Heuristic approach to guarantee safe solutions in probabilistic planning

Rémi Lacaze-Labadie, Domitile Lourdeaux and Mohamed Sallak
Sorbonne universités, Université de technologie de Compiègne, CNRS,
Heudiasyc UMR 7253, 57 avenue de Landshut – 60203 COMPIEGNE Cedex, France
Email: firstname.lastname@hds.utc.fr

***Abstract*—In this work, we propose a novel approach to solve probabilistic planning problems taking into account the risk that the decision maker is ready to accept regarding the probabilities of reaching the goals. Our approach guarantees that the probability of reaching a state satisfying the selected goals is above a certain limit threshold. To achieve this, we relax the constraints that all goals must be satisfied, and select the most valuable set of goals whose reachability probability is above the threshold. To this end, we propose a goal selection heuristic based on the reachability probability and the cost between goals that are estimated using an abstracted version of the problem. Finally, a planetary exploration problem will be used for illustrating the effectiveness of the proposed approach. Our results show that the obtained selections cover the most valuable possible goals and respect the reachability probability threshold.**

I. Introduction

Automated planning (AP) is an explicit deliberation process that chooses and organizes actions by anticipating their outcomes [1]. Starting from an initial state, the objective is to reach a final state satisfying all the goals of the problem. Probabilistic planning is a category of planning problems where the possible outcomes of actions are not deterministic. We consider such problems defined in the form of a Markov Decision Process (MDP), and which can be efficiently solved with the fixed point of the Bellman equation under the two following assumptions: the first one is that there exists at least one proper policy (a policy that reaches the goals with a probability value equal to 1); and the second one is that all costs are strictly positive so that improper policies with infinite loop accumulate cost infinitely. These two assumptions form the category of problems called Stochastic Shortest Path (SSP) [2]. If these assumptions do not hold, the Bellman equation may have no solution. Teichteil-Königsburg [3] proposes an efficient solution to solve SSP where both assumptions do not hold, which is called Stochastic Safest and Shortest Path Problem (S^3P). In S^3P, goal-oriented MDP are solved by first finding all paths that optimize the probability of reaching the goals and then selecting the one optimizing cost over them. However, even policies maximizing the probability of reaching the goals may not be satisfactory enough in situations in which only poor probability of success exists. In this paper, we adopt a different approach where we consider that in certain critical situations, the decision maker may prefer to reduce the number of satisfied goals in order to have a bounded guarantee on the probability of success. In this approach, we are looking for all solutions whose probability of reaching the goals is above a given threshold, that we will call the *safe threshold*. With this constraint, it is possible that no solution satisfies this *safe threshold*. In such situations, we propose a solution satisfying only a subset of goals with a guarantee to be reached above the *safe threshold*. Indeed, in critical and/or highly uncertain situations, the user may prefer to relax the constraint that all goals must be satisfied in order to have a solution with a better probability of success. In the remaining, a solution above the *safe threshold* will be called a *safe solution*. Since all goals may not be part of a *safe solution*, goal preferences are expressed with a score assigned to each goal. The objective of our algorithm is to find the most valuable subset of goals under the constraint that the solution is safe. This problem belongs to the category of over-constrained problems where all goals cannot be satisfied, usually because of limited resources, or in our case, because of a threshold on the probability of success.

Over-constrained problems have been studied in AP under the name of Partial Satisfaction Planning problems (PSP) [4] and Over-Subscription Planning problems (OSP) [5]. Our work is inspired by algorithms which solve over-constrained problems with the difference that we are not limited by resources of actions (cost, fuel, time, energy,...) but by a threshold on the probability of success. The challenging task of such problems is the goal selection process because a full search in the whole goal states space is unfeasible in practice. Planners have proposed good heuristics to select goals [6], [7], however they are based on limited available resources. To our best of knowledge, no heuristic has been proposed to select goals guaranteeing a minimum probability of success. The basic idea of our proposition is to build a relaxed planning problem from which we can estimate reachability probabilities and costs between goals. This relaxed plan is then transformed into a graph whose nodes represent goals, and edges represent costs and reachability probabilities between goals. The goal selection is done through a search path algorithm in the graph.

The remainder of this paper is organized as follows. In section 2, our motivations and a background of over-constrained problems are described. Then, we formalize the problem and present our heuristic algorithm in Section 3 and 4. Section 5 depicts the empirical results of a case study, followed by a discussion and the work perspectives in Section 6.

II. Motivation and Background

Our work belongs to the category of problems where it may be preferable to not satisfy the whole set of goals in order to limit the risk taken. In this section we give a motivation example, discuss on existing over-constrained problem algorithms in AP, and conclude with the main contributions of this work.

