AI Communications -1 (2018) 1–19
DOI 10.3233/AIC-180753
IOS Press

Anticipation of goals in automated planning

Raquel Fuentetaja *, Daniel Borrajo and Tomás de la Rosa
Departamento de Informática, Universidad Carlos III de Madrid, Av. Universidad 30, Leganés, Madrid, Spain
E-mails: rfuentet@inf.uc3m.es, dborrajo@ia.uc3m.es, trosa@inf.uc3m.es

Abstract. In the context of deliberative reasoning, autonomous systems should be able to explicitly reason about goals. Most research 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 in an on-line continual planning setting, additional goals may arrive over time. When 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 presented an approach that exhibited such anticipatory behavior based on hindsight optimization on a domain-dependent setup. In this paper, we tackle this problem from the point of view of domain-independent planning. The available per-step time in on-line continual problems can be very short and domain-independent planners can scale poorly in such short time. In fact, a domain-independent reimplementation of the hindsight optimization scheme may not even be applicable. We propose several alternative approaches that consider future goals in a domain-independent planning process. Experimental results in several benchmark domains suggest that these approaches exhibit a more efficient and effective behavior than a reactive approach in which goals are pursued after their arrival.

Keywords: Goal reasoning, Automated Planning, AP, Artificial Intelligence, AI

1. Introduction

Most research 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 [17].

In this paper we focus on planning problems with deterministic worlds but non-deterministic goal arrivals, as the work introduced recently by Burns *et al.* on On-line Continual Planning Problems (OCPPs) [5]. Other works have addressed the task of non-deterministic worlds, but fixed goals [15]. The approach by Burns *et al.* 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 [8,18,19], that provides a reduction from an stochastic planning problem to a deterministic one based on sampling. Burns *et al.* showed that their approach performed very well, and, in particular, better than reactive planning in several domains. However, their approach requires solving many planning tasks at every time step. If the solver can provide fast solutions, the approach works well. In their paper, they obtained fast solutions by using a domain-dependent problem solver. Given that our goal is to generalize this work to domain-independent planning, these more general planners take longer than domain-dependent ones in practice. Therefore, it remains unclear if Burns *et al.* approach is still applicable if the solver is domain-independent.

The aim of the present paper is to solve OCPPs efficiently in a domain-independent way. To define 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 several 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.

*Corresponding author. E-mail: rfuentet@inf.uc3m.es.

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 [14] to deal with soft goals in classical planning and on the one proposed by Benton et al. [2] to deal with state-dependent costs.

The paper is organized as follows. First, Section 2 introduces the general framework of the task to be solved. Next, Section 3 defines PSP-HSD tasks. Section 4 characterizes the approach by Burns *et al.* and describes our new alternative approaches, both in terms of PSP-HSD tasks. Next, Section 5 introduces the compilation of PSP-HSD tasks into numerical planning. Section 6 shows the experimental setup, some results and their analysis. Finally, Section 7 presents the conclusions of this work and introduces future work.

2. Problem description

The problem we tackle in this paper is an Online Continual Planning Problem (OCPP) in a deterministic world with uncertainty about future goals arrival, where the goal arrival probability distribution is known. We leverage the definition of OCPP by Burns *et al.* [5] as a Markov Decision Process (MDP).

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'.

The main differences with respect to the previous definition are that: (1) S is directly incorporated into the tuple, instead of the two terms, W and G; and (2) the cost function can be defined over the source state, the action and the destination state, instead of just over the source state and the action.

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 resulting state of applying a to s as $s' = \phi(a, s)$. ϕ is the transition function due to the application of actions, which is deterministic.[1] We will also 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. In the classical MDP framework [16], goals are implicitly represented with a reward function (instead of a cost function). In OCPP, goals are explicitly represented as part of the states and we use a cost function.

In this paper, we make the same assumptions as Burns *et al.* [5]: actions are deterministic on world states and the only uncertainty comes from future goal arrivals.

2.1. Transition and cost functions in our OCPPs

In this work with deal with a specific subset of OCPPs. Specifically, we assume that the arrival of each future goal is independent of the arrivals of the other goals, and that once a goal has arrived it remains in the list of known goals until its achievement, i.e. the environment does not remove arrived goals. These two assumptions are not explicitly mentioned by Burns *et al.*, but they are fullfilled by the domains used in their experiments. The problem solving task remains a difficult one even after making these assumptions. Additionally, these assumptions hold in a wide range of domains, including the ones we will use in our experiments.

Given those assumptions, the transition function expresses the probability of the next goal set given the previous state and the applied action: $T(s, a, s') = P_a(G' \mid s)$, defined as:

$$P_a\big(G' \mid s\big) = \begin{cases} 0 & \text{if } a \notin A(s) \\ \prod_{g \in G'} P_a(g \in G' \mid s) & \\ \quad \times \prod_{g \notin G'} P_a(g \notin G' \mid s) & \\ \text{otherwise} & \end{cases} \tag{1}$$

where $P_a(g \notin G' \mid s) = 1 - P_a(g \in G' \mid s)$ and

$$P_a\big(g \in G' \mid s\big) = \begin{cases} 1 & \text{if } g \in G \wedge g \notin w \wedge g \notin \phi(a, w) \\ P(g) & \text{otherwise} \end{cases} \tag{2}$$

[1] Since application of actions only affects the world state, we will use $\phi(a, s)$ or $\phi(a, w)$ interchangeably throughout the paper.

The probability is one for goals in G that have not been achieved; that is, they are not in w and the action a does not achieve them. Otherwise, it is $P(g)$, the input goal arrival distribution defining the probability that a new goal g arrives at a given time step. In this work we assume 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 $T(s, a, s')$.

Independence between goal arrivals is a realistic assumption in many real world problems, as logistics transport domains or taxi companies, where different requests are usually independent of each other. It can be generalized by considering in Equation (1) other relations among goal arrival probabilities. The assumption of probability distributions identically distributed is a simplification that can be generalized by considering time-dependent probability distributions.

Regarding the cost function of the OCPP, we have defined one 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 it in terms of the applied action and the next state $s' = (w', G')$:

$$\begin{aligned} C\big(s, a, s'\big) &= C\big(a, \big(w', G'\big)\big) \\ &= cost(a) + penalty\big(w', G'\big) \end{aligned} \tag{3}$$

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:

$$penalty\big(w', G'\big) = \sum_{g' \in G'} penalty\big(w', g'\big) \tag{4}$$

and the penalty for an individual goal is:

$$penalty\big(w', g'\big) = \begin{cases} k_{g'} & \text{if } g' \notin w' \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Goals that have arrived, were already achieved, and are unachieved again by some action also start to pay penalties once they become unachieved again.

Our cost function is based on Burns *et al.* work in the sense that we borrowed the idea of penalties paid for unachieved goals. However, their cost functions are domain-dependent and included different kinds of penalties and also rewards modeled as negative costs. Since we are in a domain-independent setting, we have opted for a simpler definition without rewards and where penalties only depend on the goals.

2.2. Action-value and value functions

Given an OCPP, action selection at every time step is a problem that can be posed as minimizing the cost of a solution over a finite horizon $0 \leqslant H < \infty$ 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. We assume a rolling or receding horizon to solve problems in an online manner, which implies that a fixed length moving horizon (H) is obtained at each decision time and the initial action selected for that horizon is executed. The intuition is that if the horizon is sufficiently long so as to provide a good estimate of the stationary behaviour of the system, the moving horizon control should perform well. In fact, it has been shown previously that the value of the rolling horizon policy converges geometrically to the optimal value [7,12].

Let s_t be the state of the system in stage t relative to the moving horizon H, where the current state is the state at stage 0. The goal is to find a policy $\pi = \{\pi_t, t = 0, \ldots, H-1\}$ among all possible policies Π, that minimizes the expected total cost for every state s in stage i, given by:

$$V_i^*(s) = \min_{\pi \in \Pi} E\left[\sum_{t=i}^{H-1} C\big(s_t, \pi_t(s_t), \phi\big(\pi_t(s_t), s_t\big)\big) \Big|_{s_i = s}\right] \tag{6}$$

where $i = 0, \ldots, H-1$, and $\pi_t : S \to A$ is a function that provides the policy action for states at stage t. $V_H^*(s) = 0$ for all states. The optimal value-function for the H-horizon MDP beginning in the current state is $V_0^*(s)$.

