ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230479

Generating Replanning Goals Through Multi-Objective Optimization in Response to Execution Observation

Alberto Pozanco[1;*], Daniel Borrajo[1;] and Manuela Veloso[1]**

[1]J.P.Morgan AI Research

Abstract. In some applications, planning-monitoring systems generate plans and monitor their execution by other agents. During execution, agents might deviate from these plans for various reasons. The deviation from the expected behavior will be observed by the planning-monitoring system, which will replan in order to provide the agent a new suggested plan. Most existing replanning approaches maintain the goals and compute a plan that achieves them under the new circumstances. This is often not realistic, as achieving the original goal might be very costly or impossible under the current conditions. Furthermore, replanning approaches usually overlook agent's behavior up to the observed deviation from the original plan. In this paper we introduce GREPLAN, a novel approach that proposes new replanning goals (and plans) by solving a multi-objective optimization problem that considers all goals within a perimeter of the original goal. Empirical results in several planning benchmarks show that GREPLAN successfully reacts to deviations from the original plan by generating new appropriate replanning goals.

1 Introduction

In some applications, planning-monitoring systems generate plans and monitor their execution by other agents. This is the case of multi-agent architectures, where a centralized entity generates plans and distributes them for execution by other agents [23, 37, 28]. More interestingly, this is also the case of successful planning applications such as navigation (Google Maps, Waze), financial planning [30] or tourist route planning systems [10], where humans are the agents executing plans generated by planning systems. During execution, agents might deviate from these plans for various reasons such as internal decisions, execution failures [16, 40], or the emergence of opportunities [9, 5, 6]. The deviation from the expected behavior will be observed by the planning-monitoring system, which will replan in order to provide the agent a new suggested plan. Most existing approaches for replanning assume the initial goal is fixed, and compute a plan that achieves it under the new conditions. This is often not realistic, as achieving the original goal might be very costly or impossible under the new circumstances. Also, in many of these cases, agents might accept relaxing the original goal as long as it translates into less costly plans.

Consider the case of a navigation tool that generates driving routes for users (the execution agent will be a human in this case). A simplified version of such domain is shown in Figure 1. A user depicted with a car is using a planning tool to go from its initial location to its goal destination, depicted with a blue cell. To do so, the car can move in the four cardinal directions. The original plan proposed by the navigation tool to the user is shown with a straight blue arrow. At some point during plan's execution, the user deviates from the original plan due to an unexpected traffic jam or a road block in the surroundings of its destination, turning left and following the plan depicted with the green arrow. These unexpected situations are common in the real world due to partial observability of the environment by the navigation tool and/or stochastic environments. If the navigation tool uses standard replanning techniques and maintains the original goal, the new plan would be very costly, i.e., it will cause the user to waste a lot of time as s/he will still be routed through the traffic jam to reach its desired destination. This is what actually happens in most current navigation tools, sometimes leading to user frustration. Instead, the user could prefer the tool to generate an alternative goal destination that is not far from the original one, aligns with the followed route, and does not take long to reach from its current location. In the example shown in Figure 1, this could mean the tool suggesting a new plan to achieve the cell depicted in green.

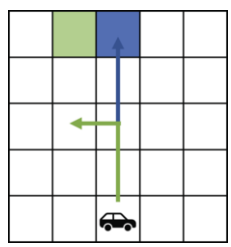

Figure 1: Replanning scenario in a navigation domain where the driver of a car deviates (green arrow) from the original plan (straight blue arrow) proposed by the navigation tool to achieve its original goal (reaching the blue cell). Green cell depicts a potential new replanning goal.

Yet another example is on financial planning tools [30] where a user might want to save some amount of money (e.g. $1000) in a given horizon (e.g. four months). The available actions are to save different quantities at each month, with increasing costs associated with higher savings levels due to the difficulty in saving more money. The suggested plan involves the user saving $250 each month. While the user followed the plan the first month, s/he faced unexpected payments in the second month, being able to save only $50. If we would keep the original goal and replan from the new state, the new plan would be very costly, as it would require the user saving extra money in the subsequent months. This would be a very unrealistic and frus-

* Corresponding Author. Email: alberto.pozancolancho@jpmorgan.com.
** On leave from Universidad Carlos III de Madrid

trating plan that the user would not like. A more realistic plan could entail relaxing the goal, i.e., propose the user a new goal/plan that seems likely to be achievable/executable given the user behavior up to the replanning step. In this case, the planner might suggest the user to save $500 by contributing $100 in the remaining two months, even if it does not reach the original goal.

Other approaches do take into account the agent's behavior up to replanning, being able to extend or even change the original goal [13, 35]. However, they assume humans can specify a set of alternative goals to be achieved under the different conditions. While this information is sometimes available, explicitly providing this knowledge upfront is impossible in many other applications where users can only specify one goal state in order to simplify user experience. This is the case of navigation tools, where users specify their destination; or financial planning tools, where users specify the money they want to save.

In this paper we introduce GREPLAN, a novel approach that proposes new replanning goals (and plans). First, GREPLAN generates the set of replanning candidate goals by computing the set of goals within a perimeter of the original goal. Then, GREPLAN solves a multi-objective optimization problem that considers the following features/objectives for each replanning candidate goal:

- Distance between the original goal and the replanning candidate goal. We compute this distance as the backtracking cost of reaching the new goal from the original one.
- Consistency of the replanning candidate goal with respect to the already executed plan. We use goal recognition approaches to estimate the consistency of a goal with a plan.
- Cost of the plan achieving the new replanning candidate goal. We compute this cost through heuristic estimations.

The rest of the paper is organized as follows. We first formalize the concepts we will use throughout the paper: automated planning, regression in planning, replanning, and planning-based goal recognition. Then, we introduce GREPLAN, our novel realistic replanning approach that generates new replanning goals. After that, we evaluate GREPLAN in different planning domains, showing its scalability and the inherent trade-offs when generating new replanning goals. Finally, we put our contribution in the context of related work, present our conclusions, and outline some future lines of research.

