The AAAI-17 Workshop on
Plan, Activity, and Intent Recognition
WS-17-13

Monitoring Plan Optimality using Landmarks and Domain-Independent Heuristics

Ramon Fraga Pereira,† Nir Oren,‡ Felipe Meneguzzi†
†Pontifical Catholic University of Rio Grande do Sul, Brazil
ramon.pereira@acad.pucrs.br
felipe.meneguzzi@pucrs.br
‡University of Aberdeen, United Kingdom
n.oren@abdn.ac.uk

Abstract

When acting, agents may deviate from the optimal plan, either because they are not perfect optimizers or because they interleave multiple unrelated tasks. In this paper, we detect such deviations by analyzing a set of observations and a monitored goal to determine if an observed agent's actions contribute towards achieving the goal. We address this problem without pre-defined static plan libraries, and instead use a planning domain definition to represent the problem and the expected agent behavior. At the core of our approach, we exploit domain-independent heuristics for estimating the goal distance, incorporating the concept of landmarks (actions which all plans must undertake if they are to achieve the goal). We evaluate the resulting approach empirically using several known planning domains, and demonstrate that our approach effectively detects such deviations.

1 Introduction

People are not perfect optimizers and routinely execute sub-optimal plans. Even when aware of the expected optimal plan, they deviate from it for a variety of reasons, ranging from being interrupted by an emergency that needs their immediate attention to simply getting their attention drawn to something else. Consequently, the ability to detect when someone deviates from an expected optimal plan can be useful, allowing for example, to help a person stay focused on a particular goal, or detecting when a person is distracted while performing routine activities. For artificial agents, detecting sub-optimal plan steps is also useful in domains such as plan recognition, where it enables the detection of concurrent plan execution and plan interleaving.

While previous approaches rely on a complex logical formalism (Fritz and McIlraith 2007) to evaluate optimality, we present a technique which exploits domain independent heuristics and planning landmarks, both of which utilize planning domain descriptions to monitor plan optimality. Our technique functions in two stages. In the first stage, we perform landmark extraction (Hoffmann, Porteous, and Sebastia 2004) (landmarks are properties or actions that cannot be avoided to achieve a goal along all valid plans). In the second stage, we perform behavior analysis by using two methods that can work independently or together to monitor plan optimality. The first method takes as input the result of an observed action for estimating the distance[1] (using any domain-independent heuristic) to the monitored goal, and thus, it analyzes possible deviations over earlier observations in a plan's execution. The second method aims to predict which actions may possibly contribute towards goal achievement (*i.e.*, reduce the distance to the goal) in the next observation, and does so by analyzing the closest (estimated distance) landmarks of the current monitored goal. By considering the distance to the monitored goal and identifying actions which do not reduce this distance, we can thus identify which actions do not contribute to goal achievement.

Apart from a formalization of the optimality detection problem, our main contribution is a set of algorithms that use planning techniques (landmarks and domain-independent heuristics) to perform plan optimality monitoring. Experiments over several planning domains show that our approach yields high accuracy at low computational cost to detect non-optimal actions (*i.e.*, which actions do not contribute to achieve a monitored goal) by analyzing both optimal and sub-optimal execution of agent plans.

This paper is structured as follows. In Section 2, we set the context for the paper, describing planning, domain-independent heuristics, plan optimality monitoring problem, and landmarks. Section 3 describes our approach for monitoring plan optimality, and Section 4 evaluates our approach. In Section 5, we survey the literature on approaches that monitor plan execution, such as goal/plan recognition, goal/-plan abandonment detection, and optimality monitoring. Finally, Section 6 concludes this paper by discussing limitations and future directions.

2 Background

2.1 Planning

Planning is the problem of finding a sequence of actions (*i.e.*, a plan) that achieves a particular goal from an initial state. We adopt the terminology of Ghallab *et al.* (Ghallab, Nau, and Traverso 2004) to represent planning domains and instances (also called planning problems) in Definitions 1–5.

[1] In the sense of the amount of effort (*e.g.*, actions) needed to achieve the goal.

Definition 1 (**Predicates and State**). *A predicate is denoted by an n-ary predicate symbol p applied to a sequence of zero or more terms (τ_1, τ_2, ..., τ_n) – terms are either constants or variables. A state is a finite set of grounded predicates (facts) that represent logical values according to some interpretation. Facts are divided into two types: positive and negated facts, as well as constants for truth ($\top$) and falsehood ($\bot$).*

Definition 2 (**Operators and Actions**). *An* operator a *is represented by a triple $\langle$name(a), pre(a), eff(a)$\rangle$: name(a) represents the description or signature of a; pre(a) describes the preconditions of a — the set of predicates that must exist in the current state for a to be executed; eff(a) represents the effects of a which modify the current state. Effects are split into eff(a)$^+$ (i.e., an add-list of positive predicates) and eff(a)$^-$ (i.e., a delete-list of negated predicates). An* action *is a grounded operator instantiated over its free variables.*

