Anticipatory Search as Partial Satisfaction Planning with State Dependent Costs

Daniel Borrajo and **Raquel Fuentetaja** and **Tomás de la Rosa**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganes (Madrid). Spain
dborrajo@ia.uc3m.es, rfuentet@inf.uc3m.es, trosa@inf.uc3m.es

Abstract

In the context of deliberative reasoning, autonomous systems should be able to explicitly reason about goals. Most works in automated planning assume that goals are initially given and fixed and planners generate plans to achieve them. However, in some real-world scenarios where planners work on an on-line continual planning setting, additional goals may arrive over time. When some information about future goal arrivals is known, it can be exploited to direct the system towards those goals even though they have not arrived yet. Recent work in systems that exhibit such anticipatory behavior has proposed to use sampling techniques in the context of a Markov Decision Process (MDP). In this paper, we tackle this problem from the point of view of domain-independent planning and propose three alternative approaches. Experimental results in several benchmark domains suggest that these approaches allow the system to exhibit a more efficient and effective anticipatory behavior.

1 Introduction

Most works on automated planning assume goals are given, do not change, and planners should only reason on the current given goals. However, autonomous systems dealing with most real world scenarios will face planning tasks where goals do arrive over time. For instance, in satellites or rovers domains, new requests to take images/obtain samples appear over time. In logistics domains, new packages should be transported elsewhere, and in elevators domains new passengers appear over time. Most planning approaches deal with dynamic goals by integrating planning with an execution module, and let the execution module force a replanning when new goals arrive [Ruml *et al.*, 2011].

In this paper we focus on planning problems with deterministic worlds but non-deterministic goals, as the work introduced recently by Burns *et al.* in the context of On-line Continual Planning Problems (OCPPs) [Burns *et al.*, 2012]. Other works have addressed the task of non-deterministic worlds, but fixed goals [Kiesel *et al.*, 2013]. Burns *et al.* approach explicitly exploits information about future goal arrivals with the purpose of building plans that anticipate the future. They select the next action to apply by a variant of hindsight optimization [Chong *et al.*, 2000; Yoon *et al.*, 2008; 2010], that provides a reduction from an stochastic planning problem to a deterministic one based on sampling. This strategy requires solving several planning tasks at every time-step.

The aim of the present paper is to solve OCPPs efficiently in a domain-independent way. To be able of defining the different approaches adequately, we first contribute with the definition of a type of planning tasks called Partial Satisfaction Planning with Horizon and State-Dependent costs (PSP-HSD). In PSP-HSD tasks there is a finite planning horizon, the goals are *soft* and the cost function is state-dependent. PSP-HSD tasks allow us to characterize Burns *et al.* approach for domain-independent planning. As a second contribution of the paper, we define three alternative and more efficient problem solving strategies than that of Burns *et al.*. They are also expressed in terms of PSP-HSD tasks that incorporate probabilistic information about goal arrivals in the cost function.

In order to solve OCPPs by means of PSP-HSD tasks in practice, and as a third contribution, we propose a compilation of these tasks into numerical planning. The compilation is based on the compilation proposed by Keyder and Geffner [2009] to deal with soft goals in classical planning; and on the one proposed by Benton et al. [2012] to deal with state dependent costs.

2 Problem Description

The problem we tackle in this paper is an On-line Continual Planning Problem (OCPP) in a deterministic world with uncertainty about future goals, where the goal arrival distribution is known. This problem has been defined before as a Markov Decision Process (MDP) [Burns *et al.*, 2012].

Definition 1 Online Continual Planning Problem (OCPP) *An OCPP is a MDP defined by tuple $\langle S, A, T, C\rangle$, where S is a set of states. Each state $s \in S$ is defined by a pair $(w, G) \in W \times \mathcal{G}$ that combines a world state $w \in W$ and a set of known goals $G \subseteq \mathcal{G}$. $\mathcal{G}$ is the set of all possible goals. A is a set of actions. T is the transition function, $T : S \times A \times S \rightarrow [0, 1]$ that defines the probability of transitioning from a state $s \in S$ to a state $s' \in S$ by applying an action $a \in A$. C is a cost function, $C : S \times A \times S \rightarrow \mathbb{R}_0^+$, that defines the cost of applying an action a in a state s obtaining a state s'.*

In OCPP, the number of possible states in S is $|W| \times 2^{|\mathcal{G}|}$. An action a is applicable in a state $s = (w, G)$ if it is applicable in w. We will denote the set of applicable actions as $A(s)$. To avoid dead-ends we assume the existence of a `no-op` action, applicable in all states, that preserves the world state. This MDP differs from the classical one [Puterman, 1994] in that the current goals are part of the current state and goal achievement is measured by a cost function.

In this paper, we make the same assumptions as in [Burns *et al.*, 2012]. We assume actions are deterministic on world states and the only uncertainty comes from future goals. We also assume that future goals are independent of world states and actions, and among them. Even making these assumptions the task is a difficult one. Also, these assumptions apply in a wide range of domains, as the ones we will use in our experiments. Finally, we assume that once a goal has arrived it remains in the list of known goals, i.e. new goals are just added to the previous ones.

