Un modèle hybride en planification probabiliste d'exploration autonome

An hybrid probabilistic model for autonomous exploration

F. Teichteil-Königsbuch P. Fabiani

ONERA-DCSD

2 Avenue Edouard-Belin
31055 Toulouse
(florent.teichteil,patrick.fabiani)@cert.fr

Résumé

Des systèmes autonomes d'exploration requièrent des capacités de planification d'actions en présence d'incertitudes. Les Processus Décisionnels de Markov (PDM) offrent un modèle probabiliste classique fondé sur une représentation énumérée et non structurée de l'espace d'état. Des travaux récents proposent des approches plus compactes et structurées. Nous présentons et discutons l'approche par factorisation en variables d'état, puis celle par décomposition en sous-régions, ainsi que nos expérimentations et comparaisons de techniques pour cette dernière. Nous proposons un nouveau modèle hybridant les deux et adapté à la planification d'exploration.

Mots Clef

planification probabiliste, robotique autonome, programmation dynamique, processus décisionnels markoviens

Abstract

Autonomous exploration systems require planning under uncertainty. Markov Decision Processes provide a classical framework based on an enumerated and unstructured state space. Recent works propose more compact and structured approaches in probabilistic planning. We present and discuss factorisation techniques using state variables, and then decomposition techniques using subregions for which we also present our experimentations and comparisons. A novel hybrid model combining both is proposed well-fitted to exploration planning.

Keywords

probabilistic planning, autonomous robotics, dynamic programming, markovian decisionnal processes

1 Introduction

Des systèmes autonomes d'exploration, qui nous intéressent, requièrent des capacités de planification d'actions en présence d'incertitudes. Les Processus Décisionnels de Markov (PDM) [11] sont un modèle stochastique classique de planification dans l'incertain, notamment les PDM complètement observables, c'est-à-dire sans incertitudes sur les observations du robot. La gestion de la prise d'information est cruciale dans les problèmes d'exploration, mais les algorithmes de résolution des PDM Partiellement Observables [9] prenant en compte le bruit d'observation, sont notablement plus coûteux en temps de calcul et encore difficilement applicables pour des applications réalistes. Nous envisagerons plus tard la prise en compte de l'observabilité partielle.

Les PDM peuvent être classiquement résolus par programmation dynamique stochastique sur l'espace d'état explicitement énuméré, mais l'efficacité des algorithmes de résolution exacte pour ces problèmes reste insuffisante pour des applications réalistes comportant souvent des espaces d'état de grande taille [4, 13]. Les techniques proposées pour répondre à ce défi comprennent des méthodes d'approximation ou d'apprentissage [2] permettant de contrôler le coût calculatoire au prix d'une erreur également contrôlée. D'autres approches consistent à utiliser la structure naturelle des problèmes de planification soit au travers de représentations factorisées plus compactes utilisant des variables d'état plus ou moins décorrélées [4, 3, 5] soit au travers d'une décomposition de l'espace d'état en sous-régions [8, 7, 10, 12] autorisant une résolution hiérarchisée parfois plus efficace, découplée au niveau local puis globalisée sur la base des résolutions partielles locales.

Les problèmes d'exploration stochastique se modélisent plus naturellement sous une forme hybride combinant une décomposition géographique et une factorisation en composantes d'état globales. Nous présentons d'abord rapidement le cadre des PDM en discutant de certains inconvénients du modèle pour un problème d'exploration. Nous exposons ensuite la théorie de factorisation des PDM puis les techniques de décomposition de PDM, que nous déve-

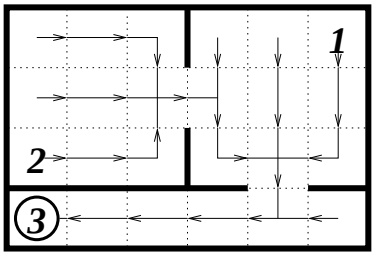

FIG. 1 – Politique optimale aveuglée par la zone de plus forte récompense

loppons dans la cinquième partie. Nous proposons ensuite dans la sixième partie un modèle de PDM pour la résolution des problèmes d'exploration ainsi qu'un algorithme hybride de résolution. Nous conclurons par l'intérêt d'un tel modèle et les perspectives que nous envisageons d'étudier.

2 Processus Décisionnels de Markov

Un PDM est une chaîne de Markov contrôlée par un agent suivant une politique associant à chaque état le choix d'une action : une action appliquée dans un même état de départ amène aléatoirement dans un autre état parmi plusieurs possibles. La probabilité que l'agent se retrouve dans un état particulier vérifie la propriété de Markov, c'est à dire que cette probabilité ne dépend que de la connaissance de l'état précédent et non de tout l'historique des états où l'agent a été.

Formellement, un PDM est un quadruplet $\langle S, A, T, R\rangle$ où S est l'ensemble des états de l'agent, A l'ensemble des actions réalisables dans tous les états, T et R les probabilités et récompenses des transitions entre états pour chaque action, qui vérifient la propriété de Markov : $T(s, a, s') = P(s'|a, s)$.

L'algorithme de résolution d'un PDM suit le principe de la programmation dynamique et dépend du critère d'optimalité choisit. Le plus classique est le critère γ-pondéré à horizon infini, qui consiste à maximiser la somme infinie des récompenses recueillies par le robot aux différents instants pondérées par un coefficient contracteur $\gamma < 1$ d'un instant sur l'autre. Bien que γ est nécessaire à la convergence de la somme des récompenses sur un horizon infini, il peut être interprété comme la probabilité d'arrêt du processus à un instant donné. Les probabilités de panne ou de destruction du robot peuvent donc être modélisées par γ. Pour ce critère, on applique une résolution itérative de la valeur ou de la politique. A chaque itération, la politique courante est améliorée en calculant l'action qui maximise la valeur de chaque état connaissant la valeur des états suivants pour cette politique [11]. La valeur d'une politique π étant solution de l'équation de Bellman [1] :

$$\forall\, s \in S\ ,\ V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s') \cdot \big(R(s, \pi(s), s') + \gamma V^{\pi}(s')\big)\ .$$

3 Problème de planification d'exploration

Dans un problème d'exploration se conjuguent, entre autres, une problématique de déplacement dans un environnement et une composante de prise d'information sur cet environnement. Dans les versions "géométriques" de cette problématique, il s'agit de réaliser une couverture satisfaisante d'une zone géométrique donnée à explorer ou surveiller au moyen de la trace au sol d'un capteur. Des solutions pratiques de ce type de problèmes existent et les plus utiles sont le plus souvent ad hoc.