A. Motivation

We illustrate our method, along this paper, using the planetary exploration problem (also known as mars rover problem), where a rover has to navigate a surface, visit different sites of more or less interest and perform tasks such as collecting samples. While this problem has been studied under limited resources (time, energy of the rover) [5], [8], we can notice that in such expensive missions, it is also important to consider the danger taken by the rover when exploring the planet. Indeed, the rover may not move as expected or remain blocked (due to bumps in the landscape or mud) and thus gets lost forever. To highlight this phenomenon, we use a slightly modified version of this planetary exploration problem. The goals are reduced to the exploration of different sites, and each goal have a score which represent the level of interest of the site. We also added the notion of risky area, meaning that every time the rover moves from a risky area, it has a probability to remain blocked (which cause the failure of the mission). To this problem, our method look for solutions where the rover has to visit as many sites as possible with a guarantee threshold that the rover will not be too greedy and remain blocked.

B. Constraint satisfaction problem in planning

The aim of classical planning is to reach a goal state satisfying all goals of the problem. However, in the real world, it is not always possible to satisfy all of the goals, due to logically conflicting goals or limited resources. We say that the problem is over-constrained and a solution can satisfy only a subset of goals. Hence, we are in presence of a problem of planning with constraints where we are looking for the best subset of goals to satisfy. Over-constrained problems in AP are known as OSP first introduced by *Smith* [5] and PSP problems introduced later by *Briel et al.* [4]. While both PSP and OSP have the same problem definition and constraint (the planner cannot achieve all goals), they have different perspectives. OSP focus on scheduling problems and are resource based. PSP problems focus on the best trade-off between goal utility and action cost.

1) Partial satisfaction planning (PSP): PSP Net Benefit [9] is a problem where some utilities/rewards are allocated to goals, and costs to actions. The solution is a plan with the best net benefit, which is the cumulative goal utility minus the cost to satisfy goals. The drawback of these methods is that they assume that action costs and goal utilities are comparable. In our approach we can't make this assumption because the probability of reaching a goal is not comparable with goal utilities.

2) Over-subscription planning (OSP): In OSP, the objective is to maximize the achieved goal utilities limited by the allocated resources/costs. Methods used in OSP are mainly based on a two-steps approach [5], [10]. First, the best subset of goals is heuristically selected and then it is used to guide the planner. The planner is given, one at a time, the selected goals in the order suggested by the selection. It stops when all the goals are reached or when no more resources are available. The challenging task of these problems is the goal selection, because it is not possible to do a search in the whole state space, except in very small problems. To simplify this process, an estimation of needed resources to reach goals is computed from an abstracted version of the problem (which factors out some details of the problem). For example, *Smith* [5] proposes to formalize an abstracted version of the problem as an orienteering problem (OP) whose nodes are goals (attached with their utilities) and edges are costs estimation between goals. The OP is solved by finding the best goal selection which does not exceed available resources. Similarly, *García et al.* [10] builds a distances matrix from a relaxed plan and applies a beam search to select goals. These methods have been applied to deterministic domains but do not handle probabilistic domains.

Meuleau et al. [8] studied OSPs applied to probabilistic domain under the name of Stochastic Over-Subscription Planning (SOSP). The first step of the method proposed by *Meuleau et al.* is based on solving a substantially smaller Markov Decision Process for each goal (that they also call sub-tasks). The second step is to schedule those sub-tasks to solve the global problem. This approach handles probabilistic aspects but the solution is risk-neutral. We define the level of safety of a solution based on the probability of entering a goal state. Since we aim at guaranteeing safe solutions, our method is based on the construction of a graph whose edges are the reachability probability (RP) between goals. It provides an estimation of the degree of safety to reach the different goals and thus to find the best subset of goals whose RP is above the threshold.

C. Our contribution

We propose an approach to guarantee a safe solution (configurable by the user) in probabilistic planning problems by reducing the number of goals to satisfy. We consider this problem as an over-constrained problem limited by the risk the user is ready to take, in the form of a probability threshold. We were inspired by OSP problems with the difference that the goal selection is based on a reachability probability estimation and that the problem is limited by the risk of the solution instead of the resources. Our contribution includes:

1) A method estimating the RP between goals to provide a good sub-optimal goal selection quickly, instead of doing a full search.
2) An algorithm that guarantees a safe solution with regard to a configurable probability threshold.

III. The goal selection heuristic

The goal selection heuristic proposed herein will be used to guide the planner. It is a three-step process resulting in the best selection of goals which can be satisfied by a safe solution. In other words, the selection of goals accumulating the highest total score, and whose reachability probability is higher or equal to the *safe threshold*. Here are the three steps:

1) Creation of a relaxed version of the problem that reduces state space, making a possible detection of goal states.
2) Estimation of the reachability probability and the cost between goal states with a value iteration based algorithm applied to the relaxed problem.
3) Search of the best goal selection in a graph modeling goal states and previously computed estimations.