The transition function, $T(s, a, s')$, where $s = (w, G)$ and $s' = (w', G')$, is zero when a is not applicable in s. Otherwise, and given the previous assumptions, $T(s, a, s') = P(G' \mid G)$, i.e. the probability of having a new set of goals G' in s' given that the set of goals in s was G, defined as:

$$P(G'|G) = \begin{cases} 0 & \text{if } G \not\subseteq G' \\ 1 & \text{if } G = G' = \mathcal{G} \\ \Big[\prod_{g \in G' - G} P(g)\Big] \times \Big[\prod_{g \in \mathcal{G} - G'} (1 - P(g))\Big] & \text{otherwise} \end{cases} \quad (1)$$

The probability is zero when $G \not\subseteq G'$, given that goals do not disappear once they have arrived; and it is one when G and G' are both the set of all possible goals, i.e. all possible goals have already arrived. Otherwise, the first product is the probability that the set of new goals $(G' - G)$ arrives in s', and the second product is the probability that the remaining possible goals (that are not in G') do not arrive in s'. $P(g)$ is the input goal arrival distribution defining the probability that a new goal g arrives at a given time-step. Each $P(g)$ is a Bernoulli process, i.e. probabilities are independent and identically distributed for all time-steps. So, given $P(g)$ for each goal g, we can completely define $T(s, a, s')$.

We use a cost function for the OCPP (Definition 1) based on Burns *et al.* work. Specifically, an instantaneous penalty $k_g \in \mathbb{R}^+$ is paid at every time-step for every unachieved goal g that has already arrived. The idea of these penalties is to promote the prompt achievement of goals. We consider the penalty is paid after the execution of actions and with respect to the goals in the next state. Thus, the cost does not depend on the source state s. We define the following cost function:

$$C(a, s') = C(a, (w', G')) = cost(a) + penalty(w', G') \quad (2)$$

where $cost(a) \in \mathbb{R}_0^+$ is the action cost. The penalty for a set of goals G' in a given state w' is defined as the sum of the penalties of its individual goals. Then:

$$penalty(w', G') = \sum_{g' \in G'} penalty(w', g') \quad (3)$$

and $penalty(w', g') = \begin{cases} k_{g'} & \text{if } g' \notin w' \\ 0 & \text{otherwise} \end{cases}$

Action-value and Value Functions

Given an OCPP, action selection at every time-step is a decision problem that can be posed as minimizing the cost of a solution over a finite horizon $H \geq 0$ starting from the current state. As the world is deterministic, the selected action should minimize the sum of the costs over the expected goal sets.

Given a state s and an applicable action a, the optimal *action-value function* for horizon H, $Q^*(s, a, H)$, provides the minimum expected cost at horizon H for the state s when a is applied. Thus, the optimal *value function*, $V^*(s, H)$, that provides the minimum expected cost at horizon H for state s [Bellman and Kalaba, 1965], is computed as:

$$V^*(s, H) = \min_{a \in A(s)} Q^*(s, a, H) \quad (4)$$

The Equation 5 in Figure 1 defines $Q^*(s, a, H)$ in the general case and the Equation 6 also in Figure 1 defines its instantiation to our problem. This instantiated equation considers that the part of the cost function involving the action cost is independent of the next state (Equation 2), and that the transition function for $a \in A(s)$ expresses $P(G' \mid G)$ (Equation 1) which is independent of the world states and actions. The optimal policy is: $\pi^*(s, H) = \operatorname{argmin}_{a \in A(s)} Q^*(s, a, H)$.

Finding this optimal policy is very costly due to the number of possible goal sets at every time-step and to the usual huge number of actions and world states. Therefore, it is not viable in practice, as was shown by Burns *et al.* 2012. In this paper we study more efficient action selection approaches that estimate the optimal value function and are based on computing minimal cost plans, as the one proposed by Burns *et al*. They evaluated their solution using domain-specific solvers. Our aim is to solve OCPPs efficiently and in a domain-independent way. Thus, we adapt their idea to domain-independent planning and propose additional alternative approaches. With this purpose we introduce first PSP-HSP planning tasks.

3 PSP-HSP Planning Tasks

Classical planners fail when they cannot achieve all goals, even when a subset of them could be achieved. This is a strong constraint for the design of alternative solutions to the OCPP involving a minimization horizon, since it would involve paying unnecessary penalties. The existence of a horizon further restricts the problem. Partial Satisfaction Planning (PSP) models can handle the so-called *soft goals* that are goals that are not strictly necessary to achieve. Therefore, PSP apparently seems more adequate than classical planning for solving OCPP tasks. Also, the planning model to be used requires to deal with *instantaneous penalties paid for unachieved goals*. Thus, the cost function should be state-dependent, since the cost depends on the unachieved goals, which depend on the state.

Definition 2 *A* **Partial Satisfaction Planning task with Horizon and State-Dependent costs (PSP-HSD)** *is defined by a tuple* $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, SG, \mathcal{C}, H)$*; where* $\mathcal{F}$ *is a finite set of fluents;* $\mathcal{A}$ *is a finite set of actions, each* $a \in \mathcal{A}$ *is defined by its preconditions,* $pre(a) \subseteq \mathcal{F}$*, positive and negative effects,*

$$Q^*(s,a,H) = \begin{cases} 0 & \text{if } H = 0 \\ \sum_{s'} T(s,a,s')\big[C(s,a,s') + V^*(s', H-1)\big] & \text{otherwise} \end{cases} \quad (5)$$

$$Q^*(s,a,H) = \begin{cases} 0 & \text{if } H = 0 \\ cost(a) + \sum_{s'=(w',G')} P(G' \mid G)\big[penalty(w',G') + V^*(s', H-1)\big] & \text{otherwise} \end{cases} \quad (6)$$

Figure 1: Generic (Equation 5) and instantiated (Equation 6) optimal action-value function for actions a that are applicable in s. States $s' = (w', G')$ are possible next states when a is applied to $s = (w, G)$; w' is the result of applying a to w.