Nous nous intéressons ici au problème d'un système devant naviguer dans un environnement mal connu et devant pour cela prendre des informations sur les différentes régions de cet environnement, par exemple à la recherche d'objets ou de lieux particuliers, avant d'y naviguer ou d'y accomplir certaines tâches. Il est évidemment plus risqué de naviguer dans une zone mal connue que dans une zone connue, mais il est plus coûteux de l'explorer avant. Une région devient aussi plus ou moins intéressante à exploiter ou dangereuse en fonction de ce que l'on y découvre (objets d'intérêt, zones dangereuses à éviter, ...). Certaines variables d'état peuvent être plus ou moins indépendantes des déplacements réalisés, mais avoir un impact décisif sur la suite, telles que la quantité d'énergie ou de carburant restante. Ce problème de planification d'exploration consiste donc à optimiser l'enchaînement d'actions de navigation et de prise d'information de façon à réaliser en séquence des objectifs intermédiaires d'exploration permettant de trouver et de naviguer jusqu'à un objectif final en minimisant les risques et les coûts. La réalisation des objectifs intermédiaires conditionne la suite de la mission, soit par l'information acquise, soit par les conséquences ayant résulté de la politique appliquée (détection d'objets, consommation de carburant, ...).

Différentes sous-régions géométriques de l'environnement peuvent donc être parcourues plusieurs fois dans la mission mais dans des contextes de navigation, de risque et d'objectif complètement différents. On peut donc découpler partiellement les aspects géométriques de la navigation du reste du problème : d'une part, chaque sous-région géométrique du PDM peut être résolue localement sous différentes hypothèses de contexte global du PDM et d'autre part, les conséquences de la mise à jour locale du modèle de l'environnement dans une sous-région particulière peuvent être limitées au re-calcul des politiques locales à cette sous-région. Ce constat incite à utiliser le modèle et les techniques des PDM décomposés.

Pour un agent robot devant faire une navigation, les PDM modélisent les imperfections des actionneurs responsables des aléas de transition vers l'état successeur de l'état courant. Ainsi, une politique optimale d'un PDM tient compte des états de faibles récompenses où le robot risque de se retrouver sans le rechercher. Cependant, les transitions d'état du robot réel dans son environnement géométrique et phy-

sique ne vérifient pas nécessairement la propriété de Markov. La modélisation du problème dans le cadre PDM doit prendre en compte les composantes d'état permettant de se ramener au cadre Markovien : ici des variables *"objectifs intermédiaires"*, *"carburant restant"*, etc.

En outre, le formalisme classique des PDM impose une énumération explicite de tous les états du modèle, ce qui limite leur utilisation à des problèmes de faible dimension et taille modérée. Or, sans l'ajout direct de variables *objectifs* à l'ensemble des états uniquement géographiques, la programmation dynamique calcule une politique aveuglée par la zone de plus grande récompense (figure 1), qui ne cherche pas à visiter les zones intermédiaires de moindres récompenses. La nécessaire prise en compte des variables *objectifs* augmente sensiblement la dimension de l'espace d'états au point que le PDM ne peut plus être résolu par des algorithmes classiques. Une modélisation factorisée des PDM [5], utilisant des réseaux bayésiens dynamiques, semble plus adaptée de ce point de vue. Cependant, la modélisation factorisée n'est efficace que si chaque variable a peu de valeurs possibles, si bien qu'il n'est pas judicieux de factoriser ainsi en variables d'état la position géographique (x, y) du robot, d'autant que x et y dans ce cas sont mutuellement très corrélées.

Ainsi, nous proposons une approche hybride consistant à appliquer pour certaines variables d'état des techniques de factorisation et pour la composante de navigation du problème des techniques de décomposition de PDM [8, 7] en sous-régions de l'espace d'états énuméré. De tels sous-PDM "géographiques" dans chaque sous-région, permettent d'obtenir plusieurs politiques locales – ou comportements locaux – qui sont autant de macro-actions élémentaires du PDM factorisé. Notre modèle comporte donc deux niveaux de résolution : un niveau géographique et un niveau factorisé.

A notre connaissance, les techniques de décomposition et de factorisation de PDM ont été chacune utilisées indépendamment dans la littérature afin d'optimiser la résolution de PDM dont l'espace d'états est soit entièrement explicité soit factorisé, mais elles n'ont pas été à ce jour imbriquées dans un algorithme hybride de résolution des problèmes d'exploration, ainsi que nous le proposons.

4 Factorisation de l'espace d'états

La factorisation de PDM consiste à utiliser une représentation plus compacte de l'espace d'états du PDM en le factorisant par composantes d'état dont les probabilités de transition sont de ce fait définies et modélisées sous forme de réseaux bayésiens dynamiques. Les techniques de résolution associées visent à éviter l'énumération explicite de tous les états discrets à chaque itération.

4.1 Réseaux bayésiens dynamiques

Un réseau bayésien dynamique est la réunion de plusieurs réseaux d'action qui décrivent chacun la relation de dépendance (probabilités conditionnelles) entre des variables stochastiques sur une échelle de temps discrète, avant et après chaque réalisation d'une action. Le réseau comprend donc des arcs (*diachroniques*) modélisant les dépendances probabilistes entre variables aléatoires entre les instants t et $t+1$. Il comprend aussi des arcs *synchrones* qui représentent la dépendance entre variables au même instant t. Ces termes de dépendances sont stockés dans une structure plus compacte qu'une matrice : un arbre de probabilités conditionnelles, décrit dans la suite. Par exemple, la figure 8 représente un réseau d'action avec un seul arc synchrone. Si l'espace d'états est de dimension n, la probabilité d'obtenir l'état s^{t+1} à un instant donné connaissant l'état $s^t = \wedge_{i=1}^{n} x_i^t$ de l'agent à l'instant précédent est alors donnée par le produit suivant :

$$T(s^t, a, s^{t+1}) = \prod_{i=1}^{n} P\left(x_i^{t+1} | a, x_i^t \wedge \left(\wedge_{j \neq i} \left(x_j^t \wedge x_j^{t+1}\right)\right)\right)$$

4.2 Arbres décisionnels

Les arbres décisionnels représentent des tables où sont stockées les valeurs numériques du problème, à savoir les probabilités de transition et les récompenses associées, qui permettent au robot de prendre ses décisions. Pour chaque action, chaque variable d'état et chaque valeur possible de cette variable, un arbre de probabilité indique la probabilité d'obtenir cette valeur de la variable à un instant donné connaissant la valeur des autres variables à l'instant précédent ou au même instant. Ainsi, dans la formule du paragraphe précédent, les arbres de probabilité correspondent aux termes $P\left(x_i^{t+1} | a, x_i^t \wedge \left(\wedge_{j \neq i} \left(x_j^t \wedge x_j^{t+1}\right)\right)\right)$. Notons aussi que $k-1$ arbres sont suffisants pour représenter les transitions d'une variable d'état à k valeurs possibles puisque la somme des probabilités de transitions pour cette variable et une action donnée est égale à 1.

D'autre part, les récompenses des transitions peuvent être ajoutées aux branches des arbres de probabilités. Chaque branche d'un arbre de probabilité est alors un couple *(probabilité,récompense)*. Cependant, le modèle de récompense est souvent très simple : consistant en une même récompense non nulle pour un sous-ensemble d'états, les autres états ne recevant pas de récompense. Or, les branches d'arbres de probabilité différents correspondent à des sous-ensembles d'états souvent identiques ou inclus l'un dans l'autre si bien que le stockage d'une valeur nulle dans les branches de ces arbres est redondant. Ainsi, plutôt que de stocker des zéros inutiles dans les récompenses des branches des arbres de probabilité, il est préférable de construire un seul arbre de récompense supplémentaire contenant une seule branche de valeur non nulle. La figure 11 donne un exemple d'arbre de probabilité pour une variable binaire ainsi que d'arbre de récompense.

