Stage de M2R (M2R SAID 2005)
Projet de Fin d'Etudes (SUPAERO)

Coordination multi-robots terrestre et aérien

Emmanuel RACHELSON
sous la direction de
Patrick FABIANI

Stage effectué à l'ONERA/DCSD Toulouse

26 août 2005

Remerciements

Je tiens tout particulièrement à remercier :

Florent Teichteil, pour m'avoir supporté dans son bureau pendant la durée du stage, pour m'avoir bien aidé et pour la période de cohabitation qui nous reste jusqu'à sa soutenance de thèse et pendant le début de la mienne. Bon courage !

Patrick Fabiani, pour son encadrement et son conseil efficace ainsi que pour m'avoir accueilli à l'ONERA.

Jean-Loup Farges, pour son soutien et son suivi discret et efficace.

Caroline Chiappa pour son encadrement depuis SUPAERO.

L'ensemble du personnel du DCSD pour les quelques mois passés et par avance pour les trois ans à venir.

Les doctorants du DCSD pour l'accueil sympathique.

Table des matières

1 Introduction **7**

2 Contexte et outils théoriques **10**
2.1 Les MDPs – Notions générales 10
2.2 Algorithmes de recherche d'une solution optimale 17
2.2.1 Programmation linéaire 17
2.2.2 Itération de la valeur 18
2.2.3 Itération de la politique 20
2.3 Introduire le temps comme paramètre du MDP 22
2.3.1 Modèle probabiliste du processus 22
2.3.2 Modèle de récompense 23
2.3.3 Fonction de valeur 24
2.3.4 Politique optimale 26
2.4 L'approche décomposée, une approche hiérarchique de l'espace d'état 26
2.4.1 La structure du MDP décomposé 27
2.4.2 MDP abstrait et macro-actions 28
2.4.3 L'approche « ϵ-grid » 29
2.4.4 Recherche dans l'espace des valeurs des états de sortie 31
2.4.5 L'encadrement par enveloppe convexe 34
2.4.6 La décomposition en étoile 37
2.4.7 Un algorithme de décomposition automatique 37
2.5 La décomposition MAXQ, une représentation hiérarchisée de la mission 37
2.5.1 Le graphe MAXQ 37
2.5.2 La résolution du problème sous forme de graphe MAXQ 38
2.5.3 Le critère γ-pondéré et l'introduction du temps dans la représentation MAXQ 41
2.6 Discussion sur le lien entre décomposition et graphe MAXQ 42
2.7 L'approche factorisée 45
2.8 Les systèmes multi-agents et l'approche décentralisée 46

3 Le problème multi-agents considéré **47**
3.1 Présentation de la problématique et ébauche de solution 47
3.2 Détail d'un cas de test : le problème des deux taxis 50
3.2.1 Présentation du problème à deux agents 50
3.2.2 Modèle de l'univers 54
3.3 Une solution considérant le temps comme variable continue 54
3.3.1 Les dates, conséquences des durées 55
3.3.2 Un modèle de SMDP avec dates 56
3.3.3 Le cas particulier des récompenses linéaires par morceaux - un exemple . 57

3.3.4 Une tentative de simplification par séparation des variables - illustration de la complexité du problème . . . 62
3.3.5 L'intégration du modèle à temps continu dans la recherche d'une politique conjointe optimale . . . 66
3.4 Résoudre les problèmes d'amélioration d'une politique commune sans introduire de variable temporelle continue . . . 66
3.4.1 Contexte et modèle . . . 66
3.4.2 Introduction du facteur temps . . . 67
3.4.3 Algorithme de résolution dans le cadre des deux agents . . . 68
3.4.4 Replanification en cours de mission . . . 70
3.5 Le problème de navigation d'un drone hélicoptère et d'un robot terrestre . . . 71
3.5.1 Le drone hélicoptère . . . 71
3.5.2 Le robot terrestre . . . 77
3.5.3 Remarques sur le déroulement d'une mission . . . 81
3.5.4 Représentation factorisée d'un problème . . . 82
3.5.5 Récapitulatif d'exécution de l'algorithme . . . 93