A. Relaxed problem

In practice, a full state space search to find the best goal selection is impossible. Indeed, the number of possible states satisfying goals can quickly become intractably large. To overcome this issue, approximation techniques use a state space abstraction of the problem [11], [12]. Usually, in OSP problems, the abstracted or relaxed version results in a graph structure where nodes are goal states labeled with a score and edges are resources or dependencies between goal states [5], [10]. The goal selection results in a search path problem where the path has to maximize the sum of node scores with limited capacity (resources attached to edges). The relaxation method we use in our heuristic is very close to the ones used by *Smith* [5] or *Benton et al.* or [6]. It is efficient under the condition that goals are sufficiently independent, which is the case in problems such as the planetary exploration involving a series of sub-tasks to realize. The method is based on the identification of interactions and shared variables between goals. It forms a state space abstraction which allows one to detect goal states in a reasonable time. In the relaxed problem, each goal state is associated to a goal and its score of the original problem. We do not cover more in details the relaxation process as it not the purpose of this article.

1) Relaxed plan definition: The relaxed planning problem is formalized as a Markov Decision Process (MDP) where the list of goals G becomes a list of identified goal states in the relaxed problem. For example, in our modified version of the planetary exploration problem presented in section II-A, the goal states are the coordinates of sites to visit. We use an MDP because it adequately models uncertainties of our problem. The relaxed problem results in an undiscounted MDP in the form $\langle S, A, S_G, p, c, u, \alpha, s_0 \rangle$ where:

- S is a finite set of states ($s_0 \in S$ is the initial state).
- A is a finite set of actions. The set of all actions applicable in a state s will be noted $A\left(s\right)$.
- $S_G \subseteq S$ is a list of goal states ($s_g \in S_G$ denotes a goal state).
- $u(s_g)$ is the utility/score of the goal state s_g ($u(s_g) > 0$).
- $p(s, a, s') \in [0, 1]$ is a transition function which gives the probability of moving from state s to state s' applying action a.
- $c(s, a) > 0$ is the cost of applying action a in state s.
- $\alpha \in [0, 1]$ is the *safe threshold* and $\beta = 1 - \alpha$ is the *risk threshold.*

B. Reachability probability and cost function

1) Background on value function computation: A *value function* is a function which links any state s to the value of a criterion (e.g. cost, reward...). This value corresponds to the expected criterion when applying the policy starting from s. In the context of MDPs, we compute the value function using *dynamic programming (DP)* [13], which refers to a collection of algorithms used when the dynamic of the environment is known (i.e. transition probability and cost). The idea is to approximate the value function by iterative refinement leading to an estimation increasingly accurate. The optimal value function V is the unique solution of the Bellman equation [14]:

$$V\left(s\right) = \min_{a\ \in\ A(s)} Q\left(s, a\right) \tag{1}$$

$$Q\left(s, a\right) = \ c\left(a, s\right) + \sum_{s'} p(s, a, s') V(s')$$

In this paper, we will use a value iteration algorithm, which is considered as a standard DP method, to compute value functions.

2) Reachability probability and cost computation: For our heuristic, we define two value functions, one for the reachability probability that we note RP and one for the cost that we note V. Both functions are estimated for each identified goal states of the relaxed planning problem. It results in two vectors of value functions, each one of the size of the number of goal states. This decomposition allows one to find the best subset of goals by selecting goals one by one until the product of the reachability probabilities between goals goes below the *safe threshold* α. While the computation of the cost function is a direct application of the Bellman equation, the reachability probability function is estimated as a variant of the Bellman equation where the value (the reachability probability) is equal to 1 when entering a goal state and 0 otherwise. Similarly to *Steinmetz et al.* [15], we define $RP(s, s_g)$ the probability of entering goal state s_g from state s and $\sigma(s, a, s_g)$ the probability of entering goal state s_g from state s applying action a as follows:

$$RP(s, s_g) = \max_{a\ \in A(s)} \sigma(s, a, s_g) \tag{2}$$

$$\sigma(s, a, s_g) = \ \sum_{s'} p(s, a, s') RP(s', s_g) \tag{3}$$

We also define the cost function $V(s, s_g)$ for each goal state in the form of the Bellman equation (cf. Eq.1) by considering that the selected action maximize the reachability probability instead of minimizing the cost. Thus, we update at the same time V and RP with the action of maximal reachability probability (i.e. the action maximizing σ). The cost function