4.3 Résolution

Les divers algorithmes de résolution des PDM factorisés sont des variantes structurées des algorithmes de résolution des PDM classiques. La plupart figurent dans l'article

initiateur [5]. Nous donnons ici quelques grandes lignes de ces méthodes qui sont l'objet de recherches actives. Ces algorithmes reposent sur une représentation des valeurs des états et de la politique sous forme d'arbres, qui permet une exploration moins explicite de l'espace d'état (par "paquets d'états") et qui nécessite les opérations d'*ajout* et de *simplification* d'arbres (cf. figure 2) :

– *ajout* : l'arbre $T_1 + T_2$ est obtenu en ajoutant T_2 à toutes les branches de T_1 et en étiquetant les nouvelles branches par les étiquettes des branches de T_1 et T_2,
– *simplification* : suppression des sous-arbres correspondant à des valeurs incompatibles avec les valeurs des nœuds de niveau inférieur.

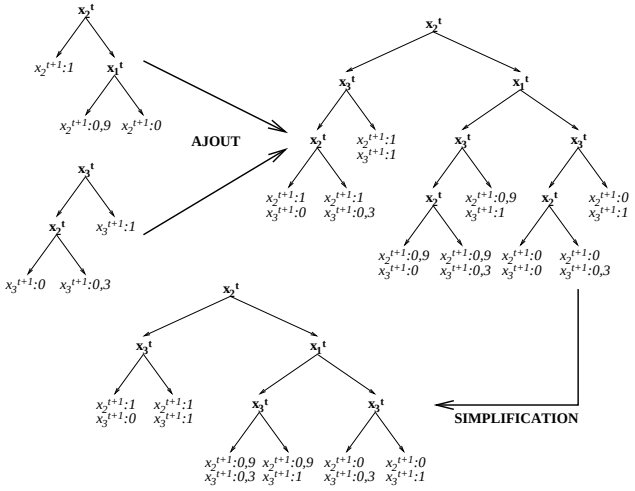

FIG. 2 – Ajout de deux arbres de probabilité et simplification de l'arbre obtenu

Remarquons tout d'abord qu'un chemin dans un arbre décisionnel correspond à un ensemble de valeurs discrètes de variables d'état qui n'expriment pas nécessairement un seul état. Les feuilles des arbres décisionnels sont donc des agrégations d'états qui ont en commun les valeurs d'un certain nombre de variables d'état. C'est d'ailleurs en ce sens que la représentation factorisée de l'espace d'états est particulièrement concise. Cette remarque est la clé des algorithmes de résolution des PDM factorisés.

Connaissant l'arbre de valeur $A(V_{t+1})$ à un instant donné du processus, nous souhaitons construire l'arbre $A(V_t^a)$ représentant la valeur de chaque état à l'instant précédent pour une action donnée. Nous devons donc identifier les états à l'instant t qui auront une même valeur à l'instant $t+1$ par application de l'action a, cette valeur commune correspondant à une feuille de l'arbre $A(V_{t+1})$. Ces états seront donc représentés par une feuille unique de l'arbre $A(V_t^a)$.

Pour cela, à partir du réseau de l'action a, nous mettons en évidence les variables pré-décisionnelles qui influencent les variables post-décisionnelles intervenant dans l'arbre de valeur $A(V_{t+1})$, ce qui constitue l'étape dite de *régression* de l'arbre de valeur. L'ajout et la simplification des arbres de probabilité de ces variables pré-décisionnelles permettent d'obtenir un arbre $PA(V_t)$ de régression dont chaque feuille est la probabilité de transiter d'un groupe d'états vers un autre groupe d'états ayant la même valeur après l'application de l'action a. Par exemple, si les arbres de probabilité de la figure 2 sont ceux de x_2^{t+1} et x_3^{t+1}, l'arbre obtenu par ajout et simplification de ces deux arbres est l'arbre $PA(V_t)$ de régression de l'arbre de valeur $A(V_{t+1})$ représenté dans la figure 3. Une propagation de Bellman pour l'action a sur les groupes d'états représentés par les feuilles de l'arbre de régression $PA(V_t)$ permet d'obtenir, connaissant l'arbre de récompense de cette action, les valeurs des états pour l'action a à l'instant t.

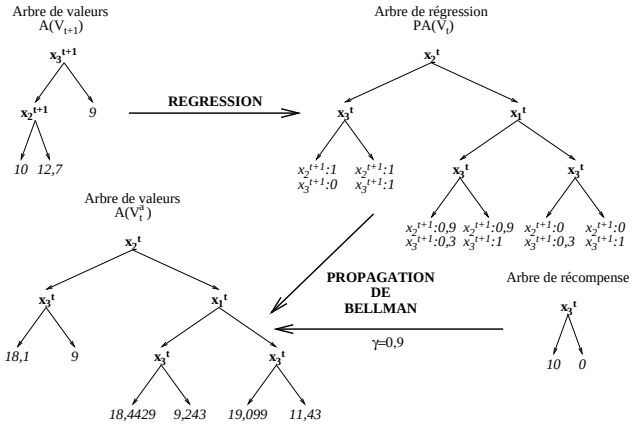

FIG. 3 – Construction de l'arbre de valeur d'une action connaissant l'arbre de valeur à l'instant suivant, l'arbre de récompense et les arbres de probabilité de la figure 2 pour cette action

Enfin, l'ajout successif de tous les arbres de valeur pour chaque action à l'instant t suivi d'une simplification renvoie un arbre dont les feuilles sont étiquetées par les valeurs des différentes actions. En choisissant la valeur maximale et en identifiant l'action correspondante pour chaque feuille, nous construisons un arbre de valeurs optimales ainsi qu'un arbre de politique optimale à l'instant t.

Par conséquent, la méthode que nous avons présenté permet d'adapter les algorithmes d'itération de la valeur ou de la politique à des structures factorisées de PDM grâce à la construction d'arbres de valeur et de politique. Les algorithmes de résolution des PDM factorisés sont donc appelés par [5] *itération de la valeur structurée* et *itération de la politique structurée*.

4.4 Problème des états géographiques

L'élargissement de l'ensemble des valeurs possibles d'une variable d'état augmente sensiblement la complexité des algorithmes de résolution des PDM factorisés. Ainsi, il est impensable de représenter les états géographiques par une seule variable prenant autant de valeurs possibles que d'états géographiques. Nous proposons donc, grâce à des techniques de décomposition de PDM [8, 7] que nous pré-

sentons dans la partie suivante, d'agréger les états géographiques au sein de macro-états bien moins nombreux qui seront les valeurs d'une variable géographique que nous appellerons *région*.
De plus, une représentation factorisée de l'espace d'états duplique le sous-espace des états géographiques au point que la restriction de la politique optimale à ce sous-espace, appelée *navigation*, est calculée inutilement plusieurs fois. Pour une mission à m objectifs, le problème de navigation pure est résolu à l'identique au moins 2^m fois. Aussi, nous proposons de calculer dans un préprocesseur des politiques de navigation pure qui seront les actions élémentaires du PDM factorisé, à l'aide des techniques de décomposition de PDM. Ainsi, la restriction de la politique optimale aux états géographiques n'est calculée qu'une seule fois.