$add(a), del(a) \subseteq \mathcal{F}$. $\mathcal{I} \subseteq \mathcal{F}$ is the initial state and $SG \subseteq \mathcal{F}$ is a set of soft goals. The states $w \in \mathcal{W}$ are collections of fluents from $\mathcal{F}$. $\mathcal{C} : \mathcal{A} \times \mathcal{W} \to \mathbb{R}_0^+$ is a state-dependent action cost function, that is computed based on the soft goals, $\mathcal{C}(a, w) = f_{SG}(a, w)$. $H \in \mathbb{N}_0$ represents a finite horizon.

An action sequence π is a plan for a PSP-HSD task Π if it is applicable in $\mathcal{I}$ and $|\pi| = H$.[1] Let $\pi = a_1, \ldots, a_H$ be a plan that generates the sequence of states $w_1, \ldots, w_H$ when applied to $\mathcal{I}$. The cost of π is defined as $c(\pi) = \sum_{i=1}^{H} \mathcal{C}(a_i, w_i)$. Optimal plans are those that minimize $c(\pi)$. PSP-HSP tasks are decidable by definition for $H < \infty$, since the set of states reachable from the initial state is finite, and the reachable part of the state transition graph can be explicitly constructed.

4 Alternative Action Selection Approaches

In this section, we present different approaches whose action selection mechanisms are based on estimations of the value function [Bellman and Kalaba, 1965]. The first one is based on hindsight optimization [Burns *et al.*, 2012], but adapted to domain-independent planning. The others are contributions of this paper.

4.1 Hindsight Optimization (HO)

The approach proposed by Burns *et al.* [2012] was based on previous work on hindsight optimization [Chong *et al.*, 2000; Yoon *et al.*, 2008; 2010]. The idea is to approximate the optimal action-value function by the expected value of minimum cost plans. Since the only uncertainty comes from the future goals, this expected value is the average of the optimal plan cost for $width$ samples $\{F^1, \ldots, F^{width}\}, F^i \subseteq \mathcal{G} \setminus G$ over the distribution of possible goal arrivals for the upcoming time-steps (the difference between the set of all goals and the already known goals, $\mathcal{G} \setminus G$). Each sample fixes a possible future in terms of goals.

More formally, if the current state is $s_0 = (w_0, G_0)$, the selected action at s_0 will be the applicable action $a \in A(s_0)$ that minimizes $\hat{Q}_{HO}(s_0, a, H)$, an estimation of the action-value function:

$$a = \operatorname*{argmin}_{a \in A(w_0)} \hat{Q}_{HO}(s_0, a, H)$$

$\hat{Q}_{HO}(s_0, a, H)$ is computed as the sum of the cost of a plus the average cost of $width$ minimal cost plans generated from the next state, one for each sampled future subset of goals F^i. Then, given F^i, this next state is $s_1^i = (w_1, G_0 \cup F^i)$, where w_1 is the resulting world state when a is applied to w_0, and the goals are the result of adding F^i to the set of goals G_0 at s_0. Each of these planning tasks is now fully deterministic and can be expressed as the PSP-HSD task $\Pi_{HO}^i(s_1^i, H-1) = (\mathcal{F}, \mathcal{A}, w_1, G_0 \cup F^i, \mathcal{C}_{HO}^i, H-1)$, where $\mathcal{F}$ is a set of fluents that define the OCPP world states W; $\mathcal{A}$ are the OCPP actions, defined in terms of their preconditions and effects on the world states; w_1 is the initial state; and $G \cup F^i$ is the set of soft goals. The cost function $\mathcal{C}_{HO}^i(a, w')$ is computed by Equation 2 (cost function for the OCPP), but taking as arguments a and a state s' composed as $s' = (w', G_0 \cup F^i)$.

Formally, for every current OCPP state s_0 and applicable action $a \in A(s_0)$, $\hat{Q}_{HO}(s_0, a, H)$ is defined as shown in Equation 6 (in Figure 2). Thus, the optimal value function for s_0 in HO is estimated as:

$$\hat{V}_{HO}(s_0, H) = \min_{a \in A(s_0)} \hat{Q}_{HO}(s_0, a, H)$$

At each time-step, the number of planning processes executed to solve the action decision problem is $width \times |A(s_0)|$. And the total number of calls to the planner in a time period of length H is: $\sum_{s_i} width \times |A(s_i)|$, where s_i are the states at each time point between s_0 and the state at $H - 1$.

4.2 Goal-Distribution-Sensitive Planning

Given that HO requires solving many planning tasks at each decision-making step, we propose Goal-Distribution-Sensitive Planning (GDS) to reduce the number of planning processes executed at each time-step. Instead of using sampling, the intuition behind GDS is to approximate the value function by the cost of a unique plan with minimum cost, obtained from a planning task that is sensitive to the goal arrival distribution.

Since each goal arrival distribution $P(g)$ is a Bernoulli process, the random variable N_g representing the *number of steps until goal arrival* follows a geometric distribution with mean $E(N_g) = \frac{1}{P(g)}$. Ideally, the penalty will be minimal if each goal is achieved before $E(N_g)$ steps. On average, goals for which $E(N_g)$ is smaller will start to pay penalty before if they are not achieved. Then, we anticipate the penalty in planning time by distributing it between the expected steps until goal arrival. At each of these steps, if a goal g has not been achieved, the penalty will be $\frac{k_g}{1/P(g)} = k_g \times P(g)$. This allows us to define a cost function that is aware of the goal arrival distribution.