It is well known that V^* can be written recursively in terms of the *action-value function* Q^*. Given a state s and an applicable action a, the optimal action-value function for stage i, $Q_i^*(s, a)$, provides the minimum expected cost for the state s in stage i when a is applied. Thus:

$$V_i^*(s) = \min_{a \in A(s)} Q_i^*(s, a) \tag{7}$$

Equation (9) in Fig. 1 defines $Q_i^*(s,a)$ in the general case and Equation (10) also in Fig. 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 (3)), and that the transition function for $a \in A(s)$ expresses $P_a(G' \mid s)$ (Equation (1)). For a given current (starting) state, s, the optimal policy for the rollout horizon is:

$$\pi_0^*(s) = \arg\min_{a \in A(s)} Q_0^*(s,a) \tag{8}$$

Finding this policy is very costly due to the number of possible goal sets at every time step and to the usual huge number of instantiated actions and world states. Therefore, it is not viable in practice, as was shown by Burns *et al.* [5]. They proposed a more efficient action selection approach that estimates the optimal value function and is based on computing several minimal cost plans with samplings of the goals set. They evaluated their solution using domain-specific solvers. Our aim is to solve OCPPs efficiently in a domain-independent way. In this paper, we adapt their idea to domain-independent planning and propose additional alternative approaches that are viable in this setting. With that 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. Therefore, these planners are not directly appropriate for the OCPP task, where the next action to execute should be chosen at every time step. When selecting the next action to execute, it is better to have a plan that achieves only a subset of goals than having no plan. In addition to this, the existence of a horizon restricts the problem because achieving all goals can be more difficult or even impossible if the number of time steps where they should be achieved is limited.

Partial Satisfaction Planning (PSP) models can handle *soft goals*, which the planner is not strictly required to achieve [3,4]. Therefore, PSP seems more adequate than classical planning for solving OCPP tasks. Additionally, the planning model to be used must 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, $add(a), del(a) \subseteq \mathcal{F}$. $\mathcal{I} \subseteq \mathcal{F}$ 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 $\mathcal{F}$. $\mathcal{C} : \mathcal{A} \times \mathcal{W} \rightarrow \mathbb{R}_0^+$ 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.

The state-dependent action cost function is defined as $\mathcal{C}(a,w) = f_{SG}(a,w)$ because the set of soft goals SG is an input of the problem and acts as a constant in the definition of the cost function.

An action sequence π is a plan for a PSP-HSD task Π if it is applicable in $\mathcal{I}$ and $|\pi| = H$.[2] 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.

In the next section we describe several approaches for action selection. These approaches are defined in terms of PSP-HSP planning tasks. One of the most important differences among them is the definition of the state-dependent cost function. After that, we will show how to compile the resulting PSP-HSP planning tasks into PDDL (Planning Domain Definition Language) which allows solving those tasks with current planners.

4. Alternative action selection approaches

In this section, we present different approaches whose action selection mechanisms are based on estimations of the value function [1]. The first one is a summary of previous work on Hindsight Optimization (HO) [5,6]. The second one is a minor contribu-

[2] Note that we assume the existence of a no-op action, whose cost depends on each specific domain. In the domains used in our experiments it is a zero-cost action.

$$Q_i^*(s,a) = \begin{cases} 0 & \text{if } i = H, \\ \sum_{s'} T(s,a,s')[C(s,a,s') + V_{i+1}^*(s')] & \text{otherwise} \end{cases} \tag{9}$$

$$Q_i^*(s,a) = \begin{cases} 0 & \text{if } i = H \\ cost(a) + \sum_{s'=(w',G')} P_a(G' \mid s)[penalty(w',G') + V_{i+1}^*(s')] & \text{otherwise} \end{cases} \tag{10}$$

Fig. 1. Generic (Equation (9)) and instantiated (Equation (10)) optimal action-value function for actions a that are applicable in s at stage i. States $s' = (w', G')$ are possible next states when a is applied to $s = (w, G)$; $w' = \phi(a, w)$, the result of applying a to w.

tion of this paper. It presents an adaptation of HO to domain-independent planning that we name Domain-Independent HO (DIHO). The other approaches are contributions of this paper.

4.1. Hindsight optimization

This section describes the key idea of the approach proposed by Burns *et al.* [5], based also on previous work on Hindsight Optimization (HO) [6,8,18,19]. Hindsight optimization is based on approximating V_1^* by an upper bound (reward maximization) or lower bound (cost minimization) and using this approximation for controlling the rolling horizon when solving an infinite horizon MDP. That bound is obtained by exchanging the order of expectation and minimization in the computation of V_1. The resulting estimate of V_1^* replaces V_1^* in the recursive definition of Q_0^* (see Equations in Fig. 1).

In the OCPP, the only uncertainty comes from the future goals. For a particular sample of the future goals, the minimization inside the approximation of $\hat{V}_1^*$ becomes the deterministic problem of choosing the action sequence with minimum cost given that sample. Then, the expected value is computed as the average of the minimum cost for several samples. Specifically, that expected value is the average of the optimal plan cost for *width* samples $\{F^1, \ldots, F^{width}\}$, $F^j \subseteq \mathcal{G} \setminus G$. These samples are generated over the distribution of possible goal arrivals for the upcoming time steps, where the possible goal arrivals is the difference between the set of all goals and the already known goals, $\mathcal{G} \setminus G$. Each sample represents one fixed possible future in terms of goals. Their implementation using a domain-dependent solver that showed a remarkable performance, outperforming a reactive planning approach in several domains.

4.2. Domain-Independent Hindsight Optimization (DIHO)

To apply the idea of Hindsight Optimization to our OCPP, we exchange expectation and minimization in V_1^* (Equation (6)) to obtain the following estimate:

$$\hat{V}_1^{DIHO}(s) = E\left[\min_{a_1,\ldots,a_H} \sum_{t=1}^{H-1} C\big(s_t, a_t, \phi(a_t, s_t)\big) \Big|_{s_1=s}\right], \tag{11}$$

where, the expectation is computed considering the different futures. From the point of view of domain-independent planning, a planning task has to be solved for every future F^j. Its solution is a plan π of length H that minimizes the cost function for the OCPP; i.e., the sum of action costs plus the total penalty:

$$\pi_j^* = \arg\min_{\pi} \left\{ \sum_{a \in \pi} cost(a) + \sum_{g \in G \cup F^j} totalPenalty_g \right\} \tag{12}$$

where the $totalPenalty_g$ of each goal g depends on the number of steps between its arrival time, t_a^g, and the planned achievement time, t_p^g, as defined in Equation (13).

$$totalPenalty_g = \begin{cases} 0 & \text{if } t_p^g \leqslant t_a^g \vee t_a^g \geqslant H \\ (\min(t_p^g, H) - t_a^g) \times k_g & \\ \text{otherwise} & \end{cases} \tag{13}$$

Therefore, this total penalty is zero either when t_p^g is smaller than t_a^g, in which case the goal has been anticipated, or when t_a^g is higher than H, in which case

the goal does not arrive before the horizon. Otherwise, when $t_p^g \leqslant H$, it is the instantaneous penalty defined for the goal (k_g) multiplied by the number of steps N_g between its arrival and its achievement, $N_g = t_p^g - t_a^g$.

In our model of DIHO, t_a^g is the current time step for both known goals and the corresponding sampled future, which means each sample F^j is a set of future goals that arrive immediately. Another possibility is to sample schedules of future goals, each with its corresponding arrival time. This would imply solving a planning task that allows the dynamic injection of goals during search such that each goal arrives at its sampled t_a^g and the instantaneous penalty for every goal is paid for every step during the planning period from its arrival to its achievement (determined during planning). The optimal plan of such planning task would minimize the expression in Equation (12). However, from a practical perspective this solution presents important drawbacks that led us to explore other alternatives:[3]

1. The simulation of goal arrivals increases the complexity of the planning task. This simulation can be represented by additional actions that mark a goal as "arrived" at its specific arrival step. However, in this case, to plan these goals the planner should select every arrival action at its arrival step among all applicable actions at that step, which is not detected easily by current heuristics.
2. It is very difficult to optimize solutions when the penalties are zero up to a depth of the search tree (the arrival steps), since current existing heuristics for state-space planners are not sufficiently accurate in this case.