2 Background

2.1 Automated Planning

Automated Planning is the task of choosing and organizing a sequence of actions such that, when applied in a given initial state, it results in a goal state [18]. Formally:

Def. 1 *A* STRIPS ***planning task*** *can be defined as a tuple* $\Pi = \langle F, A, I, G \rangle$*, where* F *is a set of propositions,* A *is a set of instantiated actions,* $I \subseteq F$ *is an initial state, and* $G \subseteq F$ *is a set of goals.*

A state consists of a set of propositions $s \subseteq F$ that are true at a given time. A state is totally specified if it assigns truth values to all the propositions in F, as the initial state I of a planning task. A state is partially specified (partial state) if it assigns truth values to only a subset of the propositions in F, as the conjunction of propositions G of a planning task. Each action $a \in A$ is described by a set of preconditions (pre(a)), which represent literals that must be true in a state to execute an action, and a set of effects (eff(a)), which are the literals that are added (add(a) effects) or removed (del(a) effects) from the state after the action execution. The definition of each action might also include a cost $c(a)$ (the default cost is one). The execution of an action a in a state s is defined by a function γ such that $\gamma(s, a) = (s \setminus \text{del}(a)) \cup \text{add}(a)$ if $\text{pre}(a) \subseteq s$, and s otherwise (it cannot be applied). The output of a planning task is a sequence of actions, called a plan, $\pi = (a_1, \ldots, a_n)$. The execution of a plan π in a state s can be defined as:

$$\Gamma(s, \pi) = \begin{cases} \Gamma(\gamma(s, a_1), (a_2, \ldots, a_n)) & \text{if } \pi \neq \emptyset \\ s & \text{if } \pi = \emptyset \end{cases}$$

A plan π is valid if $G \subseteq \Gamma(I, \pi)$. The plan cost is commonly defined as $c(\pi) = \sum_{a_i \in \pi} c(a_i)$. A plan with minimal cost is called optimal. Since solving planning tasks is challenging [8], sometimes we will be interested in using heuristic functions that estimate the cost of reaching the goal from a given state s, $h(s)$. We will use function PLANNER(Π) to compute a plan that solves planning task Π, and function HEURISTIC(Π) to estimate the cost of reaching G from I in planning task Π. We assume both functions return infinity if the planning task is not solvable, i.e., the goal is not reachable from the initial state.

2.2 Regression in Planning

Most planning algorithms search forward, expanding states from the initial state until the goal state is reached [3]. But plans can also be computed by searching backwards (regression), i.e., starting from the goal and applying the actions backwards to find a path from the goal to the initial state [2]. The applicability of actions is redefined as follows: $a \in A$ is applicable in a partial state s if it is *relevant* ($\text{add}(a) \cap s \neq \emptyset$) and *consistent* ($\text{del}(a) \cap s = \emptyset$). The resulting state obtained from regressing a in s is defined by a function γ_r such that $\gamma_r(a, s) = (s \setminus \text{add}(a)) \cup \text{pre}(a)$. Progression (searching forward) and regression (searching backward) in planning are not symmetric, as each state in the regression state space may represent a set of states of the progression state space. We use REGRESSION(Π, D) to refer to an algorithm that computes the set of partial states that can be reached from the goal G by searching backwards with a cost bound or perimeter $\leq D$.

2.3 Replanning

We are interested in scenarios where an initial plan π that solves a planning task Π has been constructed, and its execution has deviated from the expected one. As discussed, this can happen due to a number of reasons, and we do not make any assumption on the event that triggered replanning. The plan already executed up to replanning cannot be retracted, so we can always consider the current state as the new initial state and plan again. While other approaches focus on the whole initial plan π so as to perform plan repair or similar tasks [16, 4], we only focus on the agent's current state when replanning was triggered. We formally define a replanning problem as follows.

Def. 2 *A* ***replanning problem*** *is a tuple* $\Pi_R = \langle \Pi, I', G' \rangle$ *where* Π *is the original planning task, and* G' *is the new goal state to be achieved from the current state* I'*.*

Although some works allow the replanning component to modify the actions when replanning [14], we assume the set of available actions is the same as in the original planning task Π. Also, most replanning approaches assume either the new goal remains the same ($G' = G$) or the new goal is provided by humans. We will later discuss how we

relax this assumption and compute a new goal G'. The solution to a replanning problem is a new plan π that achieves the new goal G' from the agent's current state I'.

2.4 Goal Recognition

Goal Recognition is the task of inferring another agent's goals through the observation of its interactions with the environment. The problem has captured the attention of several computer science communities [1]. Among them, planning-based goal recognition approaches have been shown to be a valid domain-independent alternative to infer agents' goals [31, 21, 27]. Most approaches distill from Ramírez and Geffner [32] seminal work, which formally defines a planning-based goal recognition problem as:

Def. 3 *A **goal recognition problem** is a tuple $T = \langle F, A, I, \mathcal{G}, O, Pr \rangle$ where $\mathcal{G}$ is the set of possible goals G such that $G \subseteq F$, $O = (o_1, ..., o_m)$ is an observation sequence with each o_i being an action in A, and Pr is a prior probability distribution over the goals in $\mathcal{G}$.*

The solution to a goal recognition problem is a probability distribution over the set of goals $G_i \in \mathcal{G}$ giving the relative likelihood of each goal. The probability of each goal is computed based on the *cost difference* between the cheapest plan that can reach the goal, given the observed actions already executed, and the cheapest plan that could have reached the goal, had the agent not executed the observed actions. The costs of these optimal plans are given as $optc(G, O)$ and $optc^{\neg}(G, O)$, respectively. The cost difference is formally defined as follows:

$$costdif(G_i, O) = optc(G_i, O) - optc^{\neg}(G_i, O) \quad (1)$$

This approach needs to compile and solve two planning tasks for each candidate goal, which are more complex than the original planning task Π. To avoid this, Masters and Sardina [24] developed a less demanding approach that is observation-free, i.e., it does not need to compile the observations into two new planning tasks, just relying on the difference between the cost to achieve the goal from the initial state I and the current state of the agent I':

$$costdif(G_i, I, I') = cost(I', G_i) - cost(I, G_i) \quad (2)$$

Even though this approach might not be as accurate as Ramírez and Geffner's, we will use it in the remainder of the paper, as it is computationally more efficient and requires us to make less assumptions on the received observations.

