Dominance Pruning and Heuristics in Optimal Adversarial Non-Deterministic Planning

Rasmus G. Tollund, Álvaro Torralba

Aalborg University, Aalborg, Denmark
rgt@cs.aau.dk, alto@cs.aau.dk

Abstract

In many planning problems there are non-deterministic actions for which the outcome cannot be fully controlled by the planning agent. For critical tasks, we need to find a strategy that achieves the goal within a predictable time-frame and/or cost. Thus, we consider an adversarial planning setting and compute optimal policies that optimize the worst-case cost to reach the goal.

In this work, we introduce domain-independent optimal heuristic search algorithms for this adversarial setting. To guide the search, we show how to leverage classical planning heuristics by applying single-outcome determinization. We also generalize dominance techniques, that analyse when a state is as good as another, to the non-deterministic setting and apply them to prune the search space. Our experimental analysis shows that both methods greatly help to compute optimal policies across multiple domains.

Introduction

Fully Observable Non-Deterministic (FOND) planning, is a well-established research area that concerns finding a policy to reach the goal in environments where actions taken by the agent may have multiple outcomes (Daniele, Traverso, and Vardi 1999; Cimatti et al. 2003). A common objective is to find strong cyclic policies that guarantee to reach the goal under the fairness assumption (if an outcome is possible infinitely often, then it happens infinitely often). There are plenty of domain-independent approaches to compute such a policy by leveraging a compact description of the environment (Younes and Littman 2004), e.g., based on determinization (Kuter et al. 2008; Muise, McIlraith, and Beck 2012, 2024). However, none of these approaches optimize the cost to reach the goal.

An option to do so, is to find the policy with minimal expected cost to reach the goal (Bertsekas and Tsitsiklis 1991), which has also been extensively studied (Hansen and Zilberstein 2001; Trevizan, Thiébaux, and Haslum 2017; Klößner et al. 2022b,a). However, this still requires to assume fairness, and a known probability distribution over the outcomes of the actions. Furthermore, in critical applications optimizing the expected cost may not be enough. In some cases, we may be dealing with adversarial agents manipulating the outcomes and thus preventing us from ever reaching the goal (e.g., if we are playing a zero-sum game). In other cases, even if the environment is not entirely adversarial, we may want guarantees on what is the worst-case cost to reach the goal. Cyclic policies, such as the ones found by FOND or probabilistic planners have an unbounded worst-case cost, so they are not desirable in such applications.

In this paper, we consider Adversarial FOND, the problem of finding a policy that minimizes the worst-case cost to reach the goal. This means that we do not assume fairness, and instead we want strong acyclic policies, i.e., that guarantee reaching the goal without passing twice over the same state, no matter the non-deterministic outcomes. This setting has barely been studied in the planning literature. The only exception is the work by Bonet and Geffner (2005, 2006), which introduced the Learning Depth-First Search (LDFS) framework; a general algorithm that can be applied to multiple settings and generalizes several classical algorithms such as IDA* for deterministic settings (Korf 1985), or Alpha-Beta search for games (Plaat et al. 1996). They show that Bounded LDFS is able to find optimal policies for the adversarial setting we consider. Out setting is also similar to that of 2-player quantitative reachability games (Brihaye, Bruyere, and Pril 2010). Yet, no domain-independent methods have been proposed to leverage access to a compact description of the environment in this setting.

We fill this gap in two different ways. First, we consider how to compute heuristics in this setting using determinization. Besides the well-known all-outcome determinization, which is often used in probabilistic planning (Keller and Eyerich 2011), we show that in this setting we can use single-outcome determinization to produce more informative, and still admissible, heuristics. Second, we introduce dominance pruning techniques that can reduce the search space by eliminating alternatives that are dominated by others. In classical planning, which is fully observable and deterministic, so-called *dominance analysis* techniques (Hall et al. 2013; Torralba and Hoffmann 2015) have been used to analyse when one state is at least as good as another. We extend those techniques to the adversarial non-deterministic setting and use it to prune the search space. Our results show that both techniques can be used to significantly improve the performance of heuristic search algorithms to find optimal policies.

Background

We start by defining non-deterministic planning tasks, whose semantics are described by non-deterministic labelled transition systems.

Definition 1. *A* non-deterministic labelled transition system *(NDLTS) is a tuple* $\Theta = (S, L_A, L_O, \mathcal{L}, T, s_0, S^G, c)$ *where*

1. S *is a set of states;*
2. L_A, L_O *are sets of action labels and outcome labels, respectively;*
3. $\mathcal{L} : L_A \to (2^{L_O} \setminus \emptyset)$ *is a function mapping action labels to a non-empty set of outcome labels;*
4. $T \subseteq S \times L_A \times 2^{L_O \times S}$ *is a set of transitions, satisfying for each* $(s, \ell, O) \in T$ *:*
   (a) $\forall (s, \ell, O') \in T : O = O'$*, and*
   (b) $\forall o \in \mathcal{L}(\ell) : \exists! (o, s) \in O$*;*
5. $s_0 \in S$ *is an initial state;*
6. $S^G \subseteq S$ *is a set of goal states;*
7. $c : L_A \cup L_O \to \mathbb{N}$ *is a cost function over the labels, satisfying*
   (a) $\forall \ell \in L_A : \forall o \in \mathcal{L}(\ell) : c(\ell) + c(o) > 0$

The justifications for the restrictions are as follows. 4a ensures that there is at most one transition with a given action label per state. This simplifies some definitions and avoids confusion with different types of non-determinism. 4b similarly ensures that there is exactly one outcome for each outcome label associated to the action label in a transition. 7a ensures that there are no zero cost transitions. This is not a strictly necessary requirement as long as there are no zero-cost cycles, but it also makes the definitions easier. However, it requires non-trivial changes to allow for zero-cost cycles.

We write $s \Rightarrow_\ell O$ whenever $(s, \ell, O) \in T$ and further $s \to_{\ell,o} s'$ when $(o, s') \in O$. We write $s \not\Rightarrow$ when there exists no transition from s.