Now we express our DIHO approach formally. Let $s_0 = (w_0, G_0)$ be the current state. Then, the selected action at s_0 will be the applicable action $a \in A(s_0)$ that minimizes $\hat{Q}_0^{DIHO}(s_0, a)$, an estimation of the action-value function:

$$a_0 = \arg\min_{a \in A(s_0)} \hat{Q}_0^{DIHO}(s_0, a)$$

$\hat{Q}_0^{DIHO}(s_0, a)$ 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^j. Given F^j, this next state is $s_1^j = (w_1, G_0 \cup F^j)$, where $w_1 = \phi(a, w_0)$, the resulting world state when a is applied to w_0, and the goals are the result of adding F^j 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 following PSP-HSD task:

$$\Pi_{DIHO}^j(s_1^j, H-1) = \\ (\mathcal{F}, \mathcal{A}, w_1, G_0 \cup F^j, \mathcal{C}_{DIHO}^j, H-1),$$

where $\mathcal{F}$ is a set of fluents that defines 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 $G_0 \cup F^j$ is the set of soft goals. The cost function $\mathcal{C}_{DIHO}^j(a, w')$ is computed by Equation (3) (i.e., the cost function for the OCPP), but taking as arguments a and a state s' composed as $s' = (w', G_0 \cup F^j)$.

Formally, for every current OCPP state s_0 and applicable action $a \in A(s_0)$, $\hat{Q}_0^{DIHO}(s_0, a)$ is defined as shown in Equation (14) (in Fig. 2). Thus, the optimal value function in DIHO is estimated as:

$$\hat{V}_0^{DIHO}(s_0) = \min_{a \in A(s_0)} \hat{Q}_0^{DIHO}(s_0, a)$$

At each time step, the number of planning processes executed to solve the action selection problem is $width \times |A(s_0)|$. And the total number of calls to the planner in a time period of length T is $\sum_{s_i} width \times |A(s_i)|, i = 0, \ldots, T-1$, which is independent of the minimization horizon H.

Applying this DIHO scheme where the solver is a domain-independent planner, presents scalability problems. This is particularly true if there are restrictions on response times, since the number of planning tasks to be solved optimally at every time step can be very high. In fact, finding the optimal solution with current planners to only one problem in a reasonable time may not be feasible for medium-size problems. Hence, we will use an approximation that generates suboptimal solutions in the experiments. Even assuming suboptimal solutions, the high number of planning tasks to solve at every time step generates scalability problems. The following approaches are intended to reduce the computation time.

4.3. Goal-distribution-sensitive planning

Given that DIHO requires solving many planning tasks at each decision-making step, we propose Goal-Distribution-Sensitive Planning (GDS) to reduce the

[3] We have made several experimental tests in this direction where we confirmed these drawbacks.

$$\hat{Q}_0^{DIHO}(s_0,a) = \begin{cases} 0 & \text{if } H = 0 \\ cost(a) + \frac{1}{width}\sum_{j=1}^{width}[penalty(w_1, G_0 \cup F^j) + c(\pi^*_{\Pi^j_{DIHO}(s_1^j, H-1)})] & \text{otherwise} \end{cases} \quad (14)$$

$$\hat{Q}_0^{SE}(s_0,a) = \begin{cases} 0 & \text{if } H = 0 \\ cost(a) + penalty_{GDS}(w_1, G_0 \cup F) + c(\pi^*_{\Pi_{GDS}(s_1, H-1)}) & \text{otherwise} \end{cases} \quad (15)$$

Fig. 2. Estimation of the action-value function in DIHO (above) and GDS-SE (below) for $a \in A(s_0)$. $s_1^j = (w_1, G_0 \cup F^j)$ 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.

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 through the cost function.

The idea is to estimate V_0^* by the minimum cost of a single plan, which eliminates the effort of computing several plans and then an expected value:

$$\hat{V}_0^{GDS}(s) = \min_{a_1,\ldots,a_H}\left[\sum_{t=0}^{H-1} C\big(s_t, a_t, a_t(s_t)\big)\,\Big|_{s_0=s}\right] \quad (16)$$

Instead of computing an average of minimal plan costs for several samples, we opted for incorporating the uncertainty into the plan model. The objective is still the same, but now we want to solve a single planning task whose solution is a plan π of length $H - i$ that minimizes the sum of its actions costs plus the total penalty for all future goals:

$$\pi^* = \arg\min_{\pi}\left\{\sum_{a\in\pi} cost(a) + \sum_{g\in G\cup F} totalPenalty_g\right\} \quad (17)$$

where the $totalPenalty_g$ is defined in Equation (13).

Similarly to DIHO, in GDS t_a^g is the current time step for known goals. However, for future goals, instead of doing sampling, we estimate each t_a^g by its expected value. Since each goal arrival distribution $P(g)$ is a Bernoulli process, the random variable t_a^g representing the *number of steps until goal arrival* follows a geometric distribution with mean $E(t_a^g) = \frac{1}{P(g)}$. Ideally, the penalty will be minimal if each goal is achieved before its expected arrival time, $E(t_a^g)$. On average, goals with small arrival times that have not been achieved before their expected arrival time will start to pay their penalties before others with higher expected arrival times.

Then, the MDP can be reduced to a fully deterministic planning task considering each goal arrives at the expected step $E(t_a^g)$. Given the drawbacks related to planning tasks with dynamic injection of goals, explained in Section 4.2, we have explored an alternative technique. The key idea consists of anticipating the total penalty to be paid for every goal in planning time, such that a fraction of this penalty is paid from the initial search node through action costs. The corresponding planning task can be more easily optimized by current planners.

Since the planned achievement time t_p^g of every goal is initially unknown and the total penalty depends on that time, we propose an approach with a parameter that relates t_p^g to the arrival time t_a^g. Thus, we can compute the penalty fraction for a variety of values of that parameter, which allows us to make assumptions that can be either optimistic or pessimistic. There are also other possibilities, such as estimating t_p^g by means of a heuristic, though this is more costly from a computational point of view. This possibility has not been evaluated in this paper and it is left for future work. In the next paragraphs we explain how the fraction of the total penalty is computed for different cases.

***Optimistic case**: The planned achievement times are not higher than the arrival times ($t_p^g \leqslant t_a^g$).* In this case, no penalty will be paid since all goals are anticipated. However, intuitively, goals g with shorter t_a^g and higher penalties k_g should be achieved first. In order to give priority to those goals, we consider an instantaneous penalty in planning time to be paid for each goal at every planning step until its achievement. Here, we define a planning step as the generation of a new search node (corresponding to the next state) by the application of an action to the current search node (state) during the search. This Planning Instantaneous Penalty (*PIP*) is computed by distributing the instanta-

neous penalty among the expected steps until goal arrival. If a goal g has not been achieved, its *PIP* at every planning step will be:

$$PIP_g = \frac{k_g}{E(t_a^g)} = \frac{k_g}{1/P(g)} = k_g \times P(g) \qquad (18)$$

This definition of PIP_g is rather intuitive since it generates a cost distribution for the planning task such that the step penalty for each goal is proportional to its arrival probability. If all goals have the same instantaneous penalty, the cost of not achieving one of them would be higher as its probability increases. It gives priority to most probable goals.

***From the optimistic to the pessimistic case**: The planned achievement times are higher than the arrival times ($t_p^g > t_a^g$).* Here, we consider that $t_p^g \leqslant H$ and $t_p^g = t_a^g + N_g$ with $N_g > 0$; i.e., the planned achievement time is inside the horizon and higher than the arrival time. In that case, the total penalty will be $N_g \times k_g$ and the PIP_g is that value divided by the planned achievement time:

$$PIP_g = \frac{N_g \times k_g}{t_p^g}$$

Replacing t_p^g in the previous equation by $t_a^g + N_g$, t_a^g by its expected value, $E(t_a^g) = \frac{1}{P(g)}$, and manipulating the result, we obtain:

$$PIP_g = \frac{N_g \times P(g)}{1 + N_g \times P(g)} \times k_g \qquad (19)$$

where N_g is a parameter that allows us to modulate from a more optimistic assumption for $N_g = 1$ (where $t_p^g = t_a^g + 1$), to a more pessimistic one for $N_g = H - \frac{1}{P(g)}$ (where $t_p^g = H$).