[1] Note that we assume the existence of a no-op action.

$$\hat{Q}_{HO}(s_0, a, H) = \begin{cases} 0 & \text{if } H = 0 \\ cost(a) + \frac{1}{width} \sum_{i=1}^{width} \left[penalty(w_1, G_0 \cup F^i) + c(\pi^*_{\Pi^i_{HO}(s^i_1, H-1)}) \right] & \text{otherwise} \end{cases} \quad (6)$$

$$\hat{Q}_{SE}(s_0, a, H) = \begin{cases} 0 & \text{if } H = 0 \\ cost(a) + penalty_{GDS}(w_1, G_0 \cup F) + c(\pi^*_{\Pi_{GDS}(s_1, H-1)}) & \text{otherwise} \end{cases} \quad (7)$$

Figure 2: Estimation of the action-value function in HO (above) and GDS-SE (below) for $a \in A(s_0)$. $s^i_1 = (w_1, G_0 \cup F^i)$ and $s_1 = (w_1, G_0 \cup F)$ are the next states when a is applied to $s_0 = (w_0, G_0)$, respectively.

Given an OCPP state $s = (w, G)$, the corresponding PSP-HSD problem would be $\Pi_{GDS}(s, H) = (\mathcal{F}, \mathcal{A}, w, G \cup F, \mathcal{C}_{GDS}, H)$. This problem is also fully deterministic. $\mathcal{F}$ and $\mathcal{A}$ are defined as in HO; the soft goals are equal to the set of all posible goals: the known ones at s, G, and the future ones $F = \mathcal{G} \setminus G$. The cost function $\mathcal{C}_{GDS}$ is:

$$\mathcal{C}_{GDS}(a, w') = cost(a) + penalty_{GDS}(w', G \cup F) \quad (7)$$

where, the penalty is computed as the sum of individual goal penalties (as in Equation 3). And the individual goal penalty is defined as:

$$penalty_{GDS}(w', g) = \begin{cases} k_g \times P(g) & \text{if } g \in F \wedge g \notin w' \\ k_g & \text{if } g \in G \wedge g \notin w' \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Therefore, in GDS, $V^*(s, H)$ is approximated directly as the cost of the optimal plan $\hat{V}_{GDS}(s, H) = c(\pi^*_{\Pi_{GDS}(s,H)})$.

This approach allows for three different action selection schemes: Step Execution (SE), Long Execution (LE) and Reactive (R). In **Step Execution (SE)**, for every current OCPP state $s_0 = (w_0, G_0)$ and applicable action $a \in A(s_0)$, the next state is $s_1 = (w_1, G_0 \cup F)$, where w_1 is the resulting world state of applying a to w_0. Then, the optimal action-value function is approximated by Equation 7 (in Figure 2). The selected action $a = \text{argmin}_{a \in A(s_0)} \hat{Q}_{SE}(s_0, a, H)$. The number of planning processes executed at every current state s_0 is exactly the number of applicable actions: $|A(s_0)|$, and the total number of calls to the planner is: $\sum_{s_i} |A(s_i)|$ (s_i has the same meaning as before).

In **Long-term Execution (LE)**, given that we have a plan that provides the current state value function, $\hat{V}_{GDS}(s_0, H)$, instead of computing a different plan at each time-step, we can choose to execute the actions in that plan as long as possible. Actually, since actions are deterministic on world states, we only need to compute a new plan when the current one finishes its execution or new and non-achieved goals arrive. In those cases, a new plan is generated, taking the state at the current time-step as initial state, s_0. The number of planning processes executed at each time-step is less than or equal to one. And the total number of calls to the planner is less than or equal to the number of steps where new goals arrived before H. Given that this scheme greatly reduces the number of planning processes, and therefore the time of the action selection process, the horizon can be larger than the horizons applied in other schemes as HO.

The **Reactive (R)** scheme just plans for the current goals, ignoring any future. And it replans at each time step when new goals arrive. It is based on the reactive scheme used by Burns *et al.*. The corresponding PSP-HSD problem is similar to the ones defined for SE and LE, but without considering any future. That is, for a current state $s_0 = (w_0, G_0)$ where replanning is performed, the PSP-HSD problem to solve is: $\Pi^{\emptyset}_{GDS}(s_0) = (\mathcal{F}, \mathcal{A}, w_0, G_0 \cup \emptyset, \mathcal{C}_{GDS}, H)$, where $\emptyset$ indicates an empty set of future goals. The total number of calls to the planner is equal to the number of steps where new goals arrived before H.

5 Solving PSP-HSD Problems

Current planners are not prepared to deal directly with all features of PSP-HSD tasks: bounded horizon, soft goals and state-dependent costs. Hence, we propose to compile PSP-HSD tasks into PDDL models that can be handled by some existing planner. More specifically, we model PSP-HSD tasks as numeric planning tasks expressed in PDDL2.1 [Fox and Long, 2003]. A numeric planning task is defined by a tuple $\Psi = (PN, A, I, G, M)$, where PN is a set of propositional and numeric fluents, A is a set of actions, $I \subseteq PN$ and $G \subseteq PN$ are the initial state and the (hard) goals respectively and M is a numeric metric function [Hoffmann, 2003]. Optimal plans are those that achieve all goals with the minimum metric value.

Given the PSP-HSD task $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, SG, \mathcal{C}, H)$, where $\mathcal{C}$ is defined additively in terms of action costs and goal penalties, the corresponding numeric task Ψ_Π is generated automatically: soft goals are compiled away, the bounded horizon is modelled with numeric fluents and the metric depends on a state-dependent cost function. To compile soft goals away we have adapted Keyder&Geffner approach [2009], considering also the ideas of Benton *et al.* [2012] about state-dependent costs. The specific differences to these compilations are explained at the end of this section. The corresponding numeric planning task is $\Psi_\Pi = (\mathcal{F} \cup \mathcal{F}^+, \mathcal{A}', \mathcal{I} \cup \mathcal{I}^+, G, M)$. $\mathcal{F}^+$ contains the additional propositions and numeric functions defined in the first column of Table 1, and described in the second column. The set of actions $\mathcal{A}'$ is generated from the actions in $\mathcal{A}$ and includes additional *forgo* actions as explained below. $\mathcal{I}^+$ contains the initialization of the new facts and functions. They are specified in the last column of Table 1. G contains a fact *(collected-p args)* for every soft goal *(p args)* $\in SG$. These are considered as the hard goals of the new problem. For instance, if we have the soft goal *(at pkg1 loc3)* in the Logistics domain, it would generate the hard goal *(collected-at pkg1 loc3)*. M is the metric of the task, defined to minimize *(action-cost)* + *(goal-cost)*.

