Necessary Observations in Nondeterministic Planning

David Speck, Manuela Ortlieb, and Robert Mattmüller

Research Group Foundations of AI, University of Freiburg, Germany
{speckd,ortlieb,mattmuel}@informatik.uni-freiburg.de

Abstract. An agent that interacts with a nondeterministic environment can often only partially observe the surroundings. This necessitates observations via sensors rendering more information about the current world state. Sensors can be expensive in many regards therefore it can be essential to minimize the amount of sensors an agents requires to solve given tasks. A limitation for sensor minimization is given by essential sensors which are always required to solve particular problems. In this paper we present an efficient algorithm which determines a set of necessary observation variables. More specifically, we develop a bottom-up algorithm which computes a set of variables which are always necessary to observe, in order to always reach a goal state. Our experimental results show that the knowledge about necessary observation variables can be used to minimize the number of sensors of an agent.

Keywords: AI planning, nondeterministic planning, partial observability, observation actions

1 Introduction

An agent that interacts with a nondeterministic environment can often only partially observe the surroundings. Acting in such an environment with uncertainty necessitates observations of the current world state via sensors to obtain more information for reaching a goal state. Such sensors can be expensive with regards to battery, money, weight, maintenance, and time. Therefore it can be useful or even essential to minimize the amount of sensors necessary to solve a particular planning task. More precisely, we consider the sensors an agent needs to be fitted with. In this paper, we discuss the problem of minimizing a set of necessary sensors and *not* the problem of minimizing the amount of sensor observations. For example, in a specific robotic application, an RGB-D camera can handle all the observations a laser scanner would be used for, thus obviating the latter. Regarding an extraterrestrial mission such a sensor reduction can be essential to minimize the weight. We represent the uncertainty of a current world state as a set of world states denoted as belief state. Applying and selecting actions and observations via sensors for belief states (decision points) in such an environment is called offline partially observable nondeterministic planning. Similar to Mattmüller et al. [1] we reduce the problem of sensor minimization by assuming

that a sensor and its measured data are represented by a state variable which can be observed. This simplification induces a search for a minimal set of variables $\mathcal{O}$ where only the variables contained in $\mathcal{O}$ can be observed and still every planning task of the underlying planning domain is solvable. After the removal of a variable o from the set of observable variables $\mathcal{O}$ we call o reduced. In addition, we reduce the problem of finding such a set of variables from the planning domain level to the planning task level. As Mattmüller et al. [1] mentioned, it is possible to generalise the results if such a planning task is reasonably chosen with regard to the underlying planning domain. Clearly, necessary observation variables of a planning task Π can never be reduced without losing power with regard to solving problems because if such a necessary observation variable is not observable, at least planning task Π is not solvable anymore. Furthermore, a necessary observation variable o is an element of every minimal set of variables $\mathcal{O}$ which is still sufficient to solve every planning task of the underlying planning domain if only the variables of $\mathcal{O}$ are observable. Considering the previous example, if an RGB-D camera is necessary to track the localization of a robot, this camera can never be reduced particularly with regard to localization problems. Such a knowledge about necessary sensors can improve the runtime of a sensor reduction procedure depending on its construction. To our knowledge, three recent studies on observation minimization have been published. Two of them developed by Huang et al. [2, 3] deal with observation minimization for a fixed plan in different settings. Firstly, they presented an algorithm which calculates an approximately minimal set of observations for a given set of variables $\mathcal{V}$ and a fixed strong plan π where all variables $\mathcal{V}$ are possibly observable. The algorithm identifies all state pairs which need to be distinguished in plan π and always chooses the observation variable which distinguishes the most remaining not distinguished state pairs [2]. Secondly, Huang et al. [3] extended their results/algorithm and presented an attempt to solve the problem of observation reduction for general plans with contexts. The work of Mattmüller et al. [1] is closely connected to this work in regard to the same problem setting. They worked on a top-down approach which greedily removes observation variables by the trial and error method still sufficient to solve a particular planning task. This greedy top-down algorithm returns an inclusion minimal set of observation variables. While here, a bottom-up procedure is presented which collects stepwise necessary observation variables, i.e. variables which always have to be observed to solve a particular planning task.

2 Preliminaries

We formally define partially observable nondeterministic (POND) planning similar to the definition of Mattmüller et al. [1] using a *finite-domain representation* for the state variables. A *POND planning task skeleton* is a 5-tuple $\Pi = \langle \mathcal{V}, B_0, B_*, \mathcal{A}, \mathcal{W} \rangle$, where $\mathcal{V}$ is a finite set of *state variables*, B_0 is an *initial belief state*, B_* is a *goal description*, $\mathcal{A}$ is a finite set of *nondeterministic actions*, and $\mathcal{W} \subseteq \mathcal{V}$ is a set of possible observable variables. Every state variable