The definition of the PIP in either case (Equations (18) or (19)) allows us to define a cost function that is aware of the goal arrival distribution since it depends on the arrival probability. Given an OCPP state $s = (w, G)$, the corresponding PSP-HSD problem would be:

$$\Pi_{GDS}(s, H) = (\mathcal{F}, \mathcal{A}, w, G \cup F, \mathcal{C}_{GDS}, H)$$

As aforementioned, this problem is also fully deterministic. $\mathcal{F}$ and $\mathcal{A}$ are defined as in DIHO; the soft goals are equal to the set of all possible goals: the known ones at s, G, and the future ones $F = \mathcal{G} \setminus G$. The cost function $\mathcal{C}_{GDS}$ is:

$$\mathcal{C}_{GDS}(a, w') = cost(a) + penalty_{GDS}(w', G \cup F), \qquad (20)$$

where the penalty is computed as the sum of individual goal penalties (as in Equation (4)). And the individual goal penalty is defined as:

$$penalty_{GDS}(w', g) = \begin{cases} PIP_g & \text{if } g \in F \wedge g \notin w' \\ k_g & \text{if } g \in G \wedge g \notin w' \\ 0 & \text{otherwise} \end{cases} \qquad (21)$$

Therefore, in GDS, $V_0^*(s)$ is approximated directly as the cost of the optimal plan:

$$\hat{V}_0^{GDS}(s) = c(\pi^*_{\Pi_{GDS}(s,H)})$$

This approach allows for three different action selection schemes: Long-term Execution (LE), Step Execution (SE) and Reactive (R).

In **Long-term Execution (LE)**, we have a plan that provides the current state value function, $\hat{V}_0^{GDS}(s_0)$. Therefore, we can choose to execute the actions in that plan as long as possible, instead of computing a different plan at each time step. 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. This scheme greatly reduces the number of planning processes, and therefore the time of the action selection process.

In **Step Execution (SE)**, the idea is to plan for every time step and select the first plan action. Our implementation of SE slightly differs from this. We will often generate subpotimal plans. Hence, from a practical point of view, we decided to plan for the next step, which means to use the approximation $\hat{V}_1^{GDS}(s_1)$. Then, we make an exhaustive minimization step at the current one, similarly to DIHO. Thus, 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 $w_1 = \phi(a, w_0)$. The approximation of the opti-

mal action-value function is defined by Equation (15) (in Fig. 2), and the selected action a_0 will be:

$$a_0 = \arg\min_{a \in A(s_0)} \hat{Q}_0^{SE}(s_0, a)$$

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 in DIHO).

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.

In the GDS approaches presented in this section we propose to solve deterministic problems that assume each goal arrives at its average arrival time. We also make additional assumptions about the planned achievement time of each goal. Therefore, there are no theoretical guarantees on selecting the same actions of the ones on the optimal policy, given that we are solving approximations of the real problem. However, these approaches are justified from a practical point of view, since computing the optimal policy in a reasonable time is computationally unfeasible.

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 [9]. 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 [13]. This metric function is usually a linear combination of one or more numeric state variables. Optimal plans are those that achieve all goals with the minimum value for the metric function.

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 and Geffner approach [14], considering also the ideas of Benton *et al.* [2] 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*).

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:[4]

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

where the precondition (*not* (*sg-p args*)) ensures there is not a soft goal in Π for the fact the action achieves;

[4] 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.

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

$\mathcal{F}^+$ 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

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:

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

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. Additionally, 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:

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

where:

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

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

$$pre(a^{\sim}_{\neg p}) = pre(a) \cup pre^H \cup \{(not\ (collected\text{-}pargs))\}$$
$$add(a^{\sim}_{\neg p}) = add(a) \cup add^C \cup add^H \cup add^{\sim}$$
$$del(a^{\sim}_{\neg p}) = del(a)$$

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:

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

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

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

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:[5]

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

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, that generate the hard goal (collected-p args) for non-achieved soft goals once the horizon has been reached. They are removed from the plans during post-processing. 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:

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

The main difference between our compilation and the compilation of Keyder and Geffner [14] is that ours allows the planner to pay stepwise penalties for not achieving goals. By contrast, in the Keyder and 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.* [2] 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 [10]. However, the compilation would need an exponential number of actions in our case, which would make this approach intractable.

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

6. Experiments and results

In this section we present the setup and results of the empirical evaluation of the different action selection approaches described in previous sections. We have not included comparisons with any standard MDP-based technique, since it is easy to see they will not scale to the size of the problems we use for testing, as was also previously shown [5]. In addition, using optimal planners is out of the question on domain-independent planning of this kind of tasks, since they do not scale and there is no optimal planner that can handle state-dependent costs and numeric preconditions. So, we are restricted to using suboptimal planners. This fact limits a direct comparison to prior approaches which use optimal search algorithms. Additionally, even in the case of using suboptimal planners, very few domain-independent planners handle adequately state-dependent cost functions.

For these experiments, we chose one of the few base solvers that fulfills our constraints, LPG-td [11]. In order to reduce the impact of the selection of solver, we have used it for all techniques. Therefore, all conclusions on the following experiments will be based on the use of this solver (stochastic suboptimal planner) as opposed to using domain-dependent optimal planning as is the case of Burns *et al.* approach.

We have used three well known IPC planning benchmarks: Satellite, Rovers and TPP. We also used a fourth domain that we modelled in PDDL to simulate the surveillance UAV scenario described by Burns *et al.* [5]. In this case, we did not use rewards and penalties only depend on the goals. In this domain the vehicle can switch on/off its camera, send data, fly in the four directions and track a zone (i.e., the action of recording while it is moving). The action of tracking has a cost of two. The remaining actions have unitary costs. All actions from the IPC benchmarks have unitary costs as well. 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. In addition, we defined a no-op action with zero cost in the domain file.

For the experiments, we define a finite *execution horizon* that represents the period of time the system will be running and receiving goals. The minimization

horizon H will be at most as large as the execution horizon, but it can be smaller.

We have carried out several experiments to analyze the different aspects of the proposed approaches. Each of them is then presented in the following subsections. All 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 7.5 Gb of RAM.

6.1. Random goal arrival distribution

The idea of this experiment is to analyze the anticipatory behavior of GDS planning having a constant penalty per goal with a random distribution of goal arrivals. Since goals will have equal penalty, our hypothesis is that GDS approaches will prefer to anticipate the most probable future goals while also achieving goals that have already arrived. This behavior should lead on average to a better cost compared to a reactive approach that achieves goals after they arrive and pays penalties until achievement, without considering the probability of arrival each goal has.

For each domain we have generated a base problem. Then, we have solved these problems to set the execution horizon as 1.25 times the length of the solution (as in Burns *et al.* [5]). As a reminder, Burns *et al.* compute an optimal solution, while we compute here a suboptimal solution. For each problem, we generated three random distributions of goal arrivals. Each of those distributions was generated in the following way: we sampled the uniform distribution to assign to each goal the probability of appearing within the horizon (P_{wH}). The probability that a goal arrives within H is the same as the probability of at least one goal arrival (i.e. one success) in H Bernoulli trials. The random variable X that represents the number of successes in the first H trials follows a Binomial distribution $B(H, p)$, where p is the arrival probability at each step. Then, $P_{\text{wH}} = P(X \geqslant 1) = 1 - P(X = 0)$. Since $X \sim B(H, p)$, $P(X = 0) = (1 - p)^H$. Thus, $P_{\text{wH}} = 1 - (1 - p)^H$. In our case P_{wH} is generated automatically, and therefore it is known for every goal. Then, we compute each goal step arrival probability just by isolating the step arrival probability from the previous equation:

$$P(g) = 1 - \left(1 - P_{\text{wH}}(g)\right)^{1/H} \tag{22}$$

From a goal arrival distribution, one can simulate the future creating a *schedule of arrivals*. which 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. A schedule is generated doing Bernoulli trials with $P(g)$ at each step until succeeding or reaching the horizon. Thus, we sampled a set of 10 future goal schedules from each goal arrival distribution. They are provided as input to all planning-execution cycles, which are 120 (i.e., 4 domains $\times$ 3 goal arrival distributions $\times$ 10 schedules of arrivals).

