Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2025)

Cost-Optimal FOND Planning as Bi-Objective Best-First Search

Diego Aineto, Enrico Scala

Università degli Studi di Brescia
{diego.ainetogarcia,enrico.scala}@unibs.it

Abstract

In this paper, we tackle the problem of finding cost-optimal solutions in Fully-Observable Non-Deterministic (FOND) planning problems. First, we introduce metrics for FOND problems by interpreting solution policies under both their best and worst possible scenarios, leading to a bi-objective optimization problem. We then propose BOAND*, a novel heuristic search algorithm designed to seek Pareto-optimal solutions by navigating the space of possible policies. We conduct an empirical evaluation of the algorithm, alongside a qualitative comparison with cost-optimal solutions that consider only one objective at a time. Our findings validate this approach, paving the way for new methods of reasoning over FOND problems.

Introduction

An intelligent agent needs reasoning capabilities to select actions that achieve specific objectives (Ghallab, Nau, and Traverso 2004). In order to do so, one can provide the agent with a model of the world that expresses, in a predictive manner, how the state will look like after some action is executed. However, for some realistic scenarios, it may be difficult to anticipate the *exact* effect at planning time. So, in order to alleviate this problem, we can adopt planning models that take some uncertainty into account by relaxing the deterministic assumption of our actions. One such formulation is fully-observable non-deterministic (FOND) planning (Cimatti et al. 2003), where each action is associated to a number of possible effects; this way, the agent can take all such different outcomes into account and reason about them. Differently from conformant planning (Goldman and Boddy 1996; Bonet 2010) or POND (Rintanen 2004), in FOND the assumption is that the actual effect will be visible at execution time. That is, the agent will know exactly how the world will look like after the execution of an action; from that point onward she can then decide the proper course of action. A solution to a FOND problem is therefore a policy, that is, a function mapping states to actions. The agent can look-up this policy at run-time and always know what is that it has to do in order to reach the goal. This put FOND in stark contrast with classical planning. Indeed, policies are usually represented as trees or graphs, differently from classical planning where a solution can be encoded as a sequence.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

This paper explores FOND problems where actions have associated costs, aiming for policies that minimize these costs. Unfortunately, constructing such a policy is challenging, as it is indeed unclear how to evaluate the cost of the policy in the first place.

To address this problem, we propose a novel formalization of FOND using a multi-objective search framework. We argue that an intelligent agent should consider contingencies and seek policies providing well-grounded trade-offs. We introduce Bi-Objective FOND planning to find non-dominated policies that form a Pareto frontier, allowing the agent to select a suitable policy. Our Bi-Objective FOND framework can accommodate any two metrics, though we focus on two cases: the cost of the "best" (least expensive) and "worst" (most expensive) execution trajectories within a policy.

Building on recent work in Bi-Objective search and the AND* algorithm for FOND planning, we design a Bi-Objective heuristic search algorithm that we call BOAND*. Given a FOND planning problem with positive action costs, BOAND* returns all cost-unique Pareto-optimal policies. Together with the algorithm, we also formulate admissible heuristics which provide f-values, estimating each policy's cost by considering the current state-action mappings and those that will be required to reach the goal. Our formulation leverages admissible classical planning heuristics over the all-outcome relaxation of FOND.

We theoretically prove the algorithm's soundness, completeness, and optimality. Practically, we conduct an experimental evaluation of BOAND* equipped with the proposed heuristics, showing how they compare against each other and showcasing the value of a bi-objective approach. Our findings demonstrate the feasibility of the approach, and its competitiveness against a single-objective search (embodied by AND*) both in terms of run-time and solution quality.

To illustrate the benefits of reasoning over both best and worst trajectories, we start with a motivating example. Following a necessary background on FOND, we then formalize both cost-optimal and bi-objective FOND, and later present our bi-objective search algorithm along with its theoretical properties. Finally, we conclude with an experimental evaluation and a discussion of related work.

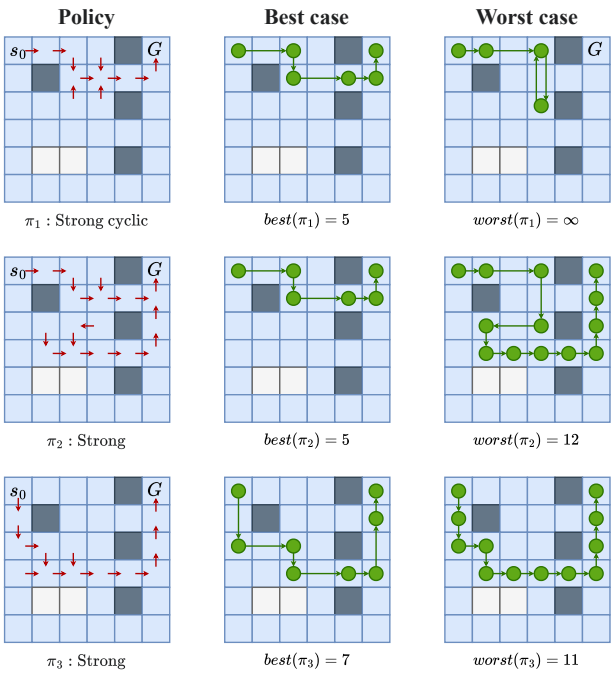


Figure 1: Three solution policies to the working example with their best and worst case trajectories.

Motivating Example

We motivate our proposal through the example of Figure 1 inspired by the well-known FROZENLAKE environment from OpenAI's Gymnasium[1]. In this example, an agent wants to navigate a grid, from an initial position s_0 to a goal position G; movements are possible along the four cardinal directions. However, the tiles of this grid are frozen, so when the agent moves it may slide past the adjacent tile and onto the one after that. For instance, if we are in position $(0, 5)$ (top-left) and we move right, we may end up in either $(1, 5)$ or $(2, 5)$. This grid also features pits where the agent may fall without any way to get out (deadends, colored in grey), and walls that block the agent from sliding (colored in white), e.g., moving down from $(1, 3)$ will always end in $(1, 2)$. Observe that, in order to reach the goal, the agent must pass between the pits using the gaps located in rows 0, 2, and 4. The shortest path goes through the gap in row 4, but taking that route might be difficult since the agent needs to position itself in tiles $(2, 4)$ or $(3, 4)$. On the other hand, going through the gap in rows 0 or 2 takes longer but the agent can use the walls to align itself.