v in $\mathcal{V}$ has a finite *domain* $\mathcal{D}_v$ and an *extended domain* $\mathcal{D}_v^+$, where $\perp$ denotes the *undefined/don't-care* value. A function s, where $s(v) \in \mathcal{D}_v^+$ for all $v \in \mathcal{V}$ is called a *partial state*. Partial state s is defined for a variable v if $s(v) \neq \perp$. The *scope* of a partial state s is the set of all variables v which are defined in s, i.e. $\mathit{scope}(s) = \{v \in \mathcal{V} \mid s(v) \neq \perp\}$. We call a partial state s a *state* if s is defined for all variables of $\mathcal{V}$ which means $\mathit{scope}(s) = \mathcal{V}$. A variable-value pair is called a fact and denoted by (v, d) or $v = d$, where $v \in \mathcal{V}$ and $d \in \mathcal{D}_v$. The set $\mathcal{S}$ represents all states over $\mathcal{V}$ and the set $\mathcal{B}$ represents all *belief states* over $\mathcal{V}$, where $\mathcal{B} = 2^{\mathcal{S}}$. We call a belief state B a *goal belief state* iff $B \subseteq B_*$. A partial state s_p can be used as a *condition* or as an *update* on a state s. We say a condition s_p is *satisfied* in a state s iff s agrees with all defined variables of s_p. An update s_p on a state s leads to a new state s' that agrees with s_p on all defined variables and with s on all other variables. An action $a \in \mathcal{A}$ is of the form $a = \langle \mathit{Pre}, \mathit{Eff} \rangle$ where the two components are a partial state Pre called *precondition* and a finite set Eff of partial states eff called *effect*. We call the partial states $\mathit{eff} \in \mathit{Eff}$ of an action a the *nondeterministic outcomes* of a. We denote the set of all facts as precondition Pre or effect Eff of an action $a = \langle \mathit{Pre}, \mathit{Eff} \rangle$ by $\mathit{pre}(a)$ and $\mathit{eff}(a)$, where $\mathit{eff}(a)$ is the union over all facts of every nondeterministic outcome $\mathit{eff} \in \mathit{Eff}$. The union over a set of actions A is analogously defined as $\mathit{pre}(A)$ and $\mathit{eff}(A)$, i.e. $\mathit{pre}(A) = \bigcup_{a \in A} \mathit{pre}(a)$ and $\mathit{eff}(A) = \bigcup_{a \in A} \mathit{eff}(a)$. *Applications* in POND planning are defined as follows: The application of a nondeterministic outcome eff to a state s is a state $\mathit{app}(\mathit{eff}, s)$ resulting from an update of s with eff. The application of an effect Eff to a state s results in a set of states $\mathit{app}(\mathit{Eff}, s) = \{\mathit{app}(\mathit{eff}, s) \mid \mathit{eff} \in \mathit{Eff}\}$. An action $a = \langle \mathit{Pre}, \mathit{Eff} \rangle$ is applicable in a state s if its precondition Pre is satisfied in s. An action $a = \langle \mathit{Pre}, \mathit{Eff} \rangle$ is applicable in a belief state B if its precondition Pre is satisfied in all states s, where $s \in B$. The application of an action $a = \langle \mathit{Pre}, \mathit{Eff} \rangle$ in a belief state B is a belief state $\mathit{app}(a, B) = \{\mathit{app}(\mathit{eff}, s) \mid \mathit{eff} \in \mathit{Eff}, s \in B\}$ if a is applicable in B and undefined otherwise. To complete the definition of partially observable nondeterministic planning we define an *observation variable* or in short *observation* as a variable $o \in \mathcal{W}$. The result of an *observation application* to a belief state B is a belief state $\mathit{app}(o, B) = \{\{s \in B \mid s(o) = d\} \mid d \in \mathcal{D}_o\} \backslash \{\emptyset\}$ where $\mathit{app}(o, B)$ is a non-empty subset of B and contains only states according to the possible values of o.

A POND *planning task* $\Pi[\mathcal{O}] = \langle \Pi, \mathcal{O} \rangle$ is a tuple, where Π is a POND planning task skeleton, and $\mathcal{O} \in \mathcal{W}$ is a finite set of observations. Actions and observations have positive unit costs for applications. In the following section we will denote a planning task $\Pi[\mathcal{O}] = \langle \Pi, \mathcal{O} \rangle$ also as Π if it is clearly understandable from the context. We call a partial mapping π from belief states to applicable actions or observations a *plan* for a given POND planning task. A plan π is *closed* if every belief state B reachable from the initial belief state B_0 following π is a goal belief state or plan π is defined for B. If from every belief state B reachable from initial belief state B_0 following π at least one goal state $B' \subseteq B_*$ is reachable following π, we call π *proper*. A plan π is a *strong cyclic plan* for a POND planning task if π is closed and proper. We denote by $\mathcal{B}_\pi$ the set

of all belief states which are non-goal states and reachable from the initial belief state B_0 following π, including B_0. An action or observation *occurs* in a plan π if there exists a belief state $B \in \mathcal{B}_\pi$, where $\pi(B) = a$ or $\pi(B) = o$ [1, 4]. Let $a = \langle Pre, \mathit{Eff} \rangle$, with $\mathit{Eff} = \{\mathit{eff}_1, \ldots, \mathit{eff}_n\}$ be an action. The *strong outcome*

$$\mathit{eff}_i^s(v) = \begin{cases} \mathit{eff}_i(v) & \text{if } v \in \mathit{scope}(\mathit{eff}_i) \\ Pre(v) & \text{otherwise} \end{cases}$$

contains all facts which are always true after applying action a with a resulting outcome of eff_i. The *strong effect* $\mathit{Eff}^s = \{\mathit{eff}_1^s, \ldots, \mathit{eff}_n^s\}$ contains all corresponding strong outcomes of Eff.