We set as penalty $k_g = 100$ for all goals. Therefore, the total cost will mainly reflect the ability of different approaches for optimizing the penalty. We set the maximum planning time per step to 60 seconds in the first experiment. If an approach does not call the planner within an execution step, this time is not accumulated for further planning episodes. We have compared the different approaches using the following configurations:

- **Reactive (R)**: it runs LPG-td with the configuration for *anytime planning*, which lets the planner spend the available time until the time bound, and then return the best solution found up to this point. H is equal to the execution horizon. We consider this configuration as the baseline for comparison.
- **GDS-LE-O**: The long-term execution of GDS planning using the optimistic case of PIP (Equation (18)). It also runs the anytime behavior of LPG-td, and H was set to coincide with the execution horizon.
- **GDS-LE-1**: The same configuration as GDS-LE-O, but using $N_g = 1$ in Equation (19) for all goals.
- **GDS-LE-H**: The pessimistic case of PIP. It uses the same configuration as GDS-LE-O, but using $N_g = H - \frac{1}{P(g)}$ in Equation (19).
- **GDS-SE**: The short-term execution of GDS planning. In this case, we only consider the optimistic case of PIP. We set $H = 8$ (i.e., a short-term horizon for deliberation). The step-timeout of 60 seconds is divided equally between the planner calls (LPG-td in anytime mode) for each applicable action.
- **DIHO-s** (Domain-Independent Hindsight Optimization): we set $H = 8$ and $width = 10$ 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).

As a reminder, GDS planning approaches pay fractions of the goal penalties at planning time as a mechanism to anticipate to goals. The practical differences between long execution configurations is how they establish the distribution of PIPs among goals

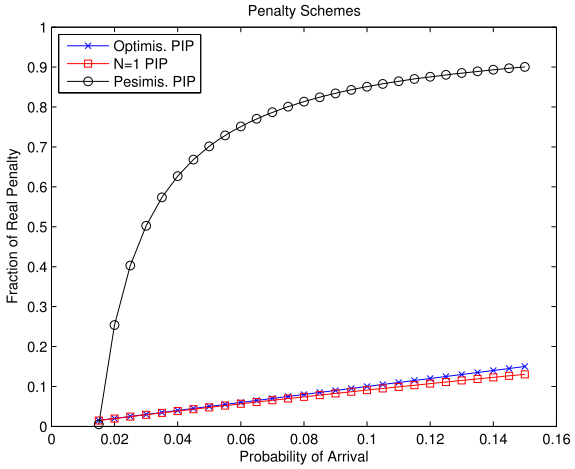

Fig. 3. Distribution of penalties for each GDS-LE planning alternative in the Satellite base problem. x-axis represents $P(g)$ and y-axis is the fraction of the penalty paid at every step until goal arrival at planning time.

depending on their step arrival probability. Figure 3 shows functions describing how that probability affects the penalty fraction for each configuration in the Satellite base problem ($H = 67$). In the optimistic case and the case for $N_g = 1$ in Equation (18), the penalty fraction is rather small compared to the real penalty paid for non-achieved goals that have already arrived. However, the pessimistic case assigns much higher penalties fractions as goals become more likely to appear in the horizon.

Table 2 shows the average total cost and the standard deviation of the 10 schedules for each distribution of goal arrivals. GDS-LE-1 achieves better total-cost averages than Reactive for all distributions, and GDS-LE-O was better in all cases minus one. GDS-LE-H also achieved better averages in 11 cases. The improvements over Reactive reflect the ability of GDS planning approaches to reduce the penalty by means of anticipating future goals.

This can be confirmed in Fig. 4 which plots the relation between the goal arrival and goal achievement time in the 10 executions of the first distribution in Satellite. All points below the diagonal represent goals achieved before arrival. Even though GDS-LE-H also outperformed Reactive in 11 cases, these results are worse than the ones exhibited by GDS-LE-1 and GDS-LE-O. We have observed that GDS-LE-H is often distracted by future goals when in fact there are real arrived goals for which a penalty is being paid. The main reason for this behavior is the anticipated high penalty GDS-LE-H pays, as we showed in Fig. 3. Since the anticipated penalty for more probable goals is close to the real penalty of arrived goals, GDS-LE-H does not have a strong preference for goals that have already arrived.

In addition we have conducted an independent sample Welch's t-test to verify the statistical significance of the mean difference of each configuration with respect to Reactive. In Table 2, the $+$ and $-$ symbols indicate a significant improvement or degradation respectively, considering a 95% confidence for the critical value of the t-distribution. The $\uparrow$ and $\downarrow$ are used with the same meaning for the cases in which the test has failed but the p-value is less than or equal to 0.10. Both GDS-LE-O and GDS-LE-1 approaches appear

Table 2

Average total cost results (in thousands) for the evaluation of three different random goal distributions per domain. Following a Welch t-test with 95% confidence, the mean difference with Reactive is significantly ($+$) better ($\uparrow$ for p-values falling in $[0.05, 0.10]$) or significantly ($-$) worse ($\downarrow$ for p-values falling in $[0.05, 0.10]$)

	goal-dist	GDS-LE-O		GDS-LE-1		GDS-LE-H		GDS-SE		DIHO-s		REACTIVE
rovers	dist-1	3.3 ± 2.2		3.1 ± 1.9		3.4 ± 1.7		5.5 ± 3.2		25.1 ± 7.40	$-$	4.2 ± 1.4
	dist-2	2.1 ± 1.1	$\uparrow$	1.9 ± 1.0	$+$	2.6 ± 1.5		4.5 ± 2.0	$-$	18.6 ± 5.80	$-$	3.0 ± 1.0
	dist-3	2.0 ± 1.1	$+$	2.1 ± 0.9	$+$	2.2 ± 0.9	$+$	3.2 ± 2.0		21.4 ± 4.10	$-$	3.6 ± 0.9
satellite	dist-1	6.6 ± 2.4	$\uparrow$	6.7 ± 2.4	$\uparrow$	9.6 ± 1.7		10.1 ± 2.9		59.0 ± 9.80	$-$	8.7 ± 2.6
	dist-2	6.5 ± 5.7		6.7 ± 5.2		8.5 ± 5.4		15.1 ± 6.2	$-$	61.5 ± 17.1	$-$	8.8 ± 3.7
	dist-3	3.2 ± 2.9	$+$	3.4 ± 2.8	$\uparrow$	3.4 ± 2.4	$+$	6.6 ± 4.4		40.8 ± 10.2	$-$	5.5 ± 1.6
tpp	dist-1	5.9 ± 3.7		4.9 ± 2.8		5.2 ± 2.9		3.6 ± 2.1	$\uparrow$	13.9 ± 5.20	$-$	5.3 ± 1.9
	dist-2	3.7 ± 2.4		3.4 ± 2.0		4.2 ± 2.7		3.0 ± 1.7		13.4 ± 6.00	$-$	4.3 ± 1.8
	dist-3	4.2 ± 3.0		4.9 ± 2.8		5.0 ± 2.5		3.5 ± 2.3		15.9 ± 6.10	$-$	5.1 ± 1.8
uav	dist-1	8.5 ± 4.8		8.1 ± 3.2		8.4 ± 3.8		7.0 ± 3.7	$+$	52.0 ± 16.6	$-$	10.2 ± 2.9
	dist-2	6.2 ± 2.8		5.8 ± 2.4	$\uparrow$	6.8 ± 2.7		6.0 ± 4.2		42.3 ± 11.8	$-$	7.9 ± 2.5
	dist-3	4.3 ± 1.9	$+$	4.7 ± 1.5	$+$	5.1 ± 2.1	$+$	4.1 ± 1.8	$+$	42.2 ± 8.30	$-$	7.4 ± 1.5