Definition 3 (**Planning Domain**). *A* planning domain definition Ξ *is a pair $\langle \Sigma, \mathcal{A} \rangle$, consisting of a finite set of facts Σ and a finite set of actions $\mathcal{A}$.*

Definition 4 (**Planning Instance**). *A* planning instance Π *is encoded by a triple $\langle \Xi, \mathcal{I}, G \rangle$, in which $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the domain definition; $\mathcal{I} \subseteq \Sigma$ is the initial state specification, which is defined by specifying the value for all facts in the initial state; and $G \subseteq \Sigma$ is the goal state specification, which represents a desired subset of facts to be achieved.*

Finally, a plan is the solution to a planning problem.

Definition 5 (**Plan**). *Let $\Pi = \langle \langle \Sigma, \mathcal{A} \rangle, \mathcal{I}, G \rangle$ be a planning instance. A plan π for Π is a sequence of actions $[a_1, a_2, ..., a_n]$ (where $a_i \in \mathcal{A}$) that modifies the initial state $\mathcal{I}$ into one in which the goal state G holds by the successive (ordered) execution of actions in a plan π.*

2.2 Domain-Independent Heuristics

Heuristics are used to estimate the cost to achieve a specific goal (Ghallab, Nau, and Traverso 2004). In classical planning, this estimate is closely related to the number of actions to achieve the goal state from a particular state. In this work, as done in classical planning, we consider that the action cost is $c(a) = 1$ for all $a \in A$. Thus, the cost for a plan $\pi = [a_1, a_2, ..., a_n]$ is $c(\pi) = \Sigma c(a_i) = n$. In general, the use of heuristics can substantially reduce search time by estimating how many actions are needed to achieve a particular goal. When a heuristic never overestimates the cost to achieve a goal, it is called *admissible* and guarantees optimal plans when used with for certain planning algorithms. An heuristic $h(s)$ is admissible if $h(s) \leq h^*(s)$ for all states, where $h^*(s)$ is the optimal cost to the goal from state s, otherwise it is called inadmissible. In this work, we use both admissible and inadmissible domain-independent heuristics for estimating the distance to a monitored goal.

2.3 Plan Optimality Monitoring Problem

We define plan optimality monitoring as the process of monitoring the execution of a plan by an agent to solve a planning instance (Definition 4) and detecting when the agent executes steps that deviate from any one of the optimal plans (Definition 6), and instead executes other actions.

Definition 6 (**Optimal Plan**). *Let $\pi = [a_1, ...a_n]$ be a plan with length $|\pi| = n$ for an instance Π, we say π is optimal, also represented as π^* if there exists no other plan $\pi^<$ such that $|\pi^<| < |\pi^*|$. A planning instance may have multiple optimal plans.*

Intuitively, given a planning instance, we aim to detect exactly which actions during a plan execution do not contribute towards a monitored goal of a planning instance. Formally, we define the task of plan optimality monitoring in Definition 7 and note that in this paper we consider that all actions are observed during a plan execution.

Definition 7 (**Plan Optimality Monitoring Problem**). *A plan optimality monitoring problem is a tuple $T_{\pi^*} = \langle \Xi, G, O \rangle$, in which $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is a planning domain definition, Σ consists of a finite set of facts and $\mathcal{A}$ a finite set of actions; $\mathcal{I}$ as the initial state; G is the monitored goal; and $O = \langle o_1, o_2, ..., o_n \rangle$ is an observation sequence of the plan execution with each observation $o_i \in O$ being an action in the set of actions $\mathcal{A}$ in domain definition Ξ.*

In order to define the solution to a plan optimality monitoring problem, we must define some auxiliary concepts.

Definition 8 (**Plan Commitment**). *Given a set of optimal plans, an agent is committed to a plan π if, given a sequence of observations $o_1, \ldots, o_m$: i) $o_k \in \pi$ where $(1 \leq k \leq m)$; and ii) if $o_k = a_j$, then $\forall i = 1 \ldots j-1$, $a_i \in O$ and a_i occurs before a_{i+1} in O. An observation o_p is non-optimal if the agent is committed to plan π and: $o_p \notin \pi$; or if $o_p = a_j$, a_k has not yet been observed where $k < j$. The solution to a plan optimality monitoring problem is then the set of non-optimal observations.*

Intuitively, the solution to a plan optimality monitoring problem are those observations that do not advance an optimal plan that the agent may be following (is committed to).

As an example of what we aim to detect, consider an instance of the LOGISTICS[2] planning problem shown in Figure 1. This example shows two cities: the city on the left contains locations A through D, and the city on the right contains location E. Locations C and E are airports. The goal within this example is to transport the box at location B to location E. From this planning problem, Table 1 shows two possible plan executions one of which is optimal and the other sub-optimal. In the sub-optimal plan execution, boxes in gray represent actions that do not contribute to goal achievement, *i.e.*, sub-optimal actions. Boxes in light-gray represent actions that must be taken to resume the plan execution to achieve the goal.