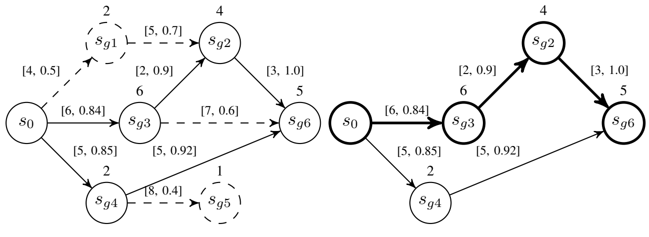

Fig. 1. Example of $SGGraph$ with $\alpha = 0.75$ and edges in the form $[V, RP]$. All dashed elements are removed because they can't be part of a safe path. The most valuable safe path is $(s_0, s_{g3}, s_{g2}, s_{g6})$.

becomes an estimation of the expected future cost when the agent acts with maximal reachability probability.

$$V(s, s_g) = Q(s, a, s_g) \text{ s.t. } a = \underset{a\, \in\, A(s)}{\text{argmax}} \ \sigma\left(s, a, s_g\right) \tag{4}$$

$$Q(s, a, s_g) = \ c\left(a, s\right) + \sum_{s'} p(s, a, s') V(s', s_g) \tag{5}$$

To ensure the Bellman equation has a solution, we assume that there exists at least one policy with a non-zero probability of termination and that all costs are strictly positive (all improper policies accumulate infinite cost). Once both value functions are solved, we have the reachability probability and the cost to go from one goal state s_{g1} to another goal state s_{g2} with $RP(s_{g1}, s_{g2})$ and $V(s_{g1}, s_{g2})$.

C. Goal selection

The goal selection process is the process of choosing and ordering the best subset of goals which will be used to guide the planner. In [5], [8], [10], the selection is done by solving an orienteering problem (OP). The OP [16], [17] is a graph whose nodes are represented with a score and whose aim is to determine a Hamiltonian path maximizing the sum of scores without exceeding a given limit. In OSP problems, the resulting path of solving the OP corresponds to the best goal selection. Using the same technique, we search for a path maximizing the sum of scores and whose product of reachability probabilities is above the *safe threshold* α. We define a directed graph called $SGGraph$ whose nodes are the initial state and goal states of the relaxed plan and edges are labeled with the reachability probabilities and the costs between goal states. An example of an $SGGraph$ is given in Fig. 1. More formally we define $SGGraph = (S, E)$ with S the set of nodes labeled with a score, being either the initial state or a goal state such that $S = S_g \cup \{s_0\}$ and E the edges labeled with the cost and the reachability probability between two nodes of the form: $\left[V\left(s_1, s_2\right),\ RP\left(s_1, s_2\right)\right]$ s.t. $s_1, s_2 \in S$. We compute $RP\left(path\right)$ as the probability of reaching all goal states of a path. It is the product of RP from the initial state to the first goal state, and then from goal state to goal state in the order given in the path. Note that a path must start at s_0.

$$path = (s_0, s_{g0}, s_{g1}, \ldots, s_{gn}) \ \text{ s.t. } s_{gi} \in S_g$$

$$RP\left(path\right) = RP\left(s_0, s_{g0}\right) \times \prod_{i=0}^{n-1} RP\left(s_{gi}, s_{gi+1}\right) \tag{6}$$

A path is safe if its reachability probability is above or equal to the *safe threshold* α:

$$RP\left(path\right) \geq \alpha \tag{7}$$

We also define the utility of a path $U(path)$ as the total utility of the path and the cost of path $C(path)$ as the total cost to go from goal state to goal state.

$$U\left(path\right) = \sum_{i=0}^{n} u(s_{gi}) \tag{8}$$

$$C\left(path\right) = V\left(s_0, s_{g0}\right) + \sum_{i=0}^{n-1} V\left(s_{gi}, s_{gi+1}\right) \tag{9}$$

The optimal path we are looking for is a safe path maximizing U as a first criterion, and minimizing C as a second criterion. We use a second criterion because several safe paths may have the same maximum utility. It is mostly the case when several paths are composed of the same goals in different orders. In this situation, considering the cost as a secondary criterion plays an important role because it enables to choose the less costly path. Finally, the goal selection is the resulting path without the initial state. The method used to solve the $SGGraph$ is given in section IV. We call $list\_g$, the goal selection resulting of the $SGGraph$:

$$list\_g = (s_{g0}, s_{g1}, \ldots, s_{gn}) \ \text{ s.t. } s_{gi} \in S_g$$

$$list\_g = \ SGGraph(s_0) \tag{10}$$

D. A small example

Here we illustrate the use of the proposed goal selection heuristic by applying it to the modified version of the planetary exploration problem that we presented in section II-A.