4 Conclusion **96**

A Démonstration de la solution au problème du passage d'une distribution sur une grille de p éléments à une grille à k éléments **98**

B Complexité de l'algorithme d'itération de la valeur **104**

Chapitre 1

Introduction

Le LAAS, le CTA et l'ONERA sont impliqués dans un programme visant à faire coopérer un robot terrestre et un robot aérien pour une mission d'exploration en environnement urbain. A partir de caméras embarquées et de traitement d'images appropriés, chaque robot trouve, au fur et à mesure de l'exploration, des zones de texture homogène plus ou moins traversables par le robot terrestre. Les deux robots sont complémentaires car le robot aérien peut voir partout mais grossièrement alors que le robot terrestre se déplace plus lentement mais voit de manière plus fine et surtout est le seul à devoir se rendre à un point donné pour réaliser une opération. L'ONERA s'intéresse à la conception des modules décisionnels des deux engins.

L'étude entreprise lors de ce stage de M2R porte donc sur les mécanismes décisionnels adaptés aux systèmes multi-agents. Le travail effectué entre dans le cadre des Processus Décisionnels de Markov (MDP), modèle de décision de plus en plus utilisé aujourd'hui afin de résoudre les problèmes de plannification dans un contexte incertain. Dans un premier temps, un tour d'horizon des différentes techniques et approches utilisées dans la littérature a été entreprise. Dans ce cadre, on a exploré les bases mathématiques des MDP et les algorithmes de résolution standards, puis on on a cherché à élargir le modèle afin d'enrichir la représentation de la réalité et on s'est intéressé aux Processus Décisionnels Semi-Markoviens (SMDP). Ensuite, les algorithmes de décomposition ont été abordés pour la structuration de l'espace d'état qu'ils proposent et la décomposition MAXQ a été explorée pour les possiblités de décision hiérarchisée qu'elle apporte. Enfin, les approches factorisées et les différentes approches décentralisées et distribuées ont été abordées. Ces diverses approches et développements sont présentés et discutés dans une première partie du rapport. Le lien avec la problématique qui nous intéresse se trouve dans l'utilisation possible de ces approches dans le cadre de notre résolution ainsi que dans la diversité des points de vue sur la manière de résoudre un problème de décision de façon optimale dans un cadre Markovien et dans les alternatives qui existent aux méthodes exactes qui peuvent vite se révéler peu performantes.

Le problème bi-agents posé s'inscrit dans un cadre multi-agents qui a été traité dans la littérature de façon assez diverse. Le travail durant le stage a été concentré sur l'impact d'une nécessaire coordination entre les agents sur la modélisation de l'univers, l'espace d'état et les algorithmes de résolution. Le cadre de la décision multi-agents en coopération, de façon décentralisée, distribuée et en ligne a été exploré au travers de la littérature et d'exemples concrets afin de construire une solution au problème posé et des cas de test - problèmes voisins représentatifs des problématiques et comportements type - ont été construits. Afin de résoudre le problème posé, deux modèles ont enfin été proposés autour d'un même algorithme de résolution afin de représenter l'univers et le problème de coopération. Le premier modèle

intègre une notion d'un temps traité comme une variable continue tandis que le second, plus orienté vers une implémentation rapide, envisage un temps discret avec un minimum de perte d'optimalité dans le calcul des politiques. Chacun de ces modèles s'insère dans un algorithme coopératif qui met en évidence l'importance primordiale du protocole de communication entre les agents. Le premier modèle exploite au maximum les conséquences de la notion de coopération et la nécessaire introduction du temps. Cette exploration des mécanismes décisionnels plongés dans un modèle à temps continu débouche sur certaines pistes intéressantes mais ses limites appraissent assez vite et assez clairement dans la complexité du traitement. Le second modèle est plus simple d'approche et synthétise l'expérience acquise avec le premier et la connaissance des différentes approches envisagées en première partie. C'est ce modèle qui pourra donner lieu à une implémentation en début des travaux de thèse et qui semble le plus prometteur comme compromis - encore améliorable - entre complexité et performances.

