Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)

A Best-First Search Algorithm for FOND Planning and Heuristic Functions to Optimize Decompressed Solution Size

Frederico Messa, André Grahl Pereira

Federal University of Rio Grande do Sul, Brazil
{frederico.messa, agpereira}@inf.ufrgs.br

Abstract

In this work, we study fully-observable non-deterministic (FOND) planning, which models uncertainty through actions with non-deterministic effects. We present a best-first heuristic search algorithm called AND* that searches the *policy-space* of the FOND task to find a solution policy. We generalize the concepts of *optimality*, *admissibility*, and *goal-awareness* for FOND. Using these new concepts, we formalize the concept of heuristic functions that can guide a policy-space search. We analyze different aspects of the general structure of FOND solutions to introduce and characterize a set of FOND heuristics that estimate how far a policy is from becoming a solution. One of these heuristics applies a novel insight. Guided by them AND* returns only solutions with the minimal possible number of mapped states. We systematically study these FOND heuristics theoretically and empirically. We observe that our best heuristic makes AND* much more effective than the straightforward heuristics. We believe that our work allows a better understanding of how to design algorithms and heuristics to solve FOND tasks.

Introduction

Many goal-directed problems can be modeled as *planning tasks* and solved with *AI planning* techniques. The solution of a planning task is a plan describing what actions should be taken in what states so that a goal state is reached from the initial state. Such a plan is also called a policy.

Fully-observable non-deterministic (FOND) planning models tasks with uncertainty through actions with non-deterministic effects. A solution for a FOND planning task is a policy that takes into account all the outcomes of taken actions, and under non-adversarial environments safely guides one to a goal state. If cyclic trajectories are allowed, states may be visited an arbitrary number of times before eventually reaching the goal. Then the solution is called a *strong-cyclic* policy. In this work, we focus on the search for strong-cyclic policies that solve the task.

There are several FOND planners that search for strong-cyclic policies. They apply a diverse set of techniques to effectively solve FOND tasks. Some do re-planning: they repeatedly try to extend policies that are solutions for the deterministic version of the task – found using *classical planning* search techniques – into policies that are solutions for the original task (Kuter et al. 2008; Fu et al. 2011; Muise, McIlraith, and Beck 2012; Pereira et al. 2022). Others solve FOND tasks by heuristically expanding an AND/OR graph that represents the task state-space and re-computing the current best action for each non-leaf state in the graph, until a solution policy is found (Hansen and Zilberstein 2001; Mattmüller et al. 2010). A planner that compiles FOND tasks into SAT instances (Geffner and Geffner 2018), planners that use Binary Decision Diagrams (Cimatti et al. 2003; Kissmann and Edelkamp 2009), and other planners (Ramirez and Sardina 2014), were also proposed. Yet, none of the existent planners in our knowledge *explicitly* search the policy-space of the FOND task to find a solution policy.

Inspired by A* (Hart, Nilsson, and Raphael 1968), we propose an algorithm, called A* with Non-Determinism (AND* for short), that searches for a solution policy in the space of policies of the FOND task. It is the analogous version for FOND planning of A*. Starting from the empty policy, it repeatedly expands the most promising stored policy into successor policies, until a solution is found. We generalize for FOND the concepts of solution optimality, heuristic admissibility, and heuristic goal-awareness.

Our main contribution is a systematic study of heuristics for *optimal-size* FOND planning. We analyze different aspects of the general structure of FOND solutions to introduce and characterize a set of FOND heuristics that estimate how far a policy is from becoming a FOND solution. We study their theoretical properties and theoretical relations of dominance. Guided by these heuristics, AND* returns only solutions with the minimal possible number of mapped states. We analyze the empirical impact of heuristic features and verify that our best heuristic, which applies a novel insight, makes AND* much more effective in practice than the straightforward heuristics. We conclude by empirically comparing AND* to some well-established planners.

Background

A FOND task Π induces a non-deterministic state-space task $\Theta_\Pi = \langle S, A, T, c, s_I, S_* \rangle$. Where S is a finite set of states, A is a finite set of *actions*, $T : S \times A \rightharpoonup 2^S$ is a partial function of *transitions* between states, $c : A \to \mathbb{R}_{\geq 0}$ is a *cost function* that maps actions to non-negative real costs, $s_I \in S$ is the *initial state*, and $S_* \subseteq S$ is the set of *goal states*. The factored description of FOND tasks allows compactly

representing certain subsets of S as *partial states*. A partial state S_p groups the set of states that share some property p.

A *trajectory* is a sequence of transitions $\langle\langle s_1, a_1, s_2\rangle, \langle s_2, a_2, s_3\rangle, \ldots, \langle s_k, a_k, s_{k+1}\rangle\rangle$, with $s_{i+1} \in T(s_i, a_i)$, $\forall i \in \{1, 2, \ldots, k\}$. It starts from s_1 and leads to s_k. The empty trajectory starts from any state s and leads to the same state s. Figure 1 depicts an example state-space task with eight states. In this example, s_* is the only goal state, there are no cyclic trajectories, and the application of the action a in s_I results in $T(s_I, a) = \{s_{b1}, s_{a1}, s_{c1}\}$. In other words, the result of applying a in s_I is not deterministic. $\langle\langle s_I, a, s_{c1}\rangle\rangle$ is one of the 37 non-empty existent trajectories in this state-space task. Trajectories represent deterministic paths in the state-space. Later, we will refer to the numbers that appear next to the states.

A *policy* is a partial function $\pi : S \rightharpoonup A$ that maps states to actions. For all $s \in \mathrm{Dom}(\pi)$, $\pi[s]$ is applicable to s (i.e. $T(s, \pi[s]) \neq \bot$). We define $|\pi| = |\mathrm{Dom}(\pi)|$. $\mathrm{Reach}(\pi) = \bigcup_{s \in \mathrm{Dom}(\pi)} T(s, \pi[s])$ is the set of states reachable from a policy π. The outgoing states $\mathrm{Out}(\pi)$ of a policy π are its unmapped reachable states – i.e., $\mathrm{Out}(\pi) = \mathrm{Reach}(\pi) \setminus \mathrm{Dom}(\pi)$. We denote $\mathrm{Out}_*(\pi)$ as the goal states of $\mathrm{Out}(\pi)$, and $\mathrm{Out}_\sim(\pi)$ as the non-goal states of $\mathrm{Out}(\pi)$.

A *π-trajectory* is a trajectory $\langle\langle s_1, a_1, s_2\rangle, \langle s_2, a_2, s_3\rangle, \ldots, \langle s_k, a_k, s_{k+1}\rangle\rangle$, with $a_i = \pi[s_i], \forall i \in \{1, 2, \ldots, k\}$. A *strong-cyclic policy* is a policy π_* with $\mathrm{Out}_\sim(\pi_*) = \emptyset$ and with the property that for any state $s \in \mathrm{Dom}(\pi_*)$ there is at least one π_*-trajectory that starts from s and leads to a goal state. These two requirements are often respectively called *closedness* and *properness*.

A *solution* for Π is a strong-cyclic policy π_* with $s_I \in S_* \cup \mathrm{Dom}(\pi_*)$. In other words, a solution is a policy that under a *non*-adversarial[1] environment safely guides one from s_I to a goal state, by addressing all the possible outcomes of the taken actions, although possibly visiting some states an arbitrary number of times before eventually reaching the goal, due to cyclic trajectories. In our example task, any solution would address the three possible outcomes of applying a in s_I and end up mapping all the states besides s_*.