$\mathcal{F}^+$ **elements**	**Description**	**Value at the initial state** $\mathcal{I}^+$
(sg-p args)	Represents the fact that *p with arguments args is a pending soft goal in* SG	One fact of this type per soft goal g *=(p args)* in SG except for those included in $\mathcal{I}$
(collected-p args)	Hard goals, one for each soft goal *p* with arguments *args*	One fact of this type per soft goal *g= (p args)* such that $g \in \mathcal{I}$
(penalty-sg-p args)	Function that represents the individual penalty for every soft goal *(p args)*	Pre-computed individual penalty to pay for a soft goal *g=(p args)* when it is not achieved in a state, as defined by $\mathcal{C}$.
(total-penalty)	Function that represents the sum of all instantaneous penalties that should be paid at each state	Initial sum of penalties for all soft goals *g=(p args)* not included in $\mathcal{I}$. It is computed as $\sum_{g \in SG, g \notin \mathcal{I}}$ *(penalty-sg-p args)*
(num-steps)	Function that represents the depth of each node in the search	0
(horizon)	Function that represents the horizon	H
(goal-cost)	Function that represents the total cost due to penalties in a state	0
(action-cost)	Function that represents the total cost of the applied actions in a state	0

Table 1: Description of the new predicates and numeric functions in $\mathcal{F}^+$ and their initial values.

The following paragraphs explain how the new set of actions $\mathcal{A}'$ is generated in terms of PDDL action schemas.

Every action $a \in \mathcal{A}$ **adding a literal that matches any soft goal** (i.e. a fact of type *(p args)*, where p is a predicate present in SG), generates two actions, $a_p^{\sim}$ and a_p, in $\mathcal{A}'$. The action $a_p^{\sim}$ represents the case where the added fact is not a pending soft goal. And a_p represents the opposite case. The action $a_p^{\sim}$ is defined as follows:[2]

- $pre(a_p^{\sim}) = pre(a) \cup pre^H \cup \{$ *(not (sg-p args))* $\}$
- $add(a_p^{\sim}) = add(a) \cup add^C \cup add^H \cup add^{\sim}$
- $del(a_p^{\sim}) = del(a)$

where the precondition *(not (sg-p args))* ensures there is not a soft goal in Π for the fact the action achieves; pre^H guarantees that the action is applied inside the horizon; add^C increases the fluent *(action-cost)* by the cost of the action, $cost(a)$; add^H updates the current depth; and $add^{\sim}$ updates the goal cost:

- $pre^H = \{$ *(< (num-steps) (horizon))* $\}$
- $add^C = \{$ *(increase (action-cost) <cost(a)>)* $\}$
- $add^H = \{$*(increase (num-steps) 1)*$\}$
- $add^{\sim} = \{$ *(increase (goal-cost) (total-penalty))* $\}$

Actions a_p are the only actions that actually achieve goals. They achieve the hard goals *(collected-p args)*, since they ensure the achievement of the soft goal *(sg-p args)* within the horizon. Besides, they update the cost for not achieving goals, considering that the penalty of the achieved goal is not paid from then on. They also decrease the total penalty by the penalty of the achieved soft goal. The add effects add^- denote a decrease of the total penalty:

- $pre(a_p) = pre(a) \cup pre^H \cup \{$ *(sg-p args)* $\}$
- $add(a_p) = add(a) \cup add^C \cup add^H \cup \{$ *(collected-p args)* $\} \cup add^-$
- $del(a_p) = del(a) \cup \{$ *(sg-p args)* $\}$

where:

$$add^- = \{\text{(increase (goal-cost) (- (total-penalty) (penalty-sg-p args))),}\ \text{(decrease (total-penalty) (penalty-sg-p args))}\}$$

Every action $a \in \mathcal{A}$ **deleting a literal that matches any soft goal**, generates also two actions, $a_{\neg p}^{\sim}$ and $a_{\neg p}$, in $\mathcal{A}'$. The action $a_{\neg p}^{\sim}$ represents the case where the deleted fact is not a collected soft goal. And $a_{\neg p}$ represents the opposite case. Then, the action $a_{\neg p}^{\sim}$ is defined as follows:

- $pre(a_{\neg p}^{\sim}) = pre(a) \cup pre^H \cup \{$ *(not (collected-p args))* $\}$
- $add(a_{\neg p}^{\sim}) = add(a) \cup add^C \cup add^H \cup add^{\sim}$
- $del(a_{\neg p}^{\sim}) = del(a)$

The action $a_{\neg p}$ deletes a collected soft goal and updates the goal cost and total penalty since the penalty for the deleted soft goals should be paid again:

- $pre(a_{\neg p}) = pre(a) \cup pre^H \cup \{$ *(collected-p args)* $\}$
- $add(a_{\neg p}) = add(a) \cup add^C \cup add^H \cup \{$ *(sg-p args)* $\} \cup add^+$
- $del(a_{\neg p}) = del(a) \cup \{$ *(not (collected-p args))* $\}$

where add^+ denotes an increase of the total penalty:

$$add^+ = \{\text{(increase (goal-cost) (+ (total-penalty) (penalty-sg-p args))),}\ \text{(increase (total-penalty) (penalty-sg-p args))}\}$$

