A Plan Optimality Monitoring Approach to Detect Commitment Abandonment

Ramon Fraga Pereira†, Nir Oren‡, and Felipe Meneguzzi†

†School of Computer Science
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil

‡Department of Computing Science
University of Aberdeen
Aberdeen, United Kingdom

ramon.pereira@acad.pucrs.br†
n.oren@abdn.ac.uk‡
felipe.meneguzzi@pucrs.br†

Abstract Assessing whether an agent has abandoned a goal is important when multiple agents are trying to achieve joint goals, or when agents commit to achieving goals for each other. Making such an inference for a single goal by observing only plan traces is not a trivial task because agents often deviate from the optimal plans for various reasons, including the pursuit of multiple goals or the inability to act optimally. In this paper, we develop an approach that uses plan optimality monitoring techniques to determine whether an agent will honour a commitment. Specifically, to determine commitment abandonment, we use these techniques with planning fact partitions (*e.g.*, dead-ends). We empirically show, for a number of representative domains, that our approach yields very high accuracy and detects commitment abandonment in nearly all cases.

Keywords: Commitment, Goal, Abandonment, Plan Execution

1 Introduction

Autonomous agents generate and execute plans in pursuit of goals, which can be intrinsic to the agent or acquired through cooperative exchanges with others. However, when carrying out its plans, an agent may execute actions that are not optimal with regards to an individual goal due to factors including indecision (*e.g.*, interleaving actions from two plans for the same goal); an imperfect planning mechanism; interleaving concurrent plans for multiple goals; and — in the most extreme case — goal or plan abandonment. Determining whether such actions have occurred is often important, especially when goal delegation has taken place; where one agent is obliged or committed to achieve a goal; and where agents are coordinating plan execution in pursuit of a joint goal. In all these cases, determining that an agent is acting sub-optimally allows other agents to re-plan, apply sanctions, or otherwise mitigate against the effects of the failure to achieve a certain state-of-affairs in a timely fashion.

We define commitment abandonment as a situation in which an agent switches from executing the actions of one plan that achieves the consequent it is committed to, to executing actions from another plan. To monitor agent behaviour and detect which actions do not contribute (*i.e.*, sub-optimal steps) to achieve a monitored goal, and hence indicate commitment abandonment, we use plan optimality monitoring techniques from the literature [14]. These techniques exploit domain-independent heuristics [4] (to analyse plan deviation at every plan step by estimating the distance to the monitored goal) and planning landmarks [7] (properties or actions that cannot be avoided to achieve a goal from an initial state). Like [14], we assume that during plan execution, all actions performed by an agent are visible, and that a goal hypothesis and a domain definition are available.

The main contribution of this paper is twofold. First, we apply plan optimality monitoring techniques to compute whether a plan is sub-optimal (which actions in this plan are sub-optimal). Second, we leverage these techniques to identify whether an agent is individually committed to achieve a particular goal, allowing us to identify whether this agent will honour a social commitment [18]. We bring these techniques together in Section 4, formalising the problem of commitment abandonment and its relation to an individual's commitment to a plan (in the BDI sense), and using it to detect whether an agent has abandoned a social commitment. In this latter context, our technique assumes that a debtor agent agrees to a certain abstract quality of execution (that some action steps might not be optimal) when they accept a commitment towards a creditor agent. This allows the creditor to ascertain, at runtime, whether and at what point in time the debtor fails to honour the commitment. Experiments over several domains (Section 5) show that our approach yields high accuracy at low computational cost to detect, in nearly all cases, whether a debtor abandoned a commitment.

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.* [4] to represent planning domains and instances (also called planning problems). A *state* is a finite set of facts that represent logical values according to some interpretation in an environment. *Facts* can be either positive or negated instantiated predicates. 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). An *operator* is represented by a triple $a = \langle name(a), pre(a), \mathit{eff}(a)\rangle$ where $name(a)$ represents the description or signature of a; $pre(a)$ describes the preconditions of a — a set of facts or predicates that must exist in the current state for a to be executed; $\mathit{eff}(\mathrm{a}) = \mathit{eff}(a)^{+} \cup \mathit{eff}(a)^{-}$ represents the effects of a, with $\mathit{eff}(a)^{+}$ an *add-list* of positive facts or predicates, and $\mathit{eff}(a)^{-}$ a *delete-list* of negative facts or predicates.

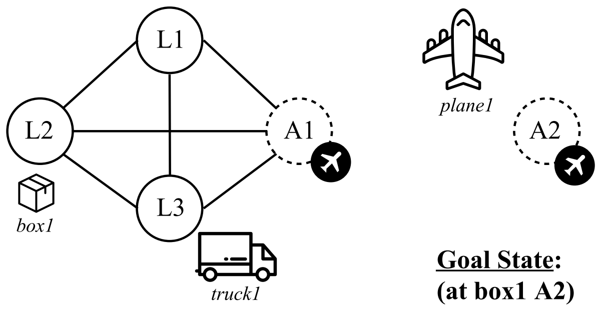


Figure 1: LOGISTICS problem example.

