Planification probabiliste : une heuristique pour garantir des solutions sûres

Rémi Lacaze-Labadie, Domitile Lourdeaux, Mohamed Sallak

Planification probabiliste : une heuristique pour garantir des solutions sûres

Rémi Lacaze-Labadie[1] Domitile Lourdeaux[1] Mohamed Sallak[1]

[1] Sorbonne universités, Université de technologie de Compiègne, CNRS,
Heudiasyc UMR 7253, 57 avenue de Landshut – 60203 COMPIEGNE Cedex, France

{remi.lacaze-labadie - domitile.lourdeaux - mohamed.sallak}@hds.utc.fr

Résumé

Dans ce papier, nous proposons une nouvelle approche pour résoudre les problèmes de planification probabiliste en prenant en compte le risque que l'utilisateur est prêt à prendre vis à vis de la probabilité d'atteindre les buts. Notre approche garantit le fait que la probabilité d'atteindre un état satisfaisant les buts du problème sera toujours supérieure à un seuil de probabilité donné. Toutefois, un tel seuil ne permet pas toujours de tous les atteindre. C'est pourquoi, notre heuristique estime la probabilité d'atteignabilité de chaque but via une version relaxée du problème, puis sélectionne le sous-ensemble de buts atteignables avec une probabilité supérieure à ce seuil. Nous illustrons nos travaux par un problème d'exploration planétaire couramment utilisé dans ce contexte. Nos résultats de simulation montrent que l'heuristique proposée sélectionne les meilleurs buts et que le seuil de probabilité de les atteindre est toujours respecté.

Mots Clef

Planification dans l'incertain, heuristique, théorie de la décision.

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 to reach a state satisfying selected goals is above a certain limit threshold. Indeed, 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.

Keywords

Planning under uncertainty, heuristic search, decision theory.

1 Introduction

La planification est une branche de l'intelligence artificielle qui consiste à sélectionner et à ordonnancer des actions afin d'atteindre un objectif [6], c'est à dire un état satisfaisant tous les buts du problème. En particulier, nous nous intéresserons à la planification probabiliste, dont les effets des actions peuvent être non déterministes. Dans ce papier, nous définirons un objectif comme l'ensemble des buts d'un problème, c'est à dire l'ensemble des prédicats devant être satisfait par la solution. Nous formaliserons ces problèmes avec des processus de décision markovien (ou Markov Decision Process en anglais, MDP). Les MDPs peuvent être efficacement résolus à l'aide de la solution unique de l'équation de Bellman sous deux hypothèses : il existe au moins une politique qui atteint tous les buts avec une probabilité de 1 et tous les coûts des actions sont strictement positifs de sorte que toutes les politiques contenant une boucle infinie accumulent un coût infini. Les MDPs formulés sous ces hypothèses sont plus connus sous le nom de problèmes de Plus Court Chemin Stochastique (SSP en anglais). Ces hypothèses garantissent d'avoir toujours une solution à l'équation de Bellman. Cependant, elles sont contraignantes et ne permettent pas de modéliser tous les types de problèmes. C'est par exemple le cas pour les problèmes qui n'ont pas de politique permettant d'atteindre avec une probabilité de 1 tous les buts. Dans ce cas, nous allons chercher à maximiser cette probabilité. Des travaux ont déjà été menés sur le sujet, comme par exemple Teichteil-Königsburg [18] qui propose une méthode efficace pour trouver le chemin le plus probable pour atteindre les buts. Cette méthode est connue dans la littérature sous le nom de Stochastic Safest and Shortest Path (S^3P) et résout les problèmes en recherchant d'abord tous les chemins optimisant la probabilité d'atteindre les buts, puis en sélectionnant celui minimisant le coût. Cependant cette méthode ne garantit pas une probabilité minimale d'atteindre les buts. L'objectif de ce papier est de proposer une heuristique sélectionnant un sous-ensemble de buts atteignable

avec un seuil de probabilité minimal choisit par le décideur. Nous positionnons nos travaux dans la catégorie des problèmes sur-contraints (*over-constrained problems* en anglais), dans lesquels les contraintes sont le manque de ressource (essence, énergie,...) empêchant tous les buts d'être atteints. En effet, nous considérons le risque, soit la probabilité de ne pas atteindre un but, comme une ressource limitant le nombre de buts atteignables. Dans cet article, nous ferons l'hypothèse que le décideur, en cas de situations à risque, préfère satisfaire moins de buts, avec cependant de meilleures chances qu'ils soient atteints. Pour ce faire, nous proposons de définir un seuil de probabilité d'atteindre les buts, que nous définirons dans le reste de l'article comme le *seuil de sûreté*. Nous proposons alors une méthode qui sélectionne ce que nous appellerons une *solution sûre* du point de vue du décideur, c'est-à-dire toutes solutions satisfaisant ce *seuil de sûreté*.

