagation chains that can occur in consistent instances, which will always be sequences of pathmax rule 2 connected by a dominance rule, and helps to show that $\mathcal{H}^{\text{fix}}$ is consistent.

Theorem 4. *On consistent instances* $\mathcal{H}^{\text{fix}}$ *is consistent.*

Proof Sketch. We show this by contradiction. Assume that there exist two states $s, t \in \mathcal{S}_{\text{gen}}$ such that $\mathcal{H}^{\text{fix}}(s) > \mathcal{H}^{\text{fix}}(t) + c(\pi_{(s,t)})$. If $\mathcal{H}^{\text{fix}}(s) < \infty$, by Lemma 3 we can show that s has a propagation chain that interleaves pathmax and dominance propagations. Also, some pair (s,t) must exist where the chain for s starts with $s \leftarrow_{\preceq} s_2$. We choose $s, t \in \mathcal{S}_{\text{gen}}$ such that $\mathcal{H}^{\text{fix}}(s) > \mathcal{H}^{\text{fix}}(t) + c(\pi_{(s,t)})$, $s \leftarrow_{\preceq} s_2$, and $c(\pi_{(s,t)})$ is minimal among all such pairs. This leads to contradiction, since $\preceq$ is a cost-simulation, there is a path from s_2 to some t_2 such that $s_2 \preceq t_2$, so it must hold that another pair s_2, t_2 exists with $c(\pi_{(s_2,t_2)}) < c(\pi_{(s,t)})$. □

Therefore, we do not re-open states (Christen et al. 2025).

5 Relation to Dominance Pruning

We now compare the number of nodes expanded by A* with dominance pruning (A$^*_{prun}$) and A* using our propagation heuristic $\mathcal{H}^{\text{fix}}$ (A$^*_{prop}$). Following Torralba (2021), we focus on consistent instances, and we borrow the definition of Type 1 optimality from Dechter and Pearl (1985).

Definition 4 (Type 1 Optimality)**.** *Let* $\mathcal{N}_{\text{exp}}(X, I, \phi)$ *be the set of nodes expanded by algorithm* X *on instance* I *with tie-breaking* ϕ*. Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be two families of algorithms.* $\mathcal{A}$ *is* ***Type 1 optimal*** *over* $\mathcal{B}$ *relative to an instance set* $\mathcal{I}$ *if for every tie-breaking* γ *of* B*, there exists some tie-breaking* θ *of* A *such that* $\forall I \in \mathcal{I}, \forall B \in \mathcal{B}, \exists A \in \mathcal{A}, \mathcal{N}_{\text{exp}}(A, I, \theta) \subseteq \mathcal{N}_{\text{exp}}(B, I, \gamma)$.

Theorem 5. A^*_{prop} *is Type 1 optimal over* A^*_{prun} *over consistent instances.*

Proof. Assume, for contradiction, that n_s is the first node expanded by A$^*_{prop}$ but $n_s \notin \mathcal{N}_{\text{exp}}(\text{A}^*_{prun}, I, \gamma)$. Let $open_{prop}$ be the open list of A$^*_{prop}$ when n_s is selected for expansion, $f_{prop}(n)$ the f-value of nodes $n \in open_{prop}$, and $f_{prun}(n)$ the f-value of open nodes in A$^*_{prun}$. Let $N = \{n \in open_{prop} \mid f_{prop}(n) = \min_{x \in open_{prop}} f_{prop}(x)\}$ be the candidates for expansion.

We consider a tie-breaking θ that gives priority to nodes in $\mathcal{N}_{\text{exp}}(\text{A}^*_{prun}, I, \gamma)$. So, when we expand n_s, there is no node in N that was expanded in A$^*_{prun}$, i.e., $f_{prop}(n_{s'}) > f_{prop}(n_s)$ for all $n_{s'} \in open_{prop}$ such that $n_{s'} \in \mathcal{N}_{\text{exp}}(\text{A}^*_{prun}, I, \gamma)$. As n_s was not expanded by A$^*_{prun}$, in A$^*_{prun}$ either (1) $f_{prun}(n_s) \geq F^*$, (2) n_s was not generated due to pruning some ancestors, or (3) n_s was pruned.