When an operator is instantiated over its free variables, it is called an *action*. A *planning instance* is represented by a triple $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, 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; $\mathcal{I} \subseteq \Sigma$ is the initial state; and $G \subseteq \Sigma$ is the goal state. A *plan* is a sequence of actions $\pi = \langle a_1, a_2, ..., a_n \rangle$ that modifies the initial state $\mathcal{I}$ into one in which the goal state G holds by the successive execution of actions in a plan π. While actions have an associated cost, as in classical planning, we assume that this cost is 1 for all actions. A plan π is considered optimal if its cost, and thus length, is minimal.

Automated planners often exploit heuristics which estimate the cost to achieve a specific goal from some state [4]. In this work, as done in classical planning, we consider that all actions have equal cost, making the cost of a plan equal to its length. When a heuristic never overestimates the cost to achieve a goal, it is called *admissible* and guarantees optimal plans when used with certain planning algorithms. A 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. Here, we use both admissible and inadmissible domain-independent heuristics for estimating the distance to a particular goal.

2.2 Landmarks

Planning landmarks are necessary properties (actions) that must be true (executed) at some point in every valid plan[1] to achieve a particular goal. Landmarks are often partially ordered by their pre-requisite dependencies. Hoffman *et al.* [7] define landmarks as follows.

Definition 1 (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 a goal state G from an initial state $\mathcal{I}$.*

[1] 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

Listing 1.1: Fact landmarks (conjunctive and disjunctive) extracted from the LOGISTICS example.

```
Fact Landmarks:
(and (at BOX1 A2))
(and (at PLANE1 A2) (in BOX1 PLANE1))
(and (at PLANE1 A1) (at BOX1 A1))
(and (at PLANE1 A2))
(and (at TRUCK1 L3))
(and (in BOX1 TRUCK1) (at TRUCK1 A1))
(and (at BOX1 L2) (at TRUCK1 L2))
(or (at TRUCK1 L1) (at TRUCK1 A1) (at TRUCK1 L3))
```

Hoffmann *et al.* [7] describe both conjunctive and disjunctive landmarks. A conjunctive landmark is a set of facts that must be true together at some state in every valid plan to achieve a goal. A disjunctive landmark is a set of facts in which one fact must be true at some state in every valid plan to achieve a goal. The process of landmark extraction both identifies conjunctive and disjunctive landmarks, and determines the temporal ordering between them (*i.e.*, identifies which landmark occurs before which). As an example of landmarks and their orderings, 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 L1, L2, L3 and an airport (A1), and the city on the right that contains another airport (A2). The goal within this example is to transport an item (box1) at location L2 to airport A2. Listing 1.1 shows the resulting fact landmarks, while Figure 2 show their ordering (edges show that a source formula must hold after its target). Note that a goal is considered to be a conjunctive landmark. From Figure 2, we see that the second landmark (stating that any valid plan must have box1 within plane1, and that plane1 must be at airport A2) must occur before the goal is achieved, and that before the box is within the plane, plane1 must be at A1, and so must box1. In this paper, we use landmarks to monitor a set of ordered facts (or actions) that cannot be avoided to achieve a goal, and therefore, if such ordered facts are not being achieved according to their ordering in a plan execution, we might infer that this plan execution is not contributing towards goal achievement.

2.3 Fact Partitioning

Pattison and Long [13] classify facts into mutually exclusive partitions to reason about whether certain observations are likely to be goals for goal recognition. Their classification relies on the fact that, in some planning domains, predicates may provide additional information that can be extracted by analyzing preconditions and effects in operator definition. We use this classification to infer whether certain observations are consistent with a particular goal. The fact partitions we use are defined as follows.

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

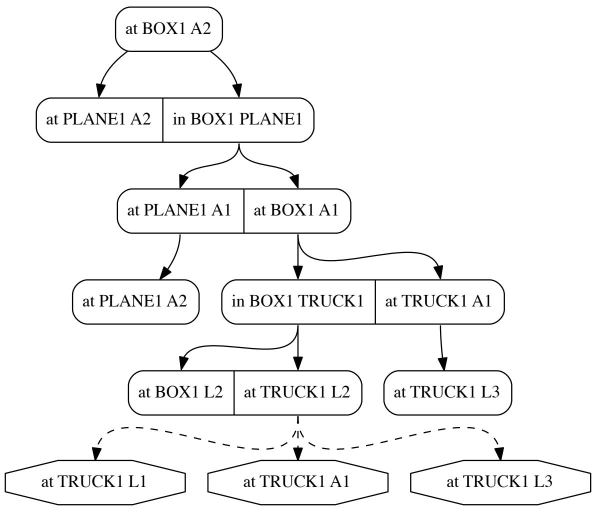


Figure 2: Ordered fact landmarks extracted from the LOGISTICS example in Figure 1. Fact landmarks that must be true together are represented by connected boxes and represent conjunctive landmarks. Disjunctive landmarks are represented by octagonal boxes connected by dashed lines.