A probability distribution that solves the goal recognition problem is derived by inputting the cost differences of all $G_i \in \mathcal{G}$ in a formula that satisfies the property that the lower the cost difference, the higher the probability, and the relative cost differences are preserved [24]. In this work we will use the following formula to compute the probability distribution. In order to simplify notation, we use $\text{cd}(G_i)$ to refer to the cost difference computed by Eq. 2, and $\text{Mcd}(G')$ and $\text{mcd}(G')$ to refer, respectively, to the maximum and minimum cost difference value across the candidate goals $G' \in \mathcal{G}$.

$$P(G_i \mid I, I') = \begin{cases} \frac{\text{Mcd}(G') - \text{cd}(G_i)}{\text{Mcd}(G') - \text{mcd}(G')}, & \text{if } \text{Mcd}(G') \neq \text{mcd}(G') \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

In this formula, most likely goals across the candidate goals will have a probability of 1, least likely goals will have a probability of 0, and the rest of candidate goals will have probabilities within the $[0, 1]$ range. We use $\text{GR}(F, A, I, G_i, I')$ to refer to a function that returns the probability of $G_i \in \mathcal{G}$ being the agent's actual goal given its initial (I) and current (I') states.

3 GREPLAN: Generating Replanning Goals through Multi-objective Optimization

We consider a planning-monitoring system that generates plans and monitors their execution by other agents. We do not make any assumption on the planning component and how it computes the plans, i.e., the accuracy of the planning model used to generate the plans wrt. the real world, or the quality of such plans. Likewise, we do not make any assumption on the executing agents. The only assumption we make is that the monitoring component is able to (i) observe the agent's plan execution; and (ii) trigger replanning when needed. We do not impose any restriction on how often the monitoring component can receive observations from the agent's execution. We do assume that when an observation is made, the planning-monitoring system can retrieve, at least, the agent's current state I'. This is not restrictive, and observations could contain more information such as the agent's executed actions or the previous agent's states. Regarding the mechanism used by the monitoring component to trigger replanning, we treat it as a black-box. Depending on the monitoring capabilities (type and frequency of observations) and the designer choice, planning-monitoring could trigger replanning when the agent deviates from the original plan, when the preconditions of the next action do not hold, or when an unexpected state is observed.

When replanning is needed, most approaches to solve replanning problems (Def. 2) assume the goal remains the same, $G' = G$. They also overlook agent's behavior up to replanning, which might reveal important information about the environment. In this section we present GREPLAN, our novel approach that proposes a new replanning goal G' (and a plan) by solving a multi-objective problem that considers three different aspects or objectives: distance, consistency and cost. GREPLAN will select G' as the one from the set of alternative potential goals that maximizes a linear function of these three objectives. When solving multi-objective problems, it is usually desirable to understand how changes to one of the objectives (or its weight if they are combined in a linear function) will affect the overall quality of a solution. This is not possible when the value of some of the objectives are unbounded a priori, as the cost of reaching the new replanning goal. In order to make our multi-objective function easier to manipulate and understand, we will bound each objective between 0 and 1. For all objectives, values closer to 1 will indicate better performance.

GREPLAN is described in Algorithm 1. It selects the **new replanning goal** as the replanning candidate goal $G' \in \mathcal{G}_r$ that maximizes the multi-objective function defined in line 13. It receives as input the original planning task Π, the agent's current state I', the distance bound D, and a set of weights $W = \langle w_1, w_2, w_3 \rangle \mid w_i \in [0, 1]$ that weight the importance of the different objectives.

First, GREPLAN generates the set of **replanning candidate goals** $\mathcal{G}_r$ (line 2) by applying the regression function that computes the partial states that can be reached from the original goal G by searching backwards with a cost bound $\leq D$. The original goal G is always inside $\mathcal{G}_r$, as it can be reached with cost 0. Then, the algorithm iterates over all the replanning candidate goals. For each $G_i \in \mathcal{G}_r$, a new hypothetical planning task Π_i is built (line 5), where the candidate goal needs to be achieved from the agent's current state I'. The estimated cost of solving this planning task $h(\pi_i)$ is computed by the HEURISTIC function. If this estimation returns infinity, i.e., G_i can-

Algorithm 1 GREPLAN

Input: $\Pi = \langle F, A, I, G \rangle, I', D, W = \langle w_1, w_2, w_3 \rangle$
Output: G'
1: $G' \leftarrow \emptyset$
2: $\mathcal{G}_r \leftarrow$ REGRESSION(Π, D)
3: $f(G_i, W) \leftarrow 0, \forall G_i \in \mathcal{G}_r$
4: **for** G_i in $\mathcal{G}_r$ **do**
5: $\quad \Pi_i \leftarrow \langle F, A, I', G_i \rangle$
6: $\quad h(\pi_i) \leftarrow$ HEURISTIC(Π_i)
7: $\quad$ **if** $h(\pi_i) \leftarrow \infty$ **then**
8: $\quad\quad f(G_i, W) \leftarrow -\infty$
9: $\quad$ **else**
10: $\quad\quad$ distance $\leftarrow 1 - bd(G, G_i)$
11: $\quad\quad$ consistency $\leftarrow \text{GR}(F, A, I, G_i, I')$
12: $\quad\quad$ cost $\leftarrow 1 - bc(\pi_i)$
13: $\quad\quad f(G_i, W) \leftarrow w_1 \times$ distance $+ w_2 \times$ consistency $+ w_3 \times$ cost
14: $\quad$ **end if**
15: **end for**
16: $G' \leftarrow \arg\max_{G_i \in \mathcal{G}_r} f(G_i, W)$
17: **return** G'

not be achieved from I', G_i is discarded from the set of replanning candidate goals; e.g. assigned the lowest possible value (line 8). Otherwise, GREPLAN computes the three objectives considered to select the new replanning goal: distance, consistency and cost.

The first objective optimized by GREPLAN is the **distance** between the original goal and the replanning candidate goal, $\mathcal{D}(G, G_i)$. We compute this distance as the cost of reaching G_i from G by searching backwards. We bound the distance term by dividing it by the cost bound D, i.e., the size of the perimeter.

$$bd(G, G_i) = \frac{\mathcal{D}(G, G_i)}{D} \tag{4}$$

The second objective is the **consistency** of the new goal with respect to the agent's behavior up to replanning. We use the observation-free goal recognition approach [24], which returns a probability distribution of all replanning candidate goals $G_i \in \mathcal{G}_r$ after observing I and I'. We use the function $\text{GR}(F, A, I, G_i, I')$ to compute this probability distribution, whose values are already bounded in the $[0, 1]$ range (see Eq. 3).