From now on, we disregard the case where s_I is a goal state because – although trivial, as the empty policy becomes a solution – it yields several edge cases in our definitions.

Heuristic Functions for FOND Planning

In this section, we formally define the concept of heuristic functions for FOND planning and characterize some of their properties by extending concepts from classical planning heuristic search.

In classical planning, all transitions are deterministic ($\forall s \in S, \forall a \in A, T(s, a) \neq \bot \implies |T(s, a)| = 1$), so any solution policy is equivalent to a single trajectory from s_I to a goal state. A* (Hart, Nilsson, and Raphael 1968) – the most popular optimal best-first search algorithm for classical planning – finds a trajectory with minimal cost from s_I to a state $s_* \in S_*$ by using f-values to guide the search. It stores trajectories with potential to become a solution and expands

[1]Also called "fair environment". (Cimatti et al. 2003)

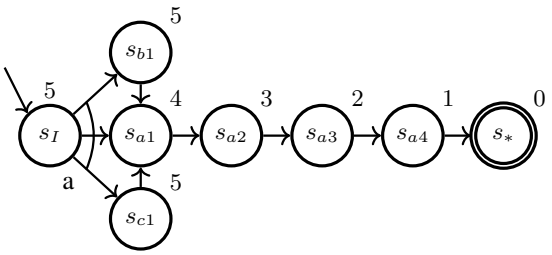


Figure 1: State-Space 1

first trajectories with least f-value. The f-value of a trajectory in classical planning is the sum of its g-value and its h-value (respectively its current accumulated cost and its estimated remaining cost to become a solution). If unitary costs are assumed, we can interpret these values as lengths. The h-values are given by a classical planning heuristic function h, which is *admissible* if it never overestimates the real remaining cost. A* guided by an admissible heuristic is guaranteed to find an optimal solution if one exists.

To define admissible heuristics for FOND planning, we need a concept of cost or value for FOND solutions. However, there are alternative metrics to evaluate solutions in FOND such as: expected number of transitions from s_I to reach the goal under a random selection of action outcomes (Mattmüller et al. 2010), number of mapped states (Fu et al. 2011), and number of mapped *partial* states (Geffner and Geffner 2018). Therefore, we generalize the concept of FOND solution cost. Given *any* metric φ that evaluates and total-preorders[2] all the possible solutions, we define a solution π_* to be optimal if it has minimal $\varphi(\pi_*)$.

In this work, we define heuristic functions for FOND planning as functions that directly return the f-values of policies, instead of giving their h-values. We use this definition because our main new heuristic reasons about how promising a policy is as a single value. For any policy π, we define $f^*(\pi)$ as the value of the best solution it may become if extended – i.e., $f^*(\pi) = \min_{\text{solution } \pi_* | \pi_* \supseteq \pi} \{\varphi(\pi_*)\}$. $f^*(\pi) = \infty$ if there is no solution $\pi_* \supseteq \pi$. A FOND heuristic f is *admissible* if $f(\pi) \leq f^*(\pi)$ for any policy π, analogously to the admissibility concept of classical planning heuristics. A FOND heuristic is *goal-aware* if $f(\pi_*) = \varphi(\pi_*)$ for all solutions π_*.

Note that any metric that allows reducing the cost of some solution by extending it into a larger one cannot possibly have a heuristic simultaneously admissible and goal-aware.

Optimal-Size FOND Heuristics

Effectively evaluating how far a given policy is from becoming a FOND solution is a fundamental step that has not been extensively explored, and that we are interested to tackle. Heuristics that provide such information would be of great utility for approaches that search for solutions

[2]A metric φ total-preorders the solutions if for all pairs of solutions π_*, π'_*, we have $\varphi(\pi_*) \leq \varphi(\pi'_*) \vee \varphi(\pi_*) \geq \varphi(\pi'_*)$, and for all triples of solutions π_*, π'_*, π''_*, we have $\varphi(\pi_*) \leq \varphi(\pi'_*) \wedge \varphi(\pi'_*) \leq \varphi(\pi''_*) \implies \varphi(\pi_*) \leq \varphi(\pi''_*)$.

in the policy-space. In this section, we present and characterize FOND heuristic functions that are admissible and goal-aware for the metric of size φ. For any solution π_*, $\varphi(\pi_*) = |\pi_*| = |\mathrm{Dom}(\pi_*)|$. The presented heuristics estimate, for a given policy π, the minimal possible size of any solution that can be made from π, thus also providing information on how far π is from becoming a solution.

We use classical planning heuristics to estimate the length – denoted $h^*(s)$ – of the shortest trajectory from any given state s to S_* – with $h^*(s) = \infty$ if there is no such trajectory. A classical planning heuristic h only returns non-negative values, and is admissible if $h(s) \leq h^*(s)$ for all states $s \in S$. We assume that all classical planning heuristics dominate the blind heuristic h^{BLIND} – i.e., that they return values at least one to non-goal states. We also assume $\mathrm{Dom}(\pi) \cap S_* = \emptyset$ for all policies π to avoid edge cases that are irrelevant when minimizing size (as we never need to map goal states).

We refer to the example state-space task presented in Figure 1. The h^*-values of the states are presented next to them. We will use the policy $\pi_{\text{e.g.}} = \{s_I \mapsto a\}$ as our recurring example in this section.

Heuristic G

A simple admissible FOND heuristic is $G(\pi) = |\pi|$. The heuristic G is trivially goal-aware, because $G(\pi_*) = |\pi_*| = \varphi(\pi_*)$ for any solution π_*. In our example, $G(\pi_{\text{e.g.}}) = 1$ because $\pi_{\text{e.g.}}$ contains a single mapped state.

Heuristic Count (C)

Another simple admissible FOND heuristic is $\mathrm{Count}(\pi) = G(\pi) + |\mathrm{Out}_\sim(\pi)| = |\mathrm{Dom}(\pi) \sqcup \mathrm{Out}_\sim(\pi)|$. The heuristic Count (C for short) is admissible because a policy π with $\mathrm{Out}_\sim(\pi) \neq \emptyset$ must map all the states in $\mathrm{Out}_\sim(\pi)$ to actions to satisfy the closedness property of a solution. Since all solutions π_* have $\mathrm{Out}_\sim(\pi_*) = \emptyset$, they all have $C(\pi_*) = G(\pi_*) = \varphi(\pi_*)$, thus C is goal-aware. In our example, heuristic $C(\pi_{\text{e.g.}}) = 4$, since $\pi_{\text{e.g.}}$ contains one mapped state plus three non-goal outgoing states $\mathrm{Out}_\sim(\pi_{\text{e.g.}}) = \{s_{b1}, s_{a1}, s_{c1}\}$.