5 Décomposition de l'espace d'états

La décomposition de PDM consiste à agréger les états d'un espace d'états entièrement explicité au sein de régions faiblement communiquantes c'est à dire dont le nombre de transitions entre les états de sortie et d'entrée est faible. Un macro-PDM est alors construit dont les états sont les régions issues de la décomposition, les actions sont des politiques locales définies dans chaque région et les transitions sont calculées à partir de ces politiques locales.
La décomposition de PDM nous semble particulièrement intéressante pour les problèmes de navigation pure, dont les états sont uniquement géographiques, car nous pouvons facilement partitionner l'espace d'états de l'environnement du robot en régions faiblement communiquantes, grâce à des algorithmes de géométrie algorithmique de terrain sur un modèle numérique de terrain ou à l'aide de techniques de partitionnement de graphes de PDM [12]. De plus, un changement du modèle dans une région en cours de mission ne nécessite que de mettre à jour les politiques locales de la région sans que la politique globale doive être recalculée pour tout le modèle, si bien qu'une re-planification en ligne est réalisable en un temps convenable.
Nous proposons un formalisme de la décomposition de PDM qui tire profit de deux modèles différents de décomposition [8, 7]. Nous avons testé deux algorithmes de génération des politiques locales dont nous comparons les performances dans la suite.

5.1 Décomposition en régions

La décomposition en régions est un partitionnement Π de l'espace d'états en régions distinctes et non vides. Pour chaque région, nous identifions l'ensemble $Sper(\mathcal{R})$ des états de sortie de la région :

$$Sper(\mathcal{R}) = \left\{ s \in S - \mathcal{R} \ / \ \exists\, s' \in \mathcal{R} \ , \ \exists\, a \in A \ , \ T(s', a, s) \neq 0 \right\}$$

Des macro-états constituant les états d'un nouveau PDM abstrait peuvent alors être définis, qui sont soit les états de sortie des régions [8] c'est à dire $\bigcup_{\mathcal{R}\in\Pi} Sper(\mathcal{R})$, soit les régions elles-mêmes [7] comme le montre la figure 4. Cependant, nous n'utilisons pas le premier modèle car il fournit des politiques sous-optimales qui ne prennent pas en compte d'éventuels comportements locaux consistant à aller chercher une récompense dans une région sans jamais en sortir, puisqu'il n'existe pas un macro-état absorbant par région permettant d'obtenir de tels comportements.

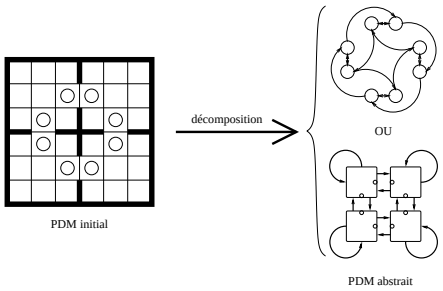

FIG. 4 – 2 PDM abstraits d'un PDM décomposé en 4 régions

5.2 PDM abstrait [7]

Supposons que nous disposons d'un certain nombre de politiques locales dans chaque région, nous pouvons alors construire un PDM abstrait $\langle S', A', T', R' \rangle$ dont les (macro)-actions sont ces politiques locales :
- $S' = \bigcup_{\mathcal{R}\in\Pi} \mathcal{R}$,
- $A' = \bigcup_{\mathcal{R}\in\Pi} \left\{ \pi_1^{\mathcal{R}}, \ldots, \pi_{m_{\mathcal{R}}}^{\mathcal{R}} \right\}$,

Pour chaque triplet $(\mathcal{R}, a, \mathcal{R}')$ dans $S' \times A' \times S'$, il existe une politique locale $\pi_j^{\mathcal{R}}$ telle que :
- $T'(\mathcal{R}, a, \mathcal{R}') = T(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}')$,
- $R'(\mathcal{R}, a, \mathcal{R}') = R(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}')$.

Le modèle de macro-transitions dépend donc du calcul des macro-probabilités $T(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}')$ et macro-récompenses $R(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}')$.

5.3 Calcul des macro-transitions

Les macro-probabilités sont les probabilités de transiter au bout d'un temps infini vers une région $\mathcal{R}'$ sachant que le robot se trouve au départ dans une région $\mathcal{R}$. Ce sont donc les moyennes mathématiques de transiter de chaque état interne de $\mathcal{R}$ vers chaque sortie vers $\mathcal{R}'$, en supposant équiprobables les états internes où se trouve le robot au départ :
- $T(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}') = \dfrac{1}{|\mathcal{R}|} \sum_{\substack{s'\in Sper(\mathcal{R})\cap\mathcal{R}' \\ s\in\mathcal{R}}} T(s, \pi_j^{\mathcal{R}}, s')$
- $R(\mathcal{R}, \pi_j^{\mathcal{R}}, \mathcal{R}') = \dfrac{1}{|\mathcal{R}|} \sum_{\substack{s'\in Sper(\mathcal{R})\cap\mathcal{R}' \\ s\in\mathcal{R}}} T(s, \pi_j^{\mathcal{R}}, s') \cdot R(s, \pi_j^{\mathcal{R}}, s')$

Les termes $T(s, \pi_j^{\mathcal{R}}, s')$ et $R(s, \pi_j^{\mathcal{R}}, s')$ sont calculés de manière itérative comme points fixes des équations :
- $T(s, \pi_j^{\mathcal{R}}, s') = \sum_{s''\in\mathcal{R}\cup Sper(\mathcal{R})} T(s, \pi_j^{\mathcal{R}}(s), s'') \cdot A_{s'', \pi_j^{\mathcal{R}}, s'}$

avec $A_{s'', \pi_j^{\mathcal{R}}, s'} = \begin{cases} T(s'', \pi_j^{\mathcal{R}}, s') & \text{si } s'' \in \mathcal{R} \\ 0 & \text{si } s'' \in Sper(\mathcal{R}) - \{s'\} \\ 1 & \text{si } s'' = s' \end{cases}$

- $R(s, \pi_j^{\mathcal{R}}, s') = \sum_{\substack{s''\in\mathcal{R}\cup\{s'\} \\ T(s'', \pi_j^{\mathcal{R}}, s')\neq 0}} T(s, \pi_j^{\mathcal{R}}(s), s'') \cdot B_{s, s'', \pi_j^{\mathcal{R}}, s'}$

avec $B_{s,s'',\pi_j^{\mathcal{R}},s'} = \begin{cases} R(s,\pi_j^{\mathcal{R}}(s),s'') + \gamma R(s'',\pi_j^{\mathcal{R}},s') & \text{si } s'' \in \mathcal{R} \\ R(s,\pi_j^{\mathcal{R}}(s),s') & \text{sinon} \end{cases}$

5.4 Génération des comportements locaux

Il existe au moins deux manières de générer les politiques locales des régions suivant l'avantage que l'on souhaite tirer de la décomposition de PDM.