Les problèmes sur-contraints ont été étudiés dans le domaine de la planification sous le nom anglais de *Partial Satisfaction Planning problems (PSP)* [19] et *Over-Subscription Planning problems (OSP)* [14]. Le challenge de ce type de problèmes est la sélection du sous-ensemble de buts satisfaisant les contraintes. En effet, une recherche complète dans l'ensemble de l'espace d'état (pour trouver les meilleurs ensembles de buts) est impossible en pratique. De bonnes heuristiques ont alors été proposées pour sélectionner les buts [2, 10], cependant elles sont toutes basées sur des contraintes relatives aux limites de ressources. À notre connaissance, aucune heuristique ne propose une sélection des buts garantissant une probabilité de les atteindre au-dessus d'un seuil donné. L'idée de base de notre algorithme est de construire une version relaxée du problème afin d'estimer le coût entre les buts ainsi que la probabilité d'atteindre un but depuis un autre but. Puis, à partir de cette version relaxée du problème, un graphe est créé dont les nœuds représentent les buts et les arcs les coûts et probabilités. Enfin, la sélection des meilleurs buts se fait par un algorithme de recherche dans ce graphe.

Ce papier est organisé de la manière suivante. Dans la section 2, nous présentons nos motivations ainsi qu'un état de l'art sur les problèmes sur-contraints en planification. Puis nous formalisons notre problème et proposons notre heuristique dans les sections 3 et 4. La section 5 décrit nos résultats empiriques sur un cas d'étude, avant de conclure par nos perceptives à la section 6.

2 Motivation et contexte

Nos travaux s'inscrivent dans une catégorie de problèmes dans laquelle le décideur préfère satisfaire moins de buts si cela augmente la probabilité qu'ils se produisent. Dans cette section, nous donnons nos motivations à travers un exemple, discutons des problèmes de satisfaction de contraintes en planification puis concluons avec nos contributions dans ce domaine.

2.1 Motivation

Nous illustrons notre méthode tout au long de ce papier par un problème d'exploration planétaire (plus connu sous le nom de *mars rover*), dans lequel un robot doit naviguer sur mars, visiter différents points d'intérêts et effectuer des opérations telles que la collecte d'échantillons. Bien que ce problème ait été étudié dans le cas de ressources limités (e.g. énergie) [14, 11], nous constatons que dans le cadre de missions qui peuvent s'avérer coûteuses, il est également important de considérer la notion de risque. En effet, le robot pourrait par exemple rester bloqué par un cratère, ce qui causerait sa perte. Pour mettre en évidence ces aspects, nous modifions légèrement ce problème en restreignant les buts à l'exploration de différents points d'intérêts, et en ajoutant la notion de risque sur les déplacements du robot. C'est à dire qu'à chaque déplacement, en fonction de la présence de dangers sur le sol, le robot a une probabilité de rester bloqué (ce qui provoque un échec de la mission). L'objectif de notre méthode sur ce problème est de trouver les meilleurs points d'intérêts à visiter, tout en garantissant que le robot ne prendra pas trop de risques et ne restera pas bloqué.

2.2 Problème de satisfaction de contraintes en planification

L'objectif de la planification classique est d'atteindre un état satisfaisant tous les buts du problème. Cependant, il n'est pas toujours possible de tous les atteindre, soit pour des raisons de conflits logiques entre les buts, soit par manque de ressources. On dit alors que le problème est sur-contraint. Dans ce cas, une solution peut seulement satisfaire un sous-ensemble de buts. On se retrouve alors dans un problème de planification sous contraintes dans lequel on cherche le meilleur sous-ensemble de buts satisfaisant les contraintes. Ces problèmes sont connus sous les noms de OSP, premièrement introduit par *Smith* [14] et de PSP introduit plus tard par *Briel et al.* [19]. Bien que ces deux types de problème ont les mêmes définitions et contraintes (l'ensemble des buts n'est pas atteignable), ils ont des perspectives différentes.

Partial satisfaction planning (PSP). Le *PSP Net Benefit* [13] est un problème dans lequel des scores (préférences) sont attribués aux différents buts, et des coûts aux actions. La solution recherchée est un plan avec le meilleur bénéfice net, c'est-à-dire le cumul des scores moins le coût pour atteindre les buts. L'inconvénient de cette approche est de supposer que les scores et les coûts sont comparables et donc du même ordre. Cependant, dans notre approche la probabilité d'atteindre un but n'est pas comparable avec un score, c'est pourquoi nous nous sommes tournés vers les OSPs.

Over-subscription planning (OSP). Dans les problèmes OSPs, l'objectif est de maximiser le score des buts atteints tout en respectant la limite imposée sur les ressources/coûts disponibles. Les méthodes utilisées pour ce

type de problèmes s'appuient souvent sur une approche en deux étapes [14, 4]. La première est de sélectionner, via une heuristique, un sous-ensemble de buts satisfaisant les contraintes du problème. La seconde étape est d'utiliser cette sélection pour guider le planificateur. Les buts sont donnés un par un au planificateur, qui planifie chaque but et s'arrête quand ils sont tous atteints ou que les ressources sont épuisées. La difficulté de cette approche est la sélection des meilleurs buts, car il n'est pas possible de faire une recherche complète dans tout l'espace d'état. Pour simplifier ce processus, nous réalisons une estimation des ressources utilisées pour atteindre les buts à l'aide d'une version relaxée du problème. Par exemple, Smith [14] propose une abstraction du problème sous la forme d'un Orienteering Problem (OP), qui est un graphe orienté dans lequel chaque nœud est un but (associé d'un score), et chaque arc l'estimation du coût entre les deux buts qu'il relie. L'OP est ensuite résolu par la recherche d'un chemin maximisant les scores sans dépasser la limite imposée sur les coûts. Similairement, *García et al.* [4] construisent une matrice de distance depuis une version relaxée du problème, et applique ensuite un algorithme de recherche en faisceau. Ces méthodes ont été appliquées sur des domaines déterministes, mais ne sont pas adaptées aux domaines probabilistes.