In case (1), we have that $f_{prop}(n_s) \geq f_{prun}(n_s) \geq F^*$, reaching a contradiction, as some node n_s in the optimal solution found by A$^*_{prun}$ should be in open. In case (2), we reach a contradiction since n_s was the first node selected for expansion but not expanded by A$^*_{prun}$. In case (3), there is a node n_t such that n_t prunes n_s in A$^*_{prun}$, i.e., $s \preceq t$ and $g(n_t) \leq g(n_s)$, so $f_{prun}(n_t) \leq f_{prun}(n_s)$. Then, by the dominance rule, $\mathcal{H}(t) \leq \mathcal{H}(s)$, so $f_{prop}(n_t) \leq f_{prop}(n_s)$, so this can only happen if all n_t (3a) have already been expanded or (3b) have not been generated by A$^*_{prop}$ when n_s is selected for expansion. Case (3a): If n_t was expanded, then since pathmax rules ensure that the f-value of any expanded node is equal to one of its successors, we know that there is a descendant n_u of n_t in open such that $f_{prop}(n_u) \leq f_{prop}(n_t) \leq f_{prop}(n_s)$. Also, as actions have positive costs $g(n_u) > g(n_t)$, so $\mathcal{H}(u) < \mathcal{H}(s)$. Therefore, $n_u \notin N$, but $f_{prop}(n_u) \leq f_{prop}(n_s)$, reaching a contradiction. Case (3b): If n_t has not been generated, consider the sequence of nodes $n_{t_1}, \ldots, n_{t_k}$ leading to n_t in A$^*_{prop}$ where t_1 is the initial state and $t_k = t$. There must be some n_{t_j} with $1 < j < k$ that was generated but not expanded under tie-breaking θ. Then, $f_{prop}(n_s) = g(n_s) + \mathcal{H}^{\text{fix}}(s) < f_{prop}(n_{t_j}) = g(n_{t_j}) + \mathcal{H}^{\text{fix}}(t_j)$. Therefore,

$$f_{prun}(n_{t_j}) \leq f_{prun}(n_t) \leq f_{prun}(n_s) \leq f_{prop}(n_s) < f_{prop}(n_{t_j}) \tag{3}$$

So, some propagation rules increase $\mathcal{H}^{\text{fix}}(t_j)$. With consistent heuristic h, pathmax alone cannot change the value of non-expanded states. Therefore, there must be dominance propagation to t_j or some ancestor t_l of t_j, which then propagated its value to t_j via pathmax 1. As pathmax preserves the f-value, $f_{prop}(n_{t_l}) = f_{prop}(n_{t_i})$ for all $i \in l, \ldots, k$.

Let u_l be a state that propagates its heuristic to t_l via the dominance rule. Since the instance is consistent, $\preceq$ must be a transitive cost-simulation. Therefore, there must exist a sequence of states $u_{l+1}, \ldots, u_k$ such that $c(u_i, u_{i+1}) \leq c(t_i, t_{i+1})$ and $t_i \preceq u_i$ for all $i \in l, \ldots, k$. Furthermore, as h is consistent with $\preceq$, no propagation can happen from u_l to t_l unless u_l has been expanded and $\mathcal{H}^{\text{fix}}(u_l)$ is determined by Pathmax rule 2. Thus, we get that $\mathcal{H}^{\text{fix}}(t_l) \leq \mathcal{H}^{\text{fix}}(u_k) + \sum_{i \in l, \ldots, k-1} c(u_i, u_{i+1})$. However, by transitivity, $s \preceq t \preceq u_k$, so $\mathcal{H}^{\text{fix}}(s) \geq \mathcal{H}^{\text{fix}}(u_k)$, and $\mathcal{H}^{\text{fix}}(s) \geq \mathcal{H}^{\text{fix}}(t_l) - \sum_{i \in l, \ldots, k-1} c(u_i, u_{i+1}) \geq \mathcal{H}^{\text{fix}}(t_k)$. This reaches a contradiction with Eq. 3. □

Thus, A^*_{prop} using the tie-breaking θ defined in Theorem 5, expands no more nodes than A^*_{prun} with any tie-breaking. The opposite does not hold: for instance, Fig. 1 shows A^*_{prop} expanding fewer nodes than A^*_{prun}, and additional examples without dead-ends are in the appendix.

6 Experiments

We implemented our approach within Fast Downward (Helmert 2006). To compute dominance relations, we use the approach by Torralba and Hoffmann (2015) with two configurations: computing the dominance on the original task (atomic), or using merge-and-shrink (*m&s*) (Helmert et al. 2014; Sievers and Helmert 2021) .