1) Relaxed problem: The relaxed problem results in states composed of the rover's position (x,y) and actions of moving to the 4 cardinal directions. This relaxed problem is similar to a grid world problem.

2) Reachability probability and cost estimation: In Fig. 1, s_0 is the initial position of the rover and $s_{gi}(i \in 1, 2, ..., 6)$ are the goal states associated with their scores, that is to say the sites to visit associated with their levels of interest. Each edge is labeled with a cost representing how far a site is from another one and a reachability probability being the probability to reach a site from another site.

3) Goal selection: In Fig. 1, there are two admissible paths: $path\_a = (s_{g3}, s_{g2}, s_{g6})$ and $path\_b = (s_{g4}, s_{g6})$. Other paths are not admissible because they have either one or more segments whose RP is lower than the safe threshold α or because they do not start by s_0. Using Eq.6 and Eq.8, we get:

- $RP(path\_a) = 0.84 \times 0.9 \times 1 = 0.756$ and $U(path\_a) = 6 + 4 + 5 = 15$.
- $RP(path\_b) = 0.85 \times 0.92 = 0.782$ and $U(path\_b) = 2 + 5 = 7$.

When fixing $\alpha = 0.75$, both $path\_a$ and $path\_b$ are safe paths ($RP(path\_a) \geq \alpha$ and $RP(path\_b) \geq \alpha$). However, the selected path is $path\_a$ because $U(path\_a) > U(path\_b)$.

```
1:  function RPValueIteration
2:      InitializeStates()                                   ▷ see Eq.11
3:      repeat
4:          $\Delta_{RP} = 0$
5:          for each $s \in \mathcal{S}$ do
6:              for each $s_g \in \mathcal{S}_G$ do          ▷ for all goal states
7:                  $safest\_a = \mathrm{argmax}_{a \in A(s)}\ \sigma(s, a, s_g)$
8:                  $V(s, s_g) = Q(s, safest\_a, s_g)$
9:                  $RP(s, s_g) = \sigma(s, safest\_a, s_g)$
10:                 $\Delta_V = \max(\Delta_V, R_V(s, s_g))$
11:                 $\Delta_{RP} = \max(\Delta_{RP}, R_{RP}(s, s_g))$
12:     until $\Delta_V < \epsilon$ & $\Delta_{RP} < \delta$          ▷ see Eq.12
```

Fig. 2. Relaxed Plan Value Iteration

Note that the cost is not used here for simplicity of the example but would be used if both paths had the same utility. We finally get the best goal selection from $path\_a$ that is $list\_g = (s_{g3}, s_{g2}, s_{g6})$.

IV. Algorithms

In this section, we first present the VI based algorithm used to estimate reachability probability and cost from the relaxed plan. We then detail our method to solve the $SGGraph$ and finish with the general solution of the planner.

A. Relaxed plan value iteration algorithm

1) Initialization: The cost function V is initialized for each state arbitrarily with a positive value. The reachability probability RP is initialized to 1 when entering a goal state and 0 otherwise. More formally, we have:

$$\forall s_i, s_j \in S \text{ and } \forall s_g \in S_G,$$
$$V(s_i, s_j) \geq 0,\ RP(s_i, s_g) = 1\ else\ \ RP(s_i, s_j) = 0 \quad (11)$$

2) Terminal condition: VI algorithm stops when the Bellman error (residual) over all states is sufficiently small. Thus, we stop our algorithm when residual of Eq.2 and Eq.4 are small enough for all s and s_g:

$$R_V(s, s_g) = |V(s, s_g) - V'(s, s_g)|$$
$$R_{RP}(s, s_g) = |RP(s, s_g) - RP'(s, s_g)|$$
$$\forall s \in S,\ \forall s_g \in S_G,\ R_V(s, s_g) <\ \epsilon, R_{RP}(s, s_g) <\ \delta \quad (12)$$

3) Update: In the algorithm presented Fig. 2, the value functions are updated repeatedly using Eq.2 and Eq.4 until the maximum residual of all updated values is small enough. Because the agent takes a safe attitude, both functions are updated with the safest action, that is to say the action maximizing the probability of reaching a goal state (line 7). In lines 8 and 9, we update V and RP with this action and store the maximum residual in lines 10 and 11. The algorithm stops when Δ_V and Δ_{RP} are small enough for an iteration. The values of ϵ and δ are discussed with our tests at section V. In the algorithm, we used the notation σ and Q for space-saving consideration, but it has to be replaced by the right part of Eq.3 for σ and Eq.5 for Q.