Dans les travaux de *Meuleau et al.* [11], le cas probabiliste a été étudié sous le nom de *Stochastic Over-Subscription Planning* (SOSP). La première étape de leur méthode est la résolution d'un MDP pour chaque but du problème (également appelé sous-tâche). Ces MDPs sont beaucoup plus petits et donc plus simple à résoudre que le problème initial. La seconde étape consiste à planifier ces sous-tâches une par une pour former la solution complète du problème. Bien que cette approche est adaptée aux domaines probabilistes, elle adopte une attitude neutre (*risk-neutral attitude*). A l'inverse, mais sur le même principe que *Geibel* [5], qui définit le risque d'une solution comme la probabilité d'entrer dans un état fatal ou non désiré, nous définissons le degré de sûreté d'une solution par sa probabilité d'entrer dans un état satisfaisant les buts sélectionnés par l'heuristique. Dans l'objectif de garantir une *solution sûre*, notre méthode de sélection des buts repose sur la construction d'un graphe dont chaque arc représente la probabilité d'atteindre un but depuis un autre. A l'aide de ce graphe, les buts sont sélectionnés un par un en respectant que la multiplication des probabilités reste au-dessus du *seuil de sûreté*.

2.3 Contribution

Comme vu dans la section précédente, ce sont les OSPs qui sont les plus proches de notre problématique, et c'est sur ces modèles que nous nous appuierons. Nous proposons une approche pour limiter les risques liés aux problèmes de planification probabiliste, en réduisant le nombre de buts à atteindre afin de garantir une *solution sûre* du point de vue du décideur. Voici les points essentiels de notre contribution :

1. Une méthode pour estimer la probabilité de but (RP), soit la probabilité d'atteindre un but depuis un autre but.
2. Un algorithme garantissant une *solution sûre*, définie avec un *seuil de sûreté* configurable.

3 Heuristique pour la sélection des buts

Chaque but a un score, qui représente son intérêt par rapport aux autres buts. L'heuristique pour la sélection des buts est en charge de choisir les meilleurs buts (avec la meilleure somme des scores) atteignables avec une probabilité supérieure au seuil de sûreté. Cette sélection est ensuite utilisée pour guider le planificateur. Notre heuristique se décompose en trois étapes :

1. Création d'une version relaxée du problème, réduisant l'espace d'état et permettant l'identification d'*états but* (état satisfaisant au moins un but du problème).
2. Une estimation au travers du problème relaxé, de la probabilité de but et du coût entre tous les états but identifiés.
3. La recherche dans un graphe (construit à l'aide des états but et estimations précédemment calculés) des meilleurs buts à atteindre.

Avant de détailler ces trois étapes, nous donnons la forme du problème relaxé utilisée pour formaliser l'heuristique.

Formalisation. Le problème relaxé est formalisé sous la forme d'un MDP [12]. L'ensemble des buts de ce MDP, que nous notons S_G, correspond aux états but identifiés lors de la phase de relaxation. Nous faisons le choix d'utiliser les MDPs car ils formalisent très bien les problèmes probabilistes. Le formalisme utilisé est donc un MDP sans facteur d'actualisation (*undiscounted MDP*) sous la forme $\langle S, A, S_G, p, c, u, \alpha, s_0 \rangle$ tel que :

- S est un ensemble fini d'états ($s_0 \in S$ étant l'état initial).
- A est un ensemble fini d'actions. Nous notons $A(s)$ le sous-ensemble des actions applicables dans un état s.
- $S_G \subseteq S$ est une liste d'états but ($s_g \in S_G$ désignant un état but).
- $u(s_g) > 0$ est le score de l'état but s_g.
- $p(s, a, s') \in [0, 1]$ est une fonction de transition qui donne la probabilité d'aller d'un état s à un état s' en appliquant l'action a.
- $c(s, a) > 0$ est le coût de l'action a dans l'état s.
- $\alpha \in [0, 1]$ est le *seuil de sûreté* et $\beta = 1 - \alpha$ est le *seuil de risque*.