Heuristic $\mathrm{Nearest}$ ($\downarrow$)

Any non-empty policy π with empty $\mathrm{Out}_*(\pi)$ is non-proper, as it does not have any π-trajectory leading to goal states. Moreover, if h is an admissible classical planning heuristic, mapping less than $\min_{s \in \mathrm{Out}(\pi)}\{h(s)\}$ new states in π is not enough for π to have such a trajectory. Therefore, a FOND heuristic $\mathrm{Nearest}(\pi) = G(\pi) + \min_{s \in \mathrm{Out}(\pi)}\{h(s)\}$ with admissible h is admissible for non-empty policies π with empty $\mathrm{Out}_*(\pi)$. $\mathrm{Nearest}$ ($\downarrow$ for short) is also admissible if $\mathrm{Out}_*(\pi)$ is not empty because in that case $\min_{s \in \mathrm{Out}(\pi)}\{h(s)\}$ is zero and $\downarrow(\pi) = G(\pi)$. Thus, it is always admissible for non-empty policies, and also is goal-aware since $\mathrm{Out}_*(\pi_*) \neq \emptyset$ for any solution π_*. We hard-define $\downarrow(\emptyset) = 0$ to obtain admissibility in general.

Example. $\downarrow(\pi_{\text{e.g.}}) = 5$ if we use $h = h^*$ because $\pi_{\text{e.g.}}$ contains one mapped state, and the minimal shortest distance from its outgoing states to a goal – $\min_{s \in \mathrm{Out}(\pi_{\text{e.g.}})}\{h(s)\} = \min\{h(s_{b1}), h(s_{a1}), h(s_{c1})\}$ – is four.

Heuristic $\mathrm{Farthest}$ ($\uparrow$)

If a policy π has some state $s \in \mathrm{Dom}(\pi)$ whose shortest distance to any goal state is n, π then needs at least n mapped states to be a solution, otherwise there will be no π-trajectory from s to any goal state. The same is true for states $s \in \mathrm{Out}_\sim(\pi)$, since all states in $\mathrm{Out}_\sim(\pi)$ will be in $\mathrm{Dom}(\pi_*)$ of any solution $\pi_* \supseteq \pi$. Therefore, a FOND heuristic $\mathrm{Farthest}(\pi) = \max_{s \in \mathrm{Dom}(\pi) \sqcup \mathrm{Out}(\pi)}\{h(s)\}$, *without* adding $G(\pi)$, is admissible if h is an admissible classical planning heuristic. $\mathrm{Farthest}$ ($\uparrow$ for short) is *not* goal-aware, though. Nevertheless, it can be combined to other FOND heuristics to improve their quality, owing to the fact that $f(\pi) = \max\{f_1(\pi), f_2(\pi)\}$ is goal-aware and admissible if both f_1 and f_2 are admissible, and any is goal-aware.

Example. $\uparrow(\pi_{\text{e.g.}}) = 5$ if we use $h = h^*$ because among the states in $\pi_{\text{e.g.}}$ and its non-goal outgoing states, the maximal shortest distance to a goal – $\max_{s \in \mathrm{Dom}(\pi_{\text{e.g.}}) \sqcup \mathrm{Out}(\pi_{\text{e.g.}})}\{h(s)\} = \max\{h(s_I), h(s_{b1}), h(s_{a1}), h(s_{c1})\}$ – is five.

Heuristic Δ

When k non-goal states have h^*-values greater than n, we need at least $k + n$ states to create a policy that has trajectories leading all of them to a goal. This is the insightful idea that rules our main heuristic, called Δ. We claim that taking a close look at the heuristic values of the states in $\mathrm{Dom}(\pi) \sqcup \mathrm{Out}_\sim(\pi)$, and aggregating them appropriately, is much more powerful than just counting them (Count) or taking their maximum ($\mathrm{Farthest}$).

To properly define Δ, we first need to define a vector of heuristic values $H^\pi = \langle h^\pi_1, h^\pi_2, \ldots, h^\pi_{C(\pi)} \rangle$ for any given policy π. H^π is the vector that contains the h-values of all states in $\mathrm{Dom}(\pi) \sqcup \mathrm{Out}_\sim(\pi)$, sorted in descending order.

We then define $\Delta(\pi) = \max_{i \in \langle 1,2,\ldots,C(\pi) \rangle}\{h^\pi_i + i - 1\}$, with H^π using an admissible classical planning heuristic h.

Example. $H^{\pi_{\text{e.g.}}} = \langle h_1^{\pi_{\text{e.g.}}}, h_2^{\pi_{\text{e.g.}}}, h_3^{\pi_{\text{e.g.}}}, h_4^{\pi_{\text{e.g.}}} \rangle = \langle 5, 5, 5, 4 \rangle$, if we use $h = h^*$. Then, $\Delta(\pi_{\text{e.g.}}) = 7$, because $\Delta(\pi_{\text{e.g.}}) = \max_{i \in \langle 1,2,\ldots,C(\pi_{\text{e.g.}}) \rangle}\{h_i^{\pi_{\text{e.g.}}} + i - 1\} = \max\{h_1^{\pi_{\text{e.g.}}} + 0, h_2^{\pi_{\text{e.g.}}} + 1, h_3^{\pi_{\text{e.g.}}} + 2, h_4^{\pi_{\text{e.g.}}} + 3\} = \max\{5+0, 5+1, 5+2, 4+3\} = 7$.

Admissibility. We prove that $\Delta(\pi) \leq f^*(\pi)$ for any given policy π by contradiction, assuming $f^*(\pi) < \Delta(\pi)$ for some arbitrary π.

Since $f^*(\pi) < \Delta(\pi)$, $f^*(\pi) \neq \infty$ and thus there is at least one solution $\pi_* \supseteq \pi$ with $|\pi_*| = f^*(\pi)$. If we prove $\Delta(\pi) \leq \Delta(\pi_*)$ and $\Delta(\pi_*) \leq \varphi(\pi_*)$, we will have our contradiction, because $\varphi(\pi_*) = |\pi_*|$ by the choice of φ.

First, we prove that $\Delta(\pi_*) \leq \varphi(\pi_*)$ for any given solution policy π_*. Since π_* is a solution, there is a π_*-trajectory from every state in $\mathrm{Dom}(\pi_*)$ to some goal state. Thus, there are at least i states $s \in \mathrm{Dom}(\pi_*)$ with $h^*(s) \leq i$ for each $i \in \{1, 2, \cdots, |\pi|\}$.[3] By admissibility of h,

[3] Suppose otherwise: suppose that there are only $k < i$ states $s \in \mathrm{Dom}(\pi_*)$ with $h^*(s) \leq i$, for some $i \in \{1, 2, \cdots, |\pi|\}$. Then there is some state $s' \in \mathrm{Dom}(\pi_*)$ with $h^*(s') > i$. Since it has π_*-trajectory to the goal, this trajectory has length at least $i + 1$, and contains i states $s'' \in \mathrm{Dom}(\pi_*)$ with distance at most i to the goal ($h^*(s'') \leq i$), contradicting the supposition.

there are also at least i with $h(s) \leq i$. In other words, $H^{\pi_*} = \langle h_1^{\pi_*}, h_2^{\pi_*}, \ldots, h_{C(\pi_*)}^{\pi_*} \rangle$ has the property of $h_i^{\pi_*} \leq C(\pi_*) + 1 - i$ for each $i \in \{1, 2, \ldots, C(\pi_*)\}$, since because $\mathrm{Out}_{\sim}(\pi_*) = \emptyset$, H^{π_*} is then defined exactly by the h-values of the states in $\mathrm{Dom}(\pi_*)$ and then $C(\pi_*) = |\pi_*|$. Therefore, $\Delta(\pi_*) = \max_{i \in \{1,2,\ldots,C(\pi_*)\}} \{h_i^{\pi_*} + i - 1\} \leq \max_{i \in \{1,2,\ldots,C(\pi_*)\}} \{(C(\pi_*) + 1 - i) + i - 1\} = \max_{i \in \{1,2,\ldots,C(\pi_*)\}} \{C(\pi_*)\} = C(\pi_*) = \varphi(\pi_*)$.