2.4 Landmarks

Planning landmarks are necessary properties (actions) that must be true (executed) at some point in every valid plan[3] to achieve a particular goal. Landmarks are often partially

[2]The LOGISTICS domain consists of airplanes and trucks transporting packages between locations (*e.g.*, airports and cities).

[3]A valid plan to achieve a goal is a successive execution of actions (ordered) that modifies an initial state into a state that contains the goal state

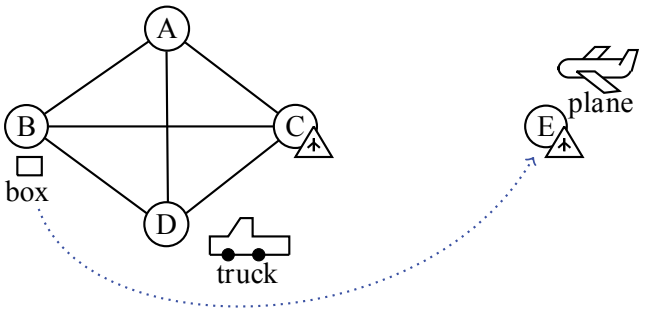

Figure 1: LOGISTICS problem example.

	Optimal Plan
0	(drive TRUCK D B CITY1)
1	(loadTruck BOX TRUCK B)
2	(drive TRUCK B AIRPORT-C CITY1)
3	(unloadTruck BOX TRUCK AIRPORT-C)
4	(fly PLANE AIRPORT-E AIRPORT-C)
5	(loadAirPlane BOX PLANE AIRPORT-C)
6	(fly PLANE AIRPORT-C AIRPORT-E)
7	(unloadAirplane BOX PLANE AIRPORT-E)

	Sub-optimal Plan
0	(drive TRUCK D B CITY1)
1	(loadTruck BOX TRUCK B)
2	(unloadTruck BOX TRUCK B)
3	(drive TRUCK B A CITY1)
4	(drive TRUCK A B CITY1)
5	(loadTruck BOX TRUCK B)
6	(drive TRUCK B AIRPORT-C CITY1)
7	(unloadTruck BOX TRUCK AIRPORT-C)
8	(fly PLANE AIRPORT-E AIRPORT-C)
9	(loadAirPlane BOX PLANE AIRPORT-C)
10	(fly PLANE AIRPORT-C AIRPORT-E)
11	(unloadAirplane BOX PLANE AIRPORT-E)

Table 1: Plan Optimality Monitoring example.

ordered by their pre-requisite dependencies. Hoffman *et al*. (2004) define landmarks as follows.

Definition 9 (**Fact Landmarks**). *Given a planning instance* $\Pi = \langle \Xi, \mathcal{I}, G \rangle$*, a formula* L *is a landmark in* Π *iff* L *is true at some point along all valid plans that achieve* G *from* $\mathcal{I}$*.*

Hoffmann *et al*. (2004) introduce two types of landmarks as formulas: conjunctive and disjunctive landmarks. A conjunctive landmark is a set of facts that must be true together at some point in every valid plan to achieve a goal. A disjunctive landmark is a set of facts in which one of facts must be true at some point in every valid plan to achieve a goal. Within LOGISTICS, an example conjunctive landmark formula is: `(at PLACE-1)` $\wedge$ `(holding BLOCK-A)`; while a disjunctive landmark example is `((at PLACE-1)` $\wedge$ `(holding BLOCK-A))` $\vee$ `((at PLACE-1)` $\wedge$ `(holding BLOCK-B))`.

Landmark extraction involves identifying conjunctive and disjunctive landmarks and the temporal ordering between them. Consider the LOGISTICS example in Figure 1. Listing 1 shows the resulting fact landmarks, while Figure 2

```
Fact Landmarks:
(and (at BOX AIRPORT-E))
(and (at PLANE AIRPORT-E) (in BOX PLANE))
(and (at PLANE AIRPORT-C) (at BOX AIRPORT-C))
(and (at PLANE AIRPORT-E))
(and (at TRUCK D))
(and (in BOX TRUCK) (at TRUCK AIRPORT-C))
(and (at BOX B) (at TRUCK B))
(or (at TRUCK A) (at TRUCK C) (at TRUCK D))
```

Listing 1: Fact landmarks (conjunctive and disjunctive) extracted from the LOGISTICS example.

show their ordering. Note that Figure 2 shows such landmarks ordered by facts that must be true together, with the goal at the top of the figure and earlier landmarks below.