Definition 2 (Strictly Activating). *A fact f is strictly activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, such that $f \notin$ eff(a)$^+ \cup$ eff(a)$^-$. Furthermore, $\exists a \in \mathcal{A}$, such that $f \in$ pre(a).*

Definition 3 (Unstable Activating). *A fact f is unstable activating if $f \in \mathcal{I}$ and $\forall a \in \mathcal{A}$, $f \notin$ eff(a)$^+$ and $\exists a \in \mathcal{A}, f \in$ pre(a) and $\exists a \in \mathcal{A}$, $f \in$ eff(a)$^-$.*

Definition 4 (Strictly Terminal). *A fact f is strictly terminal if $\exists a \in \mathcal{A}$, such that $f \in$ eff(a)$^+$ and $\forall a \in \mathcal{A}$, $f \notin$ pre(a) and $f \notin$ eff(a)$^-$.*

A *Strictly Activating* fact appears as an operator's precondition, and does not appear as an add or delete effect in any operator definition. Therefore, unless defined in the initial state, this fact can never be added or deleted by an operator. An *Unstable Activating* fact appears as both a precondition and a delete effect in two operator definitions, so once deleted, this fact cannot be re-achieved. The deletion of an unstable activating fact may prevent a plan execution from achieving a goal. A *Strictly Terminal* fact does not appear as a precondition of any operator definition, and once added, cannot be deleted. For some planning domains, this kind of fact is the most likely to be in the set of goal facts, because once added in the current state, it remains true until the final state.

The fact partitions that we can extract depend on the planning domain definition. For instance, from the DRIVER-LOG domain[3], it is not possible to

[3] DRIVER-LOG domain consists of drivers and trucks that can transport and stack packages between locations.

extract any fact partitions. However, it is possible to extract fact partitions from the EASY-IPC-GRID domain[4], such as *Strictly Activating* and *Unstable Activating* facts. We exploit fact partitions to obtain additional information on fact landmarks. For example, consider an *Unstable Activating* fact landmark L_{ua}, so that if L_{ua} is deleted from the current state, then it cannot be re-achieved. We can trivially determine that goals for which this fact is a landmark are unreachable (and consequently detect that there is no available optimal or sub-optimal plan to achieve this goal), because there is no available action that achieves L_{ua} again.

3 Monitoring Plan Optimality

Pereira *et al.* [14] have developed a plan optimality monitoring approach that uses landmarks and domain-independent heuristics. Intuitively, this approach aims to detect which actions in the execution of an agent plan do not contribute to the plan for achieving the monitored goal.

Analysing Plan Execution Deviation: We now describe an algorithm that Pereira *et al.* [14] have proposed to analyse plan execution deviation. This algorithm uses domain-independent heuristics to estimate the distance to the monitored goal for every observed action in an observation sequence (Definition 5), and infer whether there is any deviation between them. To do so, we leverage this algorithm and compute the estimated distance to the monitored goal for every state resulting from the execution of an observed action. Note that we assume full plan observability in the sense that no actions are missing from the observations from the initial state up to a time point. For example, if the optimal plan to goal G has 10 action steps, and we observe just 2 actions in the observation sequence, then $O_G = \langle o_1 \prec o_2 \rangle$ corresponds exactly to the two first actions in the plan.

Definition 5 (Observation Sequence). *Let O be a sequence $\langle o_1, o_2, ..., o_n \rangle$ of observations of a plan's execution with each observation $o_i \in O$ corresponding to an action in the set of actions $\mathcal{A}$ in domain definition Ξ.*

Given a state s, a heuristic h returns an estimated distance $h(s)$ to the goal state. If the action corresponding to observation o_i results in state s_i, we consider a deviation from a plan to occur if $h(s_{i-1}) < h(s_i)$. A deviation here may indicate that the agent behaviour is unstable, or that the agent is performing concurrent plans. The up-tick shown in Figure 3 illustrates a deviation detected using the FAST-FORWARD heuristic [6] for two different plan executions. These two plan executions (an optimal plan – blue, and a sub-optimal plan – red) are plans that achieve the goal state from the initial state in Figure 1. During the execution of the sub-optimal plan (red), deviations occur for actions leading at the observation time 2 and 3. By analysing this plan deviation, we conclude that these actions do not contribute to achieve the goal because they increase the estimated distance

[4] EASY-IPC-GRID domain consists of an agent that moves in a grid from cells to others transporting keys to open locked locations.

to the goal state. However, since heuristics may be inaccurate, we use landmarks to build a further condition of sub-optimality, predicting actions that achieve next landmarks, and consequently the monitored goal state.

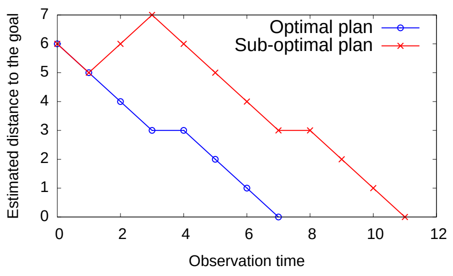


Figure 3: Plan execution deviation example using the FAST-FORWARD heuristic.