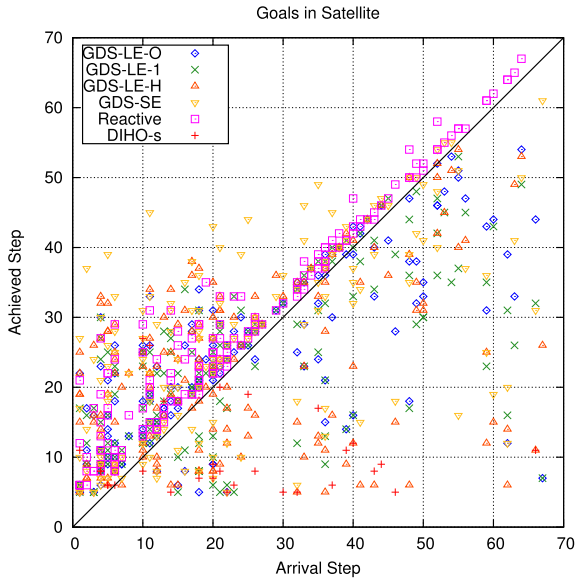

Fig. 4. Arrival vs. achievement times for goals in the 10 executions of the base problem and first goal distribution of Satellite.

with better total cost averages, and they are not considered significant around half of the cases. In the cases that failed the t-test, it was due to the high variance of the total cost. If several goals arrive in a schedule far from their expected arrival, the anticipation will pay off less than if the arrivals are around the expected value. This variability reflects that the final outcome is sensitive with respect to the actual schedule. GDS-SE varies its performance depending on the domain. The main difference with respect to the other GDS planning approaches is that there is a planner call for each execution of an action. Thus, the planner has less time to provide a good solution. This suggests that the usefulness of GDS planning also depends on the practical ability of the planner for optimizing PSP-HSD tasks.

Regarding DIHO-s, the step time bound greatly affects its performance, with very bad results with our experimental setting. It is able to anticipate some goals, as we can see in Fig. 4. 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 foresee any alternative to the approximate computation of the best action to be applied at each time step. Any implementation of DIHO has to deal with the issue of a large number of planner calls imposed by the Monte Carlo sampling. For instance, DIHO-s needed to solve 14,537 planning tasks on average in the Satellite base problem. 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. However, the plan quality degradation with regard to Reactive shown by the statistical test only holds in the context of our experimental setting in which the time bound per step limits the planner performance. This fact does not contradict Burns *et al.* findings given that in their case the optimal search algorithm is not restricted by time.

Table 3 shows the average and standard deviation of the planning time (in minutes) for the evaluated distributions and configurations. Given that the time bound is 60 seconds, another way to see these results is that the numbers in the table for long-term executions reflect the average number of steps in which a plan is needed. As expected, long-term configurations of GDS-planning spent less time than Reactive because they do not have to replan on the arrival of goals that have been already achieved. GDS-LE-H spends slightly less time, because the higher penalties allow the planner to anticipate more goals, even if they are less probable or its achievement implies more cost. The time for GDS-SE is considerably more, because it replans at each step. The time used by DIHO-s is shown as a reference, since the planner did not optimize after the first solution. In the next experiment we give more time per step, so we can show a more fair comparison between DIHO and the other approaches.

6.2. *Scalability of GDS-planning*

We wanted to analyze how GDS planning scales in larger problems. For this reason we have performed an additional experiment in which we do comparisons with problems of incremental size. For each domain, we have used the base problem of the previous section as the starting problem. Then, we generated two more problems increasing the number of possible goals as well as the number of objects needed to have these extra goals. For each problem we have generated a distribution of goal arrivals, and then 10 schedules of goal arrivals, following the same procedure as before. Given that the PSP-HSD tasks are larger we set the time bound per step to 300 seconds. In this case, we allowed Domain-Independent Hindsight Optimization to run the planner with the anytime planning behavior, and therefore its results can be compared to other approaches fairly. The 300 seconds are split between each planner call ($width = 10\times$ the number of applicable actions). For cases in which this time is still too low we set a minimum of one second for each planner call.

Table 3

Average total planning time results (in minutes) for the evaluation of three different random goal distributions per domain. DIHO-s time is reported as a reference. It is not comparable given that the planner returns the first solution without consuming the remaining time at each step

	goal-dist	GDS-LE-O	GDS-LE-1	GDS-LE-H	GDS-SE	REACTIVE	DIHO-s
rovers	dist-1	5.5 ± 1.2	5.7 ± 1.6	4.8 ± 1.5	69.7 ± 2.0	6.9 ± 1.3	14.0 ± 1.6
	dist-2	4.6 ± 1.4	4.1 ± 1.0	4.2 ± 0.9	70.3 ± 0.9	5.5 ± 1.4	16.1 ± 2.4
	dist-3	4.6 ± 1.1	4.5 ± 1.2	4.2 ± 1.1	68.8 ± 1.7	6.7 ± 0.8	12.8 ± 2.2
satellite	dist-1	10.9 ± 1.6	10.3 ± 1.4	9.3 ± 1.3	67.7 ± 0.0	13.8 ± 1.7	20.2 ± 0.5
	dist-2	11.2 ± 4.3	10.6 ± 3.7	10.2 ± 3.6	67.6 ± 0.0	16.2 ± 1.8	20.0 ± 0.3
	dist-3	6.9 ± 2.1	7.7 ± 2.2	7.5 ± 2.2	67.7 ± 0.0	12.1 ± 1.9	20.6 ± 0.8
tpp	dist-1	5.6 ± 2.2	5.8 ± 2.0	5.6 ± 1.8	51.3 ± 1.0	5.6 ± 1.8	8.5 ± 0.4
	dist-2	5.0 ± 1.7	5.0 ± 1.7	4.8 ± 1.7	52.2 ± 1.7	5.0 ± 1.8	9.0 ± 1.7
	dist-3	5.7 ± 1.5	5.8 ± 1.5	5.3 ± 1.6	51.8 ± 0.8	5.7 ± 1.0	8.1 ± 1.2
uav	dist-1	10.8 ± 2.4	11.2 ± 2.6	10.0 ± 2.2	68.6 ± 2.7	12.5 ± 1.6	8.0 ± 0.9
	dist-2	9.5 ± 2.2	9.0 ± 2.4	8.9 ± 2.8	69.6 ± 3.2	11.0 ± 2.0	8.0 ± 1.2
	dist-3	9.3 ± 2.4	9.6 ± 2.1	8.1 ± 1.7	67.4 ± 3.5	11.7 ± 2.1	8.3 ± 1.1

Table 4

Total cost results (in thousands) for the evaluation of three problems per domain of increasing size. Following a Welch t-test with 95% confidence, the mean difference with Reactive is significantly ($+$) better ($\uparrow$ for p-values falling in $[0.05, 0.10]$) or significantly ($-$) worse ($\downarrow$ for p-values falling in $[0.05, 0.10]$)

	prob.	GDS-LE-O		GDS-LE-1		GDS-LE-H		GDS-SE		DIHO		REACTIVE
rovers	prob1	3.90 ± 2.00		3.70 ± 1.90	$\uparrow$	4.30 ± 2.40		3.60 ± 1.90	$\uparrow$	20.5 ± 7.50	$-$	5.30 ± 2.10
	prob2	15.9 ± 9.60		15.0 ± 7.80		18.9 ± 10.5		30.4 ± 11.5	$-$	82.4 ± 12.0	$-$	14.2 ± 4.00
	prob3	14.9 ± 6.60		13.5 ± 9.80		16.3 ± 6.90		35.2 ± 14.3	$-$	111.9 ± 15.7	$-$	12.0 ± 3.70
satellite	prob1	3.70 ± 2.10	$+$	3.20 ± 1.50	$+$	5.10 ± 1.60		4.20 ± 2.40		38.8 ± 10.5	$-$	5.80 ± 2.00
	prob2	16.9 ± 8.80		16.2 ± 6.10		21.0 ± 7.70		28.6 ± 8.70	$-$	147.6 ± 28.5	$-$	17.0 ± 5.80
	prob3	46.9 ± 16.9		47.2 ± 13.4	$\downarrow$	45.0 ± 15.6		94.6 ± 27.0	$-$	281.9 ± 31.3	$-$	36.5 ± 10.8
tpp	prob1	6.20 ± 3.80		7.20 ± 4.00		7.00 ± 3.90		6.10 ± 3.80		17.8 ± 7.10	$-$	7.50 ± 3.30
	prob2	4.10 ± 2.20	$+$	4.80 ± 3.10		5.70 ± 2.70		3.30 ± 1.80	$+$	21.5 ± 7.10	$-$	6.10 ± 1.70
	prob3	20.0 ± 11.9		15.6 ± 7.50		21.5 ± 10.5	$\downarrow$	12.7 ± 8.80		67.4 ± 18.0	$-$	14.1 ± 4.90
uav	prob1	5.60 ± 4.00	$+$	5.90 ± 4.50	$\uparrow$	5.80 ± 3.30	$+$	4.40 ± 3.20	$+$	47.4 ± 10.6	$-$	9.30 ± 2.90
	prob2	13.1 ± 5.10		11.9 ± 4.50	$+$	16.0 ± 3.70		10.9 ± 5.20	$+$	133.5 ± 15.1	$-$	16.1 ± 3.00
	prob3	65.8 ± 25.1	$-$	62.3 ± 26.2	$-$	61.9 ± 25.9	$-$	59.9 ± 24.7	$-$	365.8 ± 61.6	$-$	31.7 ± 10.4

This configuration is named DIHO, to differentiate it from the previous configuration, DIHO-s.