Every other action $a \in \mathcal{A}$ (it that does not add/delete any literal that matches with the predicate of a soft goal), generates a new action $a' \in \mathcal{A}'$ defined as follows:[3]

- $pre(a') = pre(a) \cup pre^H$
- $add(a') = add(a) \cup add^C \cup add^H \cup add^{\sim}$
- $del(a') = del(a)$

Additional new $forgo_p$ actions are included in $\mathcal{A}'$ for each different predicate *p* in a pending soft goal. Forgo actions are fictitious actions, whose only purpose is to generate the hard goal *(collected-p args)* for non-achieved soft goals once the horizon has been reached. They are removed from the plans on a post-processing step. Since the penalty for not achieving these soft goals has already been paid, *forgo* actions do not have cost. The $forgo_p$ actions have the following definition:

- $pre(forgo_p) = \{$ *(≥ (num-steps) (horizon)), (sg-p args))* $\}$
- $add(forgo_p) = \{$ *(collected-p args)* $\}$
- $del(forgo_p) = \{$ *(sg-p args)* $\}$

The main difference between our compilation and the compilation of Keyder&Geffner 2009 is that ours allows the planner to pay the penalty for not achieving goals step by step. On the contrary, in Keyder&Geffner compilation the full penalty for not achieving a goal is paid when the goal is artificially added, i.e. in the corresponding *forgo* action. The approach by Benton *et al.* 2012 represents penalties by a continuous cost function since the time is represented by a PDDL+ process simulating continuous time. Our approach can be seen as a discretized version of this approach for numerical planning in PDDL2.1.

Instead of using numerical planning, an alternative approach would be to compile away state-dependent actions costs [Geisser *et al.*, 2015]. However, the compilation would

[2]For the sake of clarity, we assume each domain action only adds or deletes one soft goal. However, this assumption can be easily removed by just generating the corresponding cross-product.

[3]The no-op action belongs to this type.

need an exponential number of actions in our case, which would make this approach intractable.

6 Experiments and Results

Very few domain-independent planners handle adequately state-dependent cost functions. Therefore, in practice, we are restricted to these planners which in addition compute sub-optimal plans. For these experiments the base solver was LPG-td [Gerevini *et al.*, 2006]. So, we compare four alternatives: Hindsight optimization (HO), Reactive and our techniques, GDS-LE and GDS-SE. There are no other techniques, to our knowledge, to compare against. It is easy to show that any standard MDP-based technique will not scale to the size of the problems we use for testing, as was also previously shown [Burns *et al.*, 2012].

We have used three well known IPC planning benchmarks: Satellite, Rovers and TPP; and a fourth domain that we modelled in PDDL to simulate the surveillance UAV scenario described by Burns *et al.* [2012]. These domains are very appropriate for this research, since they might benefit from anticipating future goals and they fulfill all the defined assumptions. For each IPC domain we modified the available random problem generator to create a new class of problems in which the list of goals are the majority of possible goals given the problem objects. For instance, in Satellite, the goals consist of having images of all directions in all modes. Also, we defined a no-op action with zero cost.

The experiments focused on two types of scenarios. In the first one we analyze the anticipatory behaviour of GDS planning having a constant penalty per goal with a random distribution of goal arrivals. In the second one we study the effect of having a particular penalty distribution for not achieving goals, while goals have an equal probability of arriving at current the step.

6.1 Random Goal Arrival Distribution

For each domain we generated three problems of increasing size (in number of goals as shown in Table 2). We have (suboptimally) solved these problems and then set the execution horizon as 1.25 times the length of the solution (as in [Burns *et al.*, 2012]). For each problem, we generated a random set of 10 future goal schedules to be provided as input to all planning-execution cycles. Thus, the total number of instances was 120. Each schedule was generated by sampling a uniform distribution to assign to each goal the probability of appearing within the horizon. We computed each goal arrival probability $P(g)$ at each step from this value. Then, the schedule is generated doing Bernoulli trials with $P(g)$ until getting a success or reaching the horizon. At the end, each schedule is a list of ordered pairs (g_i, t_i), where $g_i \in F$ is a future goal, and t_i is the time-step when g_i will arrive.

We set as penalty $k_g = 100$ for all goals in all executions. Action costs in the domains are in the order of units. Therefore, the total cost will mainly reflect the ability of different approaches for optimizing the penalty. Experiments were run on a cluster with Intel XEON 2.93 Ghz nodes using Linux Ubuntu 12.04 LTS. Each machine was limited to use 7.5Gb of RAM. The configuration of each evaluated approach is:

Problem	**GDS-LE**		**GDS-SE**		**HO**		**Reactive**	
(# goals)	time	cost	time	cost	time	cost	time	cost
rover-1 (14)	5.6	2.3	68.8	3.9	22.5	20.4	6.7	3.7
rover-2 (22)	9.3	12.5	125.3	34.7	57.2	77.7	11.1	10.0
rover-3 (26)	9.2	14.6	160.6	41.1	87.7	92.3	12.7	9.2
satel-1 (32)	9.7	4.2	66.6	7.3	27.0	51.0	13.6	7.7
satel-2 (50)	18.0	20.1	113.1	41.6	57.7	153.1	22.7	19.2
satel-3 (72)	22.8	41.9	129.1	127.7	89.9	253.4	30.1	30.9
tpp-1 (12)	5.0	3.4	52.5	2.6	19.4	15.5	5.0	4.0
tpp-2 (18)	7.3	9.0	87.7	6.2	38.6	35.3	7.4	7.2
tpp-3 (24)	10.1	18.9	124.5	17.1	80.7	65.2	11.3	12.3
uav-1 (24)	8.1	5.0	67.0	4.9	10.6	35.9	9.7	7.3
uav-2 (40)	15.6	16.7	121.3	13.2	20.5	125.8	18.2	16.7
uav-3 (60)	22.2	59.8	181.5	63.7	68.1	286.4	27.4	21.9

Table 2: Average accumulated time in minutes and average total cost in thousand units on 10 executions.