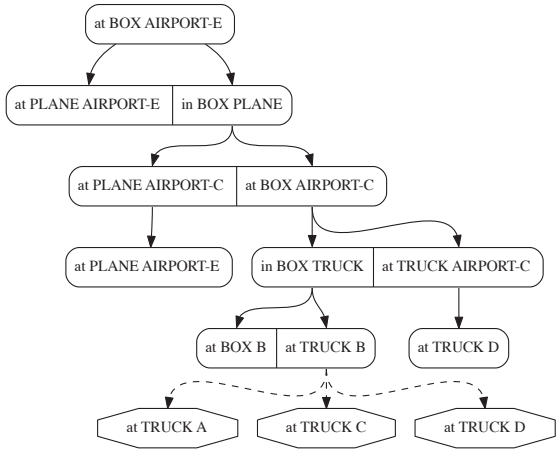

Figure 2: Ordered fact landmarks extracted from the LOGISTICS example from Figure 1. Fact landmarks that must be true together are represented by connected boxes and represent conjunctive landmarks. Disjunctive landmarks are represented by octagonal boxes connected by dashed lines.

3 Plan Optimality Monitoring

We now describe a heuristic for optimality monitoring that uses planning landmarks and domain-independent heuristics. Our approach consists of estimating the distance to the monitored goal for every observed action and then identifying which actions reduce the distance towards the goal, labeling actions that fail do so as deviations.

3.1 Analyzing Plan Execution Deviation

To analyze possible plan execution deviation, we compute the estimated distance to the monitored goal for every state resulting from the execution of an observed action. Given a state s, a heuristic h returns an estimated distance $h(s)$ to the goal state. If observation o_i results in state s_i, we consider a deviation from a plan to occur if $h(s_{i-1}) < h(s_i)$.

The uptick shown in Figure 3 illustrates a deviation detected using the FAST-FORWARD heuristic for the two plan executions shown in Table 1. Note that during the execution of the sub-optimal plan (red), deviations occur for actions leading to time s 2 and 3. By analyzing this plan deviation, we conclude that actions `(unloadTruck BOX TRUCK B)` and `(drive TRUCK B A CITY1)` do not contribute to achieving the goal because they increase the estimated distance to the goal. However, since heuristics may be inaccurate, we utilise landmarks to further validate deviations.

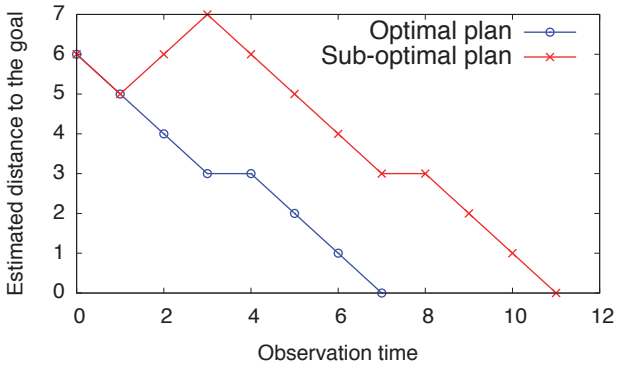

Figure 3: Plan execution deviation example using the FAST-FORWARD heuristic.

3.2 Predicting Non-regressive Actions via Landmarks

Ordered landmarks effectively provide way-points towards the monitored goal, identifying what cannot be avoided on the way to achieving the goal. It should be noted that the initial (and goal state) are themselves landmarks, as all plans begin and terminate in these states. Since all plans should pass through a landmark, we can exploit their presence to predict what actions might be executed next, either to reach the landmark, or to move towards a goal. We use such predictions to check the set of observed actions of a plan execution to determine which actions do not contribute to the monitored goal. We formalize this in Algorithm 1.

To predict which actions could reasonably be executed in the next observation, our algorithm analyzes the closest landmarks by estimating the distance to the landmarks from the current state. The algorithm uses an admissible domain-independent heuristic to estimate the distance to landmarks, namely the MAX-HEURISTIC, which we denote as h_{max}. We consider that the closest fact landmarks are those that return estimated distance $h_{max}(l) = 0$ and $h_{max}(l) = 1$. In this way, the algorithm iterates over a set of ordered fact landmarks $\mathcal{L}$ (Line 3), and, for each landmark l, the MAX-HEURISTIC estimates the distance from the current state to l. If the estimated distance to landmark l is $h_{max}(l)$ = 0 (Line 5), this means that this landmark is in the current state, and the algorithm selects those actions that contain such landmark as a precondition, because these can be executed immediately (Line 6). Otherwise, if the estimated distance to landmark l is $h_{max}(l)$ = 1 (Line 7), this means that this landmark can be reached by executing a single action, and the algorithm selects those actions that are applicable in the current state and contain such landmark as an effect (Line 8). These actions are selected because they reduce the distance to the next landmark, and consequently to the monitored goal. Thus, we can estimate, using the observed plan execution, which actions do not contribute to achieve a goal.

```
(and (at BOX AIRPORT-E))                          = 7
(and (at PLANE AIRPORT-E) (in BOX PLANE))         = 6
(and (at PLANE AIRPORT-C) (at BOX AIRPORT-C))     = 5
(and (at PLANE AIRPORT-E))                        = 0
        - An applicable action is:
                (fly PLANE AIRPORT-E AIRPORT-C)
(and (at TRUCK D))                                = 0
        - An applicable action is:
                (drive TRUCK D B CITY1)
(and (in BOX TRUCK) (at TRUCK AIRPORT-C))         = 3
(and (at BOX B) (at TRUCK B))                     = 1
(or
        (at TRUCK A) = 1
        (at TRUCK C) = 1
        (at TRUCK D) = 0
                - An applicable action is:
                        (drive TRUCK D B CITY1))
```

Listing 2: Predicted upcoming actions for the LOGISTICS example.