We can model such a domain as a FOND problem; Figure 1 presents three alternative policies, using arrows to denote which action to take in each tile. Now, the question is, which policy should we follow? One possibility is to look at the structure or size of the policy to choose the simpler policy (Fu et al. 2011; Geffner and Geffner 2018a). This approach has merit if the policy needs to be understood by a human, but it is completely blind to the effort required to reach the goal. Alternatively, we can measure this effort, e.g., actions taken to reach the goal, but given the nondeterminism of the domain, we would have to take some random realization of the policies (Mattmüller et al. 2010a). A more systematic approach is to look at the best and worst case of the policy (Shmaryahu, Shani, and Hoffmann 2019), i.e., if everything goes well (or, respectively, wrong) how many actions will the agent need to reach the goal. Let us analyze our policies using this last approach.

Policy π_1 commits to the shortest path passing through the gap in row 4, and keeps trying to align itself with that gap if it fails. Consequently, π_1 has a best case of 5, and a worst case of ∞ since it may always slide when trying to align itself in row 4. π_2 attempts to use the shortest path too but, if it fails, it defaults to using the walls to safely align itself with the gap in row 2. This allows π_2 to maintain a best case of 5 while obtaining a much better worst case of 12. Lastly, π_3 aims directly for the safer row 2 path, resulting in a best case of 7 and a worst case of 11.

Looking at the best case, we would say that both π_1 and π_2 are equivalent, and better solutions than π_3. While, if we consider the worst case, π_3 comes on top followed by π_2 and then π_1. However, one would usually prefer π_2 over π_1 given that π_2 has the same best case and a much better worst case. Indeed, this sort of reasoning is exactly what we find in multi-objective optimisation, where we would say that, under this best and worst case criteria, π_2 *dominates* π_1. At the same time, we cannot really say that π_3 is better or worse than π_1 or π_2, i.e., π_3 neither dominates nor is dominated by the other solutions. Note also that the cyclic case above can be assumed to have an end if we consider fair environments (see below for more details). Yet, it may still be useful to prefer the case with a finite and known solution cost.

FOND Planning

Let V be a set of state variables such that each variable $v \in V$ has a finite domain D_v. An assignment to V is a function s mapping each variable $v \in V$ to $D_v \cup \{\bot\}$. If $\forall v \in V : s(v) \in D_v$, then s is a *complete assignment*; otherwise, if $\exists v \in V : s(v) = \bot$, then s is a *partial assignment*. A complete assignment s is *consistent* with a partial assignment s', denoted as $s \models s'$, when $\forall v \in V$, either $s(v) = s'(v)$ or $s'(v) = \bot$.

A fully-observable non-deterministic (FOND) planning task is a tuple $P = \langle V, A, s_0, G \rangle$ where V is a set of finite-domain *state variables*, A is a set of non-deterministic *actions*, s_0 is the *initial state*, and G is the *goal condition*. A *state*, in particular s_0, is a complete assignment to V and we denote with S the state space; G is a partial assignment to V; each action $a \in A$ is represented by a tuple $\langle \mathsf{pre}_a, \mathsf{Eff}_a \rangle$, where the *precondition* pre_a is a partial assignment to V and Eff_a is a set of possible *effects* such that each effect $\mathsf{eff} \in \mathsf{Eff}_a$ is also a partial assignment to V.

An action $a \in A$ is *applicable* in a state s if $s \models \mathsf{pre}_a$. The application of an action $a \in A$ in a state s causes an effect $\mathsf{eff} \in \mathsf{Eff}_a$ to be triggered, generating the successor state $s' = succ(s, \mathsf{eff})$ such that $s'(v) = \mathsf{eff}(v)$ if

[1] https://gymnasium.farama.org/environments/toy_text/frozen_lake/

$v \in vars(\mathsf{eff})$, and $s'(v) = s(v)$, otherwise. Hence, applying action a in state s can generate the successor states $Succ(s, a) = \{succ(s, \mathsf{eff}) \mid \mathsf{eff} \in \mathsf{Eff}_a\}$.

Let $P = \langle V, A, s_0, G\rangle$ be a FOND planning problem; a *policy* π for P, is formally defined as a partial function $\pi : S \rightharpoonup A$ mapping non-goal states of S into actions such that the action $\pi(s)$ is applicable in the state s. $Dom(\pi)$ is the *domain* of π, i.e., the subset of the state space for which π is defined. $Reach(\pi) = \bigcup_{s \in Dom(\pi)} Succ(s, \pi(s))$ is the set of states reachable by following the policy π. The outgoing states $Out(\pi)$ of a policy π are the reachable states for which π is undefined, i.e., $Reach(\pi) \setminus Dom(\pi)$. We denote with $Out_G(\pi) \subseteq Out(\pi)$ the reachable goal states, and with $Out_\sim(\pi) \subseteq Out(\pi)$ the reachable non-goal states that have no action assigned in π. We assume that $Dom(\pi) \subseteq Reach(\pi) \cup \{s_0\}$, i.e., $Dom(\pi)$ only contains states that are reachable from the initial state s_0.

A policy π for P induces a set of possibly infinite state trajectories $(s_0, s_1, s_2, \ldots)$, called π-trajectories, where $s_{i+1} \in Succ(s_i, \pi(s_i))$ for $i \geq 0$. We denote with $\mathcal{T}(\pi)$ the set of all trajectories starting from s_0 and following π until the end, i.e., all finite trajectories that end with a state in $Out(\pi)$ as well as infinite trajectories. A policy is cyclic if one state can be visited more than once. As the state space is finite, infinite trajectories can only occur when the policy is cyclic. In the literature on FOND we can distinguish fair and unfair environments. Under fairness, all actions outcome will manifest in the limit, so also cyclic policies will terminate in a finite, but unbounded, number of steps (Cimatti et al. 2003).