Nous pouvons d'abord utiliser la décomposition pour résoudre un PDM initial de trop grande dimension en se ramenant à un PDM abstrait de plus petite dimension. Dans ce cas, nous souhaitons obtenir une politique optimale par région et non un ensemble de comportements locaux, soit en mettant à jour régulièrement la politique locale de chaque région au fur et à mesure de la résolution itérative du PDM abstrait [7], soit en calculant au préalable plusieurs politiques locales par région dont au moins une sera solution du PDM abstrait résolu une seule fois [8].

Une deuxième utilisation de la décomposition consiste à obtenir un ensemble de comportements locaux par région qui servent de données géographiques macroscopiques et condensées, disponibles à tout moment par l'opérateur s'il souhaite modifier momentanément le comportement macroscopique du robot, ou utilisables dans un modèle de PDM abstrait plus complexe, par exemple factorisé. C'est ce deuxième cas qui nous intéresse pour résoudre notre problème d'exploration multi-objectifs.

PDM locaux. Le calcul de plusieurs politiques locales dans chaque région nécessite de définir des PDM locaux paramétrables par des valeurs qui influent sur les politiques locales. De plus, ces politiques doivent permettre au robot soit de rester dans la région, soit d'en sortir par un ensemble de portes avec une probabilité de sortie pour chaque porte. Un PDM local $\langle S', A', T', R' \rangle^{(\lambda(s))_{s \in Sper(\mathcal{R})}}$ d'une région $\mathcal{R}$, représenté sur la figure 5, est donc défini par :

- $S' = \mathcal{R} \cup Sper(\mathcal{R}) \cup \{\alpha\}$, où α est un état absorbant,
- $A' = A$,
- $T(s,a,s') = \begin{cases} T(s,a,s') & \text{si } (s,s') \in \mathcal{R} \times (\mathcal{R} \cup Sper(\mathcal{R})) \\ 1 & \text{si } (s,s') \in (Sper(\mathcal{R}) \cup \{\alpha\}) \times \{\alpha\} \end{cases}$
- $R(s,a,s') = \begin{cases} R(s,a,s') & \text{si } (s,s') \in \mathcal{R} \times (\mathcal{R} \cup Sper(\mathcal{R})) \\ \lambda(s) & \text{si } (s,s') \in Sper(\mathcal{R}) \times \{\alpha\} \\ 0 & \text{si } (s,s') \in \{\alpha\}^2 \end{cases}$

Toute combinaison de pondérations non nulles sur certains états de sortie correspond à une politique locale optimale consistant à sortir par ces sorties avec, toutefois, une probabilité éventuellement non nulle de rester dans la région.

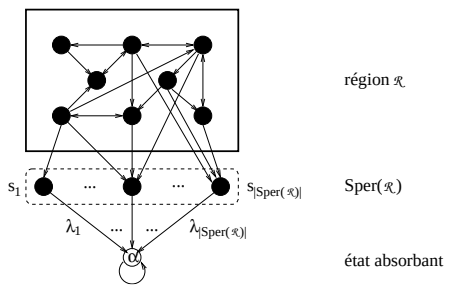

FIG. 5 – PDM local d'une région

Choix des politiques locales. Il est démontré [8] qu'une politique locale optimale d'une région pour une distribution de valeurs $(\lambda(s))_{s \in Sper(\mathcal{R})}$ égale à la valeur optimale du PDM initial sur les états de sortie correspondant est la restriction de la politique optimale du PDM initial à cette région. Cependant, nous ne connaissons qu'un encadrement de la valeur optimale V^* des états du PDM initial [8].

Nous devons donc générer un nombre exhaustif de politiques locales pour chaque combinaison possible de pondérations de sorte qu'au moins une politique locale est la restriction de la politique optimale du PDM initial à la région.

Une première méthode [8] consiste à construire une grille de valeurs de pondérations de pas δ suffisamment fine pour qu'une politique locale soit au moins $\frac{\delta}{2}$-optimale. Néanmoins, cette méthode est particulièrement inefficace car nous devons générer au moins $\prod_{s \in Sper(\mathcal{R})} \frac{V^*_{max}(s) - V^*_{min}(s)}{\delta}$ points de la grille et donc résoudre autant de PDM locaux dans cette région. De plus, nos tests ont révélé que plus de 99% des politiques calculées sont inutiles car plusieurs combinaisons de pondérations correspondent à la même politique locale.

Par conséquent, il est préférable de ne générer qu'une seule fois les politiques optimales communes à différentes combinaisons de pondération sur les sorties de la région [10]. En remarquant que la valeur des états pour une politique locale donnée est linéaire en fonction des pondérations sur les sorties, la valeur optimale des états en fonction des pondérations est une fonction convexe et linéaire par morceaux, telle que chaque morceau correspond à la politique dominante pour les pondérations de ce morceau (cf. figure 6). Ainsi, les différents morceaux de la fonction de valeur optimale peuvent être obtenus à l'aide d'un programme linéaire inspiré des algorithmes de résolution des PDM Partiellement Observables qui calcule les points de pondérations où la politique dominante peut être améliorée en créant un nouveau morceau linéaire [10]. Nous avons testé cette méthode qui s'est avérée être très intéressante puisque les politiques locales des régions sont obtenues en un temps polynomial en fonction du nombre de sorties par région : le PDM de la figure 4 a été décomposé en 0.23, 2.9 et 17.72 secondes par programmation linéaire contre 19.88, 58.89 et 92.78 secondes par discrétisation des valeurs de pondération pour respectivement 9, 25 et 49 états par région ($\gamma = 0.99$).

Enfin, une troisième méthode revient à générer des comportements locaux prédéterminés propres au problème que nous souhaitons résoudre mais qui ne sont pas nécessairement optimaux au sens du PDM initial. Dans le cas particulier d'un problème d'exploration, nous nous intéressons seulement aux politiques consistant à rester dans la région pour en recueillir la récompense ou à sortir par un ensemble de portes données afin d'aller chercher la récompense d'une autre région. Cette stratégie est a priori sous-optimale pour le problème de navigation pure car rien ne

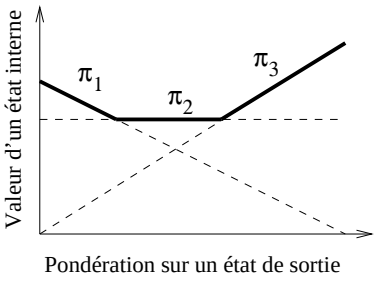

FIG. 6 – Valeur optimale d'un état interne d'une région contenant 3 politiques locales et une seule sortie

garantit que les politiques locales générées sont optimales mais elle suffit à obtenir des politiques d'exploration optimales, puisque optimales pour le PDM factorisé. C'est cette méthode que nous retenons pour notre modèle hybride de PDM.