Algorithm 1 Computing Non-regressive actions from landmarks.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, δ *current state*, and $\mathcal{L}$ *ordered fact landmarks*.
Output: $\eta_{PActions}$ *set of possible upcoming actions*.

1: **function** NONREGRESSIVEACTIONS(Ξ, δ, $\mathcal{L}$)
2: $\quad \eta_{PActions} \leftarrow \langle\,\rangle$
3: $\quad$ **for** each fact landmark l in $\mathcal{L}$ **do**
4: $\quad\quad Al \leftarrow \langle\rangle$
5: $\quad\quad$ **if** $h_{max}(l) = 0$ **then** $\triangleright$ $h_{max}(l)$ *estimates* l *from* δ.
6: $\quad\quad\quad Al \leftarrow$ all a in $\mathcal{A}$ s.t. $l \in pre(a)$
7: $\quad\quad$ **else if** $h_{max}(l) = 1$ **then**
8: $\quad\quad\quad Al \leftarrow$ all $a \in \mathcal{A}$ s.t $pre(a) \in \delta \wedge l \in eff(a)^{+}$
9: $\quad\quad$ **end if**
10: $\quad\quad \eta_{PActions} := \eta_{PActions} \cup Al$
11: $\quad$ **end for**
12: $\quad$ **return** $\eta_{PActions}$
13: **end function**

To exemplify how our algorithm predicts upcoming actions, let us consider the LOGISTICS problem in Figure 1. If the current state is the initial state, then the algorithm predicts upcoming actions that might be executed as the first observation in the plan execution. As output for this example, Listing 2 shows fact landmarks (on the left); the estimated distance from the initial state to fact landmarks (after the symbol =); and on the bottom of the fact landmarks, which applicable actions our method predicts to be the first observation. Note that, there are fact landmarks for which the estimated distance is $h_{max}(l) = 1$, because there is no applicable action in the initial state to achieve these fact landmarks.

3.3 Detecting Sub-Optimal Steps

We now develop our approach to detect sub-optimal steps, bringing together the methods that we presented before (Subsections 3.1 and 3.2). Algorithm 2 formally describes our planning-based approach to detect sub-optimal plan steps. The algorithm takes as input a plan optimality monitoring problem (Definition 7), *i.e.*, a planning domain, an initial state, a monitored goal, and a set of observed actions as the execution of an agent plan. The algorithm initially computes key information using the landmark extraction algorithm proposed by Hoffman *et al.* (2004). Afterwards, the algorithm analyzes plan execution by iterating over the set of observed actions and applying them, checking which actions do not contribute to the monitored goal. Any such action is then considered to be sub-optimal. When analyzing plan execution deviation (via the distance to the goal) our algorithm can use any domain-independent heuristic. To predict upcoming actions (via landmark consideration), we use the MAX-HEURISTIC because it is admissible and only a short distance must be estimated. Line 10 combines these techniques, labelling a step as sub-optimal if an observed action is not in the set of predicted upcoming actions, and the estimated distance of the current state is greater than the previous one.

We note that our approach iterates over observations, and during each iteration, also iterates over all fact landmarks. Since $|L| \leq |O|$, it's complexity is giving a complexity $O(|O|^2)$ in the worst case. However, we also note that this result ignores the complexity of landmark extraction (which is PSPACE-complete). We overcome this latter restriction through the use of a landmark extraction algorithm (Hoffmann, Porteous, and Sebastia 2004) that extracts only a subset of all possible landmarks.

Algorithm 2 Plan Optimality Monitoring.

Parameters: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, $\mathcal{I}$ *initial state*, G *monitored goal*, and O *observed actions*.
Output: $A_{SubOptimal}$ *as sub-optimal actions*.