Definition 1 (Closedness)**.** *A policy π is closed if it is defined for all its reachable nongoal states, i.e., $Out_\sim(\pi) = \emptyset$.*

Definition 2 (Properness)**.** *A policy π is proper if for any state $s \in Reach(\pi)$ there is at least one π-trajectory that contains s and reaches a goal state.*

Definition 3 (FOND Solutions)**.** *Let $P = \langle V, A, s_0, G\rangle$ be a FOND Problem, and π a proper policy for P. If π is acyclic, then it is a* strong *solution for P. If π is cyclic, it provides a solution for P only if the environment is fair. In that case the policy is said to be a* strong cyclic *solution for P.*

Hereinafter, we assume fair environments, so both strong and strong cyclic policies are solutions.

FOND Planning with Costs

Unlike previous works that focus on policy size, we are interested in assessing the value of a policy in terms of the cost of executing it. We use Cost-Optimal FOND Planning to refer in general terms to the problem of computing solution policies for a FOND Planning task that are optimal with respect to a policy-cost metric q. That is, given a metric q that evaluates and total-preorders all the possible solutions of a FOND Planning task P, a solution $\pi \in \Pi(P)$ is *optimal* if $q(\pi)$ is minimal. The purpose of this section is to introduce policy-cost metrics and do so by firstly equipping the FOND problem with an *action cost* function.

Definition 4 (Action cost)**.** *An action costs function $c : A \to \mathbb{N}^+$ assigns to each action a positive cost.*

Assuming hereinafter, that FOND tasks are equipped with cost, i.e., $P = \langle V, A, s_0, G, c\rangle$, we look at how to define policy-cost metrics. Observe that, unlike in classical planning, determining the cost of a policy in a non-deterministic setting depends on the outcome of the executed actions. That is, a policy has many realizations and each one incurs a different cost. The many realizations of a policy π are captured by the notion of π-trajectories, so we use the cost of a π-trajectory as a building block towards our objective of formalizing policy-cost metrics.

Definition 5 (Cost of a π-trajectory)**.** *Given a policy π for a FOND task $\langle V, A, c, s_0, G\rangle$ and an action cost function c, the cost of a π-trajectory $\tau = (s_0, s_1, s_2, \ldots)$ is*

$$cost(\tau) = \sum_{i=0}^{|\tau|-1} c(\pi(s_i)) \tag{1}$$

Building on top of Definition 5, we focus on the *best case* and the *worst case* metrics, representing optimistic and pessimistic interpretations of a policy's cost. These metrics reflect the least and most expensive trajectories, providing lower and upper bounds for the cost of achieving goals under a given policy.

Definition 6 (The Best and Worst Case Metrics)**.** *Let π be a policy. The best case and worst case metrics are defined as follows:*

$$best(\pi) = \min_{\tau \in \mathcal{T}(\pi)} cost(\tau) \tag{2}$$

$$worst(\pi) = \max_{\tau \in \mathcal{T}(\pi)} cost(\tau) \tag{3}$$

As noted by Messa and Pereira (2023), when searching over policy space it is important to have metrics such that extending the policy, i.e., by adding new state-action assignments, does not decrease the evaluation of the policy.

Assumption 1. *Let $\pi \subset \pi'$. It holds that $q(\pi) \leq q(\pi')$*

Our next theorem demonstrates that the best and worst cases satisfy this assumption.

Theorem 1. *The best and worst case are proper metrics.*

Proof (sketch). Let $\pi \subset \pi'$. For every π-trajectory τ there exists a π'-trajectory τ' such that τ is a prefix of τ'. Since action costs are positive, the best and worst case of π' can never be lower than that of π. □

Cost-optimal policies with respect to either the best or the worst case can be found using the AND* algorithm (Messa and Pereira 2023). Yet, as shown in our motivating example, focusing solely on one criterion has limitations. Optimizing for the best case can yield policies with very high worst-case costs, even when there are equally good best-case solutions with much lower worst-case costs. This is exactly what happens in Figure 1 with policies π_1 and π_2. To reason that π_2 is better than π_1, we argue that both best and worst cases should be considered simultaneously.

To enable this more robust reasoning, we formalize the bi-objective FOND Planning problem and extend the AND* algorithm for bi-objective settings in the following sections.

We begin with the general Bi-Objective FOND Planning case and then specialize it to scenarios where the two objectives are the best and worst case metrics.

Bi-Objective FOND Planning

In bi-objective optimization problems, the value of a solution is no longer a single metric but a pair of metrics, i.e., $\mathbf{q}(\pi) = \langle q_1(\pi), q_2(\pi) \rangle$.

Definition 7 (Bi-Objective FOND Planning task)**.** *A Bi-Objective FOND Planning task is a tuple $\langle V, A, s_0, G, c, \boldsymbol{q} \rangle$ where $\langle V, A, s_0, G \rangle$ is a FOND Planning task, c is an action cost function, and $\boldsymbol{q}$ is a bi-objective function.*

In a bi-objective setting, it will often be the case that there does not exist a solution that minimizes both metrics at the same time. Therefore, the aim is to find *Pareto-optimal* solutions, i.e., solutions that cannot be improved w.r.t. to one metric without degrading the other. This is formalized through the notion of (Pareto) *dominance*, which we adapt to our context in the following definition.

Definition 8 (Dominance)**.** *Let π and π' be two policies for a FOND task. We say that π* dominates *π', denoted by $\pi \prec \pi'$, if either $q_1(\pi) < q_1(\pi') \wedge q_2(\pi) \leq q_2(\pi')$ or $q_1(\pi) \leq q_1(\pi') \wedge q_2(\pi) < q_2(\pi')$. We also say that π* weakly dominates *π', denoted $\pi \preceq \pi'$, if $q_1(\pi) \leq q_1(\pi') \wedge q_2(\pi) \leq q_2(\pi')$.*

Observe that for $\pi \prec \pi'$ to hold, either $q_1(\pi) < q_1(\pi')$ or $q_2(\pi) < q_2(\pi')$ must be true, i.e., π must be strictly better in one of the metrics. A Pareto-optimal solution is one that is not dominated by any other solution.