Applying and selecting actions in deterministic environments where an initial world state is fully known is called classical planning. We define classical planning as a special case of POND planning. A *classical planning task* using *finite-domain representation* for the state variables is a 4-tuple $\Pi_{det} = \langle \mathcal{V}, \mathcal{A}_{det}, s_o, B_* \rangle$, where $\mathcal{V}$ is a finite set of state variables, s_o is an *initial world state* over $\mathcal{V}$, $\mathcal{A}_{det}$ is a finite set of *deterministic actions* over $\mathcal{V}$, and B_* is a goal description. An action $a \in \mathcal{A}_{det}$ in classical planning is a nondeterministic action a with the restriction that effect Eff contains only one nondeterministic outcome eff, i.e $|\mathit{Eff}| = 1$. We call such an action a a *deterministic action*. In classical planning actions are applied in world states and result in world states. A deterministic action $a = \langle Pre, \mathit{Eff} \rangle$ is applicable in a world state s if its precondition Pre is satisfied in s. The *application* of an action $a = \langle Pre, \mathit{Eff} \rangle$ to a world state s is the world state $app(a, s) = app(\mathit{eff}, s)$, where $\{\mathit{eff}\} = \mathit{Eff}$ if a is applicable in s and undefined otherwise. Hereafter, if we talk about preconditions and effects in classical planning we only mention facts, i.e. we ignore undefined variables. A (classical) *plan* π_{det} for a planning task $\Pi_{det} = \langle \mathcal{V}, \mathcal{A}_{det}, s_o, B_* \rangle$ is a sequence of applicable actions $a_0, \ldots, a_n$ with a world state sequence $s_0, \ldots, s_{n+1}$ where $app(a_i, s_i) = s_{i+1}$ and $s_{n+1} \in B_*$ is a goal world state. An action $a \in \mathcal{A}_{det}$ *occurs* in a plan π_{det} if it is contained in the application sequence of π_{det}. The *determinization* of a nondeterministic action $a = \langle Pre, \mathit{Eff} \rangle$, with $\mathit{Eff} = \{\mathit{eff}_1, \ldots, \mathit{eff}_n\}$, is a set of n actions $a^i = \langle Pre, \{\mathit{eff}_i\} \rangle$ generated by the function $\mathcal{A}_{det}(a)$.[1] Such an action $a^i = \langle Pre, \{\mathit{eff}_i\} \rangle$ is a deterministic action. Every POND planning task $\Pi = \langle \mathcal{V}, B_0, B_*, \mathcal{A}, \mathcal{W} \rangle$ has $|B_0|$ unique classical (deterministic) planning tasks $\Pi_{det} = \langle \mathcal{V}, \mathcal{A}_{det}, s_0, B_* \rangle$ where $\mathcal{A}_{det} = \bigcup_{a \in \mathcal{A}} \mathcal{A}_{det}(a)$ and $s_0 \in B_0$. The function $n(a^i) : \mathcal{A}_{det} \to \mathcal{A}$ maps a determinized action $a^i \in \mathcal{A}_{det}$ back to its original nondeterministic action $a \in \mathcal{A}$. A *disjunctive action landmark* or in short *landmark* of a classical planning task Π_{det} is a set of actions L such that at least one action of L occurs in every plan for Π_{det}. Originally the landmark cut procedure by Helmert and Domshlak [5] is used as a heuristic function by calculating disjunctive landmarks of a classical planning task. We will slightly modify the procedure by returning the disjunctive landmarks of a classical planning task Π_{det} instead of the heuristic value and denote the resulting landmarks by $\mathit{LM\text{-}cut}(\Pi_{det})$.

[1] For simplification we assume that every determinized action has a particular unique ID which may lead to duplications of actions.

3 Necessary Observations

Targeting observation reduction regarding a POND planning task we can search for observations that can never be reduced, i.e. observations that are always necessary for every strong cyclic plan. We call such an observation a necessary observation and define it as follows.