3.1 Relaxation du problème

Chercher la meilleure sélection des buts, via une recherche complète dans l'espace d'état, est en pratique impossible. En effet, le nombre d'états but peut vite devenir très grand. C'est pourquoi des méthodes d'abstraction de l'espace

d'état ont été utilisées pour résoudre ce type de problème [9, 8]. La plupart de ces méthodes appliquées aux OSPs créent une version relaxée du problème sous forme de graphe, dont les nœuds représentent les états but étiquetés de leurs scores et les arcs les ressources nécessaires pour aller d'un état but à un autre [14, 4]. La sélection des buts se fait alors à l'aide d'algorithmes de recherche dans un graphe, dont le chemin recherché doit maximiser la somme des nœuds tout en respectant une limite de capacité sur les arcs. La méthode de relaxation que nous utilisons dans notre heuristique est très proche de celles proposées par *Smith* ou *Benton et al.* [14, 2]. Pour être efficace, cette méthode suppose que les différents buts sont suffisamment indépendants entre eux, hypothèse que nous pouvons faire dans le cadre de notre cas d'application où les buts sont des sous-tâches indépendantes que le robot doit réaliser (visiter différents points d'intérêts). Nous ne rentrons pas dans les détails de la relaxation car ce n'est pas l'objectif de cet article, nous donnons cependant l'idée générale. Cela consiste à identifier les interactions et variables partagées entre les buts, ce qui permet ensuite de former un espace d'état restreint et donc l'identification des états but. Chaque état but identifié sur le problème relaxé correspond à un but du problème initial. Le plan relaxé ainsi formé, est une version simplifiée du problème, contenant suffisamment d'informations pour estimer les coûts et probabilités entre les buts.

3.2 Fonction probabilité de but et coût de but

S'inspirant de Teichteil-Königsbuch [17], et pour des raisons d'uniformisation, nous appellerons la fonction estimant la probabilité d'atteindre un but, la fonction probabilité de but (noté RP). Quant à la fonction coût de but (noté V), elle estime les coûts entre les états but. Ces deux fonctions sont des *fonctions de valeur*, qui associent une valeur pour chaque état. Nous les approximons via une méthode de *programmation dynamique* [16, 3] avec un algorithme *Value Iteration* appliqué au problème relaxé. L'algorithme *Value Iteration* s'appuie sur la résolution de l'équation de Bellman [1], qui approche la valeur exacte par raffinement itératif. Cette équation est définie comme suit :

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

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

En général, une fonction de valeur est une estimation de l'intérêt d'être dans un état s, par rapport à l'ensemble des buts du problème. Dans le cas sur-contraint, nous estimons les fonctions de valeur, par rapport à chaque état but du problème relaxé. C'est donc un vecteur de fonctions de valeur, de la taille du nombre d'états but.

Approximation des fonctions de valeur. Tandis que l'approximation de la fonction coût de but est une application directe de l'équation de Bellman, la fonction probabilité de but en est une variante où les valeurs initiales sont égales à 1 quand on entre dans un état but et 0 sinon. Sur le même principe que la formulation proposée par *Steinmetz et al.* [15], nous définissons $RP(s, s_g)$, la probabilité d'entrer dans un état but s_g depuis un état s et $\sigma(s, a, s_g)$ la probabilité d'entrer dans un état but s_g depuis un état s en appliquant l'action a :

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

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

La fonction coût de but $V(s, s_g)$ est défini sous la forme de l'équation de Bellman (cf. Eq.1), non pas en sélectionnant l'action qui minimise le coût mais l'action qui maximise la probabilité de but. De la sorte, les deux fonctions de valeur sont mise à jour avec la même action (i.e. qui minimise σ) et donc les mêmes décisions de l'agent. La fonction coût de but est alors une estimation des futurs coûts quand l'agent agit en maximisant sa probabilité de but.

$$V(s, s_g) = Q(s, a, s_g) \text{ s.t. } a = \operatorname*{argmax}_{a \in A(s)} \sigma(s, a, s_g) \tag{4}$$

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

Dans ce papier, nous faisons l'hypothèse qu'il existe au moins une politique avec une probabilité de succès non nulle et que tous les coûts sont strictement positifs. Ainsi toutes les politiques non-propres (*improper policies* en anglais) accumulent un coût infini, ce qui permet à l'équation de Bellman de toujours avoir une solution. Une fois que nous avons estimé la probabilité de but et le coût, on peut en déduire comme un cas particulier, la probabilité et le coût pour aller d'un état but s_{g1} à un autre état but s_{g2}, soit $RP(s_{g1}, s_{g2})$ et $V(s_{g1}, s_{g2})$.

3.3 Sélection des buts

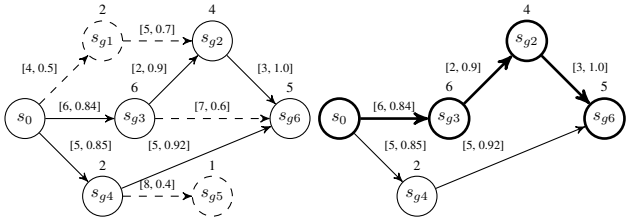

FIGURE 1 – Exemple d'un $SGGraph$ avec $\alpha = 0.75$ et des arcs étiquetés avec $[V, RP]$. Le meilleur chemin est $(s_0, s_{g3}, s_{g2}, s_{g6})$ et les meilleurs buts $\{s_{g3}, s_{g2}, s_{g6}\}$.