The third and last objective is the **cost** of the plan achieving the replanning candidate goal G_i from I'. We bound the cost term by dividing the difference between the estimated cost of reaching the given goal minus the minimum cost across all the replanning goals $G_i \in \mathcal{G}_r$, by the difference between the maximum and minimum costs across all the replanning candidate goals. In order to simplify notation, we use $\text{Mh}(G_j)$ and $\text{mh}(G_j)$ to refer, respectively, to the maximum and minimum estimated cost to achieve any of the replanning candidate goals $G_j \in \mathcal{G}_r$ from I'.

$$bc(\pi_i) = \begin{cases} \frac{h(\pi_i) - \text{mh}(G_j)}{\text{Mh}(G_j) - \text{mh}(G_j)}, & \text{if } \text{Mh}(G_j) \neq \text{mh}(G_j) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Finally, GREPLAN returns as the new replanning goal G' the one that maximizes the function $f(\cdot, \cdot)$ (line 16), which consists of a linear sum of the three objectives weighted by the set of weights W (line 13). By maximizing this multi-objective function, GREPLAN can propose new replanning goals that: are not far from the original one (w_1); (ii) align with the agent's observed behavior (w_2); and are not costly to achieve (w_3).

We would like to end the section by providing a few remarks. First, GREPLAN could avoid generating the set of replanning candidate goals if this set is given. The rest of the algorithm would remain the same, and we would only be reasoning about the distance, consistency and cost of the goals provided by the user. Second, GREPLAN uses heuristics to compute the consistency and cost terms. While they could also be computed through more precise and expensive methods, i.e., Ramírez and Geffner's goal recognition approach (Eq. 1) to compute consistency, and optimal plans to compute cost, we want GREPLAN to be as fast as possible.

3.1 Running Example

Let us exemplify how GREPLAN works by considering the navigation domain discussed in the Introduction and shown in Figure 2. The

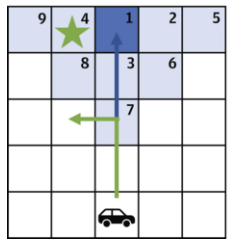

Figure 2: Navigation example previously shown in Figure 1. Blue cells conform the set of replanning candidate goals $\mathcal{G}_r$ when $D = 2$. Numbers unequivocally identify each replanning candidate goal G_i. The green star depicts the new replanning goal G' generated by GREPLAN.

blue cells represent the set of replanning candidate goals generated by the algorithm when called with a distance bound $D = 2$, i.e., those states at a distance (cost) of two or less from the original goal state. The original goal is included in this set. Then, GREPLAN finds the goal that maximizes the function $f(\cdot, \cdot)$. Hence, it computes the value of each of the objectives (distance, consistency and cost) for each replanning candidate goal, resulting in a matrix where goals are represented in the rows and the objectives in the columns. Table 1 shows an excerpt of that matrix for the navigation example when we use the FF heuristic [20] to estimate consistency and cost.

Candidate Goal	Distance	Consistency	Cost
G_1	**1.0**	0.0	0.5
G_2	0.5	0.0	0.25
G_4	0.5	**1.0**	0.75
G_8	0.0	**1.0**	**1.0**
G_9	0.0	**1.0**	0.5

Table 1: Value of the three objectives for a subset of goals in $\mathcal{G}_r$ in the navigation example shown in Figure 2. Values are computed using the FF heuristic. Bold numbers represent best scores for each metric.

The new replanning goal generated by GREPLAN will depend on the specified weights, and therefore on the behavior expected by the driver (agent). For example, setting $w_1 = 1$ (distance) and $w_2, w_3 = 0$ (consistency and cost) would translate into standard replanning where the original goal is not changed, $G' = G_1 = G$. On the other hand, if we equally balance all weights: $w_1, w_2, w_3 = 1$, G_4 would be the new replanning goal, as it maximizes the sum of the three objectives. As we can see in Figure 2, this is a goal that is not far from the original goal, is consistent with the driver's behavior up to replanning, and is not far from its current state.

3.2 GREPLAN *in the Absence of Weights*

When weights are specified, a Pareto front over the replanning candidate goals can be computed. This offers a way to visualize and

understand the trade-offs between different objectives, showing the set of solutions that are optimal (dominate) in the sense that there is no other solution that can improve any objective without worsening another. However, in some scenarios users of planning-monitoring systems cannot provide a complete definition or ranking over the weights. This can occur in cases where users do not fully understand the problem, are not aware of the different objectives trade-offs, or are simply lazy to input a set of weights.

In these cases, we propose a way of selecting the new replanning goal G' as the goal that optimizes $f(\cdot,\cdot)$ under a larger number of weights' configurations or scenarios W. We model this problem as a Mixed-Integer Linear Programming (MILP) problem as follows.

$$\max \sum_{G_i \in \mathcal{G}_r} \sum_{W \in \mathcal{W}} \alpha\Big(x_{G_i} f(G_i, W)\Big) + \beta\Big(x_{G_i} \text{WINS}(G_i, \mathcal{G}_r, \mathcal{W})\Big) \tag{6}$$

subject to:

$$\sum_{G_i \in \mathcal{G}_r} x_{G_i} = 1 \tag{7}$$

$$x_{G_i} \in \{0,1\}, \quad \forall G_i \in \mathcal{G}_r \tag{8}$$

We create one binary decision variable x_{G_i} for each replanning candidate goal $G_i \in \mathcal{G}_r$. These variables will take a value of 1 if G_i is set to be the new replanning goal G', and 0 otherwise. The set of potential weight configurations $\mathcal{W}$ can be either given explicitly as a set of allowed configurations, or implicitly as rules to compute this set, i.e., all the combinations of weights with a granularity of 0.1.

The objective function (Eq. 6) maximizes two different objectives. The first objective, weighted by a constant α, maximizes the value of $f(\cdot,\cdot)$ under all possible weight's configurations $\mathcal{W}$. The second objective, weighted by a constant β, maximizes the number of weight's configurations under which G_i is the best goal. This number is computed by the WINS function, which receives the replanning goal G_i, the set of replanning goals $\mathcal{G}_r$, and the set of potential weight configurations $\mathcal{W}$. This function returns the number of weight configurations for which G_i maximizes $f(\cdot,\cdot)$, i.e., the number of scenarios for which G_i is the best replanning candidate goal. Although these two objectives are aligned, they might yield different results, i.e., new replanning goals. While the first objective focuses on the total sum of the $f(\cdot,\cdot)$ function across all the scenarios, the second one just counts the number of scenarios for which G_i turns to be the best new replanning goal, disregarding the total sum of the function. Therefore, we would set $\alpha = 1$ and $\beta = 0$ when interested in a goal that maximizes quality margin, even if this margin occurs in a lower number of weight configurations. On the other hand, we would set $\alpha = 0$ and $\beta = 1$ when interested in a goal that maximizes the number of scenarios under which it should be chosen, regardless of the quality margin.