Now, we prove that $\Delta(\pi) \leq \Delta(\pi')$ for any pair of policies π, π' with $\pi \subseteq \pi'$. Consider that the value of $\Delta(\pi)$ is determined by the term $h_k^{\pi} + k - 1$ (i.e., $h_k^{\pi} + k - 1 \geq h_i^{\pi} + i - 1$ for all $i \in \{1, 2, \ldots, C(\pi)\}$). Then $\Delta(\pi) = h_k^{\pi} + k - 1$. Furthermore, $\Delta(\pi')$ have to be at least $h_i^{\pi} + i - 1$ for all $i \in \{1, 2, \ldots, C(\pi)\}$ since $\Delta(\pi') \geq h_i^{\pi'} + i - 1$ and $h_i^{\pi} \leq h_i^{\pi'}$ for all such i, as all states used to define H^{π} are also used to define $H^{\pi'}$. Thus, $\Delta(\pi') \geq h_k^{\pi} + k - 1 = \Delta(\pi)$. □

Goal-Awareness. As we will see next, $\Delta(\pi) \geq C(\pi)$ for any given policy π. So $\Delta(\pi_*) \geq \varphi(\pi_*)$ for any given solution π_* as C is goal-aware. But, $\Delta(\pi_*) \leq f^*(\pi_*) \leq \varphi(\pi_*)$, by the admissibility of Δ. Therefore, $\Delta(\pi_*) = \varphi(\pi_*)$. □

Heuristics Dominances

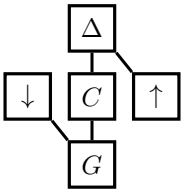


Figure 2: Hasse Diagram of FOND Heuristics Dominance

There are four established direct dominances between pairs of FOND heuristics (Figure 2), given that a same admissible classical planning heuristic h is used by all the FOND heuristics that use one, under the assumption that h dominates h^{BLIND} and that $\mathrm{Dom}(\pi) \cap S_* = \emptyset$.

- $\downarrow(\pi) \geq G(\pi)$ and $C(\pi) \geq G(\pi)$ for any given policy π because both $\downarrow(\pi)$ and $C(\pi)$ are defined as the sum of $G(\pi)$ with some always-non-negative value.
- $\Delta(\pi) \geq C(\pi)$ for any given policy π because $\Delta(\pi) = \max_{i \in \{1,2,\ldots,C(\pi)\}} \{h_i^{\pi} + i - 1\} \geq h_{C(\pi)}^{\pi} + C(\pi) - 1 \geq 1 + C(\pi) - 1 = C(\pi)$.
- $\Delta(\pi) \geq \uparrow(\pi)$ for any given policy π because $\Delta(\pi) = \max_{i \in \{1,2,\ldots,C(\pi)\}} \{h_i^{\pi} + i - 1\} \geq \max_{i \in \{1,2,\ldots,C(\pi)\}} \{h_i^{\pi}\} = \max_{s \in \mathrm{Dom}(\pi) \sqcup \mathrm{Out}(\pi)} \{h(s)\} = \uparrow(\pi)$.

As a consequence, we can obtain the best of all these five FOND heuristics by simply using a hybrid FOND heuristic $\Delta_{\downarrow}(\pi) = \max\{\Delta(\pi), \downarrow(\pi)\}$.

The examples from previous section using $\pi_{\text{e.g.}}$ also show that these four relations of dominance are actually strict ones (i.e., there are actually some policies where the dominating heuristic gives a f-value strictly greater than the dominated one). Furthermore, there are no other relations of dominance among the five basic FOND heuristics besides the four presented, and the one between Δ and G that follows by transitivity. All other combinations do not dominate each other.

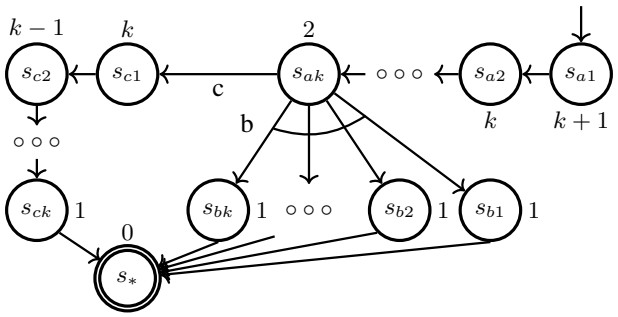


Figure 3: State-Space 2

Δ versus $\downarrow$

We now use an example to show that there is no dominance between Δ and $\downarrow$. Figure 3 depicts the example state-space. In this example, $s_{a1} = s_I$ and s_* is the only goal state. The h^*-value of the states (their shortest distances to the goal) is presented next to them. There are three actions: a, b, and c. Transitions without label are a result of the action a. There are two optimal solutions for the state-space depicted: π_*^1 and π_*^2, both with size $2 \cdot k$ (Equation 1).

$$\begin{aligned} \pi_*^1 = \{&s_{a1} \mapsto a, s_{a2} \mapsto a, \ldots, s_{ak-1} \mapsto a, \mathbf{s_{ak}} \mapsto \mathbf{b}, \\ &s_{b1} \mapsto a, s_{b2} \mapsto a, \ldots, s_{bk} \mapsto a\} \\ \pi_*^2 = \{&s_{a1} \mapsto a, s_{a2} \mapsto a, \ldots, s_{ak-1} \mapsto a, \mathbf{s_{ak}} \mapsto \mathbf{c}, \\ &s_{c1} \mapsto a, s_{c2} \mapsto a, \ldots, s_{ck} \mapsto a\} \end{aligned} \quad (1)$$

We analyze how the FOND heuristics Δ and $\downarrow$ – using $h(s) = h^*(s), \forall s \in S$ – behave for two incomplete policies: $\pi^1 \subset \pi_*^1$ and $\pi^2 \subset \pi_*^2$, both with size k (Equation 2).

$$\begin{aligned} \pi^1 &= \{s_{a1} \mapsto a, s_{a2} \mapsto a, \ldots, s_{ak-1} \mapsto a, \mathbf{s_{ak}} \mapsto \mathbf{b}\} \\ \pi^2 &= \{s_{a1} \mapsto a, s_{a2} \mapsto a, \ldots, s_{ak-1} \mapsto a, \mathbf{s_{ak}} \mapsto \mathbf{c}\} \end{aligned} \quad (2)$$

While $\downarrow(\pi^1) = |\pi^1| + \min\{h(s_{b1}), h(s_{b2}), \ldots, h(s_{bk})\}$ is just $k + 1$, Δ perfectly estimates $f^*(\pi^1)$: $\Delta(\pi^1) = 2 \cdot k$ (Equation 3), because $H^{\pi^1} = \langle k + 1, k, \ldots, 2, 1, 1, \ldots, 1 \rangle$ (with k ones). Differently of Δ, $\downarrow$ does not correlate the heuristic information from the outgoing non-goal states. We use the notation $\langle {a \atop b} \rangle$ to compactly express the sum $a + b$.