Le processus de sélection des buts est chargé de choisir et d'ordonner le meilleur sous-ensemble de buts pouvant être satisfait par une solution sûre. Dans la plupart des travaux similaires [14, 11, 4], la sélection se fait par la résolution d'un *orienteering problem* (OP). Un OP [7, 20] est un problème de graphe dans lequel chaque nœud a un score, et dont l'objectif est de trouver un chemin Hamiltonien maximisant la somme des scores sans dépasser une

certaine limite. Pour les problèmes OSPs, un graphe est construit depuis la version relaxée, ce qui permet ensuite de trouver les meilleurs buts via un algorithme de résolution d'OP. Dans notre cas, nous appelons $SGGraph$, le graphe orienté dont les nœuds sont l'état initial et les états but identifiés lors de la phase de relaxation. Les arcs du graphe représentent les dépendances entre états but (probabilité de but et coût de but). Le $SGGraph$ est ensuite résolu comme un OP, en cherchant un chemin dont le produit des probabilités de but est supérieur au seuil de sûreté α. Un exemple de $SGGraph$ est donné dans la Figure 1. Plus formellement, nous définissons $SGGraph = (S, E)$ avec S l'ensemble des nœuds tel que $S = S_g \cup \{s_0\}$ et E les arcs avec pour étiquette la probabilité de but et le coût pour aller d'un nœud s_1 à un autre nœud s_2 sous la forme : $[V(s_1, s_2),\ RP(s_1, s_2)]$. Nous définissons également $RP(path)$, la probabilité d'atteindre tous les états but d'un chemin. C'est le produit des RP pour aller de l'état initial au premier état but du chemin, puis d'état but en état but dans l'ordre donné par le chemin. A noter que tout chemin valide doit commencer par l'état initial.

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

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

Un chemin est sûr, si sa probabilité de but $RP(path)$ est au-dessus du seuil de sûreté α.

$$RP(path) \geq \alpha \quad (7)$$

Nous définissons aussi l'utilité d'un chemin $U(path)$ comme étant la somme des utilités des états but du chemin et le coût d'un chemin $C(path)$ comme le coût total pour atteindre tous les état but du chemin.

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

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

Le chemin optimal recherché, est un chemin sûr, maximisant U en premier critère et minimisant C en second. Ce second critère est utilisé lorsque plusieurs chemins sont sûrs et d'utilités équivalentes. Ce qui est fréquemment le cas quand plusieurs chemins sont composés des mêmes états but dans un ordre différent. Dans cette situation, le critère de coût permet de choisir l'ordre le moins coûteux. Enfin, nous appelons la sélection des buts $list\_g$, qui est le chemin optimal moins l'état initial. L'algorithme de recherche du chemin optimal est donnée dans la partie 4.

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

$$list\_g = SGGraph(s_0) \quad (10)$$

3.4 Un petit exemple

Nous illustrons l'utilisation de notre heuristique de sélection des meilleurs buts via la version modifiée du problème d'exploration, que nous avons présenté à la section 2.1.

Relaxation du problème. L'espace d'état est réduit à la position du robot et les actions à ses déplacements possibles dans les 4 directions cardinales (i.e. problème de type *grid world*).

Probabilité de but et coût de but. Dans la Figure 1, s_0 est la position initiale du robot, $s_{gi} (i \in 1, 2, ..., 6)$ sont les points d'intérêts à visiter (buts) étiquetés de leurs scores. Les étiquettes des arcs correspondent respectivement, à la distance pour aller d'un point à un autre (coût) et à la probabilité d'atteindre un point depuis un autre.

Sélection des buts. Dans la Figure 1, il y a 2 chemins possibles : $path\_a = (s_{g3}, s_{g2}, s_{g6})$ et $path\_b = (s_{g4}, s_{g6})$. En utilisant l'Eq.6 et l'Eq.8, nous obtenons :

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

En fixant $\alpha = 0.75$, les deux chemins $path\_a$ et $path\_b$ sont sûrs ($RP(path\_a) \geq \alpha$ et $RP(path\_b) \geq \alpha$). Cependant, le meilleur chemin est $path\_a$ car de qualité supérieure ($U(path\_a) > U(path\_b)$). Notons que le coût n'intervient pas dans cet exemple pour des raisons de simplicité, mais aurait été utilisé si les deux chemins étaient de même qualité. Enfin, nous extrayons depuis le $path\_a$, la sélection des buts, soit $list\_g = (s_{g3}, s_{g2}, s_{g6})$.

4 Algorithmes

Dans cette partie, nous commençons par présenter l'algorithme *Value Iteration* (VI) utilisé pour approximer nos deux fonctions de valeur (probabilité de but et coût de but). Nous détaillons ensuite notre méthode pour résoudre le $SGGraph$ et finissons par présenter la solution globale du problème.