Definition 1 (Necessary Observation). *A necessary observation of a POND planning task Π is an observation o such that it occurs at least once in every strong cyclic plan for Π. We call a set of necessary observations o a necessary observation set $\mathcal{N}$.*

For example, we assume a nondeterministic BLOCKSWORLD with two blocks A and B. Initially, block A is located on block B. The goal is variable B-*clear* which means that no block is on B. There exists only one action *pick-up-A-B* with two outcomes: either block A is picked up or nothing happens and block A remains on block B. Furthermore, we assume that it is only possible to observe if any block is located on block X by observation X-*clear*. Obviously, action *pick-up-A-B* has to be applied until block A is picked up for reaching the goal state. The only possibility for verifying that block A is picked up is to observe B-*clear* which is why observation B-*clear* occurs in every strong cyclic plan for the problem. Thus in this setting B-*clear* is a necessary observation and additionally the only one. However, not always necessary observations exist. We assume that the previous example contains also an observation X-*picked* which encodes if block X is picked up. Now we can construct two different strong cyclic plans – one with observation B-*clear* and one with A-*picked* because observations B-*clear* and A-*picked* verify whether B-*clear* is satisfied after applying action *pick-up-A-B*. Therefore none of these observations occurs in every strong cyclic plan and consequently no necessary observation exists. Interestingly, the reduction of an unnecessary observation can lead to additional necessary observations. Regarding the previous example, by reducing observation B-*clear* observation A-*picked* becomes necessary and vice versa. This property will be topic of upcoming research.

Necessary observation sets and cardinality or inclusion minimal observation sets sufficient to solve a POND planning task are closely connected. To formalize this connection, we need a theorem formulated by Mattmüller et al. [1].

Theorem 1 (Mattmüller et al., 2014 [1]). *Given a POND planning task skeleton Π the problem of finding a cardinality (*OBSERVECARDMIN*) or an inclusion minimal set of observations (*OBSERVEINCLMIN*) $\mathcal{O} \subseteq \mathcal{W}$ for Π such that there exists a strong cyclic plan for $\Pi[\mathcal{O}]$, or returning* NONE *if no such set $\mathcal{O}$ exists, is* 2-EXPTIME*-complete.* □

Clearly, every cardinality minimal solution is also inclusion minimal, but not vice versa. Therefore, the following results regarding the OBSERVECARDMIN problem hold also for the OBSERVEINCLMIN problem.

Theorem 2. *A necessary observation set is a subset of all solutions for the* ObserveCardMin *problem.*

Proof. A solution $\mathcal{O}$ for the ObserveCardMin problem is a cardinality minimal set such that there exists a strong cyclic plan π. By Definition 1 a necessary observation set $\mathcal{N}$ contains only necessary observation o such that o occurs in every strong cyclic plan π for Π. Therefore, a strong cyclic plan π for Π can only exist if all elements of a necessary observation set $\mathcal{N}$ are contained in $\mathcal{O}$, i.e. $\mathcal{N} \subseteq \mathcal{O}$. □

Next we proof that every planning task has a unique and well-defined maximal necessary observation set $\mathcal{N}^*$. The latter property is the main idea of the following bottom-up procedure which computes iteratively necessary observation sets and combines all corresponding necessary observations to one resulting necessary observation set.

Theorem 3. *The maximal necessary observation set $\mathcal{N}^*$ of a planning task Π is unique and well-defined.*

Proof. Every planning task Π has at least one maximal necessary observation set $\mathcal{N}^*$ (Definition 1). We assume that $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ are two maximal necessary observation sets of a planning task Π with $\mathcal{N}_1^* \neq \mathcal{N}_2^*$. Thus, there exists at least one necessary observation o which is an element of the symmetric difference $\mathcal{N}_1^* \triangle \mathcal{N}_2^*$, i.e. $o \in \mathcal{N}_1^* \triangle \mathcal{N}_2^*$. By definition, a maximal necessary observation set $\mathcal{N}^*$ contains all necessary observations which leads to a contradiction. Therefore exactly one necessary observation set $\mathcal{N}^*$ exists for a planning task Π (uniqueness). From the uniqueness of $\mathcal{N}^*$ follows that $\mathcal{N}^*$ is well-defined. □

3.1 Bottom-Up Search

We present an algorithm called Nos which computes a necessary observation set for a given POND planning task Π. The Nos algorithm is divided in two parts. First, we present a calculation of necessarily needed nondeterministic actions using determinized planning tasks Π_{det} and landmarks computed by the landmark cut procedure *LM-cut*(Π_{det}). The second part is about getting a necessary observation set from a set of landmarks $\mathcal{L}$. In this part we search for differences between desired outcomes contained in the landmarks, i.e. outcomes which lead further to a goal state, and outcomes which are undesired as they belong to the same original nondeterministic action. Figure 1 provides an overview of the different steps of the Nos algorithm.