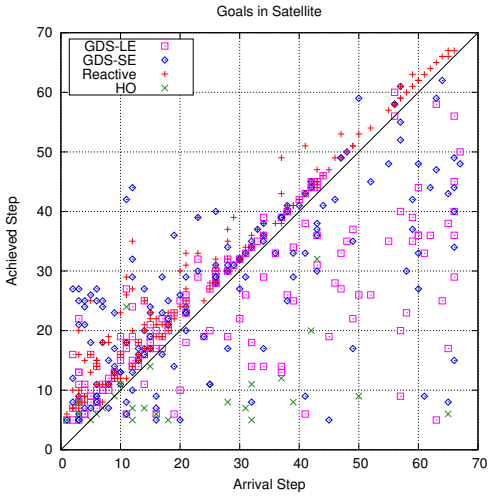

Figure 3: Arrival vs. achievement times for goals in problem satel-1 and for the 10 executions.

- **Reactive (R)**: it runs LPG-td with the configuration for anytime planning and a time bound of 60 seconds. H is equal to the execution horizon.
- **GDS-LE**: the same configuration as Reactive.
- **GDS-SE**: we set $H = 8$. The step-timeout of 60 seconds is divided equally between the planner calls (LPG-td in anytime mode) for each applicable action.
- **Hindsight Optimization (HO)**: we set $H = 8$ and $width = 20$ samples. Dividing the same 60 seconds between the number of needed planner calls would result in a very short time. Therefore, we let the planner run until it returns the first solution (LGP-td speed mode).

Table 2 shows the average total cost for each problem and the average accumulated planning time during a complete execution. GDS-LE improved the total cost of Reactive in the first problem of each domain, and GDS-LE was better than Reactive in three cases. This reflects the ability of GDS approaches to reduce the penalty by means of anticipating future goals. This can be confirmed in Figure 3 which plots the relation between the goal arrival and goal achievement time in all executions of the problem satel-1. All points below the diagonal represent goals achieved before arrival.

Regarding HO, the step time bound greatly affects its per-

Problem	GDS-LE		GDS-SE		Reactive	
(# goals)	time	cost	time	cost	time	cost
rovers-2	35.0	5.3	595.6	8.9	49.0	7.8
satel-2	88.0	16.0	562.8	20.8	110.0	18.5
tpp-2	44.0	8.6	404.2	6.8	45.5	10.3
uav-2	70.5	14.3	600.4	12.0	84.5	15.5

Table 3: Re-execution of medium size problems with 300 seconds per step. Results show average accumulated time in minutes and average total cost in thousand units on 10 executions.

Problem	GDS-LE		GDS-SE		Reactive	
(# goals)	time	cost	time	cost	time	cost
rovers-2	9.1	13.5	69.2	23.6	9.9	35.3
satel-2	16.9	16.8	64.7	24.5	20.9	41.2
tpp-2	8.3	44.4	52.0	34.7	8.0	66.8
uav-2	15.2	20.1	65.7	19.5	16.1	66.0

Table 4: Results show average accumulated time in minutes and average total cost in thousand units on 10 executions.

formance, with very bad results with our experimental setting. It is able to anticipate to some goals, as we can see in Figure 3. But, it left many others unachieved, because of the bad quality of the first solution given by the planner. In a domain-independent planning context, we do not forsee any alternative to the approximate computation of the best action to be applied at each time step. Any implementation of HO has to deal with the issue of a large number of planner calls imposed by the sampling of Monte Carlo's schemes. For instance, HO needed to solve 28,750 planning tasks on average in satel-1 and almost 78,000 in satel-3. Even though the horizon is small, there is no practical way to apply domain-independent optimal planning for these tasks in a reasonable per-step time limit.

GDS-SE and LE planning performance degrades in the second and third problems, because the number of possible future goals increases with the problem size, while the average goal arrival rate hardly increases between problems. Therefore, both GDS-LE and GDS-SE have to plan for larger goal sets, given that they plan for all problem goals in advance. However, Reactive only plans for known goals, so its problems remain relatively small and they are easier to optimize for the planner. In execution traces of larger problems we found that plans of the GDS approaches were of bad quality, mainly because the planner did not have enough time to improve its first solutions. Therefore, the strength of GDS planning will depend on the available planning time at each step. The longer time per step, the longer time the planner would have to optimize its solutions.

To verify this intuition we have re-executed the second problem of all domains (we already obtained good results with the first one), but providing a time bound of 300 seconds per time step. Thus, LPG-td has more time to optimize plan quality. Results are shown in Table 3. As we can see, the average cost for Reactive is fairly similar, but GDS planning approaches greatly improved their performance, specially GDS-SE. For instance, GDS-SE got almost a reduction of 25% of the average cost achieved with 60 seconds per step. With this setting GDS-LE was better than Reactive in all domains, and GDS-SE was better in two out of four domains.