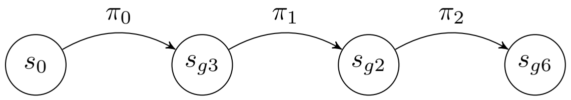

Fig. 3. The global policy $\pi = (\pi_0, \pi_1, \pi_2)$ is the aggregated policies corresponding to the selected subset of goals.

B. Solving SGGraph

The goal selection process consists in finding the optimal path of $SGGraph$ without loop (see section III-C for the definition of an optimal path). At first, we simplify the graph by removing all edges whose reachability probability is lower than the safe threshold α and all nodes that are not attached to s_0. For example in Fig. 1, all the dashed edges and nodes are removed. Indeed, these elements cannot be part of any safe path which has to start by s_0 and satisfy Eq.7. Once the graph is reduced, the next step is to search in the graph the optimal path. Finding the optimal path in a directed cyclic graph is exponential in time, so approximate methods have to be used. We apply a beam search algorithm which provides a good solution in a reasonable time, and has been used by others for the same problem [5], [10]. The beam search uses an inadmissible pruning rule to keep only the most promising nodes at each level of the search. For each level, we order all successors using a heuristic, and expand only k successors (where k denotes the beam value). The optimal solution is not guaranteed as the search is not complete, except if the beam is large enough. However, by using a good pruning heuristic, we get a good sub-optimal solution quickly. The heuristic that we have chosen gives priority to the nodes having the better immediate reachability probability (i.e., the probability from the current node to the node to expand). We use a depth-first search and avoid repeating nodes due to possible cycles, by marking them when they are visited in the recursion and removing the marks just before returning from the recursive call. For each new safe path found, we compare U (Eq.8) and if necessary C (Eq.9) to the current optimal path and store the best one.

C. Planning the selected goals

Because of the probabilistic context, we cannot simply plan each goal of the selection one by one. We have to find policies for each goal of the selection, which will be used one at a time. The idea is to plan each goal state s_{gi} of the selection as an independent sub-problem and to aggregate all solutions (cf. Fig. 3). All sub-problems have the same model definition as the original problem, that is to say the same actions, states, transition probabilities and costs of actions, but differ in the initial state and the goals. The initial state of the sub-problem solving s_{gi} is the goal state s_{gi-1} of the previous sub-problem (or s_0 of the original problem for $i = 0$). The goals G_i of the sub-problem solving s_{gi} is the set of all predicates contained in s_{gi}. For example, if a goal state of the relaxed problem is the position of the rover, then G_i will be composed of two goals: the coordinate x and y of the rover. With the same

VI algorithm presented in our heuristic, we solve each subproblem considering two value functions : 1) $V_i(s)$ the cost of reach reaching G_i and 2) $RP_i(s)$ the probability of reaching G_i. We then compute the policy π_i considering the agent acts with maximal reachability probability first, and minimal cost if some actions have the same level of reachability probability. To do so, we have to define $SA(s)$ the set of safest applicable actions, that is to say actions whose probability of reaching G_i is the highest. We finally choose the action minimizing cost among all safest actions.

$$SA\,(s) = \underset{a\,\in\,A(s)}{\operatorname{argmax}} \{\sum_{s'} p(s,a,s')RP_i(s')\}$$

$$\pi_i\,(s) = \underset{a\,\in\,SA(s)}{\operatorname{argmin}}\;\{c\,(a,s) + \sum_{s'} p(s,a,s')V_i(s')\} \qquad (13)$$

The solution of the problem is $\pi(s) = (\pi_0, \pi_1, \ldots, \pi_n)$ where each policy is used one at a time to reach each goal in the order of the goal selection. Technically, we maintain the already reached goals at the execution time so we know which of the π_i to call.

V. Numerical results

A. Case study

In this section, we present the numerical results obtained by applying our heuristic to our modified version of the planetary exploration problem presented in section II-A. It is similar to a grid world problem where the grid represents the landscape of the planet. This problem is general enough to illustrate our proposed heuristic. Indeed, the list of goals is composed of a list of cells to reach on the grid, which corresponds to the sites of the planet to visit. Each goal has a utility value, which is the level of interest of the site. We model variation on the reachability probability with two distinct types of cell (safe and risky). When the rover enters a risky cell, it has a probability to remain blocked, and thus cannot reach anymore a goal state. The rover can perform four actions : go north, go east, go south, and go west. In a safe cell, an action $a \in A$ takes the rover in a given direction with a probability of success equal to p. If the action fails, the rover can be transported to each of the three remaining directions with a probability equal to $(1-p)/3$. In a risky cell, the rover has a probability equal to q to remain blocked, and $1-q$ to move. The objective of this problem is to reach the most valuable possible cells, with a probability above the *safe threshold*. A simulation is a success if the execution of the solution reaches all the cells selected by the goal selection. We first evaluate the quality of solutions obtained by heuristically selecting goals for different thresholds, and we then compute some tests performance.