Part 1: Given a POND planning task Π we use the landmark cut procedure to generate landmarks for a corresponding determinized (classical) planning task Π_{det}. The application of a deterministic action, i.e. an action with only one outcome, of a POND planning task can never be a reason for an observation. The uncertainty of a world state presented in a belief state is evoked by an earlier

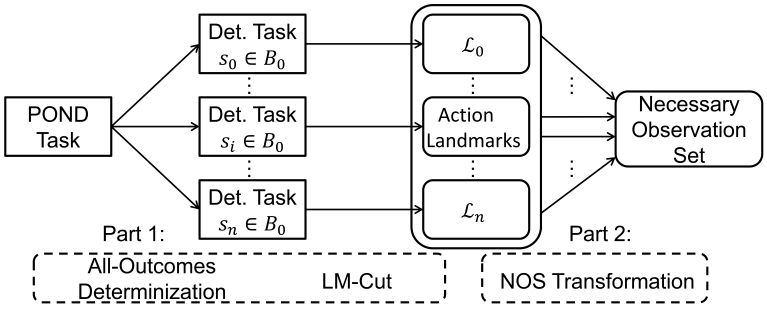

Fig. 1. Overview of the different steps of the NOS algorithm.

nondeterministic action or the uncertainty of the initial state. Therefore, we are only interested in nondeterministic actions with more than one outcome. Thus we modify a determinized (classical) planning task Π_{det} by the cost function

$$c_{det}(a) = \begin{cases} 1 & \text{if } |\mathit{Eff}| > 1 \text{ where } n(a) = \langle \mathit{Pre}, \mathit{Eff} \rangle \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

which maps all actions $a \in \mathcal{A}_{det}$ with more than one outcome (nondeterministic) in Π to cost 1 and all other actions $a' \in \mathcal{A}_{det}$ with only one outcome in Π to cost 0 (deterministic). A planning task Π_{det} with cost function c_{det} is denoted as Π^c_{det}. We compute a set of landmarks for a determinized (classical) planning task Π_{det} as

$$\mathcal{L} = \mathit{LM\text{-}cut}(\Pi^c_{det}) \ . \tag{2}$$

Every landmark of Π^c_{det} contains only outcomes of nondeterministic actions. Finally, given a POND planning task Π we compute families of landmarks for a sampled number of determinized planning task Π_{det} and collect all landmarks $\mathit{LM\text{-}cut}(\Pi^c_{det})$ as one final set of landmarks $\mathcal{L}$.

Part 2: Computing a necessary observation set out of a computed set of landmarks $\mathcal{L}$ is the second step of the NOS algorithm. As mentioned before we search for differences between desired outcomes and undesired outcomes of a nondeterministic action. For that reason we group outcomes of the same original action contained in a landmark. Such a group of outcomes is called a *parallel outcome* $P \subseteq \mathcal{A}_{det}$, where each action $a \in P$ has the same original action in the corresponding POND planning task Π, i.e. $\forall a, a' \in P : n(a) = n(a')$. Using parallel outcomes we can define grouped landmarks as follows.

Definition 2 (Grouped Landmarks). *We call $L^G = \{\mathcal{P}^a_L \mid a \in L\}$ a grouped landmark where $\mathcal{P}^a_L = \{a' \in L \mid n(a) = n(a')\}$ forms disjoint equivalence classes with nondeterministic actions of the same original actions as representatives.*

Concluding, we define undesired outcomes of a parallel outcome as $\overline{P}$ and call it *complement* of a parallel outcome set P which contains all actions $a \notin P$, where

$n(a) = n(P)$. Attached with the latter concept it becomes possible to compute necessary observations given a set of landmarks $\mathcal{L}$ (Part 1) using functions (1) and (2) and the following six functions.[2]

Function $symDiffs(\mathcal{P})$ collects all sets of facts (symmetric differences) which can be chosen to distinguish the outcomes of a parallel outcome $\mathcal{P}$ (desired) from an outcome of its complement $\overline{\mathcal{P}}$ (undesired).

$$symDiffs(\mathcal{P}) = \{eff^s(\mathcal{P}) \triangle eff^s(\{a\}) \mid a \in \overline{\mathcal{P}}\} \tag{3}$$

Transform all facts to observation variables.

$$obsVars(\mathcal{P}) = \bigcup_{D \in symDiffs(\mathcal{P})} \{\{v \in \mathcal{W} \mid \exists d \neq \bot : (v, d) \in D\}\} \tag{4}$$

Collect all variables which are necessary to distinguish the outcomes of parallel outcome $\mathcal{P}$ from the outcomes of its complement $\overline{\mathcal{P}}$.

$$singleVars(\mathcal{P}) = \{v \in D \mid D \in obsVar(\mathcal{P}) \wedge |D| = 1\} \tag{5}$$

Compute a necessary observation set for a given landmark. Remove parallel outcomes which have an empty observation choice and therefore cannot be completely distinguished.

$$nos(L) = \bigcap_{\mathcal{P} \in L^G \,:\, \emptyset \notin obsVars(\mathcal{P})} singleVars(\mathcal{P}) \tag{6}$$

Remove all landmarks from a set of landmark which contain a parallel outcome $\mathcal{P}$ with an empty complement $\overline{\mathcal{P}}$ and therefore never lead to an uncertainty by applying the original action.

$$pruneL(\mathcal{L}) = \{L \in \mathcal{L} \mid \forall \mathcal{P} \in L^G : \overline{\mathcal{P}} \neq \emptyset\} \tag{7}$$

Collect all necessary observation sets computed for every landmark individually.