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. Since all valid plans that achieve a goal state from an initial state 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. Like Pereira *et al.* [14], 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.

To predict which actions could reasonably be executed in the next observation, Pereira *et al.* [14] have developed an algorithm that estimates the distance to the closest landmarks from the current state. This algorithm uses an admissible domain-independent heuristic to estimate these distances, namely the MAX-HEURISTIC[5], which we denote as h_{max}. We use the MAX-HEURISTIC because it is admissible and only a short distance must be estimated. Pereira *et al.* [14] consider that the closest fact landmarks (either conjunctive or disjunctive landmarks) are those in the set of extracted landmarks $\mathcal{L}$ that return estimated distance $h_{max}(l) = 0$ and $h_{max}(l) = 1$, namely:

– For every fact landmark $l \in \mathcal{L}$ in which the estimated distance is 0, we select those actions $a \in \mathcal{A}$ such that $l \in pre(a)$; and
– For every fact landmark $l \in \mathcal{L}$ in which the estimated distance is 1, we select those actions $a \in \mathcal{A}$ such that $pre(a) \in$ current state and $l \in \mathit{eff}(a)^{+}$;

[5] MAX-HEURISTIC (h_{max}) is an admissible heuristic proposed by Bonet and Geffner in [2], 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.

Listing 1.2: Predicted upcoming actions for the LOGISTICS example.

```
(and (at BOX1 A2))                          = 7
(and (at PLANE1 A2) (in BOX1 PLANE1))       = 6
(and (at PLANE1 A1) (at BOX1 A1))           = 5
(and (at PLANE1 A2))                        = 0
        - An applicable action is:
                (fly PLANE1 A2 A1)
(and (at TRUCK1 L3))                        = 0
        - An applicable action is:
                (drive TRUCK1 L3 L2 CITY1)
(and (in BOX1 TRUCK1) (at TRUCK1 A1))       = 3
(and (at BOX1 L2) (at TRUCK1 L2))           = 1
(or
        (at TRUCK L1) = 1
        (at TRUCK A1) = 1
        (at TRUCK L3) = 0
                - An applicable action is:
                        (drive TRUCK1 L3 L2 CITY1))
```

These actions obtained from the closest landmarks may reduce the distance to the monitored goal and next landmarks. We call these actions as predicted actions, actions that could move towards the monitored goal and next landmarks.

To exemplify how this algorithm predicts upcoming actions, 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 1.2 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 this algorithm predicts to be the first observation. Note that there are fact landmarks for which the estimated distance is $h_{max}(l) = 1$ and there is not any predicted actions for these fact landmarks, it happens because there is no applicable action in the initial state to achieve these fact landmarks.

Detecting Sub-Optimal Steps: We now describe how Pereira *et al.* [14] uses the approaches described above to detect sub-optimal actions during a plan execution. As input, their plan optimality monitoring approach takes as input a planning domain definition, an initial state, a monitored goal, and an observation sequence (*i.e.*, set of observed actions) as the execution of an agent plan (Definition 5). The approach initially computes landmarks using the algorithm proposed by Hoffman *et al.* [7]. Afterwards, the algorithm iterates over the set of observed actions O and applies them, checking which actions do not contribute to achieve the monitored goal. Any such action is then considered to be sub-optimal. Basically, Pereira *et al.* [14] combines the techniques described above (*Analysing Plan Execution Deviation* and *Predicting Non-regressive Actions via Landmarks*), by labelling an observed action step as sub-optimal according to the following condition:

– An observed action o in O is considered sub-optimal if: $o \notin$ set of predicted actions AND $(h(s_{i-1}) < h(s_i))$.

4 Detecting Commitment Abandonment

Commitments have been used in multi-agent systems to enable autonomous agents to communicate and coordinate successfully to achieve a particular goal [10,19,1]. A commitment C(DEBTOR, CREDITOR, antecedent, consequent) formalizes that the agent DEBTOR commits to agent CREDITOR to bring about the consequent if the antecedent holds. Here, the antecedent and consequent conditions are conjunctions or disjunctions of events and possibly other commitments. In this work, we aim to monitor the DEBTOR's behaviour (*i.e.*, sequence of actions) to detect if this agent is individually committed to carrying out a plan to achieve the consequent for the CREDITOR.

In this section, we apply our approach for plan optimality monitoring to infer when agents are likely to abandon commitments to each other. We formally define the commitment abandonment problem and then develop an approach to efficiently solve this problem using fact partitions (Subsection 2.3) and the techniques from Section 3.

4.1 Commitment Abandonment Problem

We define commitment abandonment as a situation in which an agent switches from executing the actions of one plan that achieves the consequent it is committed to, by executing actions from another plan. This plan may achieve other goals, including the consequent of other commitments. Actions in a plan that do not contribute to achieve the consequent of a commitment may indicate that the debtor agent is likely to abandon the commitment that it is committed to its creditor. Reasons why an agent abandons a commitment can include dealing with conflicting commitments, in this case, the agent decides which commitment is more important given its current situation.