1) Selection of terminal condition parameters: Remember that ϵ and δ are used to control when we stop the computation of the value functions and thus the accuracy of the estimation. We measured on several normal and extreme test cases, how sensitive were our results for values going from 0.1 to 0.0001 for both parameters. We identified that, for each test case, it converges to the same result starting from 0.01 for both ϵ and δ. We then used a value of 0.01 for all of our tests.

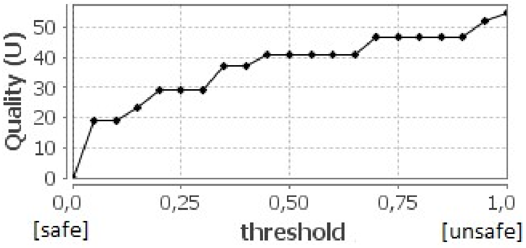

Fig. 4. Goal selection test: the quality (U) is the total goal utility.

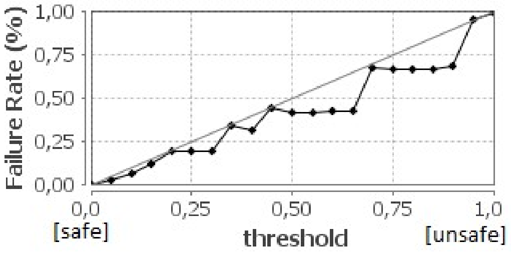

Fig. 5. Goal selection test: the failure rate is the rate of simulations not reaching all goals of the goal selection.

B. Results

In Fig. 4 and Fig. 5, we used a grid area composed of 25×25 cells, and we consider different thresholds $\beta \in [0,1]$ (from safe to unsafe). Note that we use the *risk threshold* β instead of the *safe threshold* α for simplicity of reasoning, because it represents a risk limit to no exceed. We used a problem involving 10 sites to visit (10 goals) randomly generated, and 50% of risky cells randomly selected with a probability $q = 0.1$ to remain blocked when moving. We first test the goal selection process for different thresholds β, and we then simulate each goal selection in order to verify the respect of the threshold. In Fig. 4, the x-axis represents the *risk threshold* [β], and the y-axis represents the quality denoted U that is obtained by summing all the goals utilities. We can see that the quality is monotonically increasing when β is increasing. Indeed, when we tend to an unsafe attitude, we can reach more valuable set of goals. However, in some cases (for example $\beta \in [0.45, 0.65]$), the quality remains constant, meaning that no better subset of goals is reachable even with a slightly higher value of β. In Fig. 5, we simulate each previously computed goal selection to test if the threshold is respected. For each goal selection, we simulate 1000 times our planner guided by this selection, and we compute the average failure rate defined as the percent of the simulations not reaching all the goals of the selection divided by the total number of simulations. We also tested some results with 2000 simulations and got the same precision as with 1000 simulations. As expected, the failure rates never exceed the threshold β (the diagonal line). However, there is sometimes a quite important gap between the produced solution and the threshold. This is due to the fact that the planner detects that no less costly solution can satisfy the goal selection, so the planner maintains the reachability probability as high as possible.

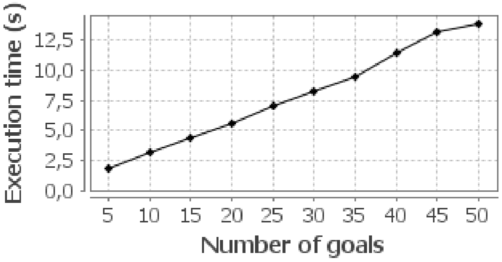

Fig. 6. Performance test: variation of the number of goals with a fixed state-space (grid size: 25 × 25).

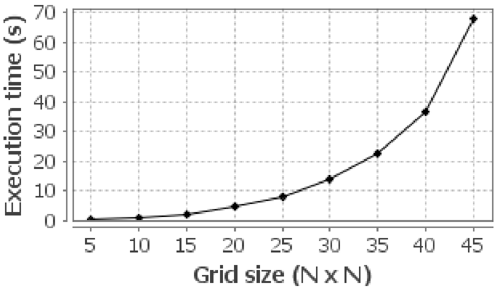

Fig. 7. Performance test: variation of state-space size with a fixed number of goals (8 goals).