$$nos(\mathcal{L}) = \bigcup_{L \in pruneL(\mathcal{L})} nos(L) \tag{8}$$

As in previous examples we assume a nondeterministic BLOCKSWORLD but with three blocks A, B, and C instead of only two blocks. There exist exactly two actions *put-on-block-B-C* and *put-tower-on-block-A-B-C* which are visualized in Figure 2. Action *put-on-block-B-C* has an effect where either block B drops down to the table (0) or block B is stacked on block C (1), and action *put-tower-on-block-A-B-C* has an effect where either tower A-B drops down to the table (0) or tower A-B is stacked on block C (1). Furthermore, it is only possible to observe if any block is located on block X by observation X-*clear*. We assume $L = \{put\text{-}on\text{-}block\text{-}B\text{-}C^1, put\text{-}tower\text{-}on\text{-}block\text{-}A\text{-}B\text{-}C^1\}$ to be a land-

[2] Notice that at least one outcome of every landmark occurs in every plan for Π_{det} and that a strong cyclic plan for the corresponding POND planning task Π always contains all outcomes of Π_{det} which additionally need to be verified by observations.

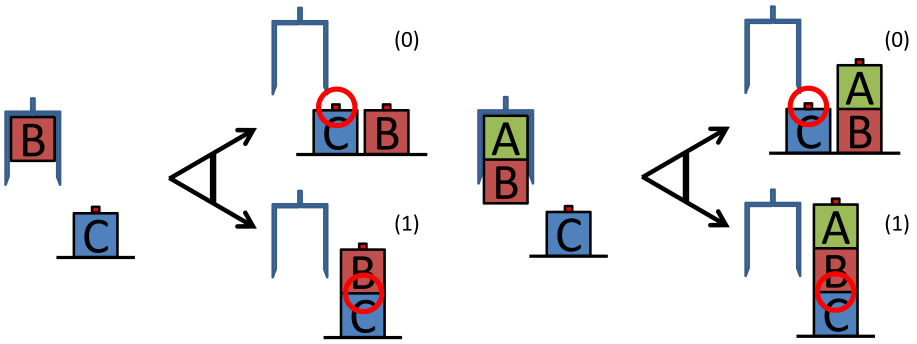

Fig. 2. Actions *put-on-block-B-C* (left) and *put-tower-on-block-A-B-C* (right) with there corresponding possible outcomes.

mark. Thus the goal is to apply an action such that any block is located on block C afterwards, i.e. $\overline{C\text{-}clear}$. The corresponding grouped landmark of L is $L^G = \{\{put\text{-}on\text{-}block\text{-}B\text{-}C^1\}, \{put\text{-}tower\text{-}on\text{-}block\text{-}A\text{-}B\text{-}C^1\}\}$. The symmetric differences of observable facts which distinguishes the desired outcomes (1) of both actions contained in L^G from their corresponding undesired outcome (0) is C-*clear* (Figure 2). Therefore to ensure that at least one outcome of L occurs observation C-*clear* has to be applied and the resulting necessary observation set is $\mathcal{N} = \{C\text{-}clear\}$.

NOS Algorithm: Using Part 1 and Part 2 it becomes possible to calculate a necessary observation set for a given POND planing task Π and a number of considered initial states k. Algorithm NOS (Algorithm 1) a computes a necessary observation set $\mathcal{N}$ which is a subset of a maximal necessary observation set $\mathcal{N}^*$. Every variable $o \in \mathcal{N}$ is observed at least once in any strong cyclic plan π for Π. Algorithm 1 iterates over a number k of determinized planning tasks with different initial states and first collects the corresponding landmarks and then computes necessary observations using the landmarks.

Algorithm 1 NOS(Π, k)

Require: $\Pi = \langle \mathcal{V}, B_0, B_*, \mathcal{A}, \mathcal{W} \rangle, k \in \mathbb{N}$
1: $\mathcal{N} = \emptyset$; $\mathcal{L} = \emptyset$;
2: **for** $\{s_0, \ldots, s_k\} \subseteq B_0$ **do**
3: $\quad \Pi_{det} = \langle \mathcal{V}, \mathcal{A}_{det}, s_i, B_* \rangle$
4: $\quad \mathcal{L} = \mathcal{L} \cup LM\text{-}cut(\Pi^c_{det})$
5: $\mathcal{N} = nos(\mathcal{L})$
6: **return** $\mathcal{N}$

In the following we show the runtime of Algorithm 1 and proof its correctness.