$$\max\{\langle {k+1 \atop 0} \rangle, \langle {k \atop 1} \rangle, \ldots, \langle {2 \atop k-1} \rangle, \langle {1 \atop k} \rangle, \langle {1 \atop k+1} \rangle, \ldots, \langle {1 \atop 2 \cdot k-1} \rangle\} = 2 \cdot k \quad (3)$$

For π^2, $\downarrow$ is the one that gives a perfect estimation. $\downarrow(\pi^2) = |\pi^2| + \min\{h(s_{c1})\} = k + k = f^*(\pi^2)$, while $\Delta(\pi^2) = k + 2$ (Equation 4), since $H^{\pi^2} = \langle k + 1, k, k, k - 1, \ldots, 2 \rangle$. Δ does not differentiate outgoing non-goal states from already mapped states, and thus does not capture the fact that the goal is still (much) more than two states far away from the current policy π^2.

$$\max\{\langle {k+1 \atop 0} \rangle, \langle {k \atop 1} \rangle, \langle {k \atop 2} \rangle, \langle {k-1 \atop 3} \rangle, \ldots, \langle {2 \atop k} \rangle\} = k + 2 \quad (4)$$

$\Delta_{\downarrow}$ combines the power of the two FOND heuristics. $\Delta_{\downarrow}(\pi^1) = \Delta_{\downarrow}(\pi^2) = 2 \cdot k$.

Algorithm 1: AND*

```
1 π_I := ∅, Open := {π_I}
2 while Open ≠ ∅ :
3     Remove from Open some policy π with least f(π)
4     return π if closed and proper // strong-cyclic
5     if π is not closed :
6         Select a state s from Out_∼(π)
7         for each action a applicable to s :
8             Insert π' = π ⊔ {s ↦ a} in Open
9 return ⊥ // unsolvable task
```

The AND* Algorithm

In this section, we present the A* with Non-Determinism (AND*) algorithm (Algorithm 1). Our algorithm exceptionally assumes $\mathrm{Out}(\emptyset) = \{s_I\}$. We call $\mathcal{T}(\pi)$ the set of all links $s \to s'$ between states $s \in \mathrm{Dom}(\pi)$, $s' \in T(s, \pi[s])$.

AND* begins the policy-space search on the empty policy $\emptyset$. Since s_I is assumed to be non-goal[4], the first step of AND* is to expand $\emptyset$ using all possible mappings of s_I to actions applicable to s_I, this way generating successor policies with size one. It then repeats this procedure on generated policies until finding a policy that is a solution. If no generated policy is a solution, AND* will return $\perp$ after analyzing all of them.

In detail, what AND* does is to repeatedly select the most promising already generated policy π (one with least f-value) among the ones stored in `Open` (Line 3), verifies if π is yet not closed, and if not, expands it, selecting some of its non-goal outgoing states to map to all possible actions, generating successor policies with size $|\pi| + 1$ (Lines 6–8). If π is instead closed, AND* will check whether π is also proper, and if it is, AND* will return π as a solution (Line 4).

Checking whether a policy is or not proper can be done simply by constructing $\mathcal{T}(\pi)$, and checking whether all the states in $\mathrm{Dom}(\pi)$ are reached regressing from $\mathrm{Out}_*(\pi)$.

Expansion. When expanding a policy π, we map only one of the states in $\mathrm{Out}_\sim(\pi)$ (small-step) – instead of all of them at once (big-step) – because it may be extremely costly to make big-step expansions, and there is no reason to rush that, since we do not even know in advance whether expanding π is useful or not, as it may even not lead to any solution. Doing a small-step expansion may generate up to $|A|$ successors policies while doing a big-step expansion may generate up to $|A|^{|\mathrm{Out}_\sim(\pi)|}$ successor policies. Regardless of which, no policy is generated twice during AND* execution, as two policies π'_1 and π'_2 successors of π have $\pi'_1[s] \neq \pi'_2[s]$ for at least one state s, and mappings are never overwritten.

Execution Example. Figure 4 depicts a state-space with ten states. In this example, $s_0 = s_I$ and s_* is the only goal state. There are three actions: a, b, and c. Transitions without label are a result of the action a. The application of the action b in s_0 results in $T(s_0, b) = \{s_*, s_{a1}\}$, while the application of the action a in s_{b1} results in $T(s_{b1}, a) = \{s_0, s_{b2}\}$. We

[4]AND* will return the solution $\emptyset$, otherwise.

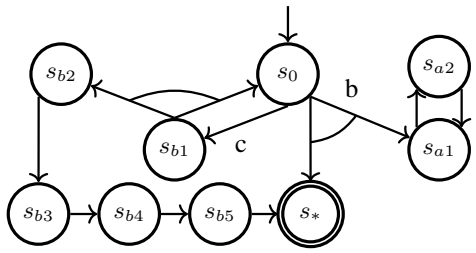


Figure 4: State-Space 3

analyze how the AND* execution would follow using some arbitrary priority function of our choice.

The algorithm starts expanding the policy $\pi_1 = \emptyset$. $\mathrm{Out}(\pi_1) = \{s_0\}$, by definition. Now, since s_0 is not a goal state, $\mathrm{Out}_\sim(\pi_1) \neq \emptyset$, thus π_1 is expanded. The algorithm selects s_0 from $\mathrm{Out}_\sim(\pi_1)$ and considers the two actions applicable to s_0, namely b and c, generating the policies $\pi_2 = \{s_0 \mapsto b\}$ and $\pi_3 = \{s_0 \mapsto c\}$. Both are added to Open.

Let's say AND* picks π_2 first. $\mathrm{Out}(\pi_2) = \{s_{a1}, s_*\}$. Since $\mathrm{Out}_\sim(\pi_2) = \{s_{a1}\} \neq \emptyset$, π_2 is expanded and the policy $\pi_4 = \pi_2 \sqcup \{s_{a1} \mapsto a\}$ is generated and stored. If AND* selects π_4 before π_3, it will expand π_4, as $\mathrm{Out}_\sim(\pi_4) = \{s_{a2}\} \neq \emptyset$, and will generate the policy $\pi_5 = \pi_4 \sqcup \{s_{a2} \mapsto a\}$, which will be discarded whenever selected, because $\mathrm{Out}_\sim(\pi_5) = \emptyset$ and $\mathrm{Out}_*(\pi_5) = \{s_*\}$, but π_5 is not proper. There is no π_5-trajectory from s_{a1} or s_{a2} to any goal state.

After the discard of π_5, there are no more selection choices, as Open will not ever have more than one policy in it again. AND* expands $\pi_3 = \{s_0 \mapsto c\}$. $\mathrm{Out}_\sim(\pi_3) = \{s_{b1}\}$, so the policy $\pi_6 = \pi_3 \sqcup \{s_{b1} \mapsto a\} = \{s_0 \mapsto c, s_{b1} \mapsto a\}$ is generated. AND* will then follow incorporating states into π_6 successfully, until finding the solution $\pi_{10} = \{s_0 \mapsto c, s_{b1} \mapsto a, s_{b2} \mapsto a, s_{b3} \mapsto a, s_{b4} \mapsto a, s_{b5} \mapsto a\}$, and returning it. $\mathrm{Out}(\pi_{10}) = \{s_*\}$.