Enfin, l'algorithme de recherche des politiques optimales pour deux agents en coopération - algorithme qui au fur et à mesure de cette étude, a pris le nom d'itération de la chronologie - cet algorithme constitue une approche assez différente de ce qui existe dans la littérature et ouvre des portes sur la possibilité de résoudre des problèmes de taille conséquentes pour des agents en contexte coopératif, quels qu'ils soient. C'est la synthèse des approches de hiérarchisation des espaces d'état et d'action vues en première partie, des approches de simplification et d'exploration des espaces des fonctions de valeurs, et enfin de la réflexion sur les implications des processus coopératifs qui a mené à la formalisation de cet algorithme. L'idée de l'itération de la chronologie, si elle semble permettre un traitement simple des problèmes de coopération a encore de nombreuses améliorations possibles et demande à être renforcée, prouvée et optimisée.

Les modèles et algorithmes qui ont été développés durant le stage constituent la base d'un travail que je souhaite poursuivre en thèse, en effet, au delà des développements proposés durant le stage, de nombreuses possibilités d'améliorations ont été abordées et des modèles plus fins et plus efficaces ont été évoqués et doivent être approfondis.

Fig. 1.1 – Le drone hélicoptère Yamaha R-MAX utilisé à l'ONERA

Chapitre 2

Contexte et outils théoriques

Ce chapitre regroupe plusieurs sections de nature très différente. L'objectif est d'introduire les différentes idées qui ont donné naissance aux modèles et algorithmes présentés au chapitre suivant. Un second objectif est de discuter des avantages, inconvénients et ouvertures proposées par les différentes approches.

La première section introduit le cadre général de la théorie de Processus Décisionnels de Markov, rappelant les principaux résultats et les mécanismes fondamentaux. La seconde section est une suite logique puisqu'elle rappelle les différentes variantes de résolution des problèmes sous forme MDP par programmation dynamique. La troisième partie concerne un problème qui se pose très tôt quand on considère une problèmatique de plannification complexe : il s'agit de gérer la durée des actions. Le modèle SMDP y est développé et sera repris dans les modèles développés plus loin comme base mathématique du traitement du temps et des durées. La quatrième section introduit les algorithmes de décomposition de l'espace d'état. Ces algorithmes représentent, outre une méthode de résolution efficace, la base de l'idée de la hiérarchisation des actions et des stratégies. Elle introduit donc une autre approche complémentaire : le graphe MAXQ présenté à la cinquième section. Dans la sixième section, on discute, dans le cadre du problème posé mais aussi dans le cadre général, des complémentarités et incompatibilités des deux approches. Tous ces développements ont pour buts d'introduire toutes les influences et les outils qui seront ré-utilisés lors de la construction des modèles et des algorithmes aux chapitre suivant. Quelques remarques sur l'approche factorisée et sur les approches multi-agents achèvent ce tour d'horizon des approches étudiées qui ont donné naissance à la contribution présentée au chapitre suivant.

2.1 Les MDPs – Notions générales

Afin de trouver le meilleur comportement pour un agent donné dans une situation donnée, il faut définir un certain nombre de notions qui restent pour l'instant intuitives. Pour commencer, il s'agit de définir le système sur lequel on travaille et donc de délimiter clairement l'agent et son environnement. On s'intéressera ici à des agents indépendants, dotés d'une certaine autonomie décisionnelle, d'un objectif, et se trouvant plongés dans un univers où le résultat de leurs actions comporte une part d'incertain et où l'état global du système n'est pas nécessairement entièrement accessible à l'agent. Dans ce contexte, l'agent va être amené à prendre une série de décisions, que l'on souhaite être les meilleures, afin de mener à bien sa mission.

A titre d'exemple, on peut considérer un drône hélicoptère affecté d'une mission d'exploration. Le résultat de ses actions de déplacement contient une part d'incertain (vent, méconnaissance du terrain) et son état ne lui est pas nécessairement entièrement accessible (pas de mesure de position par exemple). On peut également considérer l'exemple plus simple d'un robot mobile dans un labyrinthe inconnu ayant comme mission d'aller chercher une récompense à un endroit donné du labyrinthe.