6 Modèle hybride de PDM

Nous proposons ici un modèle de PDM mixte qui tire profit des deux modélisations précédentes de PDM, décomposée ou factorisée, puissantes mais qui s'appuient chacune sur une structure différente de l'espace d'états. La factorisation de PDM permet de structurer l'espace d'états en combinant des variables d'états de natures différentes, comme la position géographique du robot, les zones qu'il a déjà explorées ou encore son autonomie. Nous pouvons ainsi pallier à l'insuffisance d'un espace d'états entièrement explicité et uniquement géographique, où la mémoire des zones visitées est impossible.

Cependant, lorsque le robot doit aller explorer une zone particulière en évitant des dangers, cette technique ne traite pas efficacement le problème de navigation local dans le sous-ensemble d'états consitué des états géographiques seulement, car le nombre de valeurs possibles pour la variable d'état géographique est bien trop élevé. Nous devons donc agréger les états géographiques au sein de régions distinctes qui deviennent des macro-états géographiques à l'aide de techniques de décomposition de PDM. Des programmations dynamiques successives dans chaque région permettent alors d'obtenir des politiques locales visant chacune à récolter une récompense sur un état géographique donné de la région, et donc à explorer une zone donnée.

Les régions sont ensuite ajoutées à l'espace d'états factorisé en tant que valeurs possibles d'une même variable d'état géographique et leurs politiques locales sont les actions élémentaires du PDM factorisé. Notre modèle comprend donc un niveau de résolution géographique et un niveau de résolution factorisé.

6.1 PDM locaux et macro-PDM factorisé

Pour chaque région de la décomposition, nous construisons un PDM local dont les récompenses des transitions vers les sorties de la région sont paramétrables afin de générer plusieurs politiques locales (cf. figure 5). Nous ne souhaitons pas obtenir toutes les politiques locales possibles mais seulement celles qui permettent de recueillir les récompenses locales et de sortir par une ou plusieurs portes de la région. Par exemple, pour un ensemble de sorties données, nous avons besoin d'une seule politique parmi toutes celles qui font sortir le robot par toutes ces sorties. Si la région a m sorties, nous construisons donc 2^m politiques locales.

Nous calculons ensuite les macro-transitions correspondant à chaque politique locale, qui constituent notre modèle de transition entre chaque région. Pour l'exemple de la figure 1, nous obtenons le PDM abstrait géographique de la figure 7 où les flèches épaisses correspondent aux comportements locaux souhaités pour chaque politique locale.

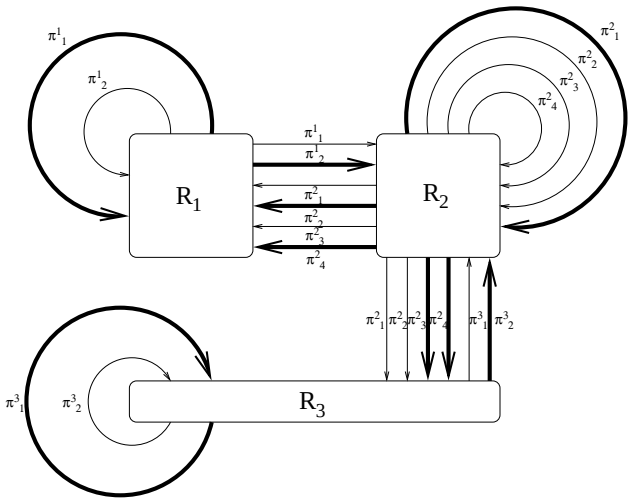

FIG. 7 – PDM abstrait géographique de la figure 1

La décomposition de l'espace d'états en région ainsi que le calcul des macro-transitions entre les régions sont effectués par un préprocesseur dont les sorties, c'est à dire les régions et leurs macro-transitions, sont injectées dans le modèle d'états et d'actions d'un macro-PDM factorisé.

6.2 Variables d'état du macro-PDM

Pour un problème d'exploration, l'espace d'états doit être décrit par au moins $|\Pi| + 2$ variables d'états où Π est l'ensemble des régions de la décomposition. En effet, outre les variables *région* et *autonomie*, nous avons besoin d'une variable binaire par région qui indique si l'objectif de la région a été atteint ou non. Ce sont précisément ces variables qui permettent de commuter entre les différentes politiques locales d'une région suivant que son objectif a été réalisé ou non, et donc de rester ou de sortir de la région pour aller vers une région où l'objectif n'a pas encore été réalisé.

D'autre part, la variable *région* est discrète et prend autant de valeurs que le nombre de régions de la décomposition. Nous voyons donc ici tout l'avantage de la décomposition en régions : si nous avions une variable *état élémen-*

taire géographique qui aurait autant de valeurs possibles que d'états élémentaires géographiques, chaque arbre de probabilité aurait énormément de branches et nous aurions beaucoup plus d'arbres de probabilité, à savoir un de plus par état géographique. Nous serions alors confronté à un problème tant de complexité informatique que de difficulté de modélisation.

L'autonomie, quant à elle, est a priori une variable d'état continue. Cependant, les modèles factorisés de PDM que nous connaissons nécessitent une discrétisation des variables qui, comme pour les états géographiques, serait impossible pour l'autonomie tant le nombre de valeurs possibles est élevé. De plus, la modélisation d'un PDM factorisé avec des variables continues n'est pas une simple extension du modèle discret puisque les transitions entre les variables d'état ne peuvent plus être représentées par des arbres. Nous souhaitons donc étudier prochainement la modélisation et la résolution de PDM factorisés par des variables continues pouvant contenir des variables discrètes. Nous pensons nous inspirer de travaux sur la modélisation et la résolution de PDM Partiellement Observables factorisés [6]. Aussi, nous supposons actuellement que l'autonomie du robot à un instant donné est une variable binaire : soit elle est suffisante pour que le robot continue la mission, soit elle est ne l'est pas et le robot s'arrête où il est.

Pour un problème d'exploration, les variables d'états sont donc au moins :

– R : région où se trouve le robot (discrète)
– $(O_i)_{1\leq i\leq|\Pi|}$: objectifs atteints (binaires)
– A : autonomie du robot (binaire)

Suivant la mission exacte à réaliser, nous pouvons rajouter des variables d'état spécifiques à la mission telles l'état des communications radio avec l'opérateur, la météo extérieure ou encore l'état général du robot.

6.3 Actions du macro-PDM

Les actions élémentaires du macro-PDM sont les macro-actions issues de la décomposition de l'espace d'états en région : ce sont donc les politiques locales générées pour chaque région. Contrairement au formalisme général de PDM factorisé proposé par [5], pour lequel chaque action est applicable dans tous les états, les macro-actions ne sont applicables que dans la région où elles ont été générées.

Ceci simplifie sensiblement le modèle bayésien de transitions car, pour une politique locale donnée, les arbres de probabilité des régions autres que celles où cette politique est définie ou peut mener sont du type no-ops, c'est à dire qu'ils n'ont que deux branches de probabilités 1 et 0 indiquant que la probabilité d'obtenir ces régions est inchangée. Pour la variable R, nous avons donc un seul arbre de probabilité de niveau supérieur à 1 par macro-action au lieu de $|\Pi|$. De même, pour une macro-action donnée, les objectifs des régions où n'est pas définie la macro-action ont un arbre de probabilité du type no-ops : soit ces objectifs sont déjà atteints et dans ce cas ils le restent, soit ils ne sont pas encore atteints et il est impossible qu'ils le soient avec cette macro-action.