We will go back to this example later in this section.

Implementation Note. The policy-space is exponentially larger than the state-space, so while we can cache state and action information, such as state-action successors or state heuristic values, the policy representation should be as compact as possible. Therefore, we represent a policy π' as the pointer to its predecessor policy π together with the information of what is the single state-action pair in $\pi' \setminus \pi$.

Soundness and Completeness

Theorem. *AND*, using a heuristic function f that is admissible and goal-aware to a metric φ, always terminates, and returns a solution π_* with minimal $\varphi(\pi_*)$ if there is one, and $\perp$ if there is none.*

Proof Sketch. The proof sketch is subdivided into four independent and sufficient parts.

1. AND* always terminates. The number of possible policies is finite since the numbers of states and actions are finite. And, as discussed before, each policy is generated at most once, thus the number of Open removals is also finite. Finally, the time processing a policy removed from Open is finite. Therefore, the execution must eventually terminate. □

2. If there is any solution, AND* does not return $\bot$. By contradiction, we suppose it does return $\bot$. Consider some strong-cyclic policy π_* with minimal size (there must exist one since there is a solution). Consider also the largest policy $\pi \subseteq \pi_*$ that was once inserted in Open. $\pi \neq \pi_*$ otherwise π would have been returned instead of $\bot$. Since π_* is a strong-cyclic policy with minimal size, all policies $\pi' \subset \pi_*$ must have $\mathrm{Out}_\sim(\pi') \neq \emptyset$. Thus π was expanded. However, any choice of $s \in \mathrm{Out}_\sim(\pi)$ in Line 6 would yield a successor policy $\pi' = \pi \sqcup \{s \mapsto \pi_*[s]\}$ in Line 8, because $\mathrm{Out}_\sim(\pi) \subseteq \mathrm{Dom}(\pi_*)$ (as $\pi \subseteq \pi_*$ and $\mathrm{Out}_\sim(\pi_*) = \emptyset$). But π is the largest subset of π_* once inserted in Open. Contradiction. □

3. When AND* returns π, it is a solution. By contradiction, assume π is not a solution. Then π is not a strong-cyclic policy or $s_I \notin \mathrm{Dom}(\pi)$. The first cannot be true because of Line 4's condition. The second case also cannot be because all the non-empty generated policies π' have $s_I \in \mathrm{Dom}(\pi')$ and because π cannot be $\emptyset$ as $\mathrm{Out}_\sim(\emptyset) \neq \emptyset$ (again failing Line 4's condition). □

4. If AND* returns a solution π_*, $\varphi(\pi_*)$ is optimal. It is optimal because all the intermediate generated policies π_i in $\emptyset = \pi_0 \subset \pi_1 \subset \cdots \subset \pi_{|\pi_*|-1} \subset \pi_{|\pi_*|} = \pi_*$ have $f(\pi_i) \leq \varphi(\pi_*)$ by the admissibility of f and would be expanded before any sub-optimal solution π'_* with $\varphi(\pi'_*) > \varphi(\pi_*)$ because $f(\pi'_*) = \varphi(\pi'_*)$ by the goal-awareness of f. □

Corollary. *AND*, using the presented admissible goal-aware optimal-size FOND heuristics, always terminates, and returns a solution π_* with the minimal possible number of mapped states if there is one, and $\bot$ if there is none.*

Early Deadlock Detection

We can always discard a newly generated policy $\pi' = \pi \sqcup \{s \mapsto a\}$ if $\exists s' \in T(s,a) \mid h(s') = \infty$, with h admissible, because then $\mathrm{Out}_\sim(\pi')$ has *dead-end states*. Besides such policies, we can detect and discard in advance some other policies that cannot become proper even with new state-actions mappings. If we verify that a policy π has some state $s \in \mathrm{Dom}(\pi)$ which has no current π-trajectory from s leading to any state $s' \in \mathrm{Out}(\pi)$, we say that s has no *escape route*, and we can for sure discard π. On the other hand, if some policy π has $\mathrm{Out}_\sim(\pi) = \emptyset$, and all states $s \in \mathrm{Dom}(\pi)$ have some escape route, then π is necessarily a strong-cyclic policy, by definition.

In particular, we can incrementally verify whether all mapped states of a successor policy π' have escape routes, given that we know that all mapped states of its predecessor policy π had escape routes (Lemma 1).

Lemma 1. *If $\pi' = \pi \sqcup \{s \mapsto a\}$ and all mapped states of π have escape routes, then all mapped states of π' have escape routes if and only if $T(s,a) \setminus \{s\} \not\subseteq \{s' \in \mathrm{Dom}(\pi) \mid \mathrm{Out}(\pi, s') = \{s\}\}$, with $\mathrm{Out}(\pi, s') = \{s'' \in \mathrm{Out}(\pi) \mid \exists \pi$-trajectory from s' leading to $s''\}$.*

Proof sketch. In on hand, if all the states resulting from the new mapping $s \mapsto a$ are either s itself or states in $\mathrm{Dom}(\pi)$ that have no escape route that leads to any state besides s in π, then s and all these states will have no escape routes in the new policy π'. They will form a *deadlock*.

Algorithm 2: Method for Early Deadlock Detection

```
1 Method still_no_deadlocks(π' = π ⊔ {s ↦ a}):
2     Stack := {s}, Marked := ∅
3     while Stack ≠ ∅ :
4         Move some state s' from Stack to Marked
5         return ⊤ if T(s', π'[s']) \ Dom(π') ≠ ∅
6         Extend Stack with T(s', π'[s']) \ (Stack ⊔ Marked)
7     return ⊥
```

On the other hand, if $T(s,a) \setminus \{s\} \not\subseteq \{s' \in \mathrm{Dom}(\pi) \mid \mathrm{Out}(\pi, s') = \{s\}\}$, there is at least one state $s' \in T(s,a)$ that either (1) is in $\mathrm{Out}(\pi')$; or (2) has $\mathrm{Out}(\pi, s') \neq \{s\}$. Note first that since all mapped states of π had escape routes in π, the only states $s'' \in \mathrm{Dom}(\pi)$ that may stop having escape routes in the successor policy π' are the ones with exactly $\mathrm{Out}(\pi, s'') = \{s\}$. In case (1), s and all these in-danger states will have some escape route in π' through s'. In case (2), they will have some escape routes in π' through the states in $\mathrm{Out}(\pi, s')$. □

The **still_no_deadlocks** method (Algorithm 2) visits the states in $T(s,a)$ and verify whether all of them are either s or states $s' \in \mathrm{Dom}(\pi)$ with $\mathrm{Out}(\pi, s') = \{s\}$ through a recursive analysis on successors. If yes, then the inclusion of the mapping $s \mapsto a$ results in a deadlock. Thereby, we can use this method between Algorithm 1's Lines 7 and 8 to early discard policies, and by doing so, we also don't need to assert properness in Algorithm 1's Line 4. In the worst case, it analyzes all the states in $\mathrm{Dom}(\pi')$ one time.