Here, we take inspiration in earlier work [1,10] that connects commitments to planning, so the domain definition Ξ represents the environment where agents can interact and act, *i.e.*, Σ is the set of environment properties and $\mathcal{A}$ is a set of available actions. Now, consider a commitment C(DEBTOR, CREDITOR, At, Ct): in order for a DEBTOR to achieve the consequent Ct from the antecedent At we define that: the antecedent At must be in the initial state $\mathcal{I}$, *i.e.*, At $\subseteq \mathcal{I}$; and the consequent Ct is the goal G. Thus, a plan π for C(DEBTOR, CREDITOR, At, Ct) is a sequence of actions $[a_1, a_2, ..., a_n]$ (where $a_i \in \mathcal{A}$) that modifies the state At $\subseteq \mathcal{I}$ into one where Ct holds by the successive (ordered) execution of actions in a plan π.

To decide if a debtor agent will abandon a commitment, we monitor its behaviour in an environment by observing its successive execution of actions. This successive execution of actions represents an observation sequence (Definition 5) that should achieve a consequent from an antecedent. When a DEBTOR settles to commit to an agent CREDITOR to bring about the consequent of a commitment, the DEBTOR should individually commit to achieving such a consequent state, and to achieve such state, the DEBTOR has to execute a plan. An observer does not have access to the DEBTOR's internal state, and consequently to what plan it has committed to. Therefore, when there are multiple optimal plans, we need to be able to determine which of those plans the DEBTOR is pursuing. Thus, in Definition 6, we formally define an individual commitment from an observer's point of view.

Definition 6 (Individual Commitment). *Given a set of plans, a DEBTOR agent is individually committed to a plan π if, given a sequence of observations $o_1, \ldots, o_n$: i) $o_k \in \pi$ where $(1 \leq k \leq n)$; 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 does not contribute to achieve a consequent Ct if the DEBTOR 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 $k < j$.*

Finally, using the notion of individual commitment, we formally define a commitment abandonment problem over a planning theory in terms of a large enough deviation from such an individual commitment in Definition 7. Note that with Definition 6, we can now think of deviations from observations that constitute a strict sub-sequence of any optimal plan that start with the initial state, allowing an agent to infer abandonment at any point in a partial plan execution.

Definition 7 (Commitment Abandonment Problem). *A commitment abandonment problem is a tuple $CA = \langle \Xi, C, \mathcal{I}, O, \theta \rangle$, in which Ξ is a planning domain definition; C is the commitment, in which C(DEBTOR, CREDITOR, At, Ct), DEBTOR is the debtor, CREDITOR is the creditor, At is the antecedent condition, and Ct is the consequent; $\mathcal{I}$ as the initial state (s.t., At $\subseteq \mathcal{I}$), the state we start the monitoring process; O is an observation sequence of the plan execution with each observation $o_i \in O$ being an action from domain definition Ξ; and θ is a threshold that represents the percentage of actions (in relation to an individually committed plan, Definition 6) in an observation sequence that do not contribute to achieving Ct from At $\subseteq \mathcal{I}$ in which the DEBTOR can execute in O.*

The solution for a commitment abandonment problem is whether an observation sequence O has deviated more than θ from the optimal plan to achieve the consequent Ct of commitment C.

4.2 Detecting Commitment Abandonment via Plan Optimality Monitoring

We now bring together the techniques developed in Section 3. To detect commitment abandonment, we infer sub-optimal steps using the techniques from Section 3 and use the concept of fact partitions from Section 2.3. Once we observe evidence of such fact partitions in the observations we can determine that a goal can no longer possible be achieved.

Algorithm 1 formalizes our approach to solve a commitment abandonment problem. The algorithm takes as input a *CA* tuple and returns whether a commitment has been abandoned, based on whether one of the following occurs during plan execution: (1) if *Strictly Activating* facts that we extracted are not in the initial state (Line 3); (2) if we observe the evidence of any *Unstable Activating* and *Strictly Terminal* facts during the execution of actions in the observations (Line 8); or (3) if the number of sub-optimal steps are greater than the threshold θ (*i.e.*, the percentage of actions away from optimal execution that the creditor allows the debtor to deviate in achieving the consequent state) defined by the creditor (Line 10). Note that the DetectSubOptimalSteps($\Xi, \mathcal{I}, Ct, O$) function corresponds to the approach described in Section 3. If none of these conditions hold, the debtor is considered to remain committed to achieving the consequent state of the commitment. Note that in condition (2), the presence of predicates from two of the fact partitions can determine that the monitored goal (or consequent) is unreachable, because there is no available action that can make the facts hold.