Ainsi, même si l'ensemble $\left(\pi_j^S\right)_{1\leq j\leq 2^{|Sper(S)|}}^{S\in\Pi}$ des macro-actions a un cardinal de $\sum_{S\in\Pi} 2^{|Sper(S)|}$, le nombre total d'arbres de probabilités générés pour toutes les actions est relativement faible.

6.4 Modèle de transition du macro-PDM

Dépendance des variables. Pour les variables *objectifs*, la dépendance vis-à-vis des autres variables dépend des macro-actions. En effet, pour une macro-action donnée, seul l'objectif de la région où la macro-action est définie est susceptible d'être atteint. Cet objectif dépend donc de sa valeur précédente et de la région où le robot se trouve alors que les autres objectifs, que le robot ne peut pas atteindre avec cette macro-action, ne dépendent que de leur valeur précédente. Dans le cas de l'exemple de la figure 7, les dépendances des variables à un instant donné vis-à-vis des variables à l'instant précédent pour les politiques définies dans la région R_1 sont représentées par le réseau d'action de la figure 8.

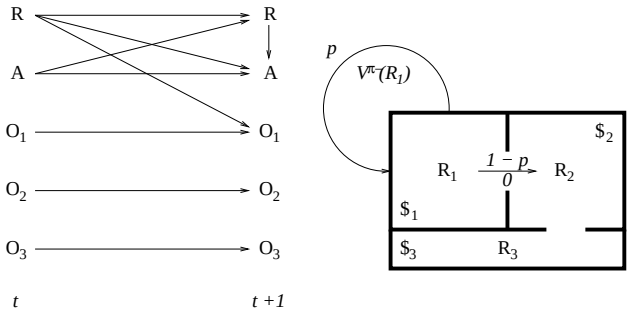

FIG. 8 – Dépendances des variables du macro-PDM factorisé et transitions du PDM géographique décomposé pour les politiques locales $\left(\pi_j^1\right)_{1\leq j\leq 2}$ de la région R_1

Arbres de probabilité. Toutes les régions inaccessibles pour une politique donnée, comme la région de sortie de la figure 7 pour les deux politiques locales $\left(\pi_j^1\right)_{1\leq j\leq 2}$ ont un arbre de probabilité du type no-ops. Pour les autres régions, la probabilité de les atteindre dépend de l'autonomie restante du robot et de la région où se trouve le robot avant d'appliquer la politique locale. Ainsi, dans le cas de notre exemple et pour les politiques de la région R_1, le modèle de transition des valeurs possibles de la variable R est représenté dans la figure 9. Nous remarquons d'ailleurs que l'arbre de R_3^{t+1} est bien le complémentaire des arbres de R_1^{t+1} et R_2^{t+1} puisque la somme des probabilités d'obtenir R_1^{t+1}, R_2^{t+1} ou R_3^{t+1} partant d'un même sous-ensemble d'états est égale à 1.

Puisque le robot ne peut atteindre que l'objectif de la région où il se trouve, les arbres de probabilité des variables $(O_i)_{1\leq i\leq|\Pi|}$ sont du type no-ops sauf celui de l'objectif correspondant à la région où est le robot. Aussi, l'arbre de probabilité de cet objectif dépend de sa valeur précédente

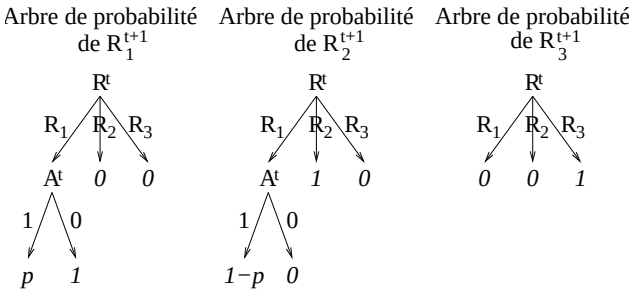

FIG. 9 – Arbres de probabilité des différentes valeurs de la variable R pour les politiques locales $\left(\pi_j^1\right)_{1\leq j\leq 2}$ de la région R_1

– objectif atteint ou non – et de la région où il se trouve. De plus, nous considérons que la probabilité de l'atteindre est la même que celle de rester dans sa région puisqu'il correspond à la seule zone absorbante de la région. La figure 10 représente les arbres de probabilité des objectifs de notre exemple à trois régions pour les politiques locales de la région R_1. Nous n'avons représenté que les arbres de probabilité de la valeur *atteint* car ces variables sont binaires si bien que les arbres de la valeur *non atteint* sont les complémentaires probabilistes des arbres représentés.

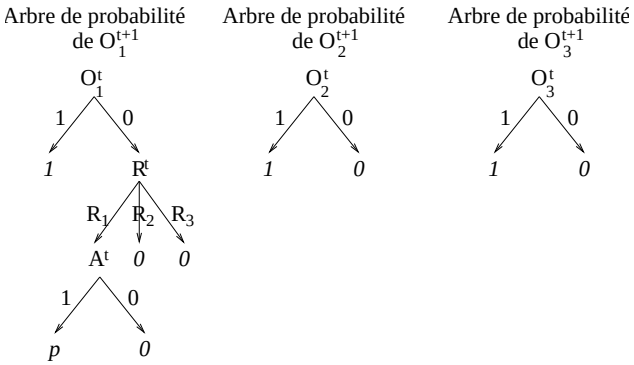

FIG. 10 – Arbres de probabilité des variables binaires $(O_i)_{1\leq i\leq|\Pi|}$ pour les politiques locales $\left(\pi_j^1\right)_{1\leq j\leq 2}$ de la région R_1

Enfin, le modèle de transition pour l'autonomie dépend du robot car la consommation d'énergie est fonction de l'énergie utilisée et de l'architecture globale du système, à savoir le nombre de modules à alimenter et la consommation de chaque module. L'arbre de probabilité de l'autonomie est donc paramétré par une fonction $f\left(R^t, R^{t+1}, \pi\right)$ qui indique la consommation du robot pour un déplacement donné par la région de départ, la région d'arrivée et la longueur du parcourt, dépendant de la politique locale π effectuée. Cet arbre est représenté dans le schéma de gauche de la figure 11 pour les politiques locales de la région R_1.