Execution Differences. We recall the execution example presented earlier in this section. We analyze how the use of **still_no_deadlocks** method affects it. When the policies $\pi_2 = \{s_0 \mapsto b\}$ and $\pi_3 = \{s_0 \mapsto c\}$ were generated from $\pi_1 = \emptyset$, the method would successfully assert the existence of escape routes for s_0 in both policies. Through s_{a1} and s_* in π_2. Through s_{b1} in π_3. However, when generating the policy $\pi_4 = \pi_2 \sqcup \{s_{a1} \mapsto a\}$ it would detect the non-existence of escape routes for s_{a1} in π_4. s_{a1} and s_{a2} form a deadlock in π_4. AND* would then discard π_4, and would not have needed to generate the non-proper closed policy π_5. The rest of the execution would follow as previously described. Note that the policy $\pi_6 = \pi_3 \sqcup \{s_{b1} \mapsto a\}$ has escape routes through s_{b2} despite having a cyclic trajectory on s_0 and s_{b1}.

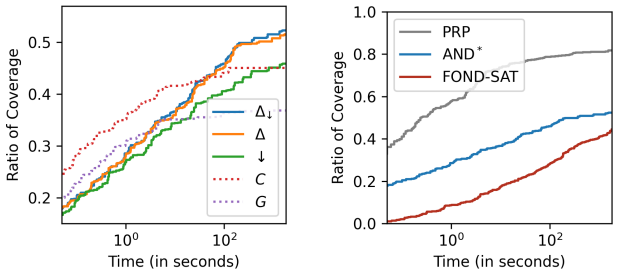
Empirical Analysis

In this section, we empirically compare the different optimal-size FOND heuristics using AND*, studying how the heuristic features impact the AND* performance. We also analyze the impact of the early deadlock detection, and compare our proposed algorithm AND* with other two FOND planners – FOND-SAT (Geffner and Geffner 2018) and PRP (Muise, McIlraith, and Beck 2012) – to provide a reference of its performance and solution compactness.

We make our empirical analysis using the same two benchmarks used by Pereira et al. (2022). One, called IPC-FOND, with 379 tasks over 12 FOND planning domains from the International Planning Competition (IPC),

	G	C	$\downarrow$	Δ	$\Delta_{\downarrow}$
-	8,858.73	3,219.35	-	-	-
$h^{\max}$	-	-	1,746.50	1,246.89	1,022.86
$h^{\text{LM-Cut}}$	-	-	793.17	535.08	460.84
h^*	-	-	650.07	459.08	372.81

Table 1: Heuristics Average Number of Generated Policies.



(a) FOND Heuristics (b) FOND Planners

Figure 5: Ratio of Coverage per Time

and the other, called NEW-FOND, that includes 211 tasks over five FOND planning domains, introduced by Geffner and Geffner (2018). We remove from comparison the tasks without solution, namely 25 tasks from the *first-responders* domain, for which all configurations, except the planner FOND-SAT, detect unsolvability. We merged *blocksworld-new* and *blocksworld-2* domains into a single domain called *blocksworld-advanced*, as they actually are equal domains.

We run our experiments on an AMD Ryzen 9 3900X, using limits of 8 GB and 30 minutes per task. Used code and data are publicly available (Messa and Pereira 2023).

We use greater policy size ($|\pi|$) as the AND* policy-selection tie-breaker, analogously to the use of the greater g-value (lower h-value) tie-breaking on classical planning. We always select to map the most recently generated outgoing non-goal state when expanding a policy. We keep dead-end state detection enabled in all experiments. We keep early deadlock detection enabled in all experiments besides the one that compares its usage impact.

All domains have the same weight in our aggregation averages, independently of the number of (solved) tasks each has. We use the ratio of solved tasks to measure the coverage of the configurations in each domain. We use arithmetic means to aggregate ratios of coverage, and geometric means to aggregate other kinds of information (to avoid giving more weight to hard tasks or domains).

Comparison of Optimal-Size FOND Heuristics

We compare the use of the four presented admissible goal-aware basic FOND heuristics – G, C, $\downarrow$, and Δ – with $\Delta_{\downarrow}$, using three classical planning heuristics as components – $h^{\max}$, $h^{\text{LM-Cut}}$ (Helmert and Domshlak 2009), and h^*.

Table 1 shows the average number of policies generated by AND* when using each heuristic combination. The averages consider only the 112 tasks that are solved by all the combinations simultaneously. Although slower than $h^{\max}$, $h^{\text{LM-Cut}}$ theoretically dominates it, and is dominated by h^*. This is reflected in practice, as the number of generated policies substantially decreases as we improve the classical planning component of the FOND heuristics. Note also that using C is equivalent to using $\Delta_{\downarrow}$ with h^{BLIND}.

We can see that FOND heuristics stronger in theory provide a better guidance in practice. AND* using the heuristic C generates 63.66% fewer policies than using G, and at least 84.33% more than using any of the FOND heuristics with classical planning heuristic information. We can also notice that using Δ is significantly more effective than using $\downarrow$, and that using their combination is the best approach.

Figure 5a shows the average ratio of coverage per time for each FOND heuristic, with $h^{\text{LM-Cut}}$ as the classical heuristic component. We can see that the FOND heuristics that do not use the classical heuristic component start better, because they are faster to be computed, but soon are overwhelmed by the superior quality of the FOND heuristics that use it. At the 30 minutes mark, $\Delta_{\downarrow}$ has 0.523 average ratio of coverage, Δ has 0.515, $\downarrow$ 0.459, C 0.451, and G 0.368.

Impact of Early Deadlock Detection

We analyzed the effect of disabling the early deadlock detection, in the performance of AND*, when using $\Delta_{\downarrow}$ with $h^{\max}$. For most of the domains, there was no significant effect. However, for five domains, disabling the early deadlock detection has a dramatic detrimental effect on the performance of AND*. In *chain-rooms*, the coverage drops from 1.000 to 0.200, and the number of generated policies increases by $4{,}447.00\%$. In *acrobatics*, the coverage drops from 1.000 to 0.500, and the number of generated policies increases by 753.51%. In *earth-observations*, *blocksworld-original*, and *blocksworld-advanced*, the number of generated policies increases by 48.09%, 39.36%, and 26.29%.

Comparison of AND* with other FOND Planners

A comparison of AND* with other FOND planners should provide an empirical reference for the results we have seen so far. However, none of the well-established FOND planners allow us an entirely fair comparison, since all of them are designed for different purposes than AND*. We choose to compare AND* using $\Delta_{\downarrow}$, $h^{\text{LM-Cut}}$ with PRP and FOND-SAT. PRP (Muise, McIlraith, and Beck 2012) is a state-of-the-art satisficing FOND planner that uses re-planning to quickly solve the traditional FOND tasks. FOND-SAT (Geffner and Geffner 2018) in turn is a FOND planner that compiles FOND tasks into SAT instances aiming for a better scalability in tasks with higher degrees of non-determinism.

Since both PRP and FOND-SAT return solutions with partial states (that we call compressed solutions), we can compute their decompressed versions, which are more comparable with the solutions returned by AND*. Figure 6 compares against AND* the sizes of both versions of the solutions returned by PRP and FOND-SAT for each task, using proportionally bigger marks for tasks from domains with fewer tasks. For most domains, the returned solutions of the three algorithms are always of similar size, even if we compare compressed versions against uncompressed ones.