Definition 9 (Pareto Optimal Solution)**.** *Given a bi-objective FOND Planning task P, a solution $\pi \in \Pi(P)$ is Pareto-optimal if for all $\pi' \in \Pi(P)$ different from π, it holds that $\pi' \not\prec \pi$.*

The set of all Pareto-optimal solutions of P is called the *Pareto frontier* and denoted with $PF(P)$. The *Pareto Coverage Set* $PCS(P)$ is a maximal subset of the Pareto frontier where no pair of solutions have the same value.

Best-Worst Case Optimal FOND Planning

Our proposal is to compute cost-optimal solutions by considering the best and the worst case metrics. We instantiate bi-objective FOND Planning with any ordering of these two metrics as bi-objective functions, that is $\mathbf{q}(\pi) = \mathbf{bw}(\pi) = (best(\pi), worst(\pi))$ or $\mathbf{q}(\pi) = \mathbf{wb}(\pi) = (worst(\pi), best(\pi))$.

This framework allows us to follow the reasoning anticipated in our motivating example on more formal grounds. Revisiting Figure 1, we can say that π_2 dominates π_1, since $best(\pi_2) = best(\pi_1)$ and $worst(\pi_2) < worst(\pi_1)$. No other dominance relationship can be established among the three depicted policies. As a matter of fact, both π_2 and π_3 are Pareto-optimal, and all other Pareto-optimal solutions to this task are weakly dominated by π_2, meaning that $\{\pi_2, \pi_3\}$ is a Pareto Coverage Set.

Looking at policies through the lenses of these bi-objective functions, we can make the following observations. A strong policy has best and worst cases that have finite cost, i.e., they are both in $\mathbb{N}$. The same is true for strong cyclic policies since, in a fair environment, all induced π-trajectories will be finite. However, it must be noted that a strong cyclic policy will always have a higher worst case than a strong policy. This is because it is always possible to find a longer finite trajectory by repeating a cycle a few more times. Therefore, we use $\bowtie$ as a symbolic upper bound on the worst case of strong cyclic policies.

Further insights follow from the above observations. An important one is that a strong policy cannot be dominated by a strong cyclic policy. Hence, if a FOND task P admits strong solutions, its Pareto Coverage Set $PCS(P)$ will always contain a strong policy. Conversely, we can also state that if P only admits strong cylic solutions, then $PCS(P)$ will be a singleton, since only one best case cost is minimal.

Computing Pareto Optimal Policies

In state-space search, it is well-known that the A* algorithm with an admissible heuristic produces optimal solutions; this result has been generalized to the multi-objective setting, allowing to compute Pareto-optimal solutions. (Stewart and White III 1991; Mandow and De La Cruz 2008).

Recently, AND*, an A* algorithm that searches over the policy-space instead of the state-space has enabled the computation of optimal solutions for FOND task under any proper metric (Messa and Pereira 2023). In typical best-first search fashion, AND* represents its search frontier through an Open list. Differently from A*, AND* associates to each policy directly an f-value considering both the cost accumulated so far, and a heuristic estimating the remaining cost to become a solution. This is due to the fact that a policy contains many trajectories, using a single estimate f is more practical. In Messa and Pereira (2023) such a cost is the policy size, and the heuristic a lower bound on the remaining states that necessarily need to be added. Policies are inserted into Open alongside their f-values, and at each iteration the policy with the lowest f-value is extracted. If the extracted policy is a solution and the f-values are admissible then it is guaranteed to be optimal. Otherwise, the policy is expanded and the generated successors added to the $Open$ list.

Our algorithm, BOAND*, mirrors the above-mentioned extensions, building on top of AND* and generalizing it to compute a Pareto Coverage Sets for bi-objective FOND Planning tasks. In the next subsections, we detail the algorithm, discuss its properties, and then show how to compute informative and admissible heuristics for the $best$ and $worst$ metrics. Similarly to AND*, a FOND heuristic is admissible if $f(\pi) \leq f^*(\pi)$ for any policy π, and goal aware if $f(\pi_*) = q(\pi_*)$ for all solutions π_*, where $f^*(\pi)$ is the minimum cost that can be obtained by extending π to become a solution. $f^*(\pi) = \infty$ if π cannot become a solution.

The BOAND* Algorithm

Algorithm 1 formalises BOAND*. The search frontier is maintained by the $Open$ list initialized in Line 4 with an empty policy $\pi_I = \emptyset$ such that $Out_\sim(\pi_I) = \{s_0\}$. Each node in $Open$ is a pair $(\mathbf{f}(\pi), \pi)$ consisting of a policy π and

its associated f-value $\mathbf{f}(\pi)$. The f-value is a pair $\langle f_1, f_2 \rangle$ representing an admissible estimates for the two metrics at hand, respectively. The BOAND* takes $\mathbf{f}$ as an input, assuming $\mathbf{f}$ admissible and goal-aware. For instance, if we are using $\mathbf{q} = \mathbf{bw}$, then f_1 is a lower bound on the best cost that may be achieved by the partial policy π.

At each iteration, BOAND* extracts the node from $Open$ with the lexicographically smallest f-value (line 6). This ensures that extracted policies are not dominated by others in $Open$. If the extracted policy is a solution (line 9) and it is not weakly dominated by previously found solutions (line 7), it is Pareto-optimal and added to the solution set (line 11). Unlike AND*, BOAND* computes a Pareto Coverage Set, which consists of multiple Pareto-optimal solutions. Therefore, the algorithm does not terminate after finding a single Pareto-optimal solution and must continue until exhausting the $Open$ list. When the extracted policy is neither a solution nor closed, BOAND* selects a state $s \in Out_{\sim}(\pi)$ and pushes in the $Open$ all child policies obtained by assigning to s each action applicable in s.