1: **function** MONITORPLANOPTIMALITY(Ξ,$\mathcal{I}$,G,O)
2: $A_{SubOptimal} \leftarrow \langle\rangle$ ▷ *Actions that do not contribute to achieve the monitored goal* G.
3: $L \leftarrow$ EXTRACTLANDMARKS($\mathcal{I}$, G)
4: $\delta \leftarrow \mathcal{I}$ ▷ δ *is the current state.*
5: $\eta_{PActions} \leftarrow$ NONREGRESSIVEACTIONS(Ξ, δ, L)
6: $D_G \leftarrow$ ESTIMATEGOALDISTANCE(δ, G) ▷ *A desired domain-independent heuristic to estimate goal* G *from* δ.
7: **for** each observed action o in O **do**
8: $\delta \leftarrow \delta$.APPLY(o)
9: $D'_G \leftarrow$ ESTIMATEGOALDISTANCE(δ, G)
10: **if** $o \notin \eta_{PActions} \wedge (D'_G > D_G)$ **then**
11: $A_{SubOptimal} \leftarrow A_{SubOptimal} \cup o$
12: **end if**
13: $\eta_{PActions} \leftarrow$ NONREGRESSIVEACTIONS(Ξ, δ, L)
14: $D_G \leftarrow D'_G$
15: **end for**
16: **return** $A_{SubOptimal}$
17: **end function**

4 Experiments and Evaluation

We evaluate our plan optimality monitoring approach over several widely used planning domains in the literature[4], most of which are inspired by real-world scenarios. The BLOCKS-WORLD domain consists of a set of stackable blocks, in which goals involve finding a sequence of actions that achieves a final configuration of blocks. The DRIVER-LOG domain consists of drivers that can walk between locations and trucks that can drive between locations, in which goals consist of transporting packages between locations. DEPOTS combines transportation and stacking, in which goals involve moving and stacking packages by using trucks and hoists between depots. EASY-IPC-GRID consists of an agent that moves in a grid from cells to others by transporting keys to open locked locations. The FERRY domain consists of set of cars that must be moved to desired locations using a ferry that can carry only one car at a time. LOGISTICS, described previously, consists of airplanes and trucks transporting packages between locations (*e.g.*, airports and cities). MICONIC involves transporting a number of passengers using an elevator to reach destination floor. SATELLITE involves using one or more satellites to make observations, by collecting data and down-linking the data to a desired ground station. SOKOBAN involves pushing a set of boxes into specified locations in a grid with walls. Finally, ZENO-TRAVEL is a domain where passengers can embark and disembark onto aircraft that can fly at two alternative speeds between locations.

For each of these domains, we selected 15-30 associated non-trivial problem instances, with each problem instance also associated to a set of observations (*i.e.*, plan executions). This set of observations can represent either an optimal or a sub-optimal plan execution. We generate plans (optimal and sub-optimal) using open-source planners, such as BLACKBOX (1998), FAST-DOWNWARD (2011), FF (2001), and LAMA (2010). For sub-optimal plans, we annotated manually the sub-optimal steps, and enumerate how many sub-optimal steps each plan has. These steps consist of actions that do not contribute for achieving the monitored goal, representing steps that our approach aims to detect. We evaluate our approach according to several metrics. *Precision* is the ratio between true positive results, and the sum of true positive and false positive results. True positive results represent the number of sub-optimal actions detected that do not contribute to achieve the monitored goal. False positive results represent the number of actions that our approach labelled as a sub-optimal action, which is in fact an optimal action. *Precision* provides the percentage of positive predictions that is correct. *Recall* is the ratio between true positive results, and the sum of the number of true positives and false negatives. Here, a false negative is a sub-optimal action that is not detected by our approach. *Recall* provides the percentage of positive cases that our approach has detected. The *F1-score* is a measure of accuracy that aims to provide a trade-off between *Precision* and *Recall*. We ran all our experiments on a dual-core Intel Core i5 processor running at 2.5 GHz with 8 GB of RAM.

[4] http://ipc.icaps-conference.org

			h_{adjsum}		$h_{adjsum2}$		$h_{adjsum2M}$	
Domain	$\|O\|$	$\|\mathcal{L}\|$	**Time**	**Precision / Recall / F1-score**	**Time**	**Precision / Recall / F1-score**	**Time**	**Precision / Recall / F1-score**
BLOCKS-WORLD (30)	15.2	20.1	0.25	63.7% / 94.8% / 76.2%	0.19	100% / 74.2% / 85.2%	0.47	74.4% / 91.4% / 82.1%
DRIVER-LOG (20)	20.1	53.6	0.71	100% / 77.7% / 87.5%	0.68	100% / 94.4% / 97.1%	1.33	100% / 100% / 100%
DEPOTS (30)	16.7	64.7	1.34	71.8% / 88.4% / 79.3%	1.22	81.2% / 100% / 89.6%	2.15	75.6% / 93.3% / 83.5%
EASY-IPC-GRID (30)	14.1	48.5	0.81	100% / 96.1% / 98%	0.77	100% / 100% / 100%	0.98	100% / 75% / 85.7%
FERRY (30)	13.8	18.1	0.23	88% / 78.5% / 83.1%	0.18	88% / 78.5% / 83.1%	0.34	80% / 42.8% / 55.8%
LOGISTICS (30)	20.8	24	0.47	100% / 85.7% / 92.3%	0.35	100% / 91.3% / 95.4%	0.89	100% / 91.3% / 95.4%
MICONIC (30)	18.1	19.4	0.29	100% / 86.9% / 93.1%	0.24	100% / 82.6% / 90.4%	0.36	100% / 82.6% / 90.4%
SATELLITE (20)	25.7	60.8	5.41	100% / 26.6% / 42.1%	4.35	87.5% / 46.6% / 60.8%	9.58	88.8% / 53.3% / 66.6%
SOKOBAN (20)	24	76.5	3.45	75% / 75% / 75%	2.26	77.7% / 58.3% / 66.6%	4.13	66.6% / 66.6% / 66.6%
ZENO-TRAVEL (15)	12.2	38.7	1.07	87.5% / 50% / 63.6%	0.86	100% / 92.8% / 96.2%	1.52	100% / 85.7% / 92.3%