The *synchronized product* of two non-deterministic labelled transition systems over the same labels and cost function is $\Theta \otimes \Theta' = (S \times S', L_A, L_O, \mathcal{L}, T'', (s_0, s'_0), S^G \times S^{G\prime}, c)$, where $((s, s'), \ell, O'') \in T''$ iff $\exists (s, \ell, O) \in T, (s', \ell, O') \in T' : O'' = \{ (o, (s'', s''')) \mid (o, s'') \in O \wedge (o, s''') \in O' \}$. Intuitively, the transitions in each transition system need to match both on the action label and the outcome labels.

Definition 2. *A* factored non-deterministic planning task *(FNDPT) is a tuple of non-deterministic labelled transition systems* $\mathcal{T} = (\Theta_1, \ldots, \Theta_n)$ *over the same labels* L_A, L_O *and cost function* c*. The state-space of a planning task is given by the synchronized product of all transition systems,* $\Theta_\mathcal{T} = \Theta_1 \otimes \cdots \otimes \Theta_n$*. A state of the state-space thus corresponds to a tuple of states for each factor, i.e.* $(s_1, \ldots, s_n)$*.*

An example FNDPT can be seen in Figure 1. We choose this formalism as it is a better fit for the dominance analysis. It has been previously used in classical planning (Helmert et al. 2014; Torralba and Sievers 2019; Büchner et al. 2024) and for stochastic shortest paths problems (Klößner et al. 2023). Often in FOND planning a SAS+ FOND task representation is used (Muise, McIlraith, and Beck 2012). This representation is easily translatable into a FNDPT.

Definition 3. *A* strategy *is a function* $\pi : S \rightharpoonup T$ *that maps states to a transition that is applicable in that state. A strategy is* complete *if it is defined for all states and* partial *otherwise. A strategy is* closed *for* s *if it is defined for* s *and for any state reachable from* s *under the strategy.*

A strategy is *strong* if all choices of non-deterministic outcomes lead to a goal state. In this paper we consider adversarial non-determinism. By this, we consider that the outcomes are chosen by an adversarial player that tries to maximize the cost of reaching a goal state. Therefore, the non-determinism is unfair and thus if any sequence of outcomes results in a cycle, the strategy will not guarantee to reach the goal. We therefore require acyclic strategies.

The solution to adversarial non-deterministic planning tasks are described by the Bellman equation (Bellman 1957)

$$V(s) = \begin{cases} 0 & \text{if } s \in S^G \\ \infty & \text{if } s \not\Rightarrow \\ \min_{s \Rightarrow_\ell O} c(\ell) + Q_V(O) & \text{otherwise} \end{cases},$$

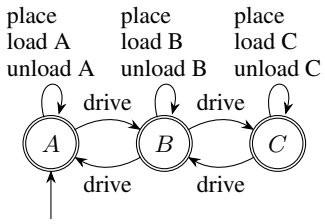
where $Q_V(O) = \max_{(o,s') \in O} c(o) + V(s')$. There is a unique minimal solution to this equation V^* (allowing for ∞ value). An optimal solution to our problem is a strong closed acyclic strategy that achieves this value for the initial state. The value of a strategy $V^\pi(s)$ is given by replacing the $\min_{s \Rightarrow_\ell O}$ operator by the choice of the policy, and ∞ when there is no (applicable) choice defined.

Observe that, for a policy to be optimal, it is enough to achieve the optimal value for the initial state. This means we do not compute an optimal solution for all reachable states, only that the cost of any run does not exceed the value of the initial state. The same holds for other algorithms, like alpha-beta pruning for the minimax algorithm (Russell and Norvig 2020). This can potentially save considerable search effort when concerned about the worst-case. Sometimes, the worst-case cost must be paid in any case, and thus not important to be optimal in non-worst-cases. It can also be used to determine the required worst-case cost, so that you can guarantee to be within this cost, and then afterwards try to improve the remaining strategy as much as possible within the allowed time, or as part of an online planning procedure.

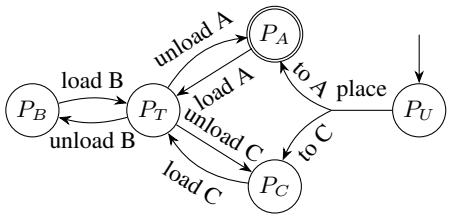
The BLDFS algorithm can be used to solve this problem (Bonet and Geffner 2005, 2006). The algorithm works by maintaining for each seen state a lower bound $V(s)$ and an upper bound $U(s)$ for the true value $V^*(s)$. It then performs a depth-first search from the initial state within the bound $V(s_0)$, attempting to find a solution within that bound. During backtracking, it updates either the upper or lower bound for each state. If no solution is found within the bound, $V(s_0)$ is increased to at least $V(s_0) + 1$. Eventually $V(s_0) = V^*(s_0)$ and an optimal solution will be found.

Heuristics for Adversarial Planning

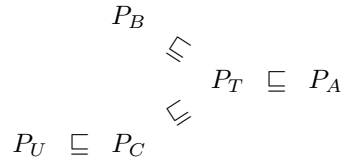
To our knowledge, no research has been done on heuristics for this type of planning task. We propose to use single outcome determinization to utilize deterministic heuristics. Let $d : L_A \to L_O$ be a determinization function, mapping each action label to a single outcome, s.t. $d(\ell) \in \mathcal{L}(\ell)$. This induces the labelled transition system Θ^d over the labels



(a) Truck factor.



(b) Package factor. The 'drive' action is not shown, but it is a self loop everywhere.



(c) Dominance hierarchy of Package factor. For Truck there is no dominance. The transitive closure is the full dominance, e.g. $P_U \sqsubseteq P_A$.

Figure 1: Planning task (1a and 1b) of a logistics example with a truck that has to deliver a package from either location A or C to location A. The 'place' action has two outcomes, placing the package either at A or C. In the truck factor these outcomes have the same target, so we do not show them explicitly. In 1c the label-dominance relation for the package factor is shown.

$L = \{ (\ell, d(\ell)) \mid \ell \in L_A \}$ and with $c((\ell, o)) = c(\ell) + c(o)$. We can then use any classical planning heuristic on Θ^d to obtain heuristic values for Θ.