Efficient dominance checks. A challenging aspect for BOAND* is to employ dominance checks to 1) validate the Pareto-optimality of found solutions, and 2) prune policies that cannot become part of the solution set. The naive way to implement these is as follows. Every time a solution π_* is found we perform a dominance check against every member of our solution set. If π_* is not dominated, we add it to our solution set and prune from the $Open$ list all nodes dominated by π_*. Instead of doing this, which requires iterating over both the solution set and the $Open$ list, we can fulfill both challenges by performing a single dominance check against the last solution found. This optimization is based on the observation that solutions added to $PCS(P)$ have strictly increasing q_1 values and strictly decreasing q_2 values. The lexicographic order guarantees that any newly extracted solution π_* will have a q_1 value that is at least as large as the last solution's. The only way for π_* not to be weakly dominated is by having a strictly lower q_2 value than any other previously found solution. This also entails that the last solution inserted in the solution set has minimal q_2, so dominance can be checked by simply comparing against the q_2 value of the last solution. To do this, we update q_2^{last} with the q_2 value of every new solution (line 11), assuming that such value is ∞ when $PCS(P)$ is empty (since all solutions will have a lower q_2 value). BOAND* performs the check at extraction time (line 7), pruning any policy (even solutions) that are weakly dominated by the solution set. This optimization matches the latest advancements in bi-objective state-space search (Hernández et al. 2023).

Summarizing, BOAND* works by exploring policy-space in a way to ensure that found solutions cannot be dominated by other yet to be seen solutions. If the solution also happens to not be dominated by those previously encountered, then it must be Pareto-optimal. We discuss its theoretical properties precisely below.

Algorithm 1: BOAND*

Input Bi-objective FOND task $P = \langle V, A, c, s_0, G, \mathbf{q} \rangle$ and admissible heuristics $\mathbf{f}$ for $\mathbf{q}$
Output A Pareto Coverage Set of P

1: $PCS(P) \coloneqq \emptyset$
2: $q_2^{last} \coloneqq \infty$
3: Create empty policy $\pi_I = \emptyset$ with $Out(\pi_I) = \{s_0\}$
4: $Open \coloneqq \{(\mathbf{f}(\pi_I), \pi_I)\}$ ▷ Initialize Open list
5: **while** $Open \neq \emptyset$ **do**
6: Extract node $(\mathbf{f}(\pi), \pi)$ with lowest $\mathbf{f}$-value
7: **if** $q_2^{last} \leq f_2(\pi)$ **then** ▷ Pruning
8: continue
9: **if** π is closed and proper **then** ▷ Solution recording
10: Add π to $PCS(P)$
11: $q_2^{last} \coloneqq f_2(\pi)$
12: **else** ▷ Expansion
13: Select a state s from $Out_{\sim}(\pi)$
14: **for** action a applicable in s **do**
15: $\pi' \coloneqq \pi \cup \{s \to a\}$
16: Insert $(\mathbf{f}(\pi'), \pi')$ in $Open$
return $PCS(P)$

Theoretical Properties

Following, we present the theoretical properties of BOAND*. Some of these properties, such as those regarding its termination, soundness, and completeness are inherited quite directly from AND* (Messa and Pereira 2023), with completeness requiring some adjustment due to the pruning in line 7. On the other hand, AND* and BOAND* vastly differ in their notions of optimality so we dedicate more attention to this aspect.

Lemma 1. *BOAND* terminates and returns a non-empty solution set iff there exists at least one solution policy.*

Proof. The policy space is finite since both S and A are finite, and no policy is generated twice. Hence, BOAND* will always terminate when the Open list is finally exhausted. If BOAND* returns a non-empty $PCS(P)$, all policies in $PCS(P)$ are solution policies, since line 9 prevents any policy that is not strong or strong cyclic from being added to the solution set. Otherwise, if $PCS(P) = \emptyset$, it signifies that the FOND task has no solution. This is because BOAND* systematically explores all successors at expansion time, and there is only one condition that prevents a policy from being expanded, the dominance check of line 7. Since $q_2^{last} = \infty$ initially, this check cannot prevent a policy from being expanded until q_2^{last} has been updated, which occurs after a solution is added to the solution set. □

Lemma 2. *If the f-values are admissible and goal-aware, an extracted solution cannot be dominated by other policies in the Open list.*

Proof. By contradiction, assume that we extract a proper policy π from $Open$, but $Open$ contains a policy π' such that $\pi' \prec \pi$. If π is extracted before π', it means that

$$\begin{aligned} &f_1(\pi) < f_1(\pi') \text{ or} \\ &f_1(\pi) = f_1(\pi') \wedge f_2(\pi) \leq f_2(\pi') \end{aligned} \tag{4}$$

On the other hand, $\pi' \prec \pi$ entails that

$$\begin{aligned} &q_1(\pi') < q_1(\pi) \wedge q_2(\pi') \leq q_2(\pi) \text{ or} \\ &q_1(\pi') \leq q_1(\pi) \wedge q_2(\pi') < q_2(\pi') \end{aligned} \tag{5}$$

Since the heuristic is admissible, it follows that

$$f(\pi) = q(\pi) \text{ and } f(\pi') \leq q(\pi') \tag{6}$$

(equality because π is closed). Substituting (6) in (4), we get

$$\begin{aligned} &q_1(\pi) < q_1(\pi') \text{ or} \\ &q_1(\pi) \leq q_1(\pi') \wedge q_2(\pi) \leq q_2(\pi') \end{aligned} \tag{7}$$

Since there is no solution for the system of inequatilities given by (5) and (7), we can conclude that our initial assumptions are in contradiction. □

Theorem 2. *BOAND* computes a Pareto Coverage Set.*

Proof (sketch). By Lemma 1, BOAND* returns a set of solution policies if one exists. By Lemma 2 and Assumption 1, an extracted solution π_* cannot be dominated by yet-to-be-seen solutions. If π_* is not weakly dominated by previous solutions (dominance check of line 7), then π_* must be Pareto-optimal and have a different cost than other policies in the solution set. Observe that BOAND* adds to the solution set all solutions that pass the dominance check. Hence, BOAND* returns a set of Pareto-optimal solutions that have a unique-cost, and there cannot exist any other Pareto-optimal solution with a different cost, i.e., BOAND* returns a Pareto Coverage Set. □