Table 2: Plan Optimality Monitoring experimental results (1).

			h_{combo}		h_{ff}		h_{sum}	
Domain	$\|O\|$	$\|\mathcal{L}\|$	**Time**	**Precision / Recall / F1-score**	**Time**	**Precision / Recall / F1-score**	**Time**	**Precision / Recall / F1-score**
BLOCKS-WORLD (30)	15.2	20.1	0.51	63.7% / 94.8% / 76.2%	0.21	100% / 74.2% / 85.2%	0.18	63.7% / 94.8% / 76.2%
DRIVER-LOG (20)	20.1	53.6	1.38	100% / 77.7% / 87.5%	0.74	100% / 94.4% / 97.1%	0.85	100% / 77.7% / 87.5%
DEPOTS (30)	16.7	64.7	2.46	71.4% / 96.1% / 81.9%	1.43	81.2% / 100% / 89.6%	1.39	71.8% / 88.4% / 79.3%
EASY-IPC-GRID (30)	14.1	48.5	1.08	100% / 96.1% / 98%	0.86	100% / 100% 100%	0.79	100% / 96.1% / 98%
FERRY (30)	13.8	18.1	0.36	80% / 78.5% / 83.1%	0.32	80% / 78.5% / 83.1%	0.19	80% / 78.5% / 83.1%
LOGISTICS (30)	20.8	24	1.11	100% / 85.7% / 92.3%	0.55	100% / 91.3% / 95.4%	0.43	100% / 85.7% / 92.3%
MICONIC (30)	18.1	19.4	0.44	100% / 86.9% / 93.1%	0.27	100% / 82.6% / 90.4%	0.21	100% / 86.9% / 93.1%
SATELLITE (20)	25.7	60.8	9.81	100% / 40% / 57.1%	4.94	87.5% / 46.6% / 60.8%	4.53	100% / 26.6% / 42.1%
SOKOBAN (20)	24	76.5	4.28	90.9% / 83.3% / 86.9%	2.22	77.7% / 58.3% / 66.6%	2.07	75% / 75% / 75%
ZENO-TRAVEL (15)	12.2	38.7	1.45	87.5% / 50% / 63.6%	0.99	100% / 92.8% / 96.2%	0.92	87.5% / 50% / 63.6%

Table 3: Plan Optimality Monitoring experimental results (2).

Since our approach can exploit any domain-independent heuristic to analyze plan execution deviation, we evaluated our approach using several admissible and inadmissible heuristics. MAX-HEURISTIC (h_{max}) is an admissible heuristic proposed by Bonet and Geffner in (2001), and is based on delete-list relaxation, in which delete-effects of actions are ignored during calculation of the heuristic cost to a goal. This calculation is the cost of a conjunctive goal, which represents the maximum cost to achieve each of the individual facts. SUM (h_{sum}) is an inadmissible heuristic proposed by Bonet and Geffner in (2001), and works in a similar manner to MAX-HEURISTIC, but is more informative. ADJUSTED-SUM (h_{adjsum}) is an inadmissible heuristic (2002) that improves SUM by considering both negative and positive interactions among facts; ADJUSTED-SUM2 ($h_{adjsum2}$) is an inadmissible heuristic (2002) that improves the ADJUSTED-SUM by combining the computation of the SET-LEVEL heuristic and the relaxed plan. The SET-LEVEL heuristic estimates the cost to a goal by returning the level in the planning graph where all facts of the goal state are reached without any mutex free (2000). ADJUSTED-SUM2M ($h_{adjsum2M}$) is an inadmissible heuristic (2002) that improves ADJUSTED-SUM2. COMBO (h_{combo}) is an inadmissible heuristic (2002) that combines the computation of the ADJUSTED-SUM and SET-LEVEL heuristics. Finally, FAST-FORWARD (h_{ff}) is a well-known inadmissible heuristic in the planning community (Hoffmann and Nebel 2001) that relies on state-space search and estimates the goal distance by using delete-list relaxation.

