# Contingent Planning Using Counter-Examples from a Conformant Planner

Sebastien Piedade*, Alban Grastien†, Charles Lesire*, Guillaume Infantes‡
*ONERA, Toulouse, France
Emails: sebastien.piedade@onera.fr, charles.lesire@onera.fr
†Australian National University, Canberra, Australia
Email: alban.grastien@anu.edu.au
‡Jolibrain, Toulouse, France
Email: guillaume.infantes@jolibrain.com

***Abstract*—Decision-making for autonomous robots in real world applications has to manage uncertainties in order to efficiently accomplish a mission. Some planning methods deal with uncertainty by improving the robustness of the plan embedded in the robot. In this paper, we propose a novel approach to one of these methods, contingent planning. Most of the existing approaches are limited by the computation complexity and the quality of the solutions they return. To deal with these limitations, we propose to limit the number of observations in the plan as observations involve an important cost in computation time and energy. The originality of our approach is that our contingent planner uses an underlying conformant planner, i.e., a planner that is not allowed to make observations, to compute conformant subplans and insert observations between conformant subplans only when a conformant plan cannot be computed. We evaluate this approach by comparing its results with respect to Contingent-FF (Contingent Fast-Forward), a well known contingent planner, on a set of benchmarks. This comparison reveals that, even if our approach has some limitations, as it is not complete, it works quite effectively in terms of solution quality on classic benchmarks of the planning community.**

***Keywords–contingent planning; autonomous decision-making; uncertainty.***

## I. INTRODUCTION

In our world, where disasters are more and more frequent, fast and effective victims rescue has become a major issue. While robots are already used by first responders in such situations to access difficult terrains or hazardous areas, the next step would be to use autonomous robots, that would be able to adapt to the situation in order to provide a fast and efficient response. This need for autonomous robotics in search and rescue has been emphasized by [1] and [2]. This capability to adapt to the environment requires us to embed into the robot platform some decision-making processes able to reason about uncertain states of the environment, as stated by [3] while reporting about the use of robots in earthquake responses.

In this paper, we propose an algorithm for planning under uncertainty that settles in the contingent planning paradigm: uncertainty is represented by sets of possible states, and the objective is to find a conditional plan, i.e., a graph of actions containing branches allowing an online decision making influenced by the results of observations of some unknown parts of the environment.

Most of the existing approaches are limited by the complexity of the plan computation and the quality of the solutions they return. One of the ways to deal with these limitations is to limit the number of observations in the plan. In fact, especially in autonomous robots missions, observations have an important cost in computation time and energy. Conformant planning [4]–[7] consists in dealing with the uncertainty of the environment by computing a plan working for every possible initial state without making any observation. Being able to solve a problem without observation is an advantage that we find interesting to study, especially as existing contingent approaches do not use a background conformant planner and its advantages.

The originality of our approach is precisely that we use in the background a conformant planner iteratively, by asking it to solve subproblems. If the conformant planner cannot find a conformant plan, then we use counter-examples returned by this planner to insert observations in-between the conformant plans obtained for such subproblems. In this version of the approach, we try to perform the observation as close as possible from the failing action of a failing plan, assuming the fact that an observation could potentially be more efficient close to the issue. The approach described in this paper has some limitations. In fact, our approach is not yet complete, but it worked quite effectively so far in terms of solution quality on classic benchmarks of the planning community.

The rest of the paper is structured as follows. In Section 2, we present some related works on planning under uncertainty. In Section 3, we present some background and notation about the problem formalisation. In Section 4, we describe and evaluate theoretically our algorithm. In Section 5, we compare our approach with respect to Contingent-FF, and we present some results on some academic benchmarks. Finally, in Section 6, we conclude this paper.

## II. RELATED WORKS

Various planning methods exist to handle uncertainty. We can separate these methods in different types depending on how uncertainty is defined. Replanning is a method consisting in computing a first plan without handling uncertainties and replanning if an event occurs during the execution of the plan [8] [9]. Fast-Forward Replan (FF-Replan) [9] is one of these methods in which the probabilities of the problem are determinized and a plan is computed with a classical planning method. If an unexpected state occurs during the execution of the plan, then the planner replans in the same determinization of the problem. When uncertainty can be represented as probabilities on state transitions or action non-deterministic effects, probabilistic planning is commonly used. Among these probabilistic methods we can cite Markov Decision Processes (MDPs) [10] that model the problem as a fully observable stochastic system. The solution for a MDP is an optimal policy mapping the best action to each state of the MDP. This optimal policy can be found by various methods, like dynamic programming [11]

 ISBN: 978-1-61208-787-0 

and some of its variants like Value iteration method or Policy iteration [12]. When the problem is only partially observable, Partially Observable MDPs (POMDPs) [13] are used instead of MDPs. POMDPs introduce a belief over the environment state which is updated by some observations. This belief is updated by a function over state transition probabilities and observation probabilities. Some solving methods for MDPs and POMDPs follow a forward search approach generally making some sampling from the initial belief, then computing only a partial policy [14]–[16]. Chanel et al. [17] have also proposed an architectural approach to compute partial policy online in bounded time by making assumptions on the current belief.