Theorem 1. *Let Θ be a NDLTS, $d : L_A \to L_O$ be a determinization function and $h : S \to \mathbb{N}_0$ be an admissible classical planning heuristic for Θ^d. h is also an admissible heuristic for Θ.*

Proof. Let $V^*_{\Theta^d}(s)$ be the optimal goal distance for Θ^d. Since h is admissible for Θ^d, $h(s) \leq V^*_{\Theta^d}(s)$. We show that $V^*_{\Theta^d}(s) \leq V^*_{\Theta}(s)$. By definition, Q_V always selects the maximum outcome, therefore, removing outcomes cannot increase the Q_V value, which means V^* cannot increase. □

A heuristic is consistent, also called monotone, whenever $\forall s \Rightarrow_\ell O : h(s) \leq c(\ell) + \max_{(o,s') \in O} c(o) + h(s')$.

Theorem 2. *Let Θ be an NDLTS, $d : L_A \to L_O$ a determinization function and $h : S \times \mathbb{N}_0$ a consistent classical planning heuristic for Θ^d. h is a consistent heuristic for Θ.*

Proof. Let $d(\ell) = o$ and $(o, s') \in O$, then $c(\ell) + c(o) + h(s') \leq c(\ell) + \max_{(o',s'') \in O} c(o') + h(s'')$. □

Importantly, the determinization distributes over synchronized product, and thus we can do the determinization on a per-factor basis for the planning task.

Adversarial Dominance

A dominance relation is a relation over states that describes when one state is better than another in terms of V^*-value.

Definition 4. *A* dominance relation *is a state relation $\sqsubseteq \subseteq S \times S$ such that $s \sqsubseteq t$ only if $V^*(t) \leq V^*(s)$.*

A dominance relation is a very general structure, and computing the maximum dominance relation requires solving the planning task. Therefore, we consider an approximate inference method based on state-space analysis using simulation relations. The core idea is that $s \sqsubseteq t$ if t can perform plans that are at least as good as the ones s can perform. Similar simulation relations have been used in reduction of non-deterministic Büchi automata (Fritz and Wilke 2005).

The intuition is that a state t simulates s, if for every transition available from s, t has a corresponding transition that is *at least as good*. Additionally, a non-goal state cannot dominate a goal state. Since the outcomes of the transitions are not controlled, we need that all outcomes of the t-transition have an outcome of the s-transition that it is at least as good as. This ensures that the worst outcome of the t-transition is at least as good as the worst outcome of the s-transition.

Definition 5. *Let $R \subseteq S \times S$ be a state relation and $\preceq \subseteq (L_A \times L_O) \times (L_A \times L_O)$ be a label-outcome-pair relation. R is a $\preceq$-game-simulation relation if whenever $s \ R \ t$ then (i) $s \notin S^G \vee t \in S^G$ and (ii)*

$$\forall s \Rightarrow_\ell O : \exists t \Rightarrow_{\ell'} O' : \forall (o', t') \in O' : \exists (o, s') \in O : (\ell, o) \preceq (\ell', o') \wedge s' \ R \ t'$$

The label-outcome-pair relation describes which combinations of action labels and outcome labels we can allow substituting for each other. When considering the entire state space, we can use the cost relation, i.e. $(\ell, o) \preceq_c (\ell', o') \iff c(\ell') + c(o') \leq c(\ell) + c(o)$. This simulation ensures that t has a strategy that is at least as good as the one of s. When all outcomes of a transition t_a dominate one outcome of t_b, we say that the transition t_a dominates t_b.

Theorem 3. *Let $\preceq$ be a label-outcome-pair relation s.t. $\preceq \subseteq \preceq_c$ and $\sqsubseteq$ be a $\preceq$-game-simulation relation. Then, $\sqsubseteq$ is a dominance relation.*

Proof. We show that $s \sqsubseteq t$ implies $V^*(t) \leq V^*(s)$. Let π be a complete optimal strategy. If $V^*(s) = \infty$, there is nothing to prove. If $V^*(s)$ is finite, we prove by strong induction on $V^*(s) = n$. The induction hypothesis is $P(n) = \forall s \in S : V^*(s) \leq n \implies (s \sqsubseteq t \implies V^*(t) \leq V^*(s))$. *Base case ($P(0)$):* s is a goal state, and by (i) of Definition 5 so is t, thus $0 = V^*(t) \leq V^*(s) = 0$. *Inductive case ($P(n) \Rightarrow P(n+1)$):* Let $s \in S$ be any state with $V^*(s) \leq n+1$. If $V^*(s) \leq n$ then $P(n)$ already covers it. If $V^*(s) = n+1$ we apply (ii) of Definition 5. Let $\pi(s) = s \Rightarrow_\ell O$, we then know there exists $t \Rightarrow_{\ell'} O'$ with $(o', t') \in O'$ that maximizes $c(o') + V^*(t')$ and there exists $(o, s') \in O$ s.t. $(\ell, o) \preceq (\ell', o')$ and $s' \sqsubseteq t'$. Since $c(\ell)+c(o) \geq 1$, $V^*(s') \leq n$ and by $P(n)$ we have $V^*(t') \leq V^*(s')$. Since $\preceq \subseteq \preceq_c$, we have $V^*(t') + c(\ell') + c(o') \leq V^*(s') + c(\ell) + c(o)$, and thus $c(\ell') + Q^*_V(O') \leq c(\ell) + Q^*_V(O)$ and since π is optimal, we have $V^*(t) \leq V^*(s)$. □

There exists a unique maximum $\preceq$-game-simulation for an NDLTS, and it can be computed in polynomial time. However, computing this on the entire state-space is infeasible. The state-space of the planning task is exponential in the size of the planning task. Thus, we consider approximations which are computed on the factors of the planning task.

Factored Computation

We want to compute dominance relations on the factors of the planning task in such a way that they compose into a dominance relation for the entire state-space. The main challenge is dealing with the effects of actions in the other factors, i.e. reasoning about whether applying an action will also lead to a better state in the other factors, without knowing the exact state in those factors.

Definition 6. *Let $\sqsubseteq$ be a dominance relation for Θ. A label-outcome-pair (ℓ', o') dominates another (ℓ, o) in Θ if $(\ell, o) \preceq_c (\ell', o')$ and $\forall s \rightarrow_{\ell,o} s' : \exists s \rightarrow_{\ell',o'} s'' : s' \sqsubseteq s''$*

Intuitively, a label-outcome-pair dominates another in a factor if in all states where the other is applicable this is also applicable and leads to a just as good state. The point being, in this factor this label-outcome is just as good. We say $\preceq$ is a label-outcome-pair dominance relation w.r.t. $\sqsubseteq$ if $(\ell, o) \preceq (\ell', o')$ only if (ℓ', o') dominates (ℓ, o) w.r.t. $\sqsubseteq$.