Table 4 presents the average total costs and the standard deviation for the execution of the 10 schedules. We conducted again the Welch's t-test for comparing the cost averages against Reactive. The results for the first problem are comparable to the results in Table 2. The only difference is the time bound for planning in a single step. Contrary to what it was observed in the base problems, the performance on larger problems shows that in many cases, the total cost average for long-term approaches is worse than the Reactive total cost average. Indeed, this degradation is statistically significant in some cases, especially in the UAV domain. The reason for the general degradation is that the number of possible future goals increases with the problem size. Therefore, long-term approaches have to plan for larger goal sets, given that they plan for all problem goals in advance. In execution traces of larger problems we found that plans of the long-term approaches were of bad quality, mainly because the planner did not have enough time to improve its first solutions. However, Reactive only plans for known goals, so its problems remain relatively small and they are easier to optimize for the planner. Therefore, the strength of GDS planning will be affected by the avail-

Table 5
Total cost results (in thousands) when using different penalty schemes. The mean difference with Reactive is marked as significantly (+) better or (−) worse

	Penalty	GDS-LE-O		GDS-LE-1		GDS-LE-H		GDS-SE		REACTIVE
rovers	equal	10.1 ± 5.9		9.2 ± 3.8		11.1 ± 3.3		12.9 ± 4.5	−	9.0 ± 2.4
	power-law	22.7 ± 17.1	+	18.3 ± 10.3	+	23.6 ± 12.2	+	32.0 ± 19.9		40.7 ± 16.5
satellite	equal	19.7 ± 6.3		20.6 ± 6.4		21.0 ± 5.6		27.7 ± 8.7		22.0 ± 7.2
	power-law	11.8 ± 4.7	+	12.1 ± 5.0	+	14.3 ± 4.2	+	26.3 ± 14.0		36.3 ± 12.3
tpp	equal	11.7 ± 2.9		12.0 ± 4.0		12.1 ± 3.9		10.5 ± 4.0		11.4 ± 2.9
	power-law	61.8 ± 43.6		52.1 ± 35.1	+	49.4 ± 31.9	+	37.8 ± 28.7	+	83.8 ± 10.2
uav	equal	13.2 ± 5.2		13.3 ± 6.1		13.8 ± 4.2		12.1 ± 5.4		15.4 ± 4.8
	power-law	23.8 ± 22.3	+	27.7 ± 22.0	+	27.6 ± 27.4	+	24.2 ± 23.7	+	56.3 ± 30.6

able planning time at each step and the ability of the planner to optimize large PSP-HSD tasks. The longer time per step, the longer the time the planner would have to optimize its solutions. A potential solution to overcome this problem would be to consider only a subset of future goals, as the ones whose probability is above a given threshold.

As it can be observed also in Table 4, the behaviour of DIHO is considerably worse than the other approaches in our setting. Therefore, it will not be considered in the next experiments.

6.3. Goal penalty distributions

In this experiment we wanted to analyze the behavior of anticipatory search when goals have the same arrival probability, but possibly different distribution of penalties. The hypothesis of this experiment is that GDS approaches will prefer to solve goals with higher penalties. Since GDS configurations will sometimes anticipate these goals, the total cost should be reduced on average. In this setting, we have used the base problems of each domain described in Section 6.1. Rather than having a distribution of goal arrivals, all goals have the probability of appearing in a schedule of 0.8. Goal arrival probability $P(g)$ was set accordingly following Equation (22), 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 base case for this experiment assigns an equal penalty $k_g = 100$ to all goals. For the comparison, we have created a different distribution of penalties. 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 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. The time bound per step was set again to 60 seconds.

Table 5 shows the average total cost and the standard deviation of the execution of 10 schedules. The results of the Welch's t-test for comparing the cost average to the Reactive case are also included. With equal arrival probabilities and penalties, long-term approaches of GDS planning show slightly better average than Reactive in three domains, but these differences are not statistically significant. However, using the power-law distribution of penalties, long-term configurations of GDS planning passed the test nine times out of 10. It is clear that when there are some few important goals, it makes sense to anticipate them for not paying high penalties. GDS planning approaches are able to recognize this situation, performing much better than Reactive in all domains.

In order to analyze the power-law distribution of penalties, we have computed Δt as the time step difference between the time at which each goal has arrived and the time at which it has been achieved. Figure 5 shows the average of Δt in the Satellite problem, sorting the goals by 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, the goal has been achieved before its arrival. All configurations show a trend towards worse Δt when the goals become less important. However, GDS planning approaches start from higher points and are able to obtain non negative Δt for the six more important goals.

Having a positive Δt means that the configuration is not paying penalties for a given goal. Nevertheless, having higher positive values of Δt does not imply better cost because the setup does not give any reward for anticipating earlier. On the other hand, having large

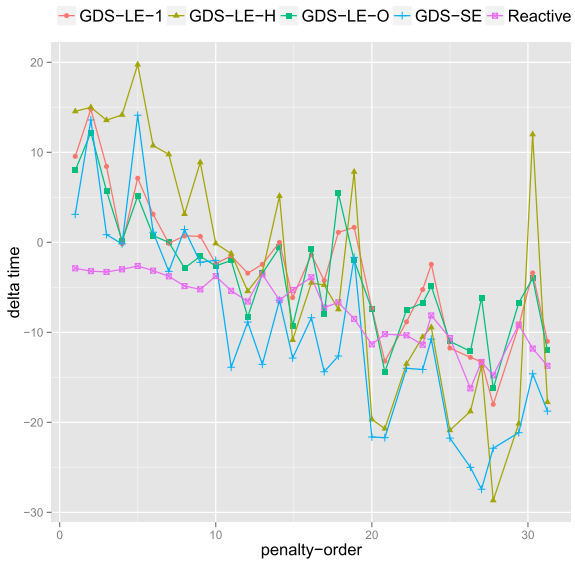

Fig. 5. Average time difference of the arrival and achievement of goals for each evaluated configuration in the Satellite problem.

negative Δt implies paying more penalties, therefore it is better to have these values closer to 0. We can verify this observation as GDS-LE-H has worse average cost than the other long term configurations. GDS-LE-H overreacts in anticipating high penalty goals. This causes the planner to achieve high penalty goals before they arrive (left portion of Fig. 5), but this does not benefit performance since the planner earns no reward for early achievement. As a consequence of overreacting to high-penalty goals, GDS-LE-H ignores lower-penalty goals (right portion of Fig. 5) and achieves worse average performance.

6.4. Goal arrival spacing

Another factor that affects the performance of anticipatory search is the spacing of goal arrivals. The rate of goal arrivals increases with higher probabilities, so the available time to anticipate might vary depending on this rate. We have conducted an additional experiment to analyze this effect. Our hypothesis is that a higher rate will dilute the cost reduction obtained by GDS approaches because the algorithms could only focus on already arrived goals, given that they continuously appear at a high rate.

To make a fair comparison, we used the same random distribution of goal arrivals in a schedule of arrivals, and created the difference in spacing by changing the length of the execution horizon, which will coincide with the length of the schedule as in other experiments. With a smaller horizon and the same probability of occurrence in a schedule, the probability of arrival at each step is necessarily higher. Thus, the arrival probabilities are proportionally increased or decreased by the selection of the execution horizon. The baseline case is 1.25 times the solution of the base problem, as in previous experiments. Then, we have evaluated the range from 0.75 to 1.75 in steps of 0.25. We have set $k_g = 100$ for all goals and a time bound of 60 seconds for planning in a step.