When uncertainty is expressed as a set of possible initial states of the system, symbolic planning methods can be used. If the agent is not able to perform any observation, conformant planning can be used to solve the planning problem [4]–[7]. The plan returned by a conformant planner is generally an action sequence. Conformant Fast-Forward (Conformant-FF) [4] is a conformant planner that explores the belief space using heuristic functions based on a relaxation of the problem actions, by ignoring the delete lists of their effects. Conformant-FF then uses the Fast-Forward (FF) planner to compute a relaxed plan for each search state. Conformant Planner via Counter-Example and Sampling (CPCES) [7] is another conformant planner that computes a conformant plan using only a subset of the initial states, which allows CPCES to reduce the problem to classical planning.

If the planning problem is (partially) observable, contingent planning can be used to solve the problem. Contingent planning [18]–[20] consists in computing a conditional plan containing branches allowing an online decision making influenced by the results of some observations of the system. The conditional plan returned by a contingent planner is generally represented as a decision tree. Contingent Fast-Forward (Contingent-FF) [18] is a contingent planner that uses the same belief space representation as Conformant-FF. In Contingent-FF, the search space is an And-Or tree and the returned plan is a sub-tree where all leaves are goal states. Belief states are represented through action-observation sequences.

The approach proposed in this paper is part of the contingent planning approaches. Indeed, in the kind of application like search and rescue missions by autonomous robots, designing a model of the complete problem by a (PO)MDP is hard, as most of the uncertainty distributions are either unknown, or with the occurence of rare events. Online Replanning methods are not really suitables for autonomous robots missions as the cost of computing a new plan during the execution has an important computation and energy cost. Conformant planning has lots of advantages, but conformant plans do not always exist for autonomous robots problems, as some observations must be done to decrease the uncertainty. However, the approach we propose can be seen as a contingent meta-planner that uses a conformant planner to compute conformant plans for subproblems when such a conformant plan exists and add observations in between the conformant subplans when observations are needed. Therefore, our approach can be useful for missions in which we need to limit the number of observations.

## III. BACKGROUND

In this section, we introduce some background and notations that we use later to present our approach. First, we introduce notations to describe the problem, including a description of states and operators. Second, we introduce some notations about belief representation, that we use in our approach to manage uncertainty. In these definitions, all actions are assumed to be deterministic and the uncertainty is assumed to lie in the initial situation only. Note that it has been proven that non-deterministic effects can be eliminated by introducing artificial initial uncertainty [21].

### A. Problem definition

The following notations are adapted from [22] [23].

**Definition 1 (Planning Problem).** A planning problem $\mathbb{P}$ is defined by a tuple $(\mathcal{L}, \mathcal{O}, I, G)$ where:

- $\mathcal{L} = \{p_1, ..., p_n\}$ is a finite set of proposition symbols; a state $s$ is then represented by a set of propositions that hold, i.e., that are true, in $s$; propositions that do not hold in $s$ are assumed to be false; we denote $\mathcal{W} = 2^{\mathcal{L}}$ as the set of all possible world states;
- $\mathcal{O}$ is a finite set of *operators*, partitioned into the set of *actions* $A$ and the set of *observations* $O$; each operator $op \in \mathcal{O}$ is defined by a precondition $pre(op) \subseteq \mathcal{L}$ and a set of effects *eff(op)*;
- $I \subseteq \mathcal{W}$ is the set of possible initial states;
- $G \subseteq \mathcal{L}$ is the set of propositions defining the goal.

**Definition 2 (Action Application).** An action $a \in A$ is *applicable* in state $s \in \mathcal{W}$ if and only if its preconditions hold in $s$, i.e.,

$$pre(a) \subseteq s \tag{1}$$

Each effect $e$ of $a$ (i.e., $e \in$ *eff(a)*) is defined by a triple $con(e) \subseteq \mathcal{L}, add(e) \subseteq \mathcal{L}, del(e) \subseteq \mathcal{L}$, where:

- $con(e)$ are the conditions in which $e$ is applied (unconditional effects are defined by $con(e) = \emptyset$);
- $add(e)$ are the propositions that will be added to the state after applying $e$;
- $del(e)$ are the propositions that will be deleted from the state after applying $e$.

If $a$ is applicable in $s$, then $T(s, a)$ is the transition function such that:

$$T(s,a) = s - \bigcup_{e \in \textit{eff(a)} \text{ s.t. } con(e) \subseteq s} del(e) \cup \bigcup_{e \in \textit{eff(a)} \text{ s.t. } con(e) \subseteq s} add(e). \tag{2}$$

We assume that the problem actions are not self-contradictory, i.e., when applying action $a$ in state $s$, for two effects $e, e' \in$ *eff(a)*, if $p \in add(e)$ and $p \in del(e')$, then $con(e)$ and $con(e')$ are not both satisfied in state $s$. We have $con(e) \cup con(e') \not\subseteq s$. Consequently, we also assume that for each effect $e$, $add(e) \cap del(e) = \emptyset$.