Tables 2 and 3 show the experimental results of our approach over the selected domains and heuristics. Each table row details results for a different domain showing averages for the number of observations $|O|$ across problem instances; number of extracted landmarks $|\mathcal{L}|$; monitoring time (in seconds); *Precision*; *Recall*, and *F1-score*). The high average number of observations made ($|O|$), ranging between 12.2 and 76.5, indicates that all plans we analyze are non-trivial in complexity. The results show that for the DRIVER-LOG and EASY-IPC-GRID domains our approach yields perfect results (100% for all metrics) when using an appropriate heuristic (ADJUSTED-SUM2M and FAST-FORWARD respectively). Apart from the SATELLITE domain that under-performs for all metrics, our approach is near-perfect for monitoring optimality while having low runtime and yielding very good results with different heuristics. We can also see that some heuristics outperform others for the same domain, for instance, ADJUSTED-SUM2M underperforms in the FERRY domain. For almost all evaluated domains (except SATELLITE) we obtain high *F1-Scores* and perfect or near-perfect accuracy (depending on the heuristic). Thus, our approach effectively detects sub-optimal plan steps in deterministic planning domains.

5 Related Work

To the best of our knowledge, the most recent prior work that monitors plan optimality was developed by Fritz and McIlraith (2007). This work formalizes the problem of monitoring plan optimality by using situation calculus, a logical

formalism to specify and reason about dynamical systems. Fritz and McIlraith seek to determine whether an execution follows an optimal plan, but — unlike our work — do not aim to determine which actions are responsible for deviation from an optimal plan.

Regarding planning-based approaches that monitor the execution of agent plans, we note four goal/plan recognition approaches. Ramírez and Geffner (2009) propose the use of planning techniques (*i.e.*, modified planners and heuristics) to eliminate goals from a candidate set due to an increasing estimated distance from the current state. Follow-up work (Ramírez and Geffner 2010) proposes a probabilistic plan recognition approach using off-the-shelf planners. Pattison and Long (2010) develop IGRAPH, a probabilistic heuristic-based goal recognition over planning domains. IGRAPH uses heuristic estimation and domain analysis to recognize which goal an observed agent is pursuing. Most recently, Pereira *et al.* (2016; 2017) develop goal/plan recognition approaches that use planning landmarks to build heuristics for recognizing goals from observations.

6 Conclusions

In this paper, we have introduced the plan optimality monitoring problem and developed a domain-independent approach to solving this problem. Our heuristic uses planning techniques, but obviating the need to execute a full planning algorithm by using landmarks and domain-independent heuristics. The resulting approach provides a basis from which numerous plan recognition and plan abandonment detection approaches can be developed and improved, since it provides useful information that can be used in planning-based approaches for recognizing goals and plans. It can also be used to repair sub-optimal plans, *e.g.*, by detecting which parts of the plan is sub-optimal, and then improve it.

As we show in our experiments and evaluation, our approach yields good results in detecting sub-optimal plan steps by dealing with realistic well-known deterministic planning domains. As future work, we intend to use more modern heuristics (Helmert and Domshlak 2009; Scherrer, Pommerening, and Wehrle 2015), and explore other planning techniques, such as symmetries in planning (Shleyfman et al. 2015), and temporal landmarks (Karpas et al. 2015).

References

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

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 144–151.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning - Theory and Practice.* Elsevier.

Helmert, M., and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Helmert, M. 2011. The Fast Downward Planning System. *Computing Research Repository (CoRR)* abs/1109.6051.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)* 14(1):253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research (JAIR)* 22(1):215–278.

Karpas, E.; Wang, D.; Williams, B. C.; and Haslum, P. 2015. Temporal landmarks: What must happen, and when. In *ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, 138–146.

Kautz, H., and Selman, B. 1998. BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In *Proceedings of Planning as Combinatorial Search (AIPS-98), Pittsburgh, Pennsylvania, USA, AAAI Press*, 58–60.

Nguyen, X., and Kambhampati, S. 2000. Extracting Effective and Admissible State Space Heuristics from the Planning Graph. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, 798–805.

Nguyen, X.; Kambhampati, S.; and Nigenda, R. S. 2002. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence* 135(1-2):73–123.

Pattison, D., and Long, D. 2010. Domain Independent Goal Recognition. In Ågotnes, T., ed., *STAIRS*, volume 222 of *Frontiers in Artificial Intelligence and Applications*, 238–250. IOS Press.

Pereira, R. F., and Meneguzzi, F. 2016. Landmark-Based Plan Recognition. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-Based Heuristics for Goal Recognition. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2017)*, To Appear.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Ramírez, M., and Geffner, H. 2009. Plan Recognition as Planning. In Boutilier, C., ed., *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, IJCAI 2009.*, 1778–1783.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39(1):127–177.

Scherrer, S.; Pommerening, F.; and Wehrle, M. 2015. Improved Pattern Selection for PDB Heuristics in Classical Planning (Extended Abstract). In *Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015*.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *AAAI 2015*, 3371–3377.