4.1 Algorithme VI

Algorithm 1 Relaxed Plan Value Iteration

function RPVALUEITERATION
  InitializeStates() ▷ see Eq.11
  repeat
    $\Delta_{RP} = 0, \Delta_{RP} = 0$
    for each $s \in \mathcal{S}$ **do**
      for each $s_g \in \mathcal{S}_G$ **do** ▷ for all goal states
        $safe\_a = \text{argmax}_{a \in A(s)}\ \sigma(s, a, s_g)$
        $V(s, s_g) = Q(s, safe\_a, s_g)$
        $RP(s, s_g) = \sigma(s, safe\_a, s_g)$
        $\Delta_V = \max(\Delta_V, R_V(s, s_g))$
        $\Delta_{RP} = \max(\Delta_{RP}, R_{RP}(s, s_g))$
  until $\Delta_V < \epsilon$ & $\Delta_{RP} < \delta$ ▷ see Eq.12

Initialisation. La fonction coût de but V est initialisée pour chaque état arbitrairement avec une valeur positive. La fonction probabilité de but RP est initialisée à 1 pour chaque état but et 0 sinon. Plus formellement :

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

Condition d'arrêt. L'algorithme VI s'arrête quand le résidu sur l'équation de Bellman entre deux itérations est suffisamment petit pour tous les états. Soit, quand le résidu sur Eq.2 et Eq.4 est suffisamment petit pour tous s et s_g.

$$R_V(s, s_g) = |V(s, s_g) - V'(s, s_g)|$$

$$R_{RP}(s, s_g) = |RP(s, s_g) - RP'(s, s_g)|$$

$$\forall s \in S,\ \forall s_g \in S_G,\ R_V(s, s_g) < \epsilon, R_{RP}(s, s_g) < \delta \tag{12}$$

Calcul des fonctions de valeur. Dans l'algorithme 1, les fonctions de valeur sont approximées en calculant itérativement Eq.2 et Eq.4 jusqu'à la condition d'arrêt définie plus haut. Pour trouver des solutions sûres, ces deux fonctions sont calculées en choisissant à chaque fois l'action la plus sûre, soit l'action maximisant la probabilité de but (ligne 7). Lignes 8 et 9, V et RP sont calculées avec cette action, puis le résidu maximal est stocké lignes 10 et 11. L'algorithme s'arrête quand Δ_V et Δ_{RP} sont suffisamment petits (nous discutons dans nos tests des valeurs de ϵ et δ). La notation σ et Q est utilisée par souci d'espace, mais σ est à remplacer par la partie droite de Eq.3 et Q par celle de Eq.5.

4.2 Résoudre SGGraph

Sur le graphe $SGGraph$, le processus de sélection des buts consiste à trouver un chemin sans boucle et optimal (voir partie 3.3 pour la définition d'un chemin optimal). Dans un premier temps, nous simplifions le graphe en retirant tout élément ne pouvant pas faire partie d'une solution valide. C'est à dire tous les arcs dont la probabilité de but est inférieure au seuil α, ainsi que tous les nœuds non reliés à l'état initial s_0 (voir éléments en pointillés sur la Figure 1). Une fois le graphe réduit, l'étape suivante est la recherche du chemin optimal. La complexité en temps de la recherche d'un chemin dans un graphe orienté avec cycle est exponentielle. Par conséquent, des méthodes de recherche heuristique doivent être utilisées. Nous faisons le choix d'une recherche en faisceau, qui permet d'obtenir une bonne solution en un temps raisonnable (c'est cette même méthode qui a été utilisée dans d'autres travaux similaires [14, 4]). Ce type de recherche n'explore qu'un ensemble limité de fils à chaque niveau, réduisant ainsi la mémoire et le temps d'exécution nécessaire. Le nombre de fils exploré à chaque étape est paramétrable et correspond à la largeur du faisceau, que nous dénotons k. L'utilisation d'une heuristique permet de choisir quels sont potentiellement les meilleurs fils à explorer. Dans notre cas, nous avons choisi une heuristique sélectionnant les fils ayant la meilleure probabilité de but direct, c'est à dire la probabilité de but entre l'état courant et le fil en question. Bien que la solution optimale ne soit pas garantie, car la recherche n'est pas complète, une bonne heuristique permet de trouver rapidement une bonne solution. Nous appliquons cet algorithme sur une recherche en profondeur, et évitons les cycles en marquant chaque nœud quand on rentre dans la récursion et en retirant la marque quand on en sort. Pour chaque chemin sûr rencontré, nous comparons l'utilité U (Eq.8) et si besoin le coût C (Eq.9) avec l'actuel meilleur chemin, puis conservons le meilleur des deux.

4.3 Planifier les buts sélectionnés

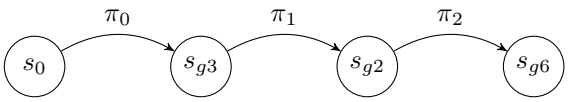

FIGURE 2 – La politique finale $\pi = (\pi_0, \pi_1, \pi_2)$ est l'agrégation des politiques trouvées pour résoudre chaque but de la sélection.