**Definition 3 (Observation Application).** An observation $o \in O$ is applicable in state $s \in \mathcal{W}$ if and only if its preconditions hold in $s$ (see (1)). The effects $o$ are defined by a proposition (*eff(o)* $\in \mathcal{L}$) that is observed when applying $o$, i.e., whose truth value is known after applying $o$. The application of an observation $o$ in state $s$ has no effect over the state $s$, i.e.,

$$T(s, o) = s \tag{3}$$

The application of an observation $o$ has no effect on the world state, but it will have an effect on the belief the agent has on the current state. Notations and definitions about belief reasoning are described below. Note that we assume that an observation $o$ observes only one proposition at a time, without loss of generality.

### B. Belief reasonning

As classically done in planning under uncertainty, we model the uncertain knowledge about the current state as a *belief* represented by a set of all the possible states (consistent with the actions/observations done so far). The following notations are taken or adapted from [23] [24].

**Definition 4 (Belief State).** The current belief $\mathcal{B} = \{s_1, \ldots, s_n\} \subseteq \mathcal{W}$ is the set of possible current states $s_1$, …, $s_n$. The initial belief $\mathcal{B}_0$ corresponds to the possible initial states $I$ of the problem.

**Definition 5 (Action Application on a Belief).** An action $a \in A$ is applicable in belief $\mathcal{B}$ if and only if $a$ is applicable for each possible state in $\mathcal{B}$, i.e., iff:

$$\forall s \in \mathcal{B}, pre(a) \subseteq s \tag{4}$$

The effect of applying action $a$ in $\mathcal{B}$ then results in a belief $\mathcal{B}'$ such that:

$$\mathcal{B}' = \{T(s,a), \text{ s.t. } s \in \mathcal{B}\} \tag{5}$$

where $T(s,a)$ is computed according to (2). By extension, the effect of applying $a$ in $\mathcal{B}$ can be noted as $T(\mathcal{B},a)$.

**Definition 6 (Observation Application on a Belief).** An observation $o \in O$ is applicable in belief $\mathcal{B}$ if and only if $o$ is applicable for each possible state in $\mathcal{B}$ (see (4)). Let $\nu(o)$ be the observation result, i.e., the observed truth value of the effects of $o$. We denote $\nu^+(o) \subseteq \mathcal{L}$ as the set of observed propositions that hold in the current state, and $\nu^-(o) \subseteq \mathcal{L}$ the set of observed propositions that do not hold. The application of observation $o$ in belief $\mathcal{B}$ does not modify the state itself, as described in Def. 3, but results in a new belief $T(\mathcal{B},o)$ such that:

$$\begin{aligned} T(\mathcal{B},o) = \{s, \text{ s.t. } & s \in \mathcal{B} \wedge \nu^+(o) \subseteq s \\ & \wedge \nu^-(o) \cap s = \emptyset\} \end{aligned} \tag{6}$$

As we assume that an observation observes only one proposition $p$ (see Def. 3), either $\nu^+(o)$ or $\nu^-(o)$ is empty, the other being equal to $p$. Also, note that (6) would work for observations whose effect has multiple propositions.

### C. Conditional Plan

A conditional plan can be represented as a graph of operators, leading an initial belief to a resulting belief (then having a flow network structure). Branchings in this graph correspond to results of observations, depending whether the observed properties hold or not in the current belief.

**Definition 7 (Conditional Plan).** Given a problem $\mathbb{P} = (\mathcal{L}, \mathcal{O}, I, G)$, a conditional plan is inductively defined by the following facts:

- The empty plan $\varepsilon$ is a conditional plan;
- $(op)$ is a conditional plan $\forall op \in \mathcal{O}$;
- if $\pi_1$ and $\pi_2$ are conditional plans, then $\pi_1;\pi_2$ is a conditional plan representing the sequence of $\pi_1$ and $\pi_2$;
- if $\pi_1, \pi_2$ are conditional plans and $o \in O$ is an observation, then the plan **if** $o$ **then** $\pi_1$ **else** $\pi_2$ is a conditional plan representing that, according to the result of observation $o$, $\pi_1$ is executed if the observed proposition is true, otherwise $\pi_2$ is executed.