We ran experiments with Lab (Seipp et al. 2017) on a cluster with AMD EPYC 7551 CPUs with memory and time limits of 4 GBs and 30 minutes, respectively. We used classical planning domains from the optimal track of International Planning Competitions from 1998 to 2023, excluding domains with conditional effects after grounding, zero cost actions, and unsolvable problems. We also evaluated our approach with the LM-Cut heuristic (Helmert and Domshlak 2009) used as the input heuristic for computing the propagation heuristic in case of A^*_{prop}, or as a standard heuristic in case of A^*_{prun}. We also evaluated with the CEGAR (Seipp and Helmert 2018) and iPDB (Haslum et al. 2007) heuristics, obtaining similar results. Code and results are published in Zenodo (Garmendia, Fišer, and Torralba 2026).

As shown, A^*_{prop} expands at most as many states as A^*_{prun} in consistent instances. In our experiments, we focus on whether dominance relations reduce expansions in practice. Fig. 4 compares the number of node expansions by A^*_{prop} and A^*_{prun} using the LM-cut heuristic, both with an atomic dominance relation, and with a *m&s* dominance relation. In general A^*_{prop} expands no more nodes than A^*_{prun}, except in cases where the heuristic is inconsistent. However, the improvement in node expansions happened only in 120 commonly solved tasks, when using the atomic relation, and in 105 tasks when using the *m&s* relation. Furthermore, the size of the reduction is mostly marginal. We can also see that the overhead in A^*_{prop} mostly does not pay off in this setting as A^*_{prun} solves more tasks: 768 vs 652 with atomic dominance, and 737 vs 576 with *m&s* dominance. Nevertheless, this may change in future when an approximation of the propagation heuristic is used, reducing the overhead.

We collected statistics on the propagation process, focusing on how often it takes long to converge. In both configurations, the heuristic bound $\mathcal{C}^*$ is reached in only six domains, showing that such cases are rare. Typically, propagation converges within a few iterations. For the *m&s* configuration, the propagation function performs an average of 14 iterations per call, with the average of the maximum number of iterations per instance being 167. In the atomic configuration, these numbers are 12 and 177, respectively.

To further understand why the reductions in node expansions achieved by A^*_{prop} are marginal in most cases, we analyze the conditions under which propagation is expected to be beneficial. Consider an instance where the only dominance found is between two states s and t such that $s \preceq t$,

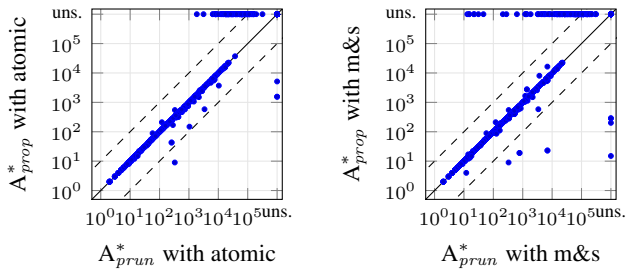

Figure 4: Number of expansions in A^*_{prop} and A^*_{prun} with the LM-Cut heuristic with atomic dominance relation (left) and with merge-and-shrink dominance relation (right).

and t cannot prune s because $g(n_t) > g(n_s)$. For A^*_{prop} to provide an improvement over A^*_{prun}, the expansion of s must be avoided, which will happen only if (1) the heuristic is inconsistent with the dominance relation, or (2) t (and possibly some descendants) is expanded before s, increasing $\mathcal{H}(t)$ enough so that now $g(n_s) + \mathcal{H}(s) = g(n_s) + \mathcal{H}(t)$ is larger than the optimal solution cost.

Although these conditions can be satisfied under our assumptions, the experimental results show that it does not happen frequently. In particular, when using the expansion order of A^*, it is likely that s is expanded before t due to having a lower g-value. This suggests that promising avenues of research to increase the impact of propagation are to (1) obtain dominance relations that are more complementary with the heuristic functions and (2) explore alternative expansion orders exploring different regions of the state space than A^*.