Theorem 4. *The runtime complexity of Algorithm 1 is bounded by* $\mathcal{O}(\|\Pi\|^4 * k)$ *where* $\Pi = \langle \mathcal{V}, B_0, B_*, \mathcal{A}, \mathcal{W} \rangle$ *is an input POND planning task, and* k *is the number of considered initial states.*

Proof. The landmark cut procedure is bounded in runtime by $\mathcal{O}(\|\Pi_{det}\|^2)$ [6] and Algorithm 1 contains k of such procedures, which leads to a runtime bounded by $\mathcal{O}(\|\Pi_{det}\|^2 * k)$. The size of a set of landmarks $\mathcal{L}$ computed by the landmark cut procedure for one determinzed planning task Π_{det} is bounded by $\mathcal{O}(\|\Pi_{det}\|)$. Overall we have k landmark cut procedures wherefore the amount of landmarks is bounded by $\mathcal{O}(\|\Pi_{det}\| * k)$. Function $nos(L)$ (6) is bounded in runtime by $\mathcal{O}(\|\Pi_{det}\|)$. Thus, function $nos(\mathcal{L})$ (8) is bounded by $\mathcal{O}(\|\Pi_{det}\|^2 * k)$. The determinization of a planning task Π_{det} can be bounded by $\mathcal{O}(\|\Pi\|^2)$ resulting in an upper bound complexity of $\mathcal{O}(\|\Pi\|^4 * k)$.

Theorem 5. *Algorithm* Nos *(Algorithm 1) returns a necessary observation set for a given POND planning task* Π*, i.e. Algorithm 1 is correct.*

Proof. Reducing the cost of an action has no side effects for other actions which is why a landmark for Π^c_{det} is also a landmark for Π_{det}. A strong cyclic plan π for Π is a composition of plans π_{det} for Π_{det}. At least one action of every landmark computed by *LM-cut*(Π^c_{det}) occurs in every plan π_{det}. To verify that one of the outcomes contained in a landmark happened, an observation is necessary. Such an observation is always contained in the symmetric difference of the strong effect of an outcome contained in a landmark and an outcome which is not contained in the landmark belonging to the same original nondeterministic action. Therefore collecting observations which are always required to distinguish all outcomes of a landmark from outcomes not contained in the landmark of the same original action results in a necessary observation set.

3.2 Observation Minimization

A necessary observation set can be used to improve the runtime of the Greedy algorithm by Mattmüller et al. [1]. The Greedy algorithm is a top-down approach which greedily removes observation variables by the trial-and-error method until a solution $\mathcal{O}$ for the ObserveInclMin problem remains. The authors mentioned that a useful extension for the Greedy algorithm is a heuristic which orders the candidate variables of removal. A precomputed necessary observation set does not order but rather eliminate such candidates. For every candidate variable of removal which is part of a necessary observation set, one removal iteration of the Greedy algorithm is eliminated. Therefore, we get the following Algorithm 2 in pseudo code. Clearly, if no necessary observation is found or if the input size of the planning task is small, there is no runtime improvement. Nevertheless, for every reduced candidate variable of removal one planning procedure is eliminated. Such a planning procedure is (even with plan reuse) 2-Exptime-complete [7]. Our results show that Algorithm 2 outperforms the original Greedy algorithm.

Algorithm 2 PRUNEDGREEDY(Π, k)

Require: $\Pi = \langle \mathcal{V}, B_0, B_*, \mathcal{A}, \mathcal{W} \rangle, k \in \mathbb{N}$
1: $\mathcal{N} = \text{NOS}(\Pi, k)$
2: set candidates of removal $\mathcal{O}$ to $\mathcal{O} \setminus \mathcal{N}$
3: $\mathcal{O} = \text{GREEDY}(\Pi)$
4: **return** $\mathcal{O}$

4 Experiments

We implemented Algorithm 1 and the GREEDY algorithm returning a solution for the OBSERVEINCLMIN problem using the MYND planner [1]. The overall 172 analysed PONDS planning tasks belong to the BLOCKSWORLDSENSE (only block clear observations), TIDYUP or FIRSTRESPONDERS domain. Every planning task belonging to the BLOCKSWORLDSENSE domain and FIRSTRESPONDERS domain has only one initial state ($|B_0| = 1$) and therefore only one determinized planning task. Whereas the planning tasks of the TIDYUP domain have an initial belief state containing up to 10^9 initial states. We used a memory limit of 4 GB. Table 1 summarizes our experimental results analysing Algorithm 1. The runtime of Algorithm 1 considering one and ten initial states was around 1 second whereas the runtime considering all initial states $s_0 \in B_0$ was up to one hour. Interestingly, for almost every task of the TIDYUP domain, there is no difference between the necessary observation set $\mathcal{N}B_0$ calculated by considering all initial states B_0 and the necessary observation set $\mathcal{N}S_1$ considering only 1 initial state. Therefore, we can argue that at least for planning tasks belonging to the TIDYUP domain, sampling over a number of initial states (e.g. 1) still leads to good results in practice. This is due to the fact that deterministic plans for different initial states of a planning task usually have similar subgoals and therefore similar landmarks.

Table 1. Cardinality of observation sets (variables). Legend: B_0 = initial belief state, $\mathcal{W}$ = possible observations, $\mathcal{N}$ = necessary observation set computed by the NOS algorithm (B_0 = considering all initial states, S_k = considering k sampled initial states).