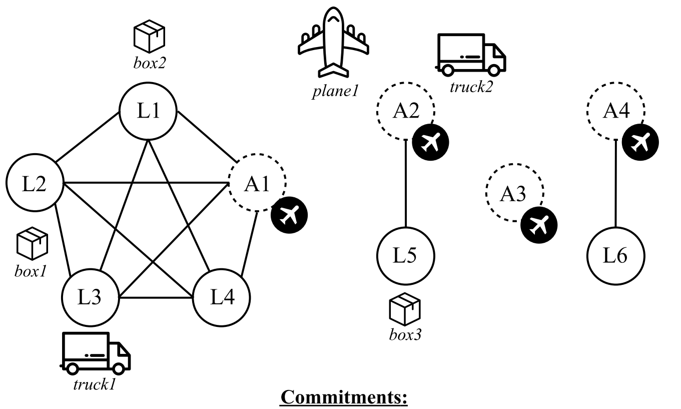
Algorithm 1 Detecting Commitment Abandonment.

Input: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, *At antecedent condition* ($At \subseteq \mathcal{I}$), *Ct consequent condition*, $\mathcal{I}$ *initial state*, *O observation sequence*, and θ *threshold.*

```
 1: function HasAbandoned($\Xi$, $At$, $Ct$, $\mathcal{I}$, $O$, $\theta$)
 2:     $\langle F_{sa}, F_{ua}, F_{st} \rangle \leftarrow$ PartitionFacts($\Sigma, \mathcal{A}$)
 3:     if $F_{sa} \cap (At \subseteq \mathcal{I}) = \emptyset$ then
 4:         return true                                ▷ Ct is no longer possible.
 5:     $\delta \leftarrow \mathcal{I}$                              ▷ Initialasing current state $\delta$.
 6:     for each observed action $o$ in $O$ do
 7:         $\delta \leftarrow \delta$.Apply($o$)    ▷ Executing the observed action $o$ in the current state $\delta$,
            i.e., $(\delta - \mathit{eff}(o)^-) \cup \mathit{eff}(o)^+$
 8:         if $(F_{ua} \cup F_{st}) \subseteq (\delta)$ then
 9:             return true                            ▷ Ct is no longer possible.
10:     $A_{SubOptimal} \leftarrow$ DetectSubOptimalSteps($\Xi$, $\mathcal{I}$, $Ct$, $O$)
11:     if $A_{SubOptimal} > (\theta * |O|)$ then
12:         return true                  ▷ Debtor has abandoned the commitment.
13:     return false               ▷ Debtor may still be committed to achieve Ct.
```

4.3 Working Example

To exemplify how our approaches detect sub-optimal steps and determine commitment abandonment, consider the Logistics problem example shown in Figure 4. This example formalizes two commitments: C1 represents that the debtor agent `TRUCK1` is committed to the creditor agent `PLANE1` to bring about the consequent `(at BOX3 L1)` when the antecedent `(at BOX3 A1)` becomes true; and C2 represents that the debtor agent `PLANE1` is committed to the creditor agent `TRUCK1` to bring about the consequent `(and (at BOX1 A3) (at BOX2 A4))` when the antecedent `(and (at BOX1 A1) (at BOX2 A1))` becomes true. Assuming that for C1 the threshold θ is 0%, and for C2 the threshold θ is 30%, Tables 1 and 2 show observed actions that we observe for C1 and C2, respectively. Rows in



Commitments:

C1: C(TRUCK1, PLANE1, (at BOX3 A1), (at BOX3 L1))

C2: C(PLANE1, TRUCK1, (and (at BOX1 A1) (at BOX2 A1)), (and (at BOX1 A3) (at BOX2 A4)))

Figure 4: LOGISTICS working example.

grey represent sub-optimal actions, and rows without a number (*i.e.*, -) represent actions executed by the creditor agent, actions that are going to achieve the antecedent state.

From the observation sequence shown in Table 1 and the threshold $\theta = 0\%$, our approach returns that the observations at time 1 and 2 are sub-optimal actions, and therefore debtor agent TRUCK1 has abandoned commitment C1, since $\theta = 0\%$ (*i.e.*, the creditor does not allow any deviation), and the agent has executed two actions that do not contribute to achieve the consequent state of C1. The observed action at time 3 is an optimal action because the agent is moving towards the location L1, where it must unload BOX3.

Now consider the observation sequence in Table 2 and a threshold $\theta = 30\%$. While our approach returns that the observations at time 3 and 4 are sub-optimal actions, the threshold (which allows 2.7 sub-optimal actions) means that the debtor agent PLANE1 is considered to remain committed to achieve the consequent of C2.

5 Experiments and Evaluation

We empirically evaluated our approaches (plan optimality monitoring and commitment abandonment) over several widely used planning domains, most of which are inspired by real-world scenarios. The DRIVER-LOG domain consists of drivers that can walk between locations and trucks that can drive between locations, and 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 for releasing agents that are at isolated cells. The FERRY

-	(loadAirplane BOX3 PLANE1 A2)
-	(fly PLANE1 A2 A1)
-	(unloadAirplane BOX3 PLANE1 A1)
0	(loadTruck BOX3 TRUCK1 A1)
1	(drive TRUCK1 A1 L4 CITY1)
2	(drive TRUCK1 L4 L2 CITY1)
3	(drive TRUCK1 L2 L1 CITY1)

Table 1: Observation sequence (1), monitoring TRUCK1.