Table 6 shows the total cost average and the standard deviation of the execution of 10 schedules. Results for the Welch's t-test have the same meaning as before. As expected, we can see that in general each configuration tends to decrease its own cost as the rate of goal arrival decreases (i.e., larger horizon, looser schedules). In addition, all GDS planning configurations show an statistically significant improvement over Reactive in looser schedules. For tighter schedules there is no degradation in the performance of GDS planning approaches. In those cases, given a higher goal arrival rate, they have to replan more, and therefore their behavior becomes similar to Reactive. Note that when a goal arrives, its instant penalty will be higher than the penalty of any goal that has not arrived, and the resulting plan will prefer to achieve goals that arrived first.

7. Summary 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. We have also 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. The benefits are specially visible in settings where there are few important goals. In these cases, anticipating relevant goals significantly reduces the total penalty. Nevertheless, the benefits of anticipating goals depend on the spacing of goal arrivals. As we showed in the experiments, GDS-planning takes advantage of anticipation when there is time to do so. Also, the experiments show that if there is no time to anticipate, GDS-planning focuses on already arrived goals, which is the right behavior.

The performance of GDS-planning approaches also depends on the ability of the base planner to produce

Table 6
Results for different goal arrival spacing. The mean difference with Reactive is marked as significantly (+) better or (−) worse

	Rate	GDS-LE-O		GDS-LE-1		GDS-LE-H		GDS-SE		REACTIVE
rovers	rate-0.75	5.9 ± 2.9		7.4 ± 3.9		7.1 ± 4.1		7.8 ± 4.9		6.2 ± 2.3
	rate-1.0	4.6 ± 2.8		4.0 ± 2.1		5.1 ± 2.7		5.7 ± 3.9		4.7 ± 2.4
	rate-1.25	2.8 ± 1.2	+	2.5 ± 1.4	+	3.5 ± 2.1		5.5 ± 2.7		4.3 ± 0.8
	rate-1.5	2.3 ± 1.8	+	1.8 ± 1.1	+	2.6 ± 1.4	+	4.5 ± 3.5		4.5 ± 1.6
	rate-1.75	1.4 ± 0.7	+	1.5 ± 0.6	+	2.3 ± 1.3	+	2.9 ± 1.1	+	4.1 ± 0.9
satellite	rate-0.75	5.8 ± 2.4		5.1 ± 1.9	+	6.6 ± 2.5		11.1 ± 6.5		7.4 ± 2.8
	rate-1.0	4.0 ± 1.9	+	3.8 ± 1.8	+	5.4 ± 2.4		7.7 ± 3.2		5.7 ± 1.5
	rate-1.25	2.6 ± 1.3	+	2.5 ± 1.2	+	4.1 ± 1.8		5.0 ± 2.9		4.7 ± 1.7
	rate-1.5	2.1 ± 1.2	+	2.4 ± 1.4	+	2.8 ± 1.6	+	5.4 ± 4.3		4.5 ± 1.2
	rate-1.75	2.4 ± 1.4	+	2.2 ± 1.3	+	3.4 ± 1.7		6.1 ± 2.5		4.8 ± 1.4
tpp	rate-0.75	7.7 ± 3.0		7.0 ± 3.3		8.2 ± 2.5		5.1 ± 2.3		6.3 ± 1.6
	rate-1.0	8.1 ± 5.1		7.1 ± 4.3		6.9 ± 4.1		5.5 ± 4.1		6.8 ± 3.0
	rate-1.25	4.9 ± 1.3		5.6 ± 1.9		5.8 ± 1.7		4.4 ± 1.8		5.8 ± 2.8
	rate-1.5	8.2 ± 4.6		7.6 ± 4.4		8.9 ± 5.0		7.5 ± 4.1		8.4 ± 3.6
	rate-1.75	3.3 ± 2.3	+	3.6 ± 2.4	+	4.5 ± 3.1		3.5 ± 2.2	+	6.1 ± 2.3
uav	rate-0.75	13.7 ± 6.7		12.4 ± 6.0		13.2 ± 4.3		11.0 ± 4.6		12.4 ± 2.9
	rate-1.0	14.6 ± 3.8		12.7 ± 5.0		13.6 ± 3.9		12.2 ± 3.4		12.9 ± 3.0
	rate-1.25	8.1 ± 3.0		7.4 ± 2.6	+	9.1 ± 3.1		9.3 ± 2.8		9.9 ± 2.0
	rate-1.5	5.4 ± 2.8	+	5.2 ± 2.6	+	6.1 ± 2.5	+	5.7 ± 3.4	+	8.7 ± 2.3
	rate-1.75	4.0 ± 3.2	+	3.9 ± 3.4	+	5.1 ± 2.8	+	3.7 ± 1.9	+	8.2 ± 1.7

good quality plans in the available time. But they do not present the DIHO drawback of solving a large number of tasks per time-step. In contrast, depending on the number of possible future goals, GDS-planning approaches can generate problems with many goals, which makes these problems difficult to solve. Then, GDS-planning is adequate when the number of goals is limited. Otherwise, some strategy to reduce the problems should be designed. For instance, a pre-selection could be done so that GDS-planning focuses on the most probable goals.

Our GDS planning schemes are not linked to any particular planner, so they can benefit from future improvements on dealing with 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 and the anonymous reviewers for their useful comments. This work has been partially supported by MICINN project TIN2014-55637-C2-1-R.

References

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

[2] 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)*, 2012, pp. 2–10.

[3] J. Benton, M. Do and S. Kambhampati, Anytime heuristic search for partial satisfaction planning, *Artificial Intelligence* **173**(5–6) (2009), 562–592. doi:10.1016/j.artint.2008.11.010.

[4] M.V.D. Briel, R. Sanchez, M.B. Do and S. Kambhampati, Effective approaches for partial satisfaction (over-subscription) planning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI Press, 2004, pp. 562–569.

[5] E. Burns, J. Benton, W. Ruml, S.W. Yoon and M.B. Do, Anticipatory on-line planning, in: *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, L. McCluskey, B. Williams, J.R. Silva and B. Bonet, eds, 2012, pp. 333–337.

[6] H.S. Chang, J. Hu, M.C. Fu and S.I. Marcus, *Simulation-Based Algorithms for Markov Decision Processes*, 2nd edn, Springer, 2013.

[7] H.S. Chang and S.I. Marcus, Approximate receding horizon approach for Markov decision processes: Average reward case, *Journal of Mathematical Analysis and Applications* **286** (2003), 636–651. doi:10.1016/S0022-247X(03)00506-7.

[8] E.K.P. Chong, R.L. Givan and H.S. Chang, A framework for simulation-based network control via hindsight optimization,

in: *IEEE Conference on Decision and Control*, 2000, pp. 1433–1438.

[9] M. Fox and D. Long, pddl2.1: An extension to pddl for expressing temporal planning domains, *Journal of Artificial Intelligence Research (JAIR)* (2003), 61–124.

[10] F. Geisser, T. Keller and R. Mattmüller, Delete relaxations for planning with state-dependent action costs, in: *Proc. of IJCAI*, 2015, pp. 1573–1579.

[11] A. Gerevini, A. Saetti and I. Serina, An approach to temporal planning and scheduling in domains with predictable exogenous events, *Journal of Artificial Intelligence Research (JAIR)* **25** (2006), 187–231.

[12] O. Hernández-Lerma and J. Lasserre, Error bounds for rolling horizon policies in discrete-time Markov control processes, *IEEE Transactions on Automatic Control* **35** (1990), 1118–1124. doi:10.1109/9.58554.

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

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

[15] S. Kiesel, E. Burns, W. Ruml, J. Benton and F. Kreimendahl, Open world planning for robots via hindsight optimization, in: *Working Notes of the ICAPS-13 Workshop on Planning and Robotics (PlanRob-13)*, 2013.

[16] M.L. Puterman, *Markov Decision Processes – Discrete Stochastic Dynamic Programming*, Wiley, New York, NY, 1994.

[17] W. Ruml, M.B. Do, R. Zhou and M.P.J. Fromherz, On-line planning and scheduling: An application to controlling modular printers, *Journal of Artificial Intelligence Research (JAIR)* **40** (2011), 415–468.

[18] 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.

[19] S. Yoon, W. Ruml, J. 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.