A conditional plan $\pi$ is *executable* in a belief $\mathcal{B}$ if its root operator is applicable in $\mathcal{B}$, and if all operators in $\pi$ are applicable in the belief corresponding to the result of their previous operators. We denote $T(\mathcal{B},\pi)$ as the result of applying an executable conditional plan in belief $\mathcal{B}$. Using a similar inductive definition, we formally say that a conditional plan $\pi$ is executable in a belief $\mathcal{B}$ if:

- $\pi = (op)$, $op \in \mathcal{O}$ such that $op$ is applicable in $\mathcal{B}$ (see Eq. 4); then $T(\mathcal{B},\pi) = T(\mathcal{B},op)$;
- $\pi = \pi_1;\pi_2$, with $\pi_1$ applicable in $\mathcal{B}$ and $\pi_2$ applicable in $T(\mathcal{B},\pi_1)$; then $T(\mathcal{B},\pi) = T(T(\mathcal{B},\pi_1),\pi_2)$;
- $\pi =$ **if** $o$ **then** $\pi_1$ **else** $\pi_2$, with $o$ applicable in $\mathcal{B}$, $\pi_1$ is applicable in $\mathcal{B}^+$ and $\pi_2$ is applicable in $\mathcal{B}^-$, where $\mathcal{B}^+$ (resp. $\mathcal{B}^-$) $= T(\mathcal{B},o)$ when $\nu^+(o)$ (resp. $\nu^-(o)$) $= eff(o)$; the result of applying $\pi$ is then $T(\mathcal{B},\pi) = T(\mathcal{B}^+,\pi_1) \bigcup T(\mathcal{B}^-,\pi_2)$.

A conditional plan $\pi$ executable in $I$ and that leads to $G$ (i.e., $G \in T(I,\pi)$) is a *solution* to problem $\mathbb{P}$. We can notice that the definition of a plan in classical formalism is equivalent to the three first points of Def. 7.

## IV. Contingent Planning Algorithm

The proposed approach settles on the use of a conformant planner that is asked to solve subproblems. To be used in our approach, the conformant planner must return either the conformant plan it has found, but also, in case no conformant plan exists, a counter-example (i.e., an initial state that caused the failure) and information about why the planner is failing on this counter-example, in the form of a plan (i.e., a sequence of actions) that fails for this counter-example.

Based on such a conformant planner, the principle of our approach is to give a problem to solve to the conformant planner and, in case of failure, use the counter-example and the failing plan to determine which observation to perform, and when, and then split the problem into subproblems taking this observation into account to reduce the uncertainty on the subproblems, and then ask the conformant planner to solve these subproblems. This process is used iteratively on the subproblems if the conformant planner fails in finding a solution. In this version of the approach, we try to perform the observation as close as possible from the failing action of the failing plan returned by the conformant planner, assuming the fact that an observation could potentially be more efficient close to the plan issue. As a conformant planner, we use CPCES [7], which fulfills our assumption: it provides a counter-example and a failing plan in case of failure. Note that we could use any conformant planner returning the same kind of information, and for conformant planners that would instead return for instance a proposition that makes the solver fail, we could integrate it with slight modification of the algorithm, without reconsidering the approach.

 ISBN: 978-1-61208-787-0 

In subsection A. of this section, we present the CPCES algorithm and the data it provides. Then, we present and describe the main algorithm of our approach, before analysing some properties.

### A. CPCES

CPCES [7] is a conformant planner that follows an iterative approach, in which a deterministic planner, namely FF [25], is used to find a plan $\pi$ (an action sequence) for a subset of the initial belief, and then the validity of this plan on the complete initial belief is checked by solving a boolean satisfiability (SAT) problem with Z3 [26]. If the plan is not valid, Z3 provides a counter-example $\gamma$, i.e., a possible initial state for which the plan is not valid. This counter-example is integrated into the initial subset, and FF is asked to solve it again. This process is used iteratively, starting from one single state of the belief, until either a valid plan is found, or FF finds no plan for the subset, in which case the counter-example and the previous plan found by FF are returned.

Figure 1 contains the CPCES algorithm taken from [7] and adapted to the notations introduced in the previous section. FF is called to compute a new plan in line 9 and Z3 is used to check the plan validity in line 4.

**Input:** $\mathbb{P} = (\mathcal{L}, A, I, G)$
**Output:** $\pi, \gamma$
1: $\mathcal{B} := \emptyset$
2: $\pi := \varepsilon$
3: **loop**
4: check validity of $\pi$
5: **if** $\pi$ is a solution for $\mathbb{P}$ **then**
6: **return** $\pi, \emptyset$
7: let $\gamma$ be a counter-example
8: $\mathcal{B} := \mathcal{B} \cup \{\gamma\}$
9: compute a new plan $\pi'$ for $\mathbb{P}' = (\mathcal{L}, A, \mathcal{B}, G)$
10: **if** no such $\pi'$ exists **then**
11: **return** $\pi, \gamma$
12: $\pi := \pi'$