Disregarding instances where there are not useful dominance relations (since this implies that dominance pruning also failed to find useful dominance), we measure the ratio of open list nodes whose priority is updated due to a change in their heuristic value, relative to the total number of states extracted from the open list. Specifically, this corresponds to the percentage of A^* steps where the heuristic value of the state popped from the open list changed due to propagation (see lines 9–11 of Algorithm 1). On average, this occurs in 14% of the cases. We also measure the proportion of heuristic value updates that result in a different state being selected for expansion, which happens on average 52% of the times.

7 Conclusion

We proposed a novel use of dominance relations in heuristic search, not only for pruning but also for propagating heuristic values across states. Our approach dynamically refines the heuristic during search, yielding a new heuristic function that retains admissibility and consistency, and can be seamlessly integrated into A^* without compromising soundness or optimality. Our theoretical analysis proves that, with appropriate tie-breaking, A^*_{prop} is at least as efficient as A^*_{prun} in terms of node expansions. While A^*_{prop} expansion order can be strictly better than A^*_{prun}, our experiments indicate that the reduction in node expansions is typically marginal and does not compensate for the computational overhead.

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

Barley, M. W.; Franco, S.; and Riddle, P. J. 2014. Overcoming the Utility Problem in Heuristic Generation: Why Time Matters. In Chien, S.; Fern, A.; Ruml, W.; and Do, M., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*. AAAI Press.

Christen, R.; Pommerening, F.; Büchner, C.; and Helmert, M. 2025. A Formalism for Optimal Search with Dynamic Heuristics. In Lipovetzky, N.; Sardina, S.; Harabor, D.; and Ramirez, M., eds., *Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2025)*, 30–39. AAAI Press.

Dechter, R.; and Pearl, J. 1985. Generalized Best-First Search Strategies and the Optimality of A*. *Journal of the ACM*, 32(3): 505–536.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced Symmetry Breaking in Cost-Optimal Planning as Forward Search. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 343–347. AAAI Press.

Eckerle, J.; Chen, J.; Sturtevant, N. R.; Zilles, S.; and Holte, R. C. 2017. Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 79–87. AAAI Press.

Franco, S.; and Torralba, Á. 2019. Interleaving Search and Heuristic Improvement. In Surynek, P.; and Yeoh, W., eds., *Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS 2019)*, 130–134. AAAI Press.

Garmendia, M. F. S.; Fišer, D.; and Torralba, Á. 2026. Code, results and appendix for: Beyond Pruning: Leveraging Dominance Relations for Heuristic Propagation. https://doi.org/10.5281/zenodo.19133258.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1007–1012. AAAI Press.

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

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

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM*, 61(3): 16:1–63.

Helmert, M.; and Röger, G. 2008. How Good is Almost Perfect? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 944–949. AAAI Press.

Holte, R. C. 2010. Common Misconceptions Concerning Heuristic Search. In Felner, A.; and Sturtevant, N., eds., *Proceedings of the Third Annual Symposium on Combinatorial Search (SoCS 2010)*, 46–51. AAAI Press.

Karpas, E.; Betzalel, O.; Shimony, S. E.; Tolpin, D.; and Felner, A. 2018. Rational deployment of multiple heuristics in optimal state-space search. *Artificial Intelligence*, 256: 181–210.

Lelis, L. H. S.; Franco, S.; Abisrror, M.; Barley, M.; Zilles, S.; and Holte, R. C. 2016. Heuristic Subset Selection in Classical Planning. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3185–3191. AAAI Press.

Mérő, L. 1984. A Heuristic Search Algorithm with Modifiable Estimate. *Artificial Intelligence*, 23(1): 13–27.

Narayanan, V.; and Likhachev, M. 2017. Heuristic Search on Graphs with Existence Priors for Expensive-to-Evaluate Edges. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 522–530. AAAI Press.

Seipp, J. 2021. Online Saturated Cost Partitioning for Classical Planning. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 317–321. AAAI Press.

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

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

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

Torralba, Á. 2017. From Qualitative to Quantitative Dominance Pruning for Optimal Planning. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4426–4432. IJCAI.

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

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

Wehrle, M.; Helmert, M.; Alkhazraji, Y.; and Mattmüller, R. 2013. The Relative Pruning Power of Strong Stubborn Sets and Expansion Core. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 251–259. AAAI Press.

Weiss, E.; Felner, A.; and Kaminka, G. A. 2023. A Generalization of the Shortest Path Problem to Graphs with Multiple Edge-Cost Estimates. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Rădulescu, R., eds., *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, 2607–2614. IOS Press.