-	(drive TRUCK1 L2 A1 CITY1)
-	(unloadTruck BOX2 TRUCK1 A1)
-	(unloadTruck BOX1 TRUCK1 A1)
0	(fly PLANE1 A2 A1)
1	(loadAirplane BOX2 PLANE1 A1)
2	(loadAirplane BOX1 PLANE1 A1)
3	(fly PLANE1 A1 A2)
4	(fly PLANE1 A2 A1)
5	(fly PLANE1 A1 A3)
6	(unloadAirplane BOX1 PLANE1 A3)
7	(fly PLANE1 A3 A4)
8	(unloadAirplane BOX2 PLANE1 A4)

Table 2: Observation sequence (2), monitoring PLANE1.

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 air-planes and trucks transporting packages between locations (*e.g.*, airports and cities). 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 10 non-trivial problem instances, in which we translated to commitment abandonment problems (10 problems for each threshold value: 0%, 5%, and 10%). Each commitment abandonment problem contains a planning domain definition, a commitment, an initial state, a set of observations (*i.e.*, plan executions), and a threshold value. For these problems, we generated plans (optimal and sub-optimal) using open-source planners, such as BLACKBOX, FAST-DOWNWARD, FF, and LAMA [15]. We generated plans that either abandoned (ultimately went to a different goal) or did not abandon their corresponding goals/consequent, varying the number of abandoned actions.

Since the plan optimality monitoring approach we use (Pereira *et al.* [14]) can exploit any domain-independent heuristic to compute whether an action contributes to goal achievement, we evaluated our commitment abandonment detection approach using several admissible and inadmissible heuristics. SUM (h_{sum}) is an inadmissible heuristic proposed by Bonet and Geffner in [2], and works in a similar manner to MAX-HEURISTIC, but is more informative. ADJUSTED-SUM (h_{adjsum}) is an inadmissible heuristic [12] that improves SUM by considering both negative and positive interactions among facts; ADJUSTED-SUM2 ($h_{adjsum2}$) is an inadmissible heuristic [12] that improves the ADJUSTED-SUM by combining the computation of the SET-LEVEL[6] heuristic and the relaxed plan heuristic.

[6] The SET-LEVEL heuristic estimates the cost to a goal by returning the level of the planning graph where all facts of the goal sate are reached without any mutex free [11].

ADJUSTED-SUM2M ($h_{adjsum2M}$) is an inadmissible heuristic [12] that improves ADJUSTED-SUM2. COMBO (h_{combo}) is an inadmissible heuristic [12] 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 [6] that relies on state-space search and estimates the goal distance by using delete-list relaxation.

We evaluated our approach using the following 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 plans that actually did abandon their expected commitments that our approach has detected correctly. False positive results represent the number of plans that actually eventually achieved the commitment consequent that our approach has detected as having abandoned the commitment. *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, False negative results represent the number of plans that would not eventually reach the commitment consequent that our approach has not detected as abandonment. *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*.

Domain	$\|O\|$	**T**	**Precision** θ **(0% / 5% / 10%)**	**Recall** θ **(0% / 5% / 10%)**	**F1-score** θ **(0% / 5% / 10%)**
DRIVER-LOG (30) $h_{adjsum2M}$	20.0	0.83	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0
DEPOTS (30) $h_{adjsum2}$	18.6	1.79	1.0 / 1.0 / 1.0	1.0 / 1.0 / 0.8	1.0 / 1.0 / 0.88
EASY-IPC-GRID (30) h_{ff}	17.3	0.95	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0
FERRY (30) $h_{adjsum2}$	13.5	0.38	1.0 / 1.0 / 1.0	1.0 / 0.8 / 0.8	1.0 / 0.88 / 0.88
LOGISTICS (30) $h_{adjsum2}$	21.0	0.56	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0
SATELLITE (30) $h_{adjsum2M}$	23.5	5.4	0.8 / 1.0 / 1.0	0.8 / 0.6 / 0.6	0.8 / 0.75 / 0.75
SOKOBAN (30) h_{combo}	22.8	5.2	0.83 / 1.0 / 1.0	1.0 / 0.6 / 0.6	0.91 / 0.75 / 0.75
ZENO-TRAVEL (30) $h_{adjsum2}$	10.0	1.1	1.0 / 1.0 / 1.0	0.8 / 0.8 / 0.8	0.88 / 0.88 / 0.88

Table 3: Experimental results for detecting commitment abandonment. $|O|$ represents the average number of observed actions in a plan execution. T is the average monitoring time (in seconds) that has been taken to detect commitment abandonment. θ is threshold value varying at 0%, 5%, and 10%.

Table 3 shows experimental results of our commitment abandonment approach over the selected domains using the heuristics that yield best results to detect sub-optimal steps. Each table row details results for a different domain showing averages for the number of observations $|O|$ across problem instances; T monitoring time (in seconds); *Precision*, *Recall*, and *F1-score*. The high average number of observations made ($|O|$), ranging between 10.0 and 23.5, indicating that all plans we analyze are non-trivial in complexity. For three domains (DRIVER-LOG, EASY-IPC-GRID, and LOGISTICS) our approach yields perfect predictions to detect commitment abandonment. Apart from the domains SATELLITE and SOKOBAN, that yield poor results (for threshold values 5% and 10%), for other domains we have near perfect prediction in detecting commitment abandonment.