This objective function is optimized subject to Constr. 7, which ensures that only one goal is chosen, i.e., we only return one new replanning goal G'. This simple model can be enhanced to include arbitrary constraints such as forcing the selected goal to have a given value in a specific metric across a number of weight configurations.

Going back to our running example, and assuming all the combinations of weights with a granularity of 0.1 in the range $[0,1]$ ($|\mathcal{W}| = 220$), the MILP returns that G_8 is the best replanning candidate goal. When we optimize quality margin, G_8 obtains a total score of 291.9 versus the 272.4 obtained by the second best goal, G_4. When we optimize number of scenarios for which the goal should be chosen, G_8 wins in 171 versus the 49 of G_4. Therefore, G_8 should be the goal the driver should be pursuing after replanning when a specific set of weights W is not provided.

4 Evaluation

4.1 Experimental Setting

We evaluate GREPLAN in different planning domains. Although the technique is general and we could evaluate it in any planning domain, we opted for selecting a varied set of domains that we believe are better suited for GREPLAN's replanning flavour. We selected the below well-known planning domains that, semantically, share the following property: the goal state is somewhat flexible, and after an event triggered replanning, agents could accept goal states that are not too far from the original one, as long as this relaxation results in less costly plans.

- DRIVERLOG. A logistics domain where drivers need to use trucks to deliver packages between locations.
- HIKING. A domain where partners do hiking routes in order to visit certain landmarks.
- ROVERS. A planetary exploration domain where a collection of rovers navigate a planet surface, finding samples and communicating them back to a lander.
- TIDYBOT. A household cleaning domain where one or more robots pick up a set of objects and put them into goal locations.
- TPP. A domain where agents need to travel to buy sets of goods at different markets.

Replanning candidate goals in these domains include states where the packages have been delivered but the truck/driver has not returned yet to its goal location in DRIVERLOG; or states where goods have been bought but not yet stored in TPP. We selected the first 10 planning tasks for each domain from the Planning Domains repository[1]. For each planning task, we generated a number of replanning problems by running Algorithm 2 with different parameters. The algorithm receives as input a planning task Π and two parameters that will affect the behavior of the agent we are simulating. The first parameter $n \in [0,1]$, represents the ratio of actions from the original plan followed by the agent until we make it deviate. In the example of Figure 2, $n = 0.5$, as the car followed half of the original plan before deviating. The second parameter is $r \in [0,1]$, which represents the ratio of actions from the original plan in which we make the agent execute random applicable actions. In the running example, $r = 0.25$, as the car executed 1 random action out of the 4 actions the original plan had. By playing with these two parameters, we are able to generate replanning scenarios where we control how much the agent follows the original plan, and how large its deviation from that plan is. First, the algorithm computes a plan π that solves the original planning task Π. This plan is computed using the `lama-first` configuration of Fast Downward [19], which runs the greedy search used in the first iteration of the LAMA any-time planner [33]. Then, it uses the PREFIX function together with the parameter n to get the

Algorithm 2 Generate Replanning Problems

Input: $\Pi = \langle F, A, I, G \rangle, n, r$
Output: Π_T
1: $\pi \leftarrow$ PLANNER(Π)
2: $\pi_n \leftarrow$ PREFIX(π, n)
3: $I_n \leftarrow \Gamma(I, \pi_n)$
4: $\pi_r \leftarrow$ RANDOMACTIONS(F, A, I_n, r)
5: $I' \leftarrow \Gamma(I_n, \pi_r)$
6: $\Pi_R = \langle \Pi, I' \rangle$ ▷ G' will be generated by GREPLAN
7: **return** Π_R

[1] https://github.com/AI-Planning/classical-domains

subset (prefix) of the plan π_n in which the agent will stick to the original plan. By executing π_n from I, the agent will be at a new state I_n from which it will deviate by executing a number of random actions given by the parameter r. This random plan π_r is generated by the RANDOMACTIONS function (line 4). After that, π_r is applied from I_n, giving us the agent's current state I'. Finally, a new replanning problem Π_R is built with the original planning task Π and I'. We run Algorithm 2 with the following parameters' combinations: $n = [0.25, 0.5, 0.75]$, and $r = [0.1, 0.2]$. We did not test higher values for r since that would imply unrealistic scenarios where the planning-monitoring system takes too long to detect the deviation and provide a new suggestion to the executing agent. This gives us $5(\text{domains}) \times 10(\text{problems}) \times 3(|n|) \times 2(|r|) = 300$ replanning problems for which GREPLAN will generate a new replanning goal.

We run GREPLAN with two distance bounds $D = [1, 2]$. Again, we did not test higher values for D as they would entail considering goals reasonably far from the original one. We consider two different versions of GREPLAN: GREPLAN$_U$, which uses a uniform set of weights $W = \langle 1, 1, 1 \rangle$; and GREPLAN$_A$ which runs GREPLAN in the absence of weights, with all the combinations of weights with a granularity of 0.1 in the range $[0, 1]$ ($|\mathcal{W}| = 220$). We set GREPLAN$_A$'s objective function weights (Eq. 6) so it lexicographically optimizes first the quality margin of the new replanning goal (α is set to a large constant), breaking ties in favour of new replanning goals that are optimal under more weights' configurations (β is set to a small non-zero constant). Both GREPLAN versions use the FF heuristic as it is implemented in Fast Downward to estimate the *cost* and *consistency* terms in the $f(\cdot, \cdot)$ function. GREPLAN$_A$ uses the CBC solver [15] to solve the MILP. After GREPLAN generates a new replanning goal G', a plan to solve the new replanning problem $\Pi_R = \langle \Pi, I', G' \rangle$ is generated using `lama-first`. Experiments were run on an Apple M1 with 8GB of memory. Code and benchmarks are available upon request.

4.2 Results

We designed GREPLAN's evaluation in order to answer the following two questions:

1. **Execution time analysis**. How much time does it take GREPLAN to generate a new replanning goal?
2. **Quality analysis**. Does GREPLAN's execution time pays off as to how *good* the new replanning goal is wrt the original one?