FOND Heuristics for Policy-Cost Metrics

As hinted at above, in order to use BOAND* to solve Best-Worst Case Optimal FOND problems, we need to equip it with an admissible f-value function that provides an estimate of the cost for both the best and the worst metrics. Following the AND* framework, the f-values that we design account for both the cost of the actions which are already part of the policy and an estimate of the cost of the actions that would be needed to extend the policy to become proper.

f-value for the best case. We start with a trivial heuristic that only uses the cost of the policy. This is akin to only using the g-value to compute the f-value in state-space search, so it can be regarded as a *blind* heuristic.

$$BlindBest(\pi) = best(\pi) \tag{8}$$

Aiming towards building more informed FOND heuristics, observe that, in order for a policy π to become a solution (i.e., to become proper), all its π-trajectories must end in a goal state. Let $h^*(s)$ denote the cost of the least expensive trajectory starting in s and ending in a goal state. We can make the heuristic more informed by considering the cost $h^*(\tau_{|\tau|})$ where $\tau_{|\tau|}$ is the last state of a π-trajectory τ. Of course, h^* is unknown, but an admissible classical planning heuristic h computed upon the all-outcome determinization of the FOND task gives us a lower bound of h^*.

$$SumMin(\pi) = best(\pi) + \min_{\tau \in \mathcal{T}(\pi)} h(\tau_{|\tau|}) \tag{9}$$

$$= \min_{\tau \in \mathcal{T}(\pi)} cost(\tau) + \min_{\tau \in \mathcal{T}(\pi)} h(\tau_{|\tau|}) \tag{10}$$

$SumMin$ is admissible because extending a policy π to become a solution policy π_* will see its best case increased by at least $\min_{\tau \in \mathcal{T}(\pi)} h^*(\tau_{|\tau|})$.

Observe that $\min_{\tau \in \mathcal{T}(\pi)} h^*(\tau_{|\tau|}) = 0$ if there are goal-reaching trajectories in $\mathcal{T}(\pi)$, making $SumMin$ as good as $BestBlind$ in such cases. Our next heuristic, $MinSum$, addresses this weakness.

$$MinSum(\pi) = \min_{\tau \in \mathcal{T}(\pi)} (cost(\tau) + h(\tau_{|\tau|})) \tag{11}$$

To see why $MinSum$ is admissible, observe that by extending π to become a solution π_*, we grow all its trajectories, i.e., $\mathcal{T}(\pi)$ contains a prefix of every trajectory in $\mathcal{T}(\pi_*)$. Moreover, given $\tau \in \mathcal{T}(\pi)$, $cost(\tau) + h^*(\tau_{|\tau|})$ is the lowest cost of any π_* trajectory that starts with τ. Therefore, $MinSum$ is a lower bound of the least costly trajectory that π_* may induce. $MinSum$ dominates $SumMin$ by the minimum of summation and the summation of minimum relationship. It is worth noting that, unlike $SumMin$, $MinSum$ blurs the distinction between the g- and h- component of an f-value, serving to illustrate why FOND heuristics directly reason on the f-values.

f-value for the worst case. Analogously to the best case we have a blind and an informed heuristic. The blind can be formalised as follows:

$$BlindWorst(\pi) = worst(\pi) \tag{12}$$

For the informed heuristic, we follow the same reasoning behind $MinSum$ but applied to the most expensive trajectory:

$$MaxSum(\pi) = \max_{\tau \in \mathcal{T}(\pi)} (cost(\tau) + h(\tau_{|\tau|})) \tag{13}$$

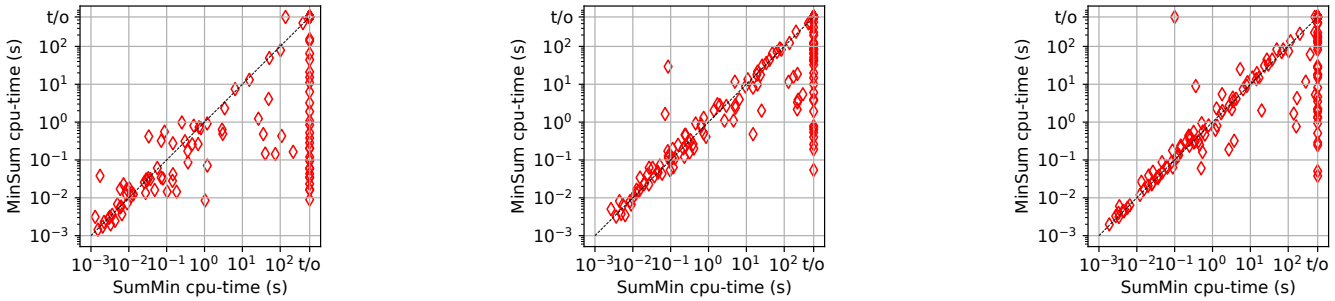
Implementation note. Generating all π-trajectories can be very costly. Instead, it is more efficient to store the cost of the least and most expensive path to every state in $Out(\pi)$, since this set includes the last state of every (finite) π-trajectory. With this method, we can compute our heuristics iterating over $Out(\pi)$ instead of $\mathcal{T}(\pi)$.

Empirical Evaluation

We run experiments to evaluate the performance of the proposed bi-objective FOND Planning framework. Our analysis is motivated by two main questions: how do the proposed policy-cost FOND heuristics perform in single-objective and bi-objective settings? What do we gain by adopting a bi-objective perspective in terms of quality of the plans?

We took a selection of 10 domains from classical FOND benchmarks (Muise 2024; Muise, McIlraith, and Beck 2024; Pereira et al. 2022; Geffner and Geffner 2018b), and added new instances of our ICYGRID domain. The domains are as follows: BEAM-WALK, BLOCKSWORLD-EX, CHAIN-OF-ROOMS, DOORS, ELEVATORS, ISLANDS, FIRST-RESPONDERS, ST_TIRES, TIREWORLD, and TIREWORLD-TRUCK.