Figure 1. CPCES Algorithm

### B. Contingent planner

**Input:** $\mathbb{P} = (\mathcal{L}, \mathcal{O}, I, G)$
**Output:** $\pi_c$
1: $\pi, \gamma := \text{conformantPlanner}(\mathbb{P})$
2: **if** $\gamma = \emptyset$ **then**
3: **return** $\pi$
4: $\mathcal{B}_o, o := \text{findObservation}(I, \mathcal{O}, \pi, \gamma)$
5: $\pi_o := \text{ContingentPlanning}((\mathcal{L}, \mathcal{O}, I, \mathcal{B}_o))$
6: $\mathcal{B}^+ := T(\mathcal{B}_o, o)$ *with* $\nu^+(o) =$ *eff(o)*
7: $\pi_p := \text{ContingentPlanning}((\mathcal{L}, \mathcal{O}, \mathcal{B}^+, G))$
8: $\mathcal{B}^- := T(\mathcal{B}_o, o)$ *with* $\nu^-(o) =$ *eff(o)*
9: $\pi_n := \text{ContingentPlanning}((\mathcal{L}, \mathcal{O}, \mathcal{B}^-, G))$
10: **return** $(\pi_o$ ;**if** $o$ **then** $\pi_p$ **else** $\pi_n)$

Figure 2. Contingent Planning Procedure

Figure 2 illustrates our main contingent plan computation algorithm. This algorithm takes as input the contingent problem $\mathbb{P}$ and returns a contingent plan $\pi_c$. We first ask a conformant planner (in our implementation, CPCES, as described in Alg. 1) to compute a conformant plan $\pi$ (line 1). If such a conformant plan exists, (line 2), we return this plan. Otherwise, the conformant planner returns a counter-example $\gamma$ and a plan $\pi$ that fails for this counter-example. From these pieces of information, we look for an observation to include in the plan. The findObservation function (further detailed in Alg. 3) returns an observation $o$ and a belief $\mathcal{B}_o$ in which this observation could be performed (line 4).

Finally, we call our algorithm again on the subproblems corresponding to: reaching the belief states in which to perform the observation from the initial state (line 5), and reaching the goal from both cases where the belief has been updated after a positive observation (line 7) and a negative observation (line 9). We finally return a plan made of the subplan to reach $\mathcal{B}_o$ followed by a branching conditionned by the observation result (line 10).

**Input:** $I, \mathcal{O}, \pi, \gamma$
**Output:** $(\mathcal{B}_o, o)$
1: $beliefList := [I]$
2: $\mathcal{B} := I$
3: **for** $a$ in $\pi$ **do**
4: **if** $a$ applicable in $\gamma$ **then**
5: $\gamma := T(\gamma, a)$
6: $\mathcal{B} := T(\mathcal{B}, a)$
7: $beliefList := beliefList + \mathcal{B}$
8: **else**
9: let $unsatPre$ be the unsatisfied preconditions of $a$
10: **break**
11: **for** $p$ in $unsatPre$ **do**
12: let $o$ be an observation for $p$ in $\mathcal{O}$
13: **for** $\mathcal{B}_o$ in $beliefList$ **do**
14: **if** $o$ applicable in $\mathcal{B}_o$ **then**
15: **return** $(\mathcal{B}_o, o)$
16: **return** $(\mathcal{B}, None)$

Figure 3. findObservation Algorithm

Figure 3 describes the findObservation algorithm. This algorithm is used to determine which observation $o$ we need to perform to discriminate the counter-example $\gamma$ from the other possible states and in which belief $\mathcal{B}_o$ we need to perform the observation. The inputs are the set of operators $\mathcal{O}$, the failing plan $\pi$ previously computed by the conformant plan and the counter-example state $\gamma$. The outputs are the observation $o$ we need to perform and the belief $\mathcal{B}_o$ in which we need to perform the observation $o$.

We first look for the action $a$ in $\pi$ that is not applicable in $\gamma$ (lines 4 to 7) by iteratively applying each action of the plan to the counter-example $\gamma$ (line 5). We also keep track of the beliefs computed by the application of each action of $\pi$ to the initial belief (lines 6 and 7). Once the failing action has been found, we get the set of propositions $unsatPre$ in the preconditions of $a$ that does not hold in the state $\gamma$ (line 9). We can notice that $unsatPre$ will never be empty because there is necessarily a failing action in the failing plan $\pi$ returned by the conformant planner. Each proposition of $unsatPre$ is a potential observable proposition allowing to discriminate the counter-example and the other states in which this proposition

 ISBN: 978-1-61208-787-0 

does not hold from the other possible states. We then try to find an observation able to observe one of these propositions in one of the beliefs computed in the belief list (lines 11 to 15). We scan the possible observations in $\mathcal{O}$ to find an observation performing an observation effect over the value of the proposition $p$ (line 12).