Consider a planning task $\mathcal{T} = (\Theta_1, \ldots, \Theta_n)$. When computing dominance in a factor, we are interested in label-outcome-pair dominance for all other factors. Therefore, we denote by $\preceq_i$ the label-outcome-pair dominance relation for factor i and by $\preceq_{\bar{i}} = \preceq_c \cap \bigcap_{j \neq i} \preceq_j$ the intersection of label-outcome-pair dominance of all other factors than i. We denote by $\sqsubseteq_i$ a dominance relation for factor i. We call a tuple of $\preceq_{\bar{i}}$-dominance relations for each factor a factored dominance relation and denote it by $(\sqsubseteq_i)_{i=1}^n$. We define the composition of two relations $R \subseteq S \times S$ and $R' \subseteq S' \times S'$ as $((s, s'), (t, t')) \in (R \otimes R')$ iff $(s, t) \in R$ and $(s', t') \in R'$.

Theorem 4. *Let $\mathcal{T} = (\Theta_1, \ldots, \Theta_n)$ be an adversarial planning task, $\sqsubseteq_1, \ldots, \sqsubseteq_n$ and $\preceq_1, \ldots, \preceq_n$ be relations s.t. $\sqsubseteq_i$ is a $\preceq_{\bar{i}}$-game-simulation relation and $\preceq_i$ is a label-outcome-pair dominance relation w.r.t. to $\sqsubseteq_i$. The relation $\sqsubseteq_1 \otimes \cdots \otimes \sqsubseteq_n$ is a dominance relation for $\mathcal{T}$.*

Proof. We prove this by induction on the composed relations so far. Let $1 \leq k \leq |\mathcal{T}|$ be the number of factors we have composed so far. Let $\Theta_{1,\ldots,k} = \Theta_1 \otimes \cdots \otimes \Theta_k$, $\sqsubseteq_{1,\ldots,k} = \sqsubseteq_1 \otimes \cdots \otimes \sqsubseteq_k$, and $\preceq_{\overline{1,\ldots,k}} = \preceq_c \cap \bigcap_{i>k} \preceq_i$. The induction hypothesis is: $\sqsubseteq_{1,\ldots,k}$ is a $\preceq_{\overline{1,\ldots,k}}$-game-simulation for $\Theta_{1,\ldots,k}$. *Base case ($k = 1$):* Trivial, since $\preceq_{\bar{1}} \subseteq \preceq_c$. *Inductive case:* In the following, we will use the shorthand $\alpha = 1, \ldots, k$ and $\beta = k + 1$. We show that $\sqsubseteq_{\alpha,\beta}$ is a $\preceq_{\overline{\alpha,\beta}}$-game-simulation for $\Theta_{\alpha,\beta}$ according to Definition 5. For (i), since a state is only a goal state if it is in all factors, this is preserved. For (ii), let $s \sqsubseteq_{\alpha,\beta} t$ and $s = (s_\alpha, s_\beta), t = (t_\alpha, t_\beta)$. In Figure 2 a visualization of this part of the proof is shown. The numbers refer to different parts of the visualization. We must show that for all transitions of s there is a transition of t that dominates it. (1) $s \Rightarrow_\ell O$ implies that $s_\alpha \Rightarrow_\ell O_{s_\alpha}$ and $s_\beta \Rightarrow_\ell O_{s_\beta}$. From the induction hypothesis, and by (ii) of Definition 5, this

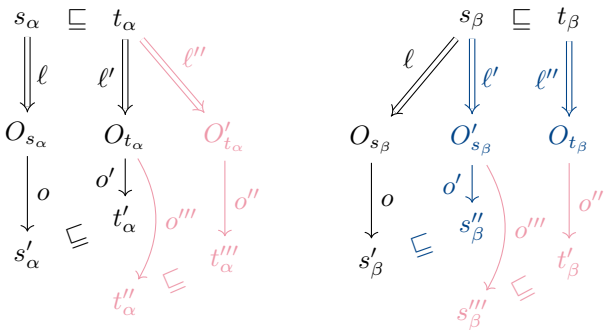


Figure 2: Proof diagram for Theorem 4. The states are named as in the proof. The diagram is coloured s.t. black is part (1), blue is part (2) and red is part (3).

means there exists $t_\alpha \Rightarrow_{\ell'} O_{t_\alpha}$ that is better in Θ_α. Since $t_\alpha \Rightarrow_{\ell'} O_{t_\alpha}$ is at least as good, there exists some $(o, s'_\alpha) \in O_{s_\alpha}$ for all $(o', t'_\alpha) \in O$ s.t. $s'_\alpha \sqsubseteq t'_\alpha$. Now, (2) since $\sqsubseteq_\alpha$ is a $\preceq_{\bar{\alpha}}$-game-simulation, this means that (ℓ', o') dominates (ℓ, o) in Θ_β. Therefore, s_β has a transition with label ℓ', $s_\beta \Rightarrow_{\ell'} O'_{s_\beta}$, and the outcome o' dominates the ℓ transition with outcome o. I.e., $(o', s''_\beta) \in O'_{s_\beta}$ and $(o, s'_\beta) \in O_{s_\beta}$ and $s'_\beta \sqsubseteq s''_\beta$. Outcomes of an action are always the same, so all outcomes of O'_{s_β} dominate an outcome of O_{s_β}. Now we apply the game-simulation again. Since $s_\beta \sqsubseteq t_\beta$ there must also be a transition $t_\beta \Rightarrow_{\ell''} O_{t_\beta}$, with some third label ℓ'', s.t. all outcomes dominate one outcome of O'_{s_β}. (3) For every outcome (o'', t'_β) of O_{t_β} there is an outcome (o''', s'''_β) of O'_{s_β} that it dominates. By label-dominance we know that in the other factors, every time there is an ℓ' transition there is an ℓ'' transition, $t_\alpha \Rightarrow_{\ell''} O'_{t_\alpha}$, in the same state where the o'' outcome dominates the o''' outcome. Since all outcomes of O_{t_α} dominate some outcome of O_{s_α}, by transitivity, also all outcomes of O'_{t_α} dominate an outcome of O_{s_α}. Finally, this means that there is a transition $t_{\alpha,\beta} \Rightarrow_{\ell''} O'$, in the product system that dominates $s_{\alpha,\beta} \Rightarrow_\ell O$, where $\forall (o', t'_{\alpha,\beta}) \in O' : \exists (o, s'_{\alpha,\beta}) \in O : s'_{\alpha,\beta} \sqsubseteq t'_{\alpha,\beta}$, and that (ℓ'', o') dominates (ℓ, o) in all factors greater than $k+1$. □