Table 2 shows the results of the **execution time analysis**, where we run the two versions of GREPLAN with different distance bounds D in replanning problems across the five different domains. We measure the execution time of each version and their respective components, as well as the time needed to solve the replanning problem once GREPLAN generates the new replanning goal (Replanning column). This is approximately the execution time of standard replanning, which does not reason about which goal should be achieved. Both GREPLAN$_U$ and GREPLAN$_A$ have similar execution times (Total (s) columns). The time needed by GREPLAN to generate a new replanning goal is related to two factors. First, GREPLAN is computing two heuristic values for each replanning candidate goal G_i: the cost of achieving it from I and I'. The number of replanning candidate goals rapidly goes up as we increase D, translating into higher execution times for both GREPLAN versions. Second, the complexity of the replanning task, for which the replanning time serves as a proxy. The more complex the planning task is, the more time GREPLAN will take to compute the two heuristic values.

Let us also take a look to how each component affects GREPLAN's execution time. The extra time needed by GREPLAN$_A$ to run the MILP is negligible, making it a good choice if a weight configuration is not provided. The generation of replanning candidate goals and computation of their distance (Distance column) as well as running the $\arg\max$ function to get G' usually take less than a second. On the other hand, computing the cost and consistency terms takes most of GREPLAN's execution time. Computing the cost term (heuristic value from I' to G_i) takes slightly longer, as it computes heuristic values for a larger number of replanning candidate goals. This is because this heuristic value is used to filter out unreachable goals, leaving the consistency term with a lower number of replanning candidate goals to reason about.

Table 3 shows the results of the **quality analysis**, where we run GREPLAN$_U$ with different distance bounds D in replanning problems across the five different domains. Results for GREPLAN$_A$ are similar and yield the same conclusions. The aim of this analysis is to better understand the differences between GREPLAN and standard replanning. We measure the ratio of times the original goal G is selected as the new replanning goal G'. We also measure the quality difference (Δ) between the new replanning goal generated by GREPLAN and the original goal. Larger values indicate G' has a better score than G in the given objective. As we can see, the number of times GREPLAN selects a goal different from the original one depends on the domain. While a new replanning goal is picked most of the times in TIDYBOT, achieving the original goal remains as the best option in most ROVERS and TPP problems. This is also influenced by the distance bound considered. In all domains a different goal tends to be selected as we increase D, since more replanning candidate goals are considered ($D = 2$). This is due two reasons. First, by increasing D we are considering more replanning candidate goals, increasing the probabilities of another one getting better values across the different objectives. Second, when $D = 1$ alternative replanning goals can only get 0 in that objective, while when $D = 2$ these goals at a distance of one from the original goal get 0.5. The number of times GREPLAN selects a goal different from the original one also depends on the weight's configuration W. In cases where we run GREPLAN$_A$, or GREPLAN with weight configurations that value more the consistency and cost terms, a new replanning goal is selected most of the times.

As expected, the original goal outperforms $\mathcal{G}'$ in the distance objective, as it always gets a value of 1 compared to the 0 and 0.5 values other replanning candidate goals can get. On the other hand, the new replanning candidate goal tends to outperform the original goal in the consistency and cost objectives. This strongly depends on the given replanning problem, which is highlighted by the high standard deviation numbers. When $G' = G$, the difference in all the objectives will be 0, lowering the average. However, when we only consider the subset of replanning problems for which GREPLAN returns a goal different than the original one, these differences increase. For example, it goes from 0 to 0.2 in DRIVERLOG when $D = 1$, and from 0.1 to 0.3 in HIKING and ROVERS when $D = 2$. This difference is even higher in certain replanning problems. For example, we observe a total difference of 0.72 in one DRIVERLOG instance, and a difference of 1.0 in various HIKING and TIDYBOT instances, with most of the difference coming from the consistency term. These results show that while in some replanning problems standard replanning (where the goal is fixed) is the best choice, in others using GREPLAN to generate a new replanning goal offers great advantages, as the new goal is closer to the agent's current state and better aligns with the agent's behavior up to replanning.

			GREPLAN$_U$				GREPLAN$_A$		Replanning
Domain	D	$\lvert\mathcal{G}_r\rvert$	Distance (s)	Consistency (s)	Cost (s)	Total (s)	MILP (s)	Total (s)	Time (s)
DRIVERLOG	1	29 ± 9.7	0.1 ± 0.0	2.3 ± 0.9	2.4 ± 0.9	4.8 ± 1.7	0.0 ± 0.0	4.8 ± 1.7	0.1 ± 0.0
	2	111 ± 55.1	0.1 ± 0.0	9.0 ± 5.0	9.0 ± 5.0	18.1 ± 10.1	0.1 ± 0.0	18.2 ± 10.1	0.1 ± 0.0
HIKING	1	3 ± 1.4	0.1 ± 0.1	0.7 ± 0.6	0.7 ± 0.6	1.5 ± 1.3	0.0 ± 0.0	1.5 ± 1.3	0.4 ± 0.3
	2	313 ± 241.0	0.1 ± 0.1	76.1 ± 91.2	76.1 ± 91.2	152.3 ± 182.5	0.3 ± 0.2	152.6 ± 182.7	0.4 ± 0.3
ROVERS	1	35 ± 26.1	0.1 ± 0.0	2.9 ± 2.3	3.4 ± 2.9	6.4 ± 5.2	0.0 ± 0.0	6.4 ± 5.2	0.1 ± 0.0
	2	103 ± 65.6	0.1 ± 0.0	8.6 ± 6.0	9.3 ± 6.7	17.9 ± 12.7	0.1 ± 0.1	18.0 ± 12.8	0.1 ± 0.0
TIDYBOT	1	6 ± 1.0	3.3 ± 0.7	27.4 ± 8.9	32.6 ± 9.0	63.3 ± 18.0	0.0 ± 0.0	63.3 ± 18.0	10.7 ± 2.9
	2	83 ± 47.7	3.5 ± 0.7	375.6 ± 249.2	1029.0 ± 300.6	1408.1 ± 491.9	0.1 ± 0.0	1408.2 ± 491.9	10.7 ± 2.9
TPP	1	12 ± 11.7	0.1 ± 0.0	1.0 ± 1.1	1.7 ± 1.6	2.8 ± 2.5	0.0 ± 0.0	2.8 ± 2.6	0.1 ± 0.0
	2	44 ± 44.3	0.1 ± 0.0	3.7 ± 3.9	6.9 ± 7.4	10.7 ± 11.1	0.0 ± 0.0	10.7 ± 11.2	0.1 ± 0.0

Table 2: Execution time results of GREPLAN$_U$ and GREPLAN$_A$ in replanning problems across the five different domains when they are run with different distance bounds D. The $\lvert\mathcal{G}_r\rvert$ column shows the average and standard deviation of the number of replanning candidate goals considered. Columns under GREPLAN$_U$ and GREPLAN$_A$ show the average and standard deviation seconds required by each component, as well as the total execution time. The last column shows the time needed to solve the replanning problem once GREPLAN generates the new replanning goal.