Dans ce qui suit, on s'attache à passer de l'intuition à la formalisation en introduisant le contexte et les notations qui vont être utilisés par la suite. On s'attache en particulier à définir de façon claire et naturelle les mots-clés : *état*, *action*, *historique*, *récompenses et coûts*, *probabilités de transition*, *fonction ou modèle de transition*, *Propriété de Markov*, *Processus Décisionnels de Markov*, *stationnarité*, *politique*, *fonction de valeur*, *équation de Bellman* et *erreur de Bellman*.

En définissant le contexte de l'agent, on a introduit un certain nombre de mots de vocabulaire intuitifs auxquels il faut maintenant fournir un cadre. On est donc mené à définir les notions suivantes :

Définition 1 (état). *L'*état *d'un agent à un instant donné est constitué des valeurs et paramètres décrivant les variables connues par l'agent sur lui-même et son environnement.*

L'espace d'état est généralement noté S et peut être de dimension infinie, cependant on se placera par la suite dans le cas d'un espace d'état dénombrable.

Définition 2 (action). *Chaque agent est capable d'effectuer un certain nombre d'*actions *dont le but est de l'emmener d'un état vers un autre en vue d'une* récompense.

Définition 3 (probabilité de transition). *L'interaction de l'agent avec l'univers n'est pas déterministe, elle est décrite par des distributions de probabilités ; on considère donc pour chaque état s_{t+1} la probabilité d'y arriver par l'action a_t connaissant l'historique h_t des états et actions jusqu'à t. On écrit cette probabilité $P_{t+1}(s_{t+1}|h_t, a_t)$.*

Définition 4 (récompense). *Afin de quantifier la notion de « meilleure action » on associe aux transitions, aux états ou aux actions des valeurs de récompenses et/ou de coût. L'agent doit prendre des* décisions*, suites d'*actions*, compte tenu des* récompenses *et des* coûts *associés pour lui à chaque action, transition et état.*

Afin de modéliser des problèmes décisionnels séquentiels et stochastiques comme ceux présentés ici, les *Processus Décisionnels de Markov* (MDP) [FFG01] fournissent un cadre adapté aussi bien au calcul hors ligne qu'en ligne.

Définition 5 (MDP). *Un MDP est défini comme un quadruplet $< S, A, T, R >$ vérifiant la propriété de Markov, soit :*

- *S l'ensemble des états,*
- *A l'ensemble des actions dont l'agent est capable,*
- *T est le modèle de transition, défini comme :*
  $$T : \begin{cases} S \times A \times S & \rightarrow & [0;1] \\ (s,a,s') & \mapsto & T(s,a,s') = P(s'|s,a) \end{cases}$$
- *R est une fonction de récompense et/ou de coût qui, dans le cas le plus général, est définie comme :*
  $$R : \begin{cases} S \times A \times S & \rightarrow & \mathbb{R} \\ (s,a,s') & \mapsto & R(s,a,s') = R(s'|s,a) \end{cases}$$

On notera que la fonction de transition ne dépend que du dernier état et de la dernière action et non de l'historique des états/actions, ce qui est conforme à la *propriété de Markov* : $P(s_{t+1}|h_t, a_t) = P(s_{t+1}|s_t, a_t)$.

Définition 6 (propriété de Markov). *Soit un système dont l'évolution est soumise à des lois aléatoires. Soit S son espace d'état et X_t l'état du système à l'instant t. La probabilité d'obtenir l'état x ($x \in S$) à l'instant $t+1$ est notée $P_{t+1}(< X_{t+1} = x >)$. On dit que ce système vérifie la* propriété de Markov *si, quel que soit l'instant t considéré, cette probabilité ne dépend que de l'état à l'instant précédent X_t. Cela s'écrit :*

$$P_{t+1}(< X_{t+1} = x >) = P_{t+1}(X_{t+1} = x|X_t)$$

Comme énoncé plus haut, cela implique entre autres que la probabilité de transition ne dépend pas de l'historique des actions et états. Comme par ailleurs on se place dans un cadre stationnaire (les distributions de probabilités sont indépendantes du temps), on retrouve directement l'égalité :

$$P_{t+1}(s_{t+1}|h_t, a_t) = P(s_{t+1}|s_t, a_t)$$

Définition 7 (stationnarité). *Un processus est dit* stationnaire *si les fonctions le décrivant sont indépendantes du temps.*

Pour un MDP cela correspond bien à la l'indépendance du modèle de transition par rapport au temps.