In addition, we modify the planning task—when computing dominance—with a *noop* action. This action has cost 0, one outcome and is a self-loop in all states. Clearly, this does not affect the solution to the planning task, but it improves the dominance analysis because it allows a state to not have a response to a transition if it is already better than the target. Importantly, this modification is only used for computing dominance, and thus does not violate the assumption that the search problem has no zero cost transitions. Consider Figure 1b. Here, $P_T \sqsubseteq P_A$ because no matter the transition of P_T, P_A can simply choose *noop* and remain in the good state. Without *noop* P_A would not have a response to $P_T \rightarrow_{\text{unload A}} P_A$, because it could only go to P_T, which is worse, or drive, which is not always better in the other factor. Figure 1c shows a table of the label-dominance relation for the Package factor.

It now remains how to compute these relations. We define an update function $\delta((\sqsubseteq_i)_{i=1}^n) = (\sqsubseteq'_i)_{i=1}^n$, where $n = |\mathcal{T}|$, that removes for each factor i the pairs of $\sqsubseteq_i$ that do not satisfy the condition in Definition 5 based on $\preceq_{\bar{i}}$. Since $\preceq_i$ is defined by $\sqsubseteq_i$ in Definition 6, we can recompute this after each update to $\sqsubseteq_i$.

Theorem 5. *A unique maximum factored dominance relation $(\sqsubseteq_i)_{i=1}^n$ exists and it is the maximum fixed point of δ.*

Proof. The set of factored dominance relations is exactly the set of fixed points of δ. The function δ is monotone w.r.t. the point-wise subset inclusion order on the dominance relations, i.e. $(\sqsubseteq_i)_{i=1}^n \leq (\sqsubseteq'_i)_{i=1}^n$ iff $\forall 1 \leq i \leq n : \sqsubseteq_i \subseteq \sqsubseteq'_i$. Furthermore, the set of dominance relations with this ordering forms a finite complete lattice with the bottom element $(\emptyset)_{i=1}^n$ and the top element $\top = (S_i \times S_i)_{i=1}^n$. By the Knaster-Tarski fixed point theorem (Tarski 1955), this means that there exists a unique maximum fixed point of δ and that it is the equal to $(\sqsubseteq^*_i)_{i=1}^n = \lim_{k\to\infty} \delta^k(\top)$. Since the lattice is finite, there exists $k \in \mathbb{N}$ s.t. $(\sqsubseteq^*_i)_{i=1}^n = \delta^k(\top)$. □

The algorithm for computing the factored dominance relation is simply to start with all state-pairs in the relation, and then refining it until no more pairs can be removed. Since there is a number of state-pairs quadratic in the number of states in the factors, it can require at most a quadratic number of iterations. Checking each state-pair can be done in polynomial time, and thus the algorithm runs in polynomial time in the size of the planning task.

Pruning

We consider 4 different methods to prune the search space using dominance relations. We describe the first 3 methods by conditions for when transition $s \Rightarrow_\ell O$ can be pruned,

$$\mathbf{PruneSource} : \exists (o, s') \in O : s' \sqsubseteq s$$

$$\mathbf{PruneInit} : \exists (o, s') \in O : s' \sqsubseteq s_0$$

$$\mathbf{PruneAction} : \exists s \Rightarrow_{\ell'} O' \in T(s) : \forall (o', s'') \in O' :$$
$$\exists (o, s') \in O : c(\ell') + c(o') \leq c(\ell) + c(o) \wedge s' \sqsubseteq s''$$

PruneSource prunes a transition if any of its outcomes are dominated by the source state. **PruneInit** if any of its outcomes are dominated by the initial state. And **PruneAction** if another transition dominates it.

The last method does not prune the entire transition but only outcomes. Therefore, we describe with a condition for when an outcome $(o, s') \in O$ can be removed,

$$\mathbf{PruneOutcome} : \exists (o', s'') \in O : c(o) \leq c(o') \wedge s'' \sqsubseteq s'$$

This method is quite different from the others as it prunes the options of non-determinism. This means that in the pruned transition system a closed optimal policy may not be closed for the original transition system. For example, in Figure 1 we see that the outcomes of *place* in the *package* factor are P_A and P_C. Since $P_C \sqsubseteq P_A$, we can prune the outcome (to A, P_A). Now, in this case the outcome is already a goal, so no more action is needed, but in general this could result in no longer having a strategy for that state. We show later how to construct a closed strategy in polynomial time.

Let **Pruning** be a pruning method and Θ' the pruned state space it induces on Θ after removing a single transition/outcome according to **Pruning**. We say that a pruning strategy is optimal-value preserving if $V^*_\Theta(s) = V^*_{\Theta'}(s)$ for all states s reached by following some optimal policy.

Theorem 6. *The pruning methods* **PruneSource**, **PruneInit**, **PruneAction** *and* **PruneOutcome** *are all optimal-value preserving.*