Each run of our experiment is given by the instance to be run, the algorithm (AND* or BOAND*, depending on the setting), and the heuristic(s) to be used. For single objective FOND, we can plan either for the best or the worst



(a) AND* best (b) BOAND* best-worst (c) BOAND* worst-best

Figure 2: Pair-wise comparison of policy-cost heuristics in single and bi-objective FOND.

case. For the best case, we have three configurations, i.e., AND* equipped with $BlindBest$, $SumMin$ or $MinSum$. For the worst case, we have $BlindWorst$ and $MaxSum$. In the case of bi-objective FOND, we can plan for best-worst or worst-best metrics, giving us a total of six BOAND* configurations for each orderings of the metrics.

As hinted at above, all these heuristics are obtained by an all-outcome relaxation of the FOND problem, upon which we run the h_{max} heuristic (Bonet and Geffner 2001). We also tried h_{lmcut} (Helmert and Domshlak 2009), but this turned out very inefficient in our setting.

All instances were solved on an Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz running Ubuntu 22.04.1 LTS, with memory and time limits of 8GB and 10 min, respectively. Code and benchmarks are available at https://github.com/daineto/CostFOND.

Heuristic Configuration Analysis

Our first analysis tries to answer the first experimental question, by looking at the performance of the presented FOND heuristics. We focus on informed heuristics, as the coverage of the blind ones is always lower. Hence, we set the worst case heuristic to $MaxSum$ and compare $SumMin$ against $MinSum$ in both single and bi-objective tasks. Figure 2 presents a pair-wise comparison of these heuristics, reporting the cpu-time required to solve an instance.

Single-objective FOND Looking first at the results for single-objective FOND tasks reported in Fig. 2a, we can see that $MinSum$ tends to be faster than $SumMin$ with a few exceptions generally concentrated on the below 1 second range. More importantly, AND* with $MinSum$ is able to solve many more instances as evidenced by all the points in the right edge of the figure.

Bi-Objective FOND For the bi-objective setting, we evaluate BOAND* under both the best-worst (Fig. 2b) and worst-best (Fig. 2c) bi-objective functions. We can observe that the results are very similar in both cases and aligned with our single-objective results. Most points either fall in the diagonal or represent timeouts. There are very few cases where $SumMin$ outperforms $MinSum$ and they tend to correspond to smaller instances where a more informed heuristics may not necessarily pay off.

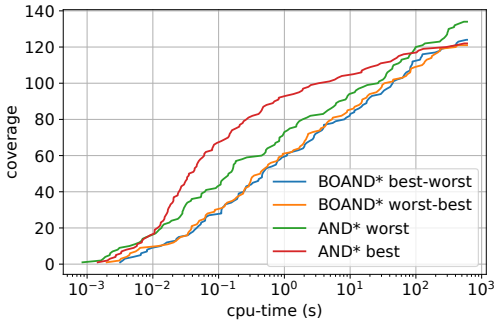


Figure 3: Coverage over time: single vs bi-objective search

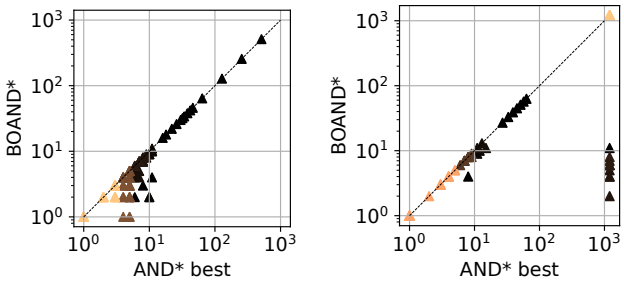
As expected, $MinSum$ displays overall better performance than $SumMin$ in all tested settings. This is likely because, as mentioned before, $SumMin$ dominates $MinSum$.

Single Objective vs Bi-Objective Search

To evaluate the difference between single and bi-objective search, we collect two pieces of information: the coverage and cpu-time spent to find a solution (we hereby focus on the fastest configurations); the cost of the solutions found by the two approaches.

Coverage and time Figure 3 shows a coverage vs time comparison. Unexpectedly, our findings reveal that: BOAND*, in both its best-worst and worst-best configurations, achieves similar coverage than AND*-best and is only slightly behind AND*-worst. A break-down of coverage per domain is reported in Table 1. Indeed, AND*-best solves 122 tasks against the 124 and 121 of the two BOAND* configurations. AND*-worst comes on top with 134 solved instances, obtaining the best coverage in all but three domains.

Quality Comparison In this analysis we use AND* to compute an optimal solution with respect to one metric and evaluate its cost with respect to the other metric. That is, we compute best-case optimal solutions and evaluate their worst case, and vice-versa. To evaluate the secondary metric, we inspect the Pareto Coverage Set returned by BOAND*,



(a) *best* case cost (b) *worst* case cost

Figure 4: Comparison of policy cost

Domain (# of instances)	BO bw	BO wb	SO w	SO b
BEAM-WALK (11)	**8**	7	**8**	**8**
BLOCKSWORLD-EX (15)	9	10	**11**	8
CHAIN-OF-ROOMS (11)	8	7	**10**	2
DOORS (15)	**7**	**7**	**7**	**7**
ELEVATORS (15)	3	3	**6**	3
ICYGRID (30)	28	28	29	**31**
ISLANDS (30)	18	18	**21**	19
FIRST_RESPONDERS (17)	9	10	**12**	10
ST_TIRES (14)	**12**	11	11	11
TIREWORLD (15)	12	10	9	**13**
TIREWORLD-TRUCK (20)	**10**	**10**	**10**	**10**
Total	124	121	**134**	122

Table 1: Coverage of BOAND* (BO) and AND* (SO) across different domains

since it is guarantee to contain a solution that weakly dominates the one found by AND*. Intuitively, we want see how beneficial it is to adopt a Bi-objective perspective. Indeed, AND* ignores the second metric and so can produce solutions which are optimal for the target metric, but very bad for the other, hidden metric. Figure 4 reports such an analysis for the best (left) and the worst (right) case metrics, using lighter colors to denote more density of points. As it is possible to observe, there are several solutions produced by BOAND* having a much lower best-case cost than those produced by AND*-worst. Similarly, for the worst case metric, we observe that many solutions found by AND*-best induce cyclic solutions even in those cases where a strong solution actually exists. This can be seen by inspecting all the dots on the right side of the right sub-figure.