Arbre de récompense. Le modèle de récompense est primordial car c'est lui qui doit permettre au robot de chan-

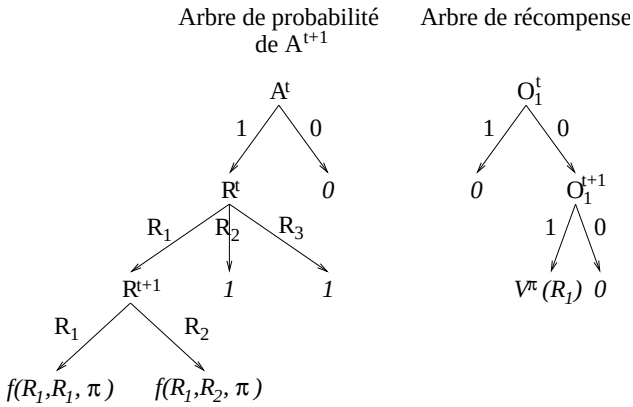

FIG. 11 – Arbre de probabilité de la variable A et arbre de récompense pour les politiques locales $\left(\pi_j^1\right)_{1\leq j\leq 2}$ de la région R_1

ger de politique locale lorsqu'il a atteint l'objectif d'une région. Le robot ne doit pouvoir recevoir la récompense d'une région que s'il ne l'a pas encore obtenue. S'il a déjà atteint l'objectif d'une région, il ne pourra plus recevoir la récompense de cet objectif de sorte qu'il sélectionne une politique locale qui l'oblige à sortir de la région et l'amène vers l'objectif non atteint d'une autre région. Ainsi, la récompense reçue par le robot suite à l'application d'une politique locale dans une région donnée dépend de la valeur de l'objectif de cette région avant et après l'application de la politique locale, comme le montre le schéma de droite de la figure 11.

6.5 Algorithme de résolution

La résolution proposée du problème d'exploration repose d'abord sur un préprocesseur qui décompose l'espace d'états géographiques en régions contenant chacune un objectif intermédiaire à accomplir. Cette décomposition est basée sur la théorie des graphes dans [12], mais d'autres techniques (géométriques) sont envisageables. Puis on construit pour chaque région un PDM local qui lui permet de générer, par résolutions successives du PDM local avec diverses pondérations sur les sorties de la région, toutes les politiques locales qui consistent chacune à sortir simultanément par une combinaison différente de sorties.

La résolution globale est ensuite faite au niveau d'un PDM factorisé contenant les régions de la décomposition, leurs politiques locales comme actions élémentaires, les objectifs atteints ainsi que l'autonomie restante du robot comme autres variables d'état. L'approche proposée donne l'algorithme décrit en figure 12. Ainsi, les techniques de résolution des PDM décomposés et factorisés sont utilisées respectivement par le préprocesseur et le solveur.

Nous avons testé cet algorithme sur l'exemple de la figure 1 dont la solution prend trop de place pour être présentée ici. Le PDM géographique, décomposé en 0.23 secondes par programmation linéaire, a produit 7, 6 et 2 politiques

Préprocesseur

```
Partitionner l'espace d'états en régions
Pour chaque région
    Construire PDM local
    Pour chaque combinaison de sorties
        Résoudre PDM local
        Calculer macro-transitions
```

Solveur

```
Construire PDM factorisé vide
Ajouter variables 'régions','objectifs','autonomie'
Ajouter actions 'politiques locales des régions'
Générer arbres de probabilités et de récompenses
Résoudre PDM factorisé
```

Solution

```
Pour chaque état factorisé
    Identifier région et politique locale optimale
    Pour chaque état géographique de la région
        Appliquer action de la politique locale
```

FIG. 12 – Algorithme de résolution d'un modèle hybride de PDM pour les problèmes d'exploration

locales pour les régions respectivement R_1, R_2 et R_3. Le macro-PDM factorisé a ensuite été construit en moins de 0.01 secondes et résolu en 2.31 secondes par itération de la politique. De plus, la compacité de la solution est apparente puisque 31 macro-états parmi 48 sont explicitement énumérés dans l'arbre de politique, ce qui représente moins de deux tiers des états.

7 Conclusion

Dans cet article, nous avons posé un problème d'exploration en environnement mal connu pour un système autonome et envisagé sa résolution. Nous proposons un modèle hybride en planification probabiliste fondé sur des travaux récents sur les PDM et consistant à utiliser des représentations factorisées par variables d'état pour certaines composantes du problème, et des représentations extensives et énumératives des états pour d'autres composantes (la navigation par exemple). La distinction entre les différentes composantes de l'espace d'états repose sur des critères de dépendances et d'indépendances probabilistes (corrélations) entre variables, ainsi que sur des critères de compacité et de structure des représentations obtenues : lorsque le nombre de valeurs possibles d'une variable est trop grand, ou bien lorsque deux variables sont fortement liées par des contraintes ou des corrélations, il vaut mieux garder une représentation en extension. Nous avons ainsi commencé à tester certaines techniques de décomposition de l'espace d'états permettant de réduire la complexité du problème dans la partie "énumérée" de l'espace d'états. Nous proposons enfin un algorithme permettant de combiner simplement ces techniques de décomposition avec les techniques de factorisation exposées précédemment. Ces travaux seront poursuivis dans la voie de l'amélioration de l'efficacité de notre approche par rapport à un PDM classique totalement énuméré, qui reste difficile à modéliser et résoudre en raison de sa grande taille.

Références

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

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-dynamic Programming* Athena, 1996.

[3] C. Boutilier, R. I. Brafman and C. Geib, *Structured Reachability Analysis for Markov Decision Processes* Uncertainty in Artificial Intelligence : Proceedings of the Fourteenth Conference (UAI-1998), pp. 24-32, 1998.

[4] C. Boutilier, T. Dean and S. Hanks, *Decision-Theoretic Planning : Structural Assumptions and Computational Leverage* Journal of Artificial Intelligence Research, Vol. 11, pp. 1-94, 1999.

[5] C. Boutilier, R. Dearden and M. Goldszmidt, *Stochastic dynamic programming with factored representations* Artificial Intelligence, Vol. 121, pp. 49-107, 2000.

[6] C. Boutilier and D. Poole, *Computing Optimal Policies for Partially Observable Decision Processes Using Compact Representations* Proceedings of the 13^{th} National Conference on Artificial Intelligence, pp. 1168-1175, 1996.

[7] T. Dean and S.-H. Lin, *Decomposition Techniques for Planning in Stochastics Domains* Reinforcement and Markov Models : Proceedings of the Fourteenth International Joint conference on Artificial Intelligence (IJCAI-1995), pp. 1121-1129, 1995.

[8] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean and C. Boutilier, *Hierarchical Solution of Markov Decision Processes using Macro-actions* Uncertainty in Artificial Intelligence : Proceedings of the Fourteenth Conference (UAI-1998), pp. 220-229, 1998.

[9] W. S. Lovejoy, *A survey of algorithmic methods for Partially Observed Markov Decision Processes* Annals of Operation Research, Vol. 28, pp. 47-66, 1991.

[10] R. Parr, *Flexible Decomposition Algorithms for Weakly Coupled Markov Decision Problems* Uncertainty in Artificial Intelligence : Proceedings of the Fourteenth Conference (UAI-1998), pp. 422-430, 1998.

[11] M. L. Puterman, *Markov Decision Processes* John Wiley & Sons INC, 1994.

[12] R. Sabbadin, *Graph partitioning techniques for Markov Decision Processes decomposition* Proceedings of the 15^{th} European Conference on Artificial Intelligence, pp. 670-674, 2002.

[13] G. Verfaillie, F. Garcia and L. Péret, *Deployment and Maintenance of a Constellation of Satellites : a Benchmark* Workshop on Planning under Uncertainty and Incomplete Information of the 13^{th} International Conference on Automatic Planning and Scheduling, 2003.