Proof. Let Θ be an NDLTS and Θ' be the pruned NDLTS with the corresponding method. We prove that the V^*-value is preserved for all states when removing a transition/outcome of transition $s \Rightarrow_\ell O$ with $(o, s') \in O$. **PruneSource**: Since $s' \sqsubseteq s \Rightarrow V^*_\Theta(s) \leq V^*_\Theta(s')$, otherwise $V^*_\Theta(s) \leq Q_V(O) + c(\ell)$ and since $c(\ell) > 0$, $V^*_\Theta < Q_{V^*_\Theta}(O)$. Therefore, either $s \in S^G$ or there is a better transition. **PruneInit**: We know $V^*(s_0) \leq V^*(s')$ and thus this state cannot possibly be part of an optimal solution, since we do not consider 0-cost actions. **PruneAction**: We know that there is another transition $s \Rightarrow_{\ell'} O'$ that is better. Let $(o', s'') \in O$ be the maximum outcome of that transition, then there is $(o, s') \in O$ s.t. $V^*(s'') \leq V^*(s')$ and $c(\ell') + c(o') \leq c(\ell) + c(o)$, therefore, $c(\ell') + c(o') + V^*(s'') \leq c(\ell) + \max_{(o,s')\in O} c(o) + V^*(s')$, which means $V^*(s)$ will not change by removing $s \Rightarrow_\ell O$. **PruneOutcome**: In this case, we do not remove $s \Rightarrow_\ell O$ but instead remove outcomes from O, resulting in the transition $s \Rightarrow_\ell O'$. Let $(o, s') \in O$ be the outcome that maximizes $c(o) + V^*(s)$ that is not in O'. Then there exists $(o', s'') \in O'$ and $c(o) + V^*(s') \leq c(o') + V^*(s'')$, thus (o, s') is not the maximum outcome of O and $Q_{V^*_\Theta}(O) = Q_{V^*_\Theta}(O')$. □

Corollary 6.1. *Pruning several transitions/outcomes with the pruning methods is optimal-value preserving.*

Proof. The pruning method relies on the dominance relation for Θ, therefore we need to show that this is also a dominance relation for Θ'. Looking at Definition 4, this is guaranteed by the fact that the pruning methods are optimal-value preserving. □

The order in which the pruning is performed matters, since **PruneAction** requires some other just as good action to be left and **PruneOutcome** some other no better outcome. If the transitions/outcomes are equally good, i.e. they dominate each other, it is not clear which to remove. In practice, we just remove the first in the order they appear.

Reconstructable Strategies

In this paper, we desire to reduce the search effort by pruning parts of the state-space. Computing complete optimal strategies requires having a strategy from every state in the state-space, which is infeasible. Therefore, we consider closed strategies, i.e. strategies that are defined for the initial state and for all states reachable under the strategy. In this setting, we can prune the state-space as long as it leaves an optimal closed strategy. However, even a full closed strategy can be bigger than wanted. Therefore, we also consider more compact representations, where we only require that the strategy for a state can be computed in polynomial time.

We call a sequence of action-outcome labels $((\ell_i, o_i))_{i=1}^m$ a *trace* of s_0, if there exist $s_{i-1} \rightarrow_{\ell_i,o_i} s_i$ for all $1 \leq i \leq m$. We say the trace *ends* in s_m. A trace under a strategy is a trace where the transitions agree with the strategy.

Definition 7. *An algorithm A is a* polynomial-time constructor *of a complete strategy π_c if given any trace of s_0 under π_c resulting in state s, $\pi_c(s)$ is computed by A in polynomial time in the size of the planning task and the trace.*

The pruning methods **PruneSource** and **PruneInit** only prune transitions that can never be part of an optimal strategy, **PruneAction** only prunes transitions where an at least as good alternative exists. Therefore, these methods preserve a closed optimal strategy. However, the **PruneOutcome** method prunes the non-deterministic outcomes, and thus might prune states reachable under the optimal policy.

Lemma 7. *Let $\sqsubseteq$ be a $\preceq_c$-game-simulation and $s, \hat{s} \in S$ be states such that $s \sqsubseteq \hat{s}$ and π a partial strategy s.t. $\pi(s)$ is defined. There exists a strategy $\hat{\pi}$ with $V_{\hat{\pi}}(\hat{s}) \leq V_\pi(s)$ where the value of $\hat{\pi}(\hat{s})$ is computable in polynomial-time in the size of the planning task given π, s and $\sqsubseteq$.*

Proof. If $\hat{s}$ is a goal state, there is nothing to do. Otherwise, let $\pi(s) = s \Rightarrow_\ell O$. Since $s \sqsubseteq \hat{s}$, $\hat{s}$ must have a transition $\hat{s} \Rightarrow_{\hat{\ell}} \hat{O}$ that dominates $s \Rightarrow_\ell O$. We find this transition by looking in the dominance relation for a transition where all outcomes dominate one outcome of the other. This must exist, otherwise we could not have $s \sqsubseteq \hat{s}$. Now, we need to choose the transition that minimizes $c(\hat{\ell}) + \max_{(\hat{o},\hat{s}') \in \hat{O}} c(\hat{o})$, which will ensure that $V_{\hat{\pi}}(\hat{s}) \leq V_\pi(s)$. We then let $\hat{\pi}(\hat{s}) = \hat{s} \Rightarrow_{\hat{\ell}} \hat{O}$. Since there is only polynomially many transitions of a state, this can be done in polynomial-time. □

Theorem 8. *Let $\sqsubseteq$ be a $\preceq_c$-game-simulation on Θ, Θ' be the result of pruning outcomes with* **PruneOutcome***, and let π be a closed optimal strategy for Θ'. There exists an algorithm $A_{\sqsubseteq,\pi}$ that is a polynomial-time constructor of an optimal strategy for Θ.*

Proof. We provide the algorithm $A_{\sqsubseteq,\pi}$ by describing how to compute a strategy π_c. By Lemma 7, with a dominated state for which π is defined we can compute $\pi_c(\hat{s})$ in polynomial time for any state $\hat{s}$. Therefore, we just need to compute this state. Let $((\ell_i, o_i))_{i=1}^m$ be the trace from s_0 that ends in $\hat{s}$. We prove by induction on the length of the trace, with induction hypothesis $P(m)$: for any trace $((\ell_i, o_i))_{i=1}^m$ of length m we can compute in polynomial time a state s_m s.t. $\pi(s_m)$ is defined, $s_m \sqsubseteq \hat{s}_m$ and $V_\pi(s_m) \leq V^*(s_0) - \sum_{i \leq m} c(\ell_i) + c(o_i)$. *Base case ($m = 0$):* The trace is empty, so $s_0 = \hat{s}_0$ trivially holds. *Inductive case ($P(m) \Rightarrow P(m+1)$):* From s_m we show how to construct s_{m+1}. We look at the strategy $\pi(s_m) = s_m \Rightarrow_\ell O$, and find an outcome $(o, s_{m+1}) \in O$ s.t. $s_{m+1} \sqsubseteq \hat{s}_{m+1}$ and $c(\ell_{m+1}) + c(o_{m+1}) \leq c(\ell) + c(o)$. This exists since $\sqsubseteq$ is a $\preceq_c$-game-simulation and **PruneOutcome** only prunes outcomes that dominate another outcome. Since π is closed for Θ' then $\pi(s_{m+1})$ is defined. Since π is optimal and by $P(m)$, we know that $V_\pi(s_{m+1}) + c(\ell) + c(o) \leq V_\pi(s_m) \leq V^*(s_0) - \sum_{i \leq m} c(\ell_i) + c(o_i)$. Since $c(\ell_{m+1}) + c(o_{m+1}) \leq c(\ell) + c(o)$ then $V_\pi(s_{m+1}) \leq V^*(s_0) - (\sum_{i \leq m} c(\ell_i) + c(o_i)) - c(\ell_{m+1}) - c(o_{m+1})$. This proves the induction.