6.2 Different Goal Penalties

In this experiment we wanted to analyze the behaviour of anticipatory search when goals have the same arrival probability, but they have different penalties. We have used the small problems of each domain. For each problem we shuffled the goal list and then assigned to each goal i the penalty $1000 \times 1/r_i$, where r_i is the position of goal i in the list of goals. Using this formula we tried to reproduce a power law

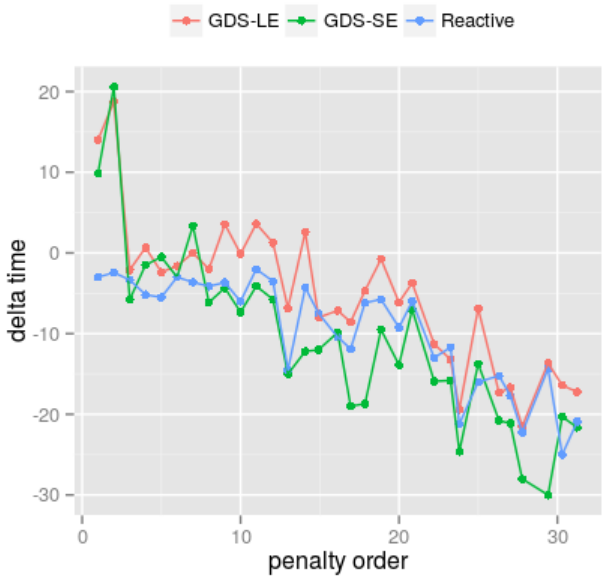

Figure 4: Average time difference of the arrival and achievement of goals for each of the evaluated configurations in the satel-1 problem.

distribution of penalties, in which very few goals are very important, but the majority of goals have low penalty per step if they are not achieved. For each problem, all goals have the probability of appearing in a future schedule of 0.8. Goal arrival probability was set accordingly taking into account the length of the schedule (i.e., 1.25 times the length of the solution of the original problem, as in the first experiment). The time bound per step was set again to 60 seconds.

Table 4 shows the average accumulated time and cost for this setting. Clearly, when there are some few important goals, it makes sense to anticipate to them for not paying high penalties. GDS planning approaches are able to recognize this situation, performing much better than Reactive in all domains. Looking for more details, we have computed Δt as the time step difference between the times when each goal has arrived and it has been achieved. Figure 4 shows the average of Δt in the satel-1 problem, sorting the goals in terms of their penalties. The x-axis is the goal order (i.e., r_i position used to generate its penalty). The y-axis is the average of Δt for each goal. Positive values of Δt indicate that on average this goal has been achieved before its arrival. All configurations show a trend towards worse Δt when the goals

become less important. However, GDS-LE and GDS-SE start from higher points and are able to obtain positive Δt for some goals, especially for the two goals with a higher penalty.

7 Conclusions and Future Work

We have introduced new approaches to solve OCPPs from a domain-independent planning perspective. We have characterized them in terms of PSP-HSD tasks, that provide a general setting that is useful to model other approaches based on deterministic planning. Also, we have proposed a compilation of these tasks into numerical planning that allows us to solve them with current planners. The results are very promising, regarding their anticipatory behavior in the short allotted time. If the base planner is able to provide good quality plans in the available time, GDS-planning approaches perform better than a hindsight optimization based approach. Our GDS planning schemes are not linked to any particular planner, so they can benefit from future improvements on state-dependent costs. PSP-HSD is a general framework that permits a wide variety of extensions: incorporating goal deadlines and priorities, studying different replanning schemes; or extending it to temporal and/or multi-agent planning.

Acknowledgements

The authors would like to thank Wheeler Ruml for his help on understanding their approach. This work has been partially supported by MICINN project TIN2014-55637-C2-1-R.

References

[Bellman and Kalaba, 1965] R. Bellman and R. Kalaba. *Dynamic Programming and Modern Control Theory*. Academic Press, 1965.

[Benton *et al.*, 2012] J. Benton, A. J. Coles, and A. I. Coles. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, June 2012.

[Burns *et al.*, 2012] Ethan Burns, J. Benton, Wheeler Ruml, Sung Wook Yoon, and Minh Binh Do. Anticipatory on-line planning. In Lee McCluskey, Brian Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[Chong *et al.*, 2000] Edwin K. P. Chong, Robert L. Givan, and Hyeong Soo Chang. A framework for simulation-based network control via hindsight optimization. In *IEEE Conference on Decision and Control.*, 2000.

[Fox and Long, 2003] Maria Fox and Dereck Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, pages 61–124, 2003.

[Geisser *et al.*, 2015] Florian Geisser, Thomas Keller, and Robert Mattmüller. Delete relaxations for planning with state-dependent action costs. In *Proc. of IJCAI*, pages 1573–1579, 2015.

[Gerevini *et al.*, 2006] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research (JAIR)*, 25:187–231, 2006.

[Hoffmann, 2003] Jörg Hoffmann. The METRIC-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)*, 20:291–341, 2003.

[Keyder and Geffner, 2009] Emil Keyder and Hector Geffner. Soft goals can be compiled away. *Journal of Artificial Intelligence Research (JAIR)*, 36:547–556, 2009.

[Kiesel *et al.*, 2013] Scott Kiesel, Ethan Burns, Wheeler Ruml, J. Benton, and Frank Kreimendahl. Open world planning for robots via hindsight optimization. In *Proceedings of the ICAPS-13 Workshop on Planning and Robotics (PlanRob-13)*, 2013.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY., 1994.

[Ruml *et al.*, 2011] Wheeler Ruml, Minh Binh Do, Rong Zhou, and Markus P. J. Fromherz. On-line planning and scheduling: An application to controlling modular printers. *Journal of Artificial Intelligence Research (JAIR)*, 40:415–468, 2011.

[Yoon *et al.*, 2008] S. Yoon, A. Fern, R. Givan, and S Kambhampati. Probabilistic planning via determinization in hindsight. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.

[Yoon *et al.*, 2010] S. Yoon, W. Ruml, J.and Benton, and M. B. Do. Improving determinization in hindsight for online probabilistic planning via determinization in hindsight. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2010.