Finally, we verify if the observation $o$ is applicable in one of the computed beliefs in $beliefList$ (lines 13 to 15). If $o$ is applicable in the current belief $\mathcal{B}_o$, then we return $\mathcal{B}_o$ and $o$. In the other case, we verify if the observation $o$ is applicable in one of the previously computed beliefs in $beliefList$. If $o$ is not applicable in any belief of $beliefList$, then we try to find another observation able to observe another proposition $p$ of $unsatPre$. If there is no observation able to discriminate the counter-exemple, then we return $None$, meaning that there is no possible observation.

### C. Theoretical evaluation

Our method is sound because, if there is a solution to the problem, then the plan found is a contingent or conformant solution to the problem under the assumption that the conformant planner used is sound. If there is no solution, then the method terminates returning a no solution message that does not appear in the algorithms above. The algorithm always terminates because, if there is a contingent solution, then the size of the search space is convergent due to the splitting of the search space after each observation. If there is a conformant solution, then it is returned directly after the first call of CPCES, and if there is no solution, the algorithm terminates with an exit message. Our method is not complete, particularly because there is no backtracking. We are currently working on a version including backtracking, but it will be a second version of the algorithm.

## V. Results

We have evaluated our algorithm by comparing its performance with respect to Contingent-FF on a set of benchmarks provided by Contingent-FF. The results are given on Table I. Computation times are given in seconds. TO indicates that the computation timed out after 5 min. NO indicates that the planner did not find an applicable observation. *Size* gives the number of actions in the plan, *depth* the maximal depth of the plan, and *observations* the number of observations. For our approach, we also compute the depth of the *shortest* path of the plan. This comparison is especially interesting given that Contingent-FF uses the same PDDL (Planning Domain Description Language) input language as our method. For this evaluation, we limited the computation time of the two approaches to 5 minutes, and we used the heuristic option of Contingent-FF that provided the fastest results (otherwise, the solver times out on most of the benchmarks). The results are given on Table I.

First, we can notice that for some benchmarks we find the same results as Contingent-FF, except for computation time, namely *ebtcs, grid p2, egrid p2, elogistics p1 and p3*. Sometimes neither Contingent-FF nor our approach are able to find a solution, like in *egrid p3* and *p4*, where our approach does not succeed to find an applicable observation and Contingent-FF times out.

We can notice in Table I that Contingent-FF has clearly better results in *blocks* where our approach finds plans with the same number of observations, but with a bigger size and a longer depth. In *erovers* $p4$, we obtain the same result in size as Contingent-FF, but our approach computes a longer plan in depth. Moreover, in *erovers p6*, Contingent-FF finds a solution with less observations, even if our solution is shorter.

We can observe in Table I that our approach is better than Contingent-FF in benchmarks where a conformant solution exists, namely *btcs, grid, rovers, logistics*. In that case, as we rely on CPCES, we find a conformant plan whereas Contingent-FF includes observations in its solution. Moreover, it generally results in finding a shorter plan, except for *logistics* and *btcs* where Contingent-FF finds a plan with a shorter depth. In problems like *elogistics p5, p7* and *erovers p2 and p8*, we find plans having the same number of observations than Contingent-FF, but the plans we find are shorter in size and depth. Moreover, our approach succeeds in solving *egrid p5* problem, while Contingent-FF times out.

One of the only drawbacks of our approach with respect to Contingent-FF is the computation time needed to solve some of the problems. First, we can notice these computation times have the same order of magnitude than Contingent-FF and do not seem to grow exponentially when increasing the size of the problems. Second, this computation time partly comes from the fact that we use CPCES as a "black-box" conformant planner, itself considering FF as a "black-box" planner. This induces a lot of access to files for writing/reading problems for these solvers during our process, while Contingent-FF does all the computation in memory. Our approach is not complete, which implies that in some problems we cannot find any applicable observations. In fact, in the current version of the approach, there is no backtracking in the failing plan computation and in the observation computation process. Our approach does not succeed when no observation is applicable in any belief computed from the application of the failing plan computed by CPCES. However, an interesting fact we can notice in these results is that our approach is better on benchmarks *rovers, logistics and elogistics*, which are closer to autonomous robots problems in which we need to navigate and explore an environment in order to pick or analyze some items.

A simulation of an autonomous robot scenario is currently in development. This simulation will allow us to evaluate the performance of our approach and see the behavior of the robot during the plan execution.

## VI. Conclusion

In this paper, we have proposed a new contingent planner with an original approach, as we use a conformant planner to find conformant subplans when possible. Our approach consists in asking CPCES, a conformant planner, to solve a problem. If no conformant plan exists for this problem, CPCES returns a counter-example and a failing plan for this counter-example. We use this information to first add an observation in the plan to reduce the uncertainty related to this counter-example, and second to decompose the problem into subproblems with less uncertainty. These subproblems are sent to CPCES again to find a conformant plan, and the process iterates until a complete conditional plan has been found.