Finally, given s_m we compute $\pi_c(\hat{s}_m)$ by the method of Lemma 7. Since $V_{\pi_c}(\hat{s}_m) \leq V_\pi(s_n) \leq V^*(s_0) - \sum_{i \leq m} c(\ell_i) + c(o_i)$, π_c is optimal. □

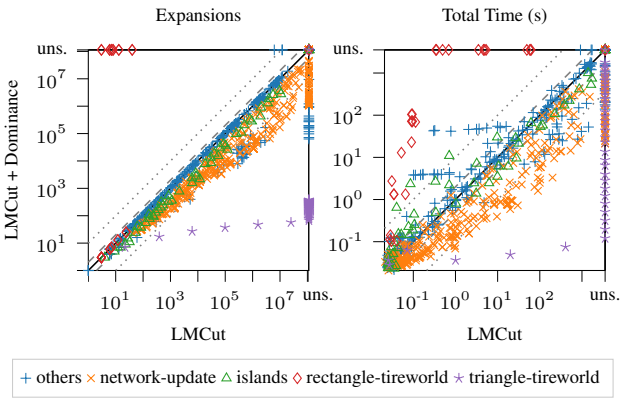


Figure 3: Comparison of LMCut in all outcome determinization with (y-axis) and without (x-axis) dominance pruning.

Experiments

In this section, we provide an experimental analysis of the pruning techniques and the outcome determinization technique developed in this paper. We structure this section into the following research questions (RQ). *RQ1:* Which of the four pruning techniques are best?. *RQ2:* What is the effect of different choices for the single-outcome determinization? *RQ3:* What is the effect of using dominance pruning?

We implemented the BLDFS algorithm on top of probFD (Steinmetz, Hoffmann, and Buffet 2016), an extension of Fast Downward (Helmert 2006) for probabilistic planning. The code and experimental data is publicly available (Tollund and Torralba 2025). We use Lab (Seipp et al. 2017) to run the experiments on AMD EPYC 7642 CPU with 30 min. time limit and 10 GB memory limit in Ubuntu 22.04.5.

Unfortunately, there exists no benchmark set for this setting. We therefore use benchmarks from FOND planning, specifically from the PR2 planner (Muise, McIlraith, and Beck 2024), and assume unit cost for all actions, and only use those with a strong acyclic policy. However, these are of poor quality. Most domains are simply deterministic problems where there is an added outcome that either does nothing or somehow "kills" the agent. This means that these actions are trivially unusable in our setting. Therefore, we contribute two new domains.

Food Delivery, inspired by Figure 1, is a domain where a food delivery person needs to deliver food to the customer within a time-limit. If the food does not arrive within the time-limit, it is free, so the customer chooses the restaurants to maximize the time it takes to deliver the food. The agent requests a new order and the non-deterministic outcome is which restaurant the order will be from.

Network Router Update, is a domain where the agent

Pruning Method	Act	Init	Out	Sou	No Pruning					Prune by Outcome & Source				
Heuristic	Blind				Blind	LMCut				Blind	LMCut			
Outcome det.						All	First	Last	Rand		All	First	Last	Rand
blocksworld-ex (15)	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
elevators (16)	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
islands (60)	50	51	51	51	51	50	50	**60**	48	51	50	50	**60**	51
miner (51)	23	11	20	41	11	25	27	**51**	32	41	43	44	47	47
rectangle-tireworld (30)	16	16	16	16	**28**	**28**	**28**	**28**	**28**	16	16	16	16	16
st-tireworld (14)	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
tireworld-truck (74)	58	58	58	58	58	73	**74**	72	73	59	73	**74**	72	72
triangle-tireworld (40)	6	6	**39**	6	6	6	6	6	6	37	**39**	**39**	**39**	38
network-update (858)	318	283	290	329	283	299	386	300	361	341	353	**395**	355	**395**
food-delivery (90)	37	**62**	**62**	**62**	**62**	51	51	48	48	**62**	50	49	47	50
Weighted Total	9.9	9.9	10.9	10.6	10.3	10.7	10.8	11.3	10.8	11.4	11.5	11.6	**11.7**	11.6
Total	610	589	639	666	601	635	725	668	699	710	728	771	740	**773**

Table 1: Coverage for pruning methods, without pruning and with dominance pruning (**PruneOutcome** and **PruneSource**) for blind heuristic and LMCut with various outcome determinization strategies. The coverage is shown for each domain having in parentheses the number of instances and in bold is highlighted the configurations that achieve the best coverage. Weighted Total is the sum of the mean of the coverage in each domain. Domains chain-of-rooms (10), doors (15), st-blocksworld (30), tireworld-spiky (14) are omitted, as all configurations solved all instances. We only count coverage on solvable instances.

schedules router updates in a communication network in batches. Updates within batches occur in a non-deterministic order, and the routing must remain functional at any point. This models a real-world problem in software defined networking, using a modified and simplified version of the Petri net encoding by Johansen et al. (2022). The agent adds routers to a batch, which is then non-deterministically accepted or tested. When tested, a package is simulated through the network, with each router in the batch routing non-deterministically to either the pre- or post-update target. The objective is to minimize total number of batches.

RQ1: In Table 1 under Pruning Methods we see a table over coverage on the different pruning methods. Comparing with No Pruning Blind, we can observe that **PruneSource** and **PruneOutcome** provide a significant increase in coverage, **PruneInit** does not provide enough reduction to warrant the overhead, and **PruneAction** some time works but is sometimes detrimental and we also found that **PruneSource** often captures the same pruning. **PruneSource** and **PruneOutcome** are orthogonal techniques, so we can expect that using both will perform even better. Therefore, we henceforth call these two techniques dominance pruning.