Related Work

The formulation of planning with non-deterministic actions with full observability was firstly introduced by Cimatti et al. (2003) and solved using Binary Decision Diagrams. Since then, a number of methods to tackle the problem computationally have been studied, resulting in a range of FOND planners. PR2, like its predecessor PRP, along with NDP and FIP before them, operates on the all-outcome determinization of the FOND task, addressing the different outcomes of an action through iterative replanning (Muise, McIlraith, and Beck 2024, 2012; Kuter et al. 2008; Fu et al. 2011). MyND explores an AND/OR graph representation of the task (Mattmüller et al. 2010b), while FOND-SAT leverages a SAT encoding (Geffner and Geffner 2018b). Additionally, PALADINUS implements an iterative depth-first search (Pereira et al. 2022), and GRENDEL adopts a backward search approach (Ramirez and Sardina 2014).

Traditionally, policy quality has not been a primary focus and has largely been either ignored completely (e.g., (Muise, McIlraith, and Beck 2012, 2024)) or defined in terms of policy size (Geffner and Geffner 2018b; Messa and Pereira 2023). Recently, the AND* algorithm has emerged as an A* variant that explicitly navigates policy space. In principle, AND* can compute optimal policies under any metric, provided that an admissible heuristic is available; however, it has primarily been examined in relation to policy-size metrics only. Other comparative criteria have been suggested in related literature (Shmaryahu, Shani, and Hoffmann 2019), which can be broadly categorized into syntactic criteria – assessing the structure and size of policies (e.g., simplicity and compactness) – and semantic criteria, which evaluate the cost of achieving the goal when following the policy (considering best, worst, and average cases). To the best of our knowledge, our work represents the first effort to explore these alternative notions of policy quality.

In the realm of multi-objective search, the field has seen significant advancements over the past three decades, leading to the development of various state-space best-first search algorithms. The A* algorithm was initially extended to the multi-objective context with the introduction of MOA* (Stewart and White III 1991). Over time, new insights have enhanced the efficiency of multi-objective search, resulting in variants such as NAMOA* (Mandow and De La Cruz 2008) and NAMOA*dr (Pulido, Mandow, and Pérez-de-la Cruz 2015). Specifically for bi-objective search, BOA* (Hernández et al. 2023) has significantly improved search efficiency through fast dominance checks. Our BOAND* algorithm generalizes AND* in a manner analogous to how BOA* generalizes A*.

Summary and Future Work

In this paper, we studied the problem of planning with full observability and non-deterministic actions in the presence of a cost function. We observed that the use of cost introduces a representational and a computational difficulty. In order to solve the representation problem, we introduce FOND planning with cost as bi-objective search problem, then computationally, we present an optimal algorithm based on Bi-Objective heuristic search. We studied our work both theoretically and practically. As a future work, we look at two different directions. First, we could improve the performance of the algorithm by introducing more powerful heuristics; second, we want to explore our formulation in an oversubscription setting (Keyder and Geffner 2009), so as to make the planner autonomously choose which goals cannot be achieved.

Acknowledgements

This work was supported by project MUR PRIN-2020 project RIPER (n. 20203FFYLK).

References

Bonet, B. 2010. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3-4): 245–269.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2): 5–33.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2): 35–84.

Fu, J.; Ng, V.; Bastani, F. B.; and Yen, I. 2011. Simple and Fast Strong Cyclic Planning for Fully-Observable Nondeterministic Planning Problems. In *IJCAI*, 1949–1954.

Geffner, T.; and Geffner, H. 2018a. Compact policies for fully observable non-deterministic planning as SAT. In *ICAPS*, 88–96.

Geffner, T.; and Geffner, H. 2018b. Compact policies for fully observable non-deterministic planning as SAT. In *ICAPS*, 88–96.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: theory and practice*. Elsevier.

Goldman, R. P.; and Boddy, M. S. 1996. Expressive Planning and Explicit Knowledge. In *AIPS*, 110–117.

Helmert, M.; and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*, 162–169.

Hernández, C.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazo, L.; Koenig, S.; and Salzman, O. 2023. Simple and efficient bi-objective search algorithms via fast dominance checks. *Artificial intelligence*, 314: 103807.

Keyder, E.; and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36: 547–556.

Kuter, U.; Nau, D.; Reisner, E.; and Goldman, R. P. 2008. Using classical planners to solve nondeterministic planning problems. In *ICAPS*, 190–197.

Mandow, L.; and De La Cruz, J. L. P. 2008. Multiobjective A* search with consistent heuristics. *Journal of the ACM (JACM)*, 57(5): 1–25.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010a. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, 105–112.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010b. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, 105–112.

Messa, F.; and Pereira, A. G. 2023. A Best-First Search Algorithm for FOND Planning and Heuristic Functions to Optimize Decompressed Solution Size. In *ICAPS*, 277–285.

Muise, C. 2024. FOND Benchmarks. https://github.com/AI-Planning/fond-domains. [Online; accessed 26-October-2024].

Muise, C.; McIlraith, S.; and Beck, C. 2012. Improved non-deterministic planning by exploiting state relevance. In *ICAPS*, volume 22, 172–180.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2024. PRP Rebooted: Advancing the State of the Art in FOND Planning. In *AAAI*, 20212–20221.

Pereira, R. F.; Pereira, A. G.; Messa, F.; and De Giacomo, G. 2022. Iterative depth-first search for FOND planning. In *ICAPS*, 90–99.

Pulido, F.-J.; Mandow, L.; and Pérez-de-la Cruz, J.-L. 2015. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64: 60–70.

Ramirez, M.; and Sardina, S. 2014. Directed fixed-point regression-based planning for non-deterministic domains. In *ICAPS*, 235–243.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *ICAPS*, 345–354.

Shmaryahu, D.; Shani, G.; and Hoffmann, J. 2019. Comparative criteria for partially observable contingent planning. *Autonomous Agents and Multi-Agent Systems*, 33: 481–517.

Stewart, B. S.; and White III, C. C. 1991. Multiobjective a. *Journal of the ACM (JACM)*, 38(4): 775–814.