We compared our approach with Contingent-FF on a set of benchmarks and, despite the fact that we generally have higher computation time, we get some concluding results. First, on benchmarks where a conformant solution exists, we

 ISBN: 978-1-61208-787-0 

TABLE I. RESULTS OF A COMPARISON WITH CONTINGENT-FF ON SOME BENCHMARKS.

| Problem | Contingent Planning with counter-examples | | | | | Contingent-FF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | time (s) | size | depth | shortest | observations | time (s) | size | depth | observations |
| blocks/p3 | 0.94 | 6 | 4 | 3 | 1 | 0.00 | 6 | 4 | 1 |
| blocks/p7 | 5.6 | 89 | 16 | 10 | 7 | 0.05 | **55** | **9** | 7 |
| blocks/p11 | 6.4 | 169 | 29 | 20 | 7 | 0.43 | **117** | **18** | 7 |
| blocks/p15 | 8.05 | 244 | 39 | 27 | 7 | 3.20 | **163** | **25** | 7 |
| btcs/p10 | 0.76 | 19 | 19 | 19 | **0** | 0.02 | 19 | 10 | 9 |
| btcs/p30 | 2.36 | 59 | 59 | 59 | **0** | 0.8 | 59 | 30 | 29 |
| btcs/p50 | 8.13 | 99 | 99 | 99 | **0** | 9.79 | 99 | 50 | 49 |
| btcs/p70 | 24.11 | 139 | 139 | 139 | **0** | 57.31 | 139 | 70 | 69 |
| ebtcs/p10 | 6.21 | 19 | 10 | 2 | 9 | 0.01 | 19 | 10 | 9 |
| ebtcs/p30 | 22.73 | 59 | 30 | 2 | 29 | 0.42 | 59 | 30 | 29 |
| ebtcs/p50 | 56.8 | 99 | 50 | 2 | 49 | 4.93 | 99 | 50 | 49 |
| ebtcs/p70 | 156.11 | 139 | 70 | 2 | 69 | 29.10 | 139 | 70 | 69 |
| grid/p2 | 3.61 | 9 | 9 | 9 | 0 | 0.01 | 9 | 9 | 0 |
| grid/p3 | 4.05 | **19** | **19** | 19 | **0** | 9.78 | 174 | 43 | 15 |
| grid/p4 | 21.24 | **45** | **45** | 45 | **0** | 227 | 464 | 68 | 17 |
| grid/p5 | 18.64 | **31** | **31** | **31** | **0** | TO | - | - | - |
| egrid/p2 | 3.96 | 9 | 9 | 9 | 0 | 0.01 | 9 | 9 | 0 |
| egrid/p3 | NO | - | - | - | - | TO | - | - | - |
| egrid/p4 | NO | - | - | - | - | TO | - | - | - |
| egrid/p5 | 73.24 | **185** | **31** | **23** | **7** | TO | - | - | - |
| rovers/p2 | 0.38 | **8** | **8** | 8 | **0** | 0.00 | 13 | 10 | 1 |
| rovers/p4 | 0.52 | **13** | **13** | 13 | **0** | 0.00 | 23 | 14 | 3 |
| rovers/p6 | 0.86 | **23** | **23** | 23 | **0** | 0.11 | 448 | 66 | 11 |
| rovers/p8 | 0.65 | **23** | **23** | 23 | **0** | 0.03 | 170 | 83 | 3 |
| erovers/p2 | 1.09 | **11** | **9** | 5 | 1 | 0.00 | 13 | 10 | 1 |
| erovers/p4 | 3.39 | 23 | 17 | 5 | 3 | 0.00 | 23 | **14** | 3 |
| erovers/p6 | 15.52 | **144** | **27** | 21 | 11 | 0.09 | 346 | 48 | **7** |
| erovers/p8 | 3.34 | **44** | **21** | 15 | 3 | 0.01 | 95 | 36 | 3 |
| logistics/p1 | 0.39 | **9** | 9 | 9 | **0** | 0.01 | 10 | 7 | 1 |
| logistics/p3 | 0.49 | **14** | 14 | 14 | **0** | 0.01 | 18 | 8 | 2 |
| logistics/p5 | 0.58 | **29** | 29 | 29 | **0** | 0.054 | 172 | 26 | 7 |
| logistics/p7 | 0.75 | **31** | 31 | 31 | **0** | 0.2 | 247 | 27 | 11 |
| elogistics/p1 | 1.07 | 10 | 7 | 4 | 1 | 0.00 | 10 | 7 | 1 |
| elogistics/p3 | 1.8 | 18 | 8 | 5 | 2 | 0.00 | 18 | 8 | 2 |
| elogistics/p5 | 9.02 | **138** | **22** | 20 | 7 | 0.12 | 172 | 26 | 7 |
| elogistics/p7 | 10.63 | **185** | 26 | 21 | 11 | 0.13 | 247 | 26 | 11 |

always find a conformant plan. Second, on most of the other benchmarks, we either get solutions with less observations, or with less actions in the plan. Moreover, ou approach is better on benchmarks close to autonomous robots problems in which we need to navigate and explore uncertain environments.