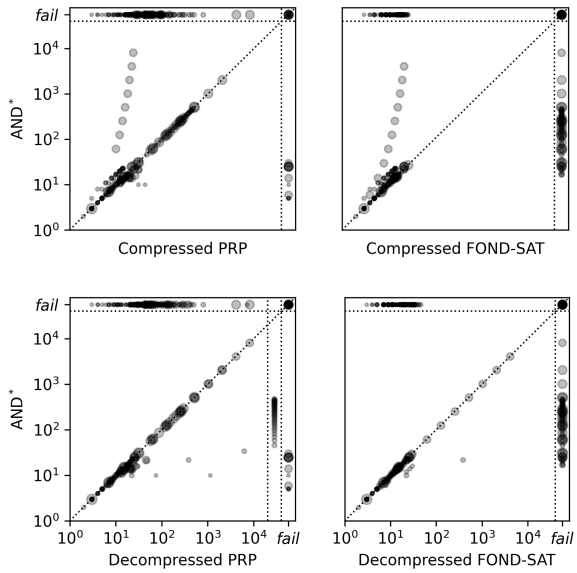


Figure 6: Comparison of Solutions Sizes

Next, any presented information about solutions size takes into account only *non-goal* mapped states and averages only over the tasks solved by all three algorithms simultaneously.

Returned Size. If we compare the uncompressed solution size of solutions returned by AND* against the compressed size of solutions returned by FOND-SAT (which are never larger than the ones returned by PRP for the solved tasks), we notice that their size are equal in ten of the 16 domains! They are similar in three of the six remaining domains: $+7.41\%$ in *blocksworld-advanced*, $+13.64\%$ in *tireworld-spiky*, and $+21.89\%$ in *tireworld-truck*. Whereas, in *tireworld-triangle* AND* maps 45.75% more states, and $+67.28\%$ in *faults*. *Doors* is the only domain for which using partial states is indispensable for scalability. In *doors*, AND* needs exponentially more states ($4 \cdot 2^i - 2$ states against $2 \cdot i + 2$ for the i-th instance).

Uncompressed Size. For 12 of the 16 domains, the average *uncompressed* size is similar: 15.43 for AND*, 15.52 for FOND-SAT, and 15.92 for PRP. The major differences appear on the remaining four domains: *zenotravel*, *miner*, *tireworld-truck*, and *tireworld-triangle*, for which AND* returns solutions with significantly fewer states than PRP (resp. -31.24%, -32.34%, -49.52%, and -94.75%). In *miner* and *tireworld-triangle*, differences also exist in relation to FOND-SAT (resp. -27.37% and -94.75%). In particular, for the i-th instance of *tireworld-triangle*, FOND-SAT and PRP return a compressed solution size with respectively $8 \cdot i - 1$ and $12 \cdot i - 2$ partial states, but both with $\frac{3}{2} \cdot 16^i - 2$ states when decompressed, while AND* returns uncompressed solutions with only $12 \cdot i - 2$ states. Because they get too large when decompressed, a special bin contains the fourth instance onwards of *tireworld-triangle* for *Decompressed PRP* in Figure 6 (FOND-SAT does not solve these). The strategy found by AND* is to always change the car's tires after moving, regardless of outcomes. These results using the decompressed versions of PRP and FOND-SAT solutions do not imply any downside for these planners. They just elucidate the existence of potential gains in AND* solution compactness if partial states were also used by it.

Coverage. With regard to satisfying performance, PRP overwhelms AND* and FOND-SAT in IPC-FOND, solving all the tasks, but loses to FOND-SAT and virtually ties with AND* in NEW-FOND, having a 0.420 average ratio of coverage for NEW-FOND. AND* is better than FOND-SAT in IPC-FOND (0.575 vs. 0.306) and worse in NEW-FOND (0.410 vs. 0.810). Figure 5b shows the average ratio of coverage per time for each algorithm, considering all domains. The results show that AND* is an effective optimal solver.

Discussion

In this work, we step into a different research direction for FOND planning. We examined a policy-space-based way to search for FOND solutions, providing a simple best-first search algorithm for FOND planning with strong theoretical guarantees, called AND*, using as reference the well-established A* algorithm. We also made a systematic study of policy-space search heuristics that factor different aspects of the general structure of FOND solutions to inform the search. With the best of our proposed heuristics, AND* is able to effectively find solutions with the minimal possible number of mapped complete states. We intend to study how to create more informed heuristics. However, this raises the question of why don't we use partial states.

As shown by the experiments, solutions over partial states can be much more compact than solutions over complete states, but this is rare in the current benchmark. And, although there are some strong arguments for using partial-state representations of policies, such as the scalability of planners and the interpretability of generated solutions, classical planning heuristics reason over complete states. Therefore, if we want to use classical planning heuristics, we are initially limited to working with estimates regarding quantities of complete states. Moreover, there are scenarios where optimizing the uncompressed policy is essential. For example, if we need to store any information per complete state (e.g. the expected distance from each to the goal), optimizing over partial states will not be good enough for us because the number of complete states can be exponentially worse in some domains if we decide to optimize over partial states. In such cases, we really need to reason about complete states, and AND* will be useful. Notwithstanding, we believe that studying heuristics that are capable of estimating the number of mapped complete states that remains for an arbitrary policy to become a FOND solution helps to better understand FOND planning, and that our work, as a whole, allows a better understanding of how to design algorithms and heuristics to solve FOND tasks.

As another future work, we intend to analyze how to compress solutions returned by AND* using partial states, and to understand what theoretical guarantees such compressed solutions would have with regard to how far they are from being optimally compressed.

Acknowledgments

We thank UFRGS, CNPq, CAPES, and FAPERGS for partially funding this research. The present work was carried out with the support of CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brazil. We acknowledge support from FAPERGS with project 21/2551-0000741-9. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147: 35–84.

Fu, J.; Ng, V.; Bastani, F.; and Yen, I.-L. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *Proceedings of the International Joint Conference On Artificial Intelligence*, volume 22, 1949–1954.

Geffner, T.; and Geffner, H. 2018. Compact policies for fully observable non-deterministic planning as SAT. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 88–96.

Hansen, E. A.; and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129: 35–62.

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: 100–107.

Helmert, M.; and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, 162–169.

Kissmann, P.; and Edelkamp, S. 2009. Solving fully-observable non-deterministic planning problems via translation into a general game. In *Proceedings of the Annual German Conference on Artificial Intelligence*, volume 32, 1–8.

Kuter, U.; Nau, D.; Reisner, E.; and Goldman, R. P. 2008. Using classical planners to solve nondeterministic planning problems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 18, 190–197.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern database heuristics for fully observable nondeterministic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 20, 105–112.

Messa, F.; and Pereira, A. G. 2023. And-Star-Project. https://doi.org/10.5281/zenodo.7738316.

Muise, C.; McIlraith, S.; and Beck, C. 2012. Improved non-deterministic planning by exploiting state relevance. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 22, 172–180.

Pereira, R. F.; Pereira, A. G.; Messa, F.; and De Giacomo, G. 2022. Iterative Depth-First Search for FOND Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 90–99.

Ramirez, M.; and Sardina, S. 2014. Directed fixed-point regression-based planning for non-deterministic domains. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, 235–243.