Dans le cadre du MDP ainsi défini, on cherche à définir la meilleure manière d'agir pour l'agent. On définit ainsi deux notions supplémentaires :

Définition 8 (politique). *Une* politique*, ou* stratégie *est une procédure de contrôle de l'agent. Elle est définie par une fonction π_t qui fournit, dans toutes les situations, l'action à appliquer par l'agent.*
En notant h_t l'historique des i actions exécutées et des i états traversés jusqu'à t, on a :

$$\pi_t : \left\{ \begin{array}{ccc} S^i \times A^i & \rightarrow & A \\ h_t & \mapsto & \pi_t(h_t) = a_t \end{array} \right.$$

Certaines politiques sont dites aléatoires car elles affectent à l'historique non pas une action mais une distribution de probabilité sur les actions de A. On note alors :

$$\pi_t : \left\{ \begin{array}{ccc} S^i \times A^i & \rightarrow & \mathcal{P}(A) \\ h_t & \mapsto & \pi_t(h_t) = q_{\pi_t} \end{array} \right.$$

avec $\mathcal{P}(A)$ l'ensemble des distributions sur A.

L'ensemble des politiques pour un problème donné est noté Π. On n'utilisera pas les politiques aléatoires dans cette étude, l'incertitude sur le résultat des actions étant déjà introduite par le modèle de transition.
On s'intéresse de plus à des politiques stationnaires, c'est-à-dire dont l'expression ne dépend pas de l'instant d'exécution, on a donc $\pi_t(h_t) = \pi(h_t)$.

La « meilleure » manière d'agir ou la meilleure politique pour l'agent est un critère subjectif qu'il faut quantifier. Cela revient en général à estimer les gains qu'on espère obtenir en suivant la politique en question.

On peut définir différents critères basés sur la fonction de récompense et l'état initial ; ces critères sont utilisés pour comparer de façon quantitative des politiques, ils définissent des *fonctions de valeur* des politiques :

Définition 9 (fonction de valeur). *Etant donné un critère C d'évaluation de la politique et une politique π, une fonction de valeur V^π est une application de S dans $\mathbb{R}$ qui associe à chaque état s ($s \in S$) la valeur du critère C si on applique π en partant de s.*

$$V^\pi : \begin{cases} S & \rightarrow & \mathbb{R} \\ s & \mapsto & C(\pi, s) \end{cases}$$

Les critères couramment utilisés sont :

- le critère fini optimise les gains potentiels de l'agent sur un horizon donné. En posant r_t^π la récompense obtenue à l'instant t en suivant la politique π à partir de l'état inital s_0, le critère fini s'exprime :

$$V_N^\pi = E\left(\sum_{t=0}^{N-1} r_t^\pi | s_0\right)$$

- le critère γ-pondéré prend en compte les gains sur un horizon infini avec un facteur de rabais γ ($\gamma \in [0;1]$)sur les gains éloignés dans le temps. Ce facteur γ assure notamment toujours de l'existence du critère. Le critère s'écrit :

$$V_\gamma^\pi = E\left(\sum_{t=0}^{\infty} \gamma^t r_t^\pi | s_0\right)$$

- le critère total est le cumul des gains atteignables sur un horizon infini avec la politique π, ce critère peut éventuellement être infini. Il s'exprime :

$$V_1^\pi = E\left(\sum_{t=0}^{\infty} r_t^\pi | s_0\right)$$

- le critère moyen estime le gain moyen par unité de temps qu'on peut espérer obtenir sur un horizon infini. Il s'exprime :

$$V_m^\pi = \lim_{n\rightarrow\infty} \frac{1}{n} E\left(\sum_{t=0}^{n-1} r_t^\pi | s_0\right)$$

On considère également l'ensemble $\mathcal{V}$ des fonctions de S dans (R). On munit cet ensemble de la norme max : $\| V \| = max_{s\in S} \mid V(s) \mid$, ainsi que d'un ordre partiel :

$$\forall\, U, V \in \mathcal{V}, \quad U \leq V \Leftrightarrow \forall\, s \in S \quad U(s) \leq V(s)$$

Cela nous permettra plus tard de définir une fonction max sur $\mathcal{V}$.