		GREPLAN$_U$				
Domain	D	ΔDistance	ΔConsistency	ΔCost	ΔTotal	$G' = G$
DRIVERLOG	1	-0.0 ± 0.2	0.0 ± 0.1	0.0 ± 0.2	0.0 ± 0.0	0.97
	2	-0.3 ± 0.3	-0.1 ± 0.2	0.5 ± 0.5	0.1 ± 0.2	0.45
HIKING	1	-0.2 ± 0.4	0.1 ± 0.2	0.2 ± 0.4	0.1 ± 0.2	0.78
	2	-0.2 ± 0.3	0.1 ± 0.2	0.3 ± 0.4	0.1 ± 0.2	0.63
ROVERS	1	-0.0 ± 0.2	0.0 ± 0.0	0.0 ± 0.2	0.0 ± 0.0	0.97
	2	-0.2 ± 0.3	-0.0 ± 0.2	0.4 ± 0.5	0.1 ± 0.2	0.57
TIDYBOT	1	-0.6 ± 0.5	0.2 ± 0.2	0.6 ± 0.5	0.2 ± 0.2	0.37
	2	-0.4 ± 0.2	0.2 ± 0.2	0.6 ± 0.3	0.4 ± 0.3	0.20
TPP	1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1.00
	2	-0.1 ± 0.3	0.0 ± 0.2	0.2 ± 0.4	0.1 ± 0.2	0.83

Table 3: Quality results of GREPLAN$_U$ in replanning problems across the five different domains when running with different distance bounds D. Columns under GREPLAN$_U$ show the average and standard deviation quality difference in the given objective between the new replanning goal G' generated by GREPLAN and the original goal G. Larger values indicate G' has a better score than G in the given objective. The last column shows the ratio of times the original goal is selected as the new replanning goal.

5 Related Work

Most replanning literature has focused on how to compute new plans to achieve the original goal [26]. Approaches to do so include completely disregarding the original plan and planning from scratch [22]; or generating a new plan that is similar to the original one [38, 16]. More recent works have questioned this assumption, considering replanning as a broader problem where goals, or even models [11], can be changed when replanning.

Cushing *et al.* [13] propose an oversubscription planning [34] approach, where the utility of goals is changed according to a reward-penalty model that penalizes agents not respecting commitments made by the original plan. During each replanning epoch, the agent selects objectives considering the current state, the partially executed plan, and all the available goals, including any opportunities and commitments (with their rewards and penalties). These goals (and their rewards) need to be specified a priori, while GREPLAN can generate new goals by only considering the agent's initial and current states and the original goal.

Closer to our approach, other works also use goal recognition in the replanning process [35], or acknowledge the trade-offs of different metrics when replanning [36]. These works use goal recognition in the context of human-robot coordination, as a way of modelling beliefs: the robot will extend its set of goals and replan based on the human's observed behavior. Regarding the trade-offs, they focus on three aspects: (i) computational efficiency, i.e., time to replan; (ii) plan stability, i.e., minimize changes with respect to the original plan; and (iii) commitments, i.e., soft constraints that are induced by an executing plan. Like [13], they compile these different replanning flavours into an oversubscription planning task, and discuss how these objectives compete with each other, i.e., you cannot do fast replanning if you consider plan stability. In our case, GREPLAN is solving a standard planning task, where the new goal is generated by solving a multi-objective problem that optimizes three objectives not previously considered in the literature: distance from the original goal, consistency with the already executed plan and cost from the current state.

Although we have mainly focused on architectures where planning-monitoring is decoupled from execution, GREPLAN could also be used by autonomous agents capable of creating/managing their own goals [25, 39]. While other works on Goal Reasoning and Goal-Driven Autonomy generate new goals based on rule-based systems [12], models predicting the appearance of goals in the future [7, 17], or based on the behavior of opponent agents [29], we allow agents to generate their own goals when replanning by considering different objectives.

6 Conclusions and Future Work

We have introduced GREPLAN, a novel approach that proposes new goals for replanning by solving a multi-objective optimization problem that considers all goals within a perimeter of the original goal. GREPLAN optimizes three objectives: (i) the distance between the original goal and the replanning candidate goal; (ii) the consistency of the replanning candidate goal with respect to the already executed plan; and (iii) the cost of the plan achieving the new replanning candidate goal. Experimental results in replanning problems across five different domains show that, although GREPLAN requires some extra time to solve the optimization problem, in many tasks the generated new replanning goal compensates this overhead by being better suited for replanning, as it is not far from the original goal, aligns with the agent's observed behavior, and is not costly to achieve.

In future work we would like to alleviate GREPLAN's computation time by reducing the number of candidate goals considered. Potential ways of doing so include randomly sampling a subset of goals, or devising heuristics that help us decide which goals should be filtered. Finally, we would also like to run user studies to understand how humans value the suggestions made by the system and how they weight each of the objectives, as well as identify any other objective humans consider important when replanning.

Acknowledgements

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("J.P. Morgan") and is not a product of the Research Department of J.P. Morgan. J.P. Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

[1] Stefano V. Albrecht and Peter Stone, 'Autonomous agents modelling other agents: A comprehensive survey and open problems', *Artif. Intell.*, **258**, 66–95, (2018).

[2] Vidal Alcázar, Daniel Borrajo, Susana Fernández, and Raquel Fuentetaja, 'Revisiting regression in planning', in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pp. 2254–2260. IJCAI/AAAI, (2013).

[3] Blai Bonet and Hector Geffner, 'Planning as heuristic search', *Artif. Intell.*, **129**(1-2), 5–33, (2001).

[4] Daniel Borrajo and Manuela Veloso, 'Probabilistically reusing plans in deterministic planning', in *Proceedings of ICAPS Workshop on Heuristics and Search for Domain-Independent Planning*, pp. 17–25, (2012).