Domain (#Tasks)	$\varnothing\lvert B_0\rvert$	$\varnothing\lvert\mathcal{W}\rvert$	$\varnothing\lvert\mathcal{N}B_0\rvert$	$\varnothing\lvert\mathcal{N}S_1\rvert$	$\varnothing\lvert\mathcal{N}S_{10}\rvert$
BWSENSE(30)	1.00	10.00	6.53	-	-
FRPONDERS(75)	1.00	10.36	2.61	-	-
TIDYUP(67)	$\approx 2 * 10^4$	23.30	5.61	5.61	5.61
All(172)	$\approx 8 * 10^3$	15.34	4.46	4.46	5.61

Concluding Figure 3 visualizes our experimental results analysing the runtime improvement of Algorithm 2 with respect to the original GREEDY algorithm [1]. We consider only instances which are solved by both algorithms and aborted the calculation after one hour using a memory limit of 4 GB. Furthermore, we display planning tasks where at least one of the two algorithms was able to solve

that task in time. Overall, Algorithm 2 solves *eight* more planning tasks within one hour computation time. A few instances have a shorter runtime using the original GREEDY algorithm. This can be either traced back to a small state space (ID: 7), where the additional computation of a necessary observation set takes more time then it saves; or it can be due to nondeterministic decisions of the MYND planner (ID: 39). The latter also causes that one planning task (ID: 43) was only solved by the original GREEDY algorithm and was not solved in time by the extended version. Finally, considering the runtime of unsolved instances as (at least) 60 minutes, Algorithm 2 (PRUNEDGREEDY) was on average 526 seconds (over 8.5 minutes) faster then the original GREEDY Algorithm [1].

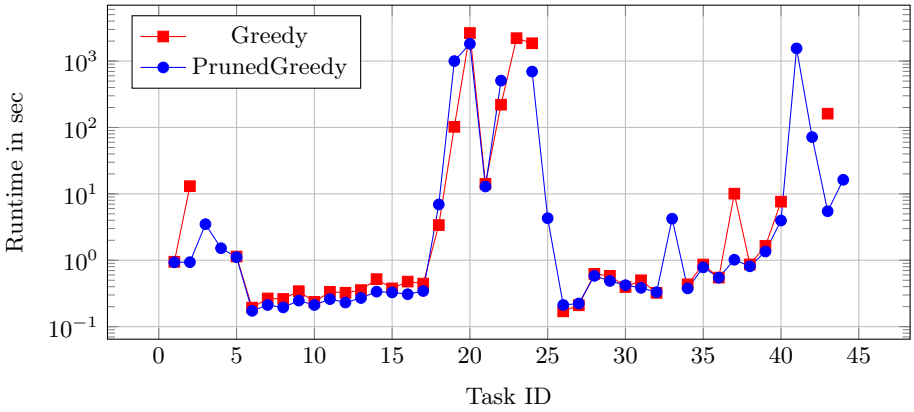

Fig. 3. Overall runtime of the GREEDY Algorithm [1] and Algorithm 2 (PRUNEDGREEDY). Legend: 1-5 = BLOCKSWORLDSENSE, 6-25 = FIRSTRESPONDERS, 26-44 = TIDYUP.

5 Conclusion and Future Work

We introduced the concept of necessary observations and discussed its connection to the recent results of observation minimization. Additionally, we presented an efficient bottom-up algorithm (NOS) for finding a set of necessary observations of a given POND planning task with polynomial runtime in size of the input size of the planning task and considered initial states. Furthermore, we extended the top-down GREEDY algorithm of Mattmüller et al. [1] with the NOS Algorithm which leads to a smaller set of candidates for removal by precomputing a necessary observation set. Our experiments show that the NOS algorithm is a useful extension for the GREEDY algorithm and is superior in terms of runtime.

For future work we plan to compute necessary observation sets in every iteration step of the GREEDY algorithm. In addition, we want to use necessary observations computed by the presented algorithm as a heuristic value for planning which possibly leads to plans with less observations.

References

1. Mattmüller, R., Ortlieb, M., Wacker, E.: Minimizing necessary observations for nondeterministic planning. In: Proceedings of the 37th German Conference on Artificial Intelligence (KI 2014). pp. 309-320 (2014)
2. Huang, W., Wen, Z., Jiang, Y., Wu, L.: Observation reduction for strong plans. In: Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). pp. 1930-1935 (2007)
3. Huang, W., Wen, Z., Jiang, Y., Peng., H.: Structured plans and observation reduction for plans with contexts. In: Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 1721-1727 (2009)
4. Cimatti, A., Pistore, M., Roveri, M., Traverso, P.: Weak, strong, and strong cyclic planning via symbolic model checking. Artificial Intelligence 147(1-2), 35-84 (2003)
5. Helmert, M., Domshlak, C.: LM-cut: Optimal planning with the landmark-cut heuristic. In: Seventh International Planning Competition (IPC 2011). pp. 103-105 (2011)
6. Helmert, M., Domshlak, C.: Landmarks, critical paths and abstraction: What's the difference anyway?. In: Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 162-169 (2009)
7. Rintanen, J.: Complexity of planning with partial observability. In: Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004). pp. 345-354 (2004)