RQ2: In Table 1, we see the coverage on the blind heuristic and LMCut heuristic with different outcome determinizations. All refers to the all-outcome determinization (Rintanen 2003), where we create transitions for each outcome. First, Last and Random refers to single-outcome determinization choosing the first, last and a random, respectively, outcome. We see that the all outcome determinization is always worse than any single outcome determinization. This is expected, as the heuristic will want to pick the best outcome, not the worst. Which single outcome determinization is best depends on the domain, and we can observe that it makes a big difference. This is because the worst outcome depends on how the problem was modelled. However, even in network-update, where there is no outcome that is always worse, we still see a big difference. Likely because the first outcome is often the worst, or because it anyway is most informative for the heuristic. In future work, it would be interesting to study how to choose these outcomes, for example by using the dominance information.

RQ3: In Table 1, we can see that in general using dominance pruning increases coverage significantly. In some domains, choosing the correct outcome outperforms dominance pruning. Only in the rectangle-tireworld domain is dominance pruning significantly worse. This domain has a lot of labels which makes the dominance computation take a long time, and at the same time the search is very easy. However, in 98.6% of instances the dominance computation finishes in time. In miner, triangle-tireworld and network-update, dominance pruning works very well. Interestingly, none of the techniques work in food-delivery. In Figure 3, we see scatter plots over expanded states and total time comparing configurations with LMCut and with/without dominance pruning. Here we use the all outcome determinization in order to avoid the domain specific performance of first and last outcome and the randomness of the random outcome. We see that in general dominance pruning expands no more nodes than LMCut, and in some domains a lot fewer.

Conclusion

In this paper, we formalized the adversarial non-deterministic planning setting. We then showed how to construct single-outcome determinization heuristics. We showed how to extend label-dominance to this setting, and how to apply this in four transition pruning methods. In the experimental evaluation we found that using dominance pruning can significantly reduce the search effort in many domains, and that the heuristics also perform well but that the outcome chosen is very important.

Acknowledgements

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

References

Bellman, R. E. 1957. *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research*, 16: 580–595.

Bonet, B.; and Geffner, H. 2005. An Algorithm Better than AO*? In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*. AAAI Press.

Bonet, B.; and Geffner, H. 2006. Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings, and Its Application to MDPs. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 142–151. AAAI Press.

Brihaye, T.; Bruyere, V.; and Pril, J. D. 2010. Equilibria in quantitative reachability games. In *International Computer Science Symposium in Russia*, 72–83. Springer.

Büchner, C.; Ferber, P.; Seipp, J.; and Helmert, M. 2024. Abstraction Heuristics for Factored Tasks. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 40–49. AAAI Press.

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.

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong Cyclic Planning Revisited. In Biundo, S.; and Fox, M., eds., *Recent Advances in AI Planning. 5th European Conference on Planning (ECP 1999)*, volume 1809 of *Lecture Notes in Artificial Intelligence*, 35–48. Heidelberg: Springer-Verlag.

Fritz, C.; and Wilke, T. 2005. Simulation relations for alternating Büchi automata. *Theor. Comput. Sci.*, 338: 275–314.

Hall, D.; Cohen, A.; Burkett, D.; and Klein, D. 2013. Faster Optimal Planning with Partial-Order Pruning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 100–108. AAAI Press.

Hansen, E. A.; and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1–2): 35–62.

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

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

Johansen, N. S.; Kær, L. B.; Madsen, A. L.; Nielsen, K. Ø.; Srba, J.; and Tollund, R. G. 2022. Kaki: Concurrent update synthesis for regular policies via petri games. In *International Conference on Integrated Formal Methods*, 249–267. Springer.

Keller, T.; and Eyerich, P. 2011. A Polynomial All Outcome Determinization for Probabilistic Planning. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 331–334. AAAI Press.

Klößner, T.; Álvaro Torralba; Steinmetz, M.; and Sievers, S. 2023. A Theory of Merge-and-Shrink for Stochastic Shortest Path Problems. In Koenig, S.; Stern, R.; and Vallati, M., eds., *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)*, 203–211. AAAI Press.

Klößner, T.; Pommerening, F.; Keller, T.; and Röger, G. 2022a. Cost Partitioning Heuristics for Stochastic Shortest Path Problems. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 193–202. AAAI Press.

Klößner, T.; Steinmetz, M.; Torralba, Á.; and Hoffmann, J. 2022b. Pattern Selection Strategies for Pattern Databases in Probabilistic Planning. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 184–192. AAAI Press.

Korf, R. E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27(1): 97–109.

Kuter, U.; Nau, D. S.; Reisner, E.; and Goldman, R. P. 2008. Using Classical Planners to Solve Nondeterministic Planning Problems. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 190–197. AAAI.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2024. PRP Rebooted: Advancing the State of the Art in FOND Planning. In Dy, J.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024)*, 20212–20221. AAAI Press.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 172–180. AAAI Press.

Plaat, A.; Schaeffer, J.; Pijls, W.; and de Bruin, A. 1996. Best-First Fixed-Depth Minimax Algorithms. *Artificial Intelligence*, 87(1–2): 255–293.

Rintanen, J. 2003. Expressive Equivalence of Formalisms for Planning with Sensing. In Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, 185–194. AAAI Press.

Russell, S.; and Norvig, P. 2020. *Artificial Intelligence: A Modern Approach*. Pearson.

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

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal Probability Analysis in MDP Probabilistic Planning: Exploring and Enhancing the State of the Art. *Journal of Artificial Intelligence Research*, 57: 229–271.

Tarski, A. 1955. A LATTICE-THEORETICAL FIXPOINT THEOREM AND ITS APPLICATIONS. *Pacific Journal of Mathematics*, 5: 285–309.

Tollund, R.; and Torralba, A. 2025. Artefact for Dominance Pruning and Heuristics in Optimal Adversarial Non-Deterministic Planning.

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

Torralba, Á.; and Sievers, S. 2019. Merge-and-Shrink Task Reformulation for Classical Planning. In Kraus, S., ed., *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5644–5652. IJCAI.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 306–315. AAAI Press.

Younes, H. L. S.; and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, School of Computer Science.