Future works consist first in improving the completeness of our method by performing a backtracking in the failing plan computation and in the observation computation process if we fail to find an observation applicable in a belief computed from the current failing plan. Second, we would like to apply our method to autonomous robots problems closer to reality with navigation and exploration by implementing it on a real robot.

## REFERENCES


[1] A. Birk and S. Carpin, "Rescue robotics a crucial milestone on the road to autonomous systems," Advanced Robotics, vol. 20, no. 5, 2006, pp. 595–605.

[2] R. R. Murphy et al., "Search and rescue robotics," Springer handbook of robotics, 2008, pp. 1151–1173.

[3] H. Kitano and S. Tadokoro, "Robocup rescue: A grand challenge for multiagent and intelligent systems," AI magazine, vol. 22, no. 1, 2001, pp. 39–39.

[4] J. Hoffmann and R. I. Brafman, "Conformant planning via heuristic forward search: A new approach," vol. 170, no. 6-7. Elsevier, 2006, pp. 507–541.

[5] D. Bryce, S. Kambhampati, and D. E. Smith, "Planning graph heuristics for belief space search," Journal of Artificial Intelligence Research (JAIR), vol. 26, 2006, pp. 35–99.

[6] A. Albore, M. Ramirez, and H. Geffner, "Effective heuristics and belief tracking for planning with incomplete information," in International Conference on Automated Planning and Scheduling (ICAPS), Freiburg, Germany, June 2011, pp. 2–9.

[7] A. Grastien and E. Scala, "Intelligent belief state sampling for conformant planning." in IJCAI, 2017, pp. 4317–4323.

[8] U. Kuter, D. Nau, E. Reisner, and R. Goldman, "Using classical planners to solve nondeterministic planning problems," in Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling, 2008, pp. 190–197.

[9] S. W. Yoon, A. Fern, and R. Givan, "Ff-replan: A baseline for probabilistic planning." in ICAPS, vol. 7, 2007, pp. 352–359.

[10] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[11] R. Bellman, "Dynamic programming," Science, vol. 153, no. 3731, 1966, pp. 34–37.

[12] Mausam and A. Kolobov, "Planning with markov decision processes: An ai perspective," Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 6, no. 1, 2012, pp. 1–210.

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," Artificial intelligence, vol. 101, no. 1-2, 1998, pp. 99–134.

[14] E. Hansen and S. Zilberstein, "LAO$^*$: A heuristic search algorithm that finds solutions with loops," Artificial Intelligence Journal (AIJ), vol. 129, no. 1-2, 2001, pp. 35–62.

[15] F. Teichteil-Königsbuch, U. Kuter, and G. Infantes, "Incremental plan aggregation for generating policies in MDPs," in International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Toronto, Canada, May 2010, pp. 1231–1238.


 ISBN: 978-1-61208-787-0

[16] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces," in Robotics: Science and Systems (RSS). Zurich, Switzerland, July 2008.

[17] C. P. C. Chanel, A. Albore, J. THooft, C. Lesire, and F. Teichteil-Königsbuch, "Ample: an anytime planning and execution framework for dynamic and uncertain problems in robotics," Autonomous Robots, vol. 43, no. 1, 2019, pp. 37–62.

[18] J. Hoffmann and R. Brafman, "Contingent planning via heuristic forward search with implicit belief states," in Proc. ICAPS, vol. 2005, 2005.

[19] A. Albore, H. Palacios, and H. Geffner, "A translation-based approach to contingent planning," in International Joint Conference on Artificial Intelligence (IJCAI), Providence, RI, USA, July 2009.

[20] B. Bonet, H. Palacios, and H. Geffner, "Automatic derivation of memoryless policies and finite-state controllers using classical planners," in International Conference on Automated Planning and Scheduling (ICAPS), Thessaloniki, Greece, September 2009, pp. 34–41.

[21] A. Albore, H. Palacios, and H. Geffner, "Compiling uncertainty away in non-deterministic conformant planning," in European Conference on Artificial Intelligence (ECAI), Lisbon, Portugal, August 2010, pp. 465–470.

[22] M. Ghallab, D. Nau, and P. Traverso, Automated Planning: theory and practice. Elsevier, 2004.

[23] A. Albore and H. Geffner, "Acting in partially observable environments when achievement of the goal cannot be guaranteed," in Proc. of ICAPS Workshop on Planning and Plan Execution for Real-World Systems. Citeseer, 2009.

[24] P. E. U. de Souza, C. P. C. Chanel, and F. Dehais, "Momdp-based target search mission taking into account the human operator's cognitive state," in 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2015, pp. 729–736.

[25] J. Hoffmann, "Ff: The fast-forward planning system," AI magazine, vol. 22, no. 3, 2001, pp. 57–57.

[26] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2008, pp. 337–340.

 ISBN: 978-1-61208-787-0 