C. Performances

The execution time in Fig. 6 and Fig. 7 is in seconds. This execution time is actually very close to the execution time of the algorithm estimating value functions (Fig. 2). It is because the time of building the $SGGraph$ and finding the best path is less than 2% of the total time with an efficient beam search. We tested several beam widths with a reasonable number of goals to be able to compare the beam solution with the exact optimal solution provided by a full search. We noticed that on average, the algorithm converges to a unique solution with a beam width equal to $0.25\times$ the number of goals. Using this beam width with an efficient heuristic, the search in the graph takes always a time duration lower than 250 ms for all the tested state space sizes and number of goals (until 50 goals or until a state space size of 45×45 states).

VI. Conclusion

Many real probabilistic problems produce unsafe solutions which can be considered as critical. It is for example the case in problems such as NASA problems where a rover or any other equipment is very expensive. In this work, we have described a heuristic to find solutions satisfying the most valuable subset of goals under a configurable limited amount of risk in the form of a threshold. A method for goal selection has been proposed to select goals which can be reached with a certain probability of success. The goal selection is based on a value iteration algorithm which provides an estimation of cost and reachability probability from a relaxed plan of the problem.

Our work has created new problems: a reasonable next step is to work out how safe thresholds can themselves be chosen by the planners. An idea would be to based the choice of the threshold on Hurwitz criterion, which is a balanced choice between safe and unsafe attitudes. Moreover, VI takes much time in large state spaces as it needs to loop over the whole state space. We aim to apply our goal selection method to anytime behavior algorithm such as RTDP or L-RTPD which can converge much faster to some near optimal solutions. Finally, we aim to apply the proposed heuristic to the planning problems considering the formation of agents in virtual environment in presence of some critical situations.

Acknowledgment

This work was carried out in the context of the VICTEAMS project. The VICTEAMS project (ANR-14-CE24-0027) has received funding from French National Research Agency (ANR), the French Defence Procurement Agency (DGA) and is labelled by the Labex MS2T. The authors would like to thank the Picardie region and the European Regional Development Fund (ERDF) 2014/2020 for the funding of this work.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting*. Cambridge University Press, 2016.

[2] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific., 1995.

[3] F. Teichteil-Königsbuch, "Stochastic Safest and Shortest Path Problems." *Association for the Advancement of Artificial Intelligence*, pp. 1825–1831, 2012.

[4] M. van den Briel, R. Sanchez, and S. Kambhampati, "Over-subscription in Planning: A Partial Satisfaction Problem," *ICAPS 2004 Workshop on Integrating Planning into Scheduling*, 2004.

[5] D. Smith, "Choosing Objectives in Over-Subscription Planning." *International Conference on Automated Planning and Scheduling*, pp. 393–401, 2004.

[6] J. Benton, M. Do, and S. Kambhampati, "Anytime heuristic search for partial satisfaction planning," *Artificial Intelligence*, vol. 173, pp. 562–592, 2009.

[7] L. Kramer and S. Smith, "Maximizing Flexibility: A Retraction Heuristic for Oversubscribed Scheduling Problems," *International Conference on Automated Planning and Scheduling*, 2005.

[8] N. Meuleau, R. Brafman, and E. Benazera, "Stochastic Over-Subscription Planning Using Hierarchies of MDPs," *International Conference on Automated Planning and Scheduling*, pp. 121–130, 2006.

[9] R. Sanchez-Nigenda and S. Kambhampati, "Planning Graph Heuristics for Selecting Objectives in Over-subscription Planning Problems," in *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 2005, pp. 192–201.

[10] A. García-Olaya, T. De La Rosa, and D. Borrajo, "Using the relaxed plan heuristic to select goals in oversubscription planning problems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7023 LNAI, 2011, pp. 183–192.

[11] M. Katz and D. Carmel, "Implicit abstraction heuristics," *Journal of Artificial Intelligence Research*, vol. 39, pp. 51–126, 2010.

[12] M. Helmert, P. Haslum, J. Hoffmann, and R. Nissim, "Merge-and-Shrink Abstraction : A Method for Generating Lower Bounds in Factored State Spaces," *Journal of the ACM*, vol. 61, no. 3, p. 63, 2014.

[13] D. Bertsekas, "Dynamic Programming and Optimal Control 3rd Edition, Volume II Chapter 6 Approximate Dynamic Programming 6 Approximate Dynamic Programming," 2011.

[14] R. Bellman, "Dynamic Programming," *Princeton University Press*, 1957.

[15] M. Steinmetz, J. Hoffmann, and O. Buffet, "Revisiting Goal Probability Analysis in Probabilistic Planning," *International Conference on Automated Planning and Scheduling*, no. 0, pp. 299–307, 2016.

[16] B. Golden, L. Levy, and R. Vohra, "The Orienteering Problem," 1987.

[17] P. Vansteenwegen, W. Souffriau, and D. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, pp. 1–10, 2011.