Étant dans un contexte de planification probabiliste, nous ne pouvons pas simplement atteindre un par un les buts de la sélection, nous devons donc définir une politique pour chaque état but. L'idée est de planifier tous les états but s_{gi} de la sélection, comme des sous-problèmes indépendants, puis d'agréger les sous-solutions pour former la solution finale (cf. Figure 2). La définition de chaque sous-problème est la même que le problème initial en terme d'actions, états, fonction de transitions et coûts des actions. Elle diffère par contre pour l'état initial ainsi que les buts. L'état initial d'un sous-problème et l'état final du sous-problème précédent. Les buts G_i du sous-problème i sont l'ensemble des prédicats de l'état but s_{gi} correspondant à ce sous-problème. Nous résolvons chaque sous-problème avec un algorithme VI, qui estime $V_i(s)$ et $RP_i(s)$, respectivement le coût et la probabilité d'atteindre un état satisfaisant G_i, avec la même approche que celle présenté précédemment sur le problème relaxé. Nous déterminons ensuite la politique π_i choisissant les actions qui maximisent d'abord la probabilité de but, puis le coût en cas d'égalité. Pour cela, nous définissons $SA(s)$ l'ensemble des actions maximisant la probabilité d'atteindre G_i et π_i tel que :

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

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

La solution du problème s'exprime sous la forme $\pi(s) = (\pi_0, \pi_1, \ldots, \pi_n)$. Chaque politique est utilisée l'une après l'autre pour atteindre chaque but dans l'ordre de la sélection. Techniquement, les buts déjà atteints sont stockés en mémoire pour connaître à chaque moment la prochaine politique à utiliser.

5 Résultats numériques

5.1 Cas d'étude

Dans cette partie, nous présentons les résultats numériques que nous avons obtenus en appliquant notre heuristique sur la version simplifiée du problème d'exploration planétaire présentée précédemment. Ce problème, proche d'un problème de type *grid world*, permet d'illustrer nos travaux. Il est composé d'une grille pour modéliser la planète, dont chaque case est une position possible du robot. Les buts du problème sont des cases représentants les sites d'intérêts que le robot doit atteindre. Comme présenté dans la section 2.1, nous ajoutons la notion de risque à ce problème en distinguant deux types de case : les cases sûres et les cases risquées. L'idée est qu'à chaque fois que le robot part d'une case risquée, il a une certaine probabilité de rester bloqué. Le robot peut effectuer quatre actions : aller vers le nord, l'est, le sud ou l'ouest. Sur une case sûre, une action $a \in A$ déplacera le robot sur la case désirée avec une probabilité p, ou sur l'une des trois autres directions avec une probabilité $(1-p)/3$ si l'action échoue (définition classique d'un problème de type *grid world* en environnement probabiliste). Sur une case risquée, le robot a une probabilité q de rester bloqué, et donc $1-q$ de se déplacer. L'objectif pour le robot est de visiter les sites avec les scores les plus élevés, tout en garantissant une probabilité de succès supérieur au seuil de sûreté. Une simulation est réussie si le robot atteint tous les sites sélectionnés sans rester bloqué. Dans un premier temps, nous évaluons pour différents seuils, la qualité des sélections des buts obtenue par l'heuristique, puis nous effectuons quelques tests de performance.

Sélection des paramètres de condition d'arrêt. Nous rappelons que ϵ et δ, sont les paramètres utilisés pour contrôler le résidu sur l'équation de Bellman, et donc la précision des estimations sur les fonctions de valeur. Pour choisir les valeurs les plus appropriées, nous avons mesuré la sensibilité de nos résultats pour des valeurs allant de 0.1 à 0.0001. En effectuant des tests sur des cas nominaux et extrêmes, nos résultats convergeaient vers les mêmes estimations à partir de 0.01 pour les deux paramètres, c'est donc la valeur que nous avons choisi.

5.2 Résultats

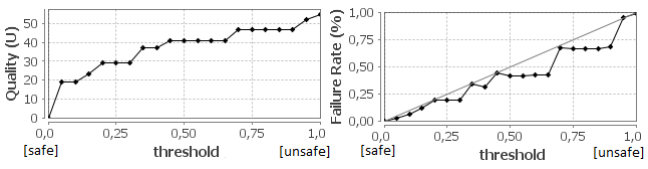

FIGURE 3 – Test sur la sélection des buts. La qualité (U) est la somme des scores des buts. Le taux d'échec, et le pourcentage de simulations terminant sans avoir atteint les buts prévus par la sélection.

Dans nos tests présentés dans la Figure 3, nous avons utilisé une grille de 25×25, et différents seuils de risque β entre 0 et 1. Nous utilisons β au lieu de α pour des raisons de simplicité de raisonnement, ce qui représente ainsi une limite de risque à ne pas dépasser. Nous avons défini dix buts générés de manière aléatoire, et 50% de cases risquées avec une probabilité $q = 0.1$ de rester bloqué. Nous avons d'abord testé la sélection des buts pour différents seuils puis nous avons simulé chaque sélection pour vérifier le bon respect du seuil. Sur le graphe de gauche de la Figure 3, l'axe x représente le seuil de risque et l'axe y la qualité U obtenue en sommant les scores des buts de la sélection. On peut voir que la qualité croît de façon monotone quand le risque croît. Dans certain cas (par exemple pour $\beta \in [0.45, 0.65]$), la qualité reste constante, signifiant qu'aucune autre meilleure sélection des buts est possible jusqu'à un certain seuil. Dans le deuxième graphe, nous testons via des simulations que chaque sélection des buts générée respecte bien le seuil de risque imposé. Pour ce faire, nous simulons chaque planification de la sélection 1000 fois (nous avons obtenu la même précision en testant avec 2000 simulations), et calculons le taux moyen d'échec, soit le taux de simulation n'ayant pu atteindre tous les buts prévu par la sélection. Comme prévu, le taux d'échec n'excède jamais le seuil β (la ligne diagonale). Toutefois, on repère parfois des intervalles relativement importants entre cette ligne et le taux d'échec. Ceci est dû au fait que le planificateur détecte qu'il ne peut ni augmenter la qualité, ni baisser le coût, avec le risque supplémentaire qui lui est alloué. Il maintient donc la probabilité de but aussi haute que possible.