6 Related Work

In [3], Geib and Goldman develop a formal model of goal and plan abandonment detection. This formal model is based on plan libraries and estimates the probability that a set of observed actions in a sequence contributes to the goal being monitored. Unlike our work, which requires no prior knowledge of an agent's plan library, they assume knowledge about possible plan decompositions available to each observed agent.

Proposed by Kafali *et al.* [8], GOSU is an algorithm that uses commitments to regulate agents' interactions in an environment for achieving their goals. By using commitments as contractual agreements, this algorithm allows an agent to decide whether it can achieve his goals for a given set of commitments and the current state. GOSU does not use any planning approach to reason about agents' goals, it uses a depth-first search algorithm.

Most recently, in [9], Kafali and Yolum propose a monitoring approach called PISAGOR that can determine whether a set of business interactions are progressing as expected in an e-commerce system. These business interactions are represented as commitments with deadlines. The authors also propose a set of operational rules for the observed agent in order to create expectations based on its commitments. Thus, PISAGOR monitors and detects whether the observed agent is progressing well, and therefore it identifies what is the problem during the interactions.

7 Conclusions

In this paper, we have developed an approach for detecting commitment abandonment that uses plan optimality monitoring techniques. As we show in experiments and evaluation, our approach yields very accurate results for detecting commitment abandonment, dealing with realistic well-known deterministic planning domains. As future work, we intend to explore partial observability (*i.e.*, missing observations), interleaving plans, more modern heuristics [5,16], and explore other planning techniques, such as symmetries in planning [17].

References

1. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Programming with commitments and goals in JaCaMo+. In: Proceedings of the 2015 International

Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015. pp. 1705–1706 (2015)

2. Bonet, B., Geffner, H.: Planning as heuristic search. Journal of Artificial Intelligence Research (JAIR) 129(1-2), 5–33 (2001)
3. Geib, C.W., Goldman, R.P.: Recognizing Plan/Goal Abandonment. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003. pp. 1515–1517 (2003)
4. Ghallab, M., Nau, D.S., Traverso, P.: Automated Planning - Theory and Practice. Elsevier (2004)
5. Helmert, M., Domshlak, C.: Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2009) (2009)
6. Hoffmann, J., Nebel, B.: The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research (JAIR) 14(1), 253–302 (May 2001)
7. Hoffmann, J., Porteous, J., Sebastia, L.: Ordered Landmarks in Planning. Journal of Artificial Intelligence Research (JAIR) 22(1), 215–278 (Nov 2004)
8. Kafalı, O., Günay, A., Yolum, P.: Gosu: Computing goal support with commitments in multiagent systems. In: Proceedings of the Twenty-first European Conference on Artificial Intelligence. pp. 477–482. ECAI'14, IOS Press (2014)
9. Kafali, Ö., Yolum, P.: PISAGOR: a proactive software agent for monitoring interactions. Knowledge and Information Systems 47(1), 215–239 (2016)
10. Meneguzzi, F., Telang, P.R., Singh, M.P.: A First-Order Formalization of Commitments and Goals for Planning. In: AAAI. pp. 697–703 (2013)
11. Nguyen, X., Kambhampati, S.: Extracting Effective and Admissible State Space Heuristics from the Planning Graph. In: Proceedings of the AAAI, 2000, Austin, Texas, USA. pp. 798–805 (2000)
12. Nguyen, X., Kambhampati, S., Nigenda, R.S.: Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. Artif. Intell. 135(1-2), 73–123 (2002)
13. Pattison, D., Long, D.: Domain Independent Goal Recognition. In: ÃĚgotnes, T. (ed.) STAIRS. Frontiers in Artificial Intelligence and Applications, vol. 222, pp. 238–250. IOS Press (2010)
14. Pereira, R.F., Oren, N., Meneguzzi, F.: Monitoring Plan Optimality using Landmarks and Domain-Independent Heuristics. In: The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition (2017)
15. Richter, S., Westphal, M.: The LAMA planner. Using landmark counting in heuristic search. In: Procs. IPC-6 (2008)
16. Scherrer, S., Pommerening, F., Wehrle, M.: Improved Pattern Selection for PDB Heuristics in Classical Planning (Extended Abstract). In: Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015 (2015)
17. Shleyfman, A., Katz, M., Helmert, M., Sievers, S., Wehrle, M.: Heuristics and symmetries in classical planning. In: AAAI 2015. pp. 3371–3377 (2015)
18. Singh, M.P.: Semantical considerations on dialectical and practical commitments. In: Proceedings of the 23rd Conference on Artificial Intelligence (AAAI). pp. 176–181. AAAI Press, Chicago (Jul 2008)
19. Telang, P., Meneguzzi, F., Singh, M.: Hierarchical Planning about Goals and Commitments. In: Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2013. pp. 877–884 (2013)