[5] Daniel Borrajo and Manuela Veloso, 'Computing opportunities to augment plans for novel replanning during execution', in *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pp. 51–55, (2021).

[6] Daniel Borrajo and Manuela Veloso, 'Intelligent execution through plan analysis', in *Proceedings of IROS'21*, Prague, (2021).

[7] Ethan Burns, J Benton, Wheeler Ruml, Sungwook Yoon, and Minh B Do, 'Anticipatory on-line planning', in *Twenty-Second International Conference on Automated Planning and Scheduling*, (2012).

[8] Tom Bylander, 'The computational complexity of propositional strips planning', *Artificial Intelligence*, **69**(1-2), 165–204, (1994).

[9] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, and Bram Ridder, 'Opportunistic planning in autonomous underwater missions', *IEEE Transactions on Automation Science and Engineering*, **15**(2), 519–530, (2017).

[10] Isabel Cenamor, Tomás de la Rosa, Sergio Núñez, and Daniel Borrajo, 'Planning for tourism routes using social networks', *Expert Systems with Applications*, **69**, 1–9, (2017).

[11] Daniel Ciolek, Nicolás D'Ippolito, Alberto Pozanco, and Sebastian Sardiña, 'Multi-tier automated planning for adaptive behavior', in *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 66–74, (2020).

[12] Alexandra Coddington, 'Motivations for madbot: a motivated and goal directed robot', in *Proceedings of the Twenty-Fifth Workshop of the UK Planning and Scheduling Special Interest Group*, pp. 39–46. Citeseer, (2006).

[13] William Cushing, J Benton, and Subbarao Kambhampati, 'Replanning as a deliberative re-selection of objectives', *Arizona State University CSE Department TR*, (2008).

[14] William Cushing and Subbarao Kambhampati, 'Replanning: A new perspective', *Proceedings of the International Confer-ence on Automated Planning and Scheduling Monterey, USA*, 13–16, (2005).

[15] John Forrest and Robin Lougee-Heimer, 'Cbc user guide', in *Emerging theory, methods, and applications*, 257–277, INFORMS, (2005).

[16] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina, 'Plan stability: Replanning versus plan repair', in *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*, pp. 212–221. AAAI, (2006).

[17] Raquel Fuentetaja, Daniel Borrajo, and Tomás de la Rosa, 'Anticipation of goals in automated planning', *AI Communications*, **31**(2), 117–135, (2018).

[18] Malik Ghallab, Dana S. Nau, and Paolo Traverso, *Automated planning - theory and practice*, Elsevier, 2004.

[19] Malte Helmert, 'The fast downward planning system', *J. Artif. Intell. Res.*, **26**, 191–246, (2006).

[20] Jörg Hoffmann and Bernhard Nebel, 'The FF planning system: Fast plan generation through heuristic search', *J. Artif. Intell. Res.*, **14**, 253–302, (2001).

[21] Gal A. Kaminka, Mor Vered, and Noa Agmon, 'Plan recognition in continuous domains', in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 6202–6210. AAAI Press, (2018).

[22] Sven Koenig, Maxim Likhachev, and David Furcy, 'Lifelong planning A*', *Artificial Intelligence*, **155**(1-2), 93–146, (2004).

[23] Jonas Kvarnström, 'Planning for loosely coupled agents using partial order forward-chaining', in *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21, pp. 138–145, (2011).

[24] Peta Masters and Sebastian Sardina, 'Cost-based goal recognition in navigational domains', *Journal of Artificial Intelligence Research*, **64**, 197–242, (2019).

[25] Matthew Molineaux, Matthew Klenk, and David Aha, 'Goal-driven autonomy in a navy strategy simulation', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1548–1554, (2010).

[26] Bernhard Nebel and Jana Koehler, 'Plan reuse versus plan generation: A theoretical and empirical analysis', *Artificial intelligence*, **76**(1-2), 427–454, (1995).

[27] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi, 'Landmark-based approaches for goal recognition as planning', *Artif. Intell.*, **279**, (2020).

[28] Alberto Pozanco and Daniel Borrajo, 'Fairness in multi-agent planning', *arXiv preprint arXiv:2212.00506*, (2022).

[29] Alberto Pozanco, Yolanda E-Martín, Susana Fernández, Daniel Borrajo, et al., 'Counterplanning using goal recognition and landmarks.', in *IJCAI*, pp. 4808–4814, (2018).

[30] Alberto Pozanco, Kassiani Papasotiriou, Daniel Borrajo, and Manuela Veloso, 'Combining heuristic search and linear programming to compute realistic financial plans', in *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pp. 527–531, (2023).

[31] Miquel Ramírez and Hector Geffner, 'Plan recognition as planning', in *Twenty-First international joint conference on artificial intelligence*, (2009).

[32] Miquel Ramírez and Hector Geffner, 'Probabilistic plan recognition using off-the-shelf classical planners', in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, (2010).

[33] Silvia Richter and Matthias Westphal, 'The lama planner: Guiding cost-based anytime planning with landmarks', *Journal of Artificial Intelligence Research*, **39**, 127–177, (2010).

[34] David E Smith, 'Choosing objectives in over-subscription planning.', in *ICAPS*, volume 4, p. 393, (2004).

[35] Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz, and Subbarao Kambhampati, 'Coordination in human-robot teams using mental modeling and plan recognition', in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2957–2962. IEEE, (2014).

[36] Kartik Talamadupula, David E Smith, and Subbarao Kambhampati, 'The metrics matter! on the incompatibility of different flavors of replanning', *arXiv preprint arXiv:1405.2883*, (2014).

[37] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba, 'Cooperative multi-agent planning: A survey', *ACM Computing Surveys (CSUR)*, **50**(6), 1–32, (2017).

[38] Roman Van Der Krogt and Mathijs De Weerdt, 'Plan repair as an extension of planning.', in *ICAPS*, volume 5, pp. 161–170, (2005).

[39] Swaroop Vattam, Matthew Klenk, Matthew Molineaux, and David W Aha, 'Breadth of approaches to goal reasoning: A research survey', Technical report, Naval Research Lab Washington DC, (2013).

[40] Sung Wook Yoon, Alan Fern, and Robert Givan, 'Ff-replan: A baseline for probabilistic planning.', in *ICAPS*, volume 7, pp. 352–359, (2007).