5.3 Performances

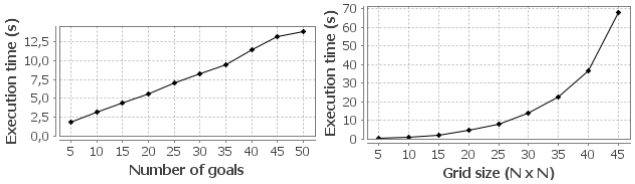

FIGURE 4 – Test de performances : sur la gauche, nous faisons varier le nombre de buts sur un espace d'état constant (grille : 25×25). A l'inverse, sur la droite, c'est l'espace d'état qui varie et le nombre de buts qui est fixe (8).

Dans la Figure 4, nous traçons le temps d'exécution en seconde du processus de sélection des buts en fonction du nombre de buts total. Nous avons remarqué que ce temps d'exécution est en réalité très proche du temps d'exécution de l'algorithme 1 qui estime les fonctions de valeur. En effet, construire le graphe $SGGraph$ et rechercher le meilleur chemin ne prend pas plus de 2% du temps total d'exécution de l'heuristique. Ceci à condition d'avoir une recherche en faisceau efficace. Dans notre cas nous avons testé plusieurs largeurs de faisceau et conclus qu'en moyenne, l'heuristique converge vers une solution unique avec une largeur de faisceau de $0.25\times$ le nombre total de buts. Avec cette largeur de faisceau, nous avons obtenu une recherche toujours inférieure à 250 ms pour une grille al-

lant jusque 45×45 états et 50 buts.

6 Conclusion

Beaucoup de problèmes de planification probabilistes en condition réelle produisent des solutions non sûres, ce qui peut être critiques dans certains domaines tels que le domaine spatial (un robot explorateur peut coûter très cher, etc.). Dans ce papier, nous avons décrit une heuristique pour trouver des solutions satisfaisant les meilleurs buts possibles, tout en limitant la prise de risque. Notre méthode sélectionne les meilleurs buts qui pourront être atteints avec une probabilité de succès supérieure à un seuil donné. Cette sélection s'appuie sur un algorithme de type *Value Itération* qui estime la probabilité de but ainsi que le coût entre chaque but à l'aide d'une version relaxée du problème.
Nous travaux ont soulevé de nouveaux problèmes : une prochaine étape serait de travailler sur un choix autonome du seuil par le planificateur. Une solution envisagée serait d'avoir un seuil qui repose sur le critère de Hurwitz, permettant ainsi d'avoir un compromis variable entre une attitude sûre et une attitude risquée. De plus, l'algorithme VI, qui explore tout l'espace d'état, prend beaucoup de temps. Nous prévoyons donc d'appliquer notre méthode de sélection sur des algorithmes dit *Anytime* tels que RTDP ou L-RTDP qui peuvent converger beaucoup plus rapidement vers des solutions presque optimales. Et enfin, nous prévoyons d'appliquer notre heuristique sur des domaines plus complexes et larges, tels que la planification de situations critiques en environnement virtuel pour la formation de médecins.

Références

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

[2] J. Benton, Minh Do, and S. Kambhampati. Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence*, 173 :562–592, 2009.

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

[4] 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)*, volume 7023 LNAI, pages 183–192, 2011.

[5] P. Geibel. Reinforcement learning with bounded risk. *Proceedings of the Eighteenth International Conference on Machine Learning*, 9(D) :162–169, 2001.

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

[7] BL. Golden, L. Levy, and R. Vohra. The Orienteering Problem. 1987.

[8] 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 ACM*, 61(3) :63, 2014.

[9] M. Katz and D. Carmel. Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, 39 :51–126, 2010.

[10] LA. Kramer and SF. Smith. Maximizing Flexibility : A Retraction Heuristic for Oversubscribed Scheduling Problems. *International Conference on Automated Planning and Scheduling*, 2005.

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

[12] Groupe PDMIA. Processus décisionnels de Markov en intelligence artificielle. *Edité par Olivier Bu et et Olivier Sigaud*, 1, 2008.

[13] 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*, pages 192–201, 2005.

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

[15] M. Steinmetz, J. Hoffmann, and O. Buffet. Revisiting Goal Probability Analysis in Probabilistic Planning. *International Conference on Automated Planning and Scheduling*, (0) :299–307, 2016.

[16] RS. Sutton and AG Barto. Reinforcement learning. *Learning*, 3(9) :322, 2012.

[17] F. Teichteil-Königsbuch. Problèmes de Plus Sûr et Plus Court Chemin Stochastique.

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

[19] 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.

[20] P. Vansteenwegen, W. Souffriau, and DV. Oudheusden. The orienteering problem : A survey. *European Journal of Operational Research*, 209 :1–10, 2011.