En général, on retient le critère γ-pondéré pour la définition de la fonction de valeur. C'est le choix que l'on fait ici aussi dans la mesure où l'on se situe dans une problématique à horizon infini. On ne considèrera que ce critère par la suite et on ometra l'indice γ : $V^\pi = V_\gamma^\pi$. Ce choix se justifie par ailleurs par l'interprétation physique suivante : le facteur γ peut être vu d'un point de vue physique comme la probabilité de fonctionnement de l'agent entre deux états de mission ; $1 - \gamma$ est la probabilité de panne entre deux instants consécutifs, c'est pourquoi on

affecte la récompense potentielle à t d'un facteur γ^t qui pondère ce gain par la probabilité que l'agent soit toujours en fonctionnement à t.
On définit alors la politique optimale π^* comme la politique qui maximise le critère (ou qui le minimise dans le cas où on a défini une fonction de coût). La fonction de valeur associée à π^* est notée V^*, c'est la fonction de valeur qui prend les valeurs maximales quel que soit l'état s dans lequel on la considère.

Définition 10 (histoire-dépendance). *Une politique ou stratégie est dite histoire-dépendante si elle dépend de l'ensemble de l'historique des actions/états.*

On définit par opposition la notion de politique Markovienne qui ne prend en argument que l'état courant. On peut alors montrer ([FFG01]) que pour chacun des critères ci-dessus :

Proposition 1. *Soit π une politique histoire-dépendante (déterministe ou aléatoire), soit V^π une fonction de valeur de cette politique (basée sur un des critères précédents), quel que soit l'état de départ s appartenant à S, il existe une politique Markovienne π' (respectivement déterministe ou aléatoire) telle que $V^\pi(s) = V^{\pi'}(s)$.*

Démonstration. On effectue la preuve dans le cas du critère γ-pondéré et pour une politique aléatoire.
On veut prouver que quelle que soit la politique aléatoire et histoire-dépendante π que l'on considère et quel que soit le point de départ $x \in S$ que l'on considère, il existe une politique π' Markovienne telle que $V^\pi(s) = V^{\pi'}(s)$. Or :

$$V_\gamma^\pi = \sum_{t=0}^{\infty} E^\pi \left[\gamma^t r_t(s_t, a_t) | s_0 = x\right]$$

Donc :

$$V^\pi(s) = \sum_{t=0}^{\infty} \sum_{s \in S} \sum_{a \in A} \gamma^t r_t(s,a) P^\pi(s_t = s, a_t = a | s_0 = x)$$

Et également :

$$V^{\pi'}(s) = \sum_{t=0}^{\infty} \sum_{s \in S} \sum_{a \in A} \gamma^t r_t(s,a) P^{\pi'}(s_t = s, a_t = a | s_0 = x)$$

Il s'agit donc de montrer que :

$$P^\pi(s_t = s, a_t = a | s_0 = x) = P^{\pi'}(s_t = s, a_t = a | s_0 = x)$$

Soit donc $x \in S$ et π une politique aléatoire histoire-dépendante. A chaque instant t, on construit la politique aléatoire Markovienne π' telle que :

$$\forall t \in \mathbb{N}, \quad \forall s \in S, \quad \forall a \in A, \quad P^{\pi'}(a_t = a | s_t = s) = P^\pi(a_t = a | s_0 = x, s_t = s)$$

On va démontrer le résultat souhaité par récurrence.
On a immédiatement (en $t = 0$) :

$$P^\pi(s_0 = s, a_0 = a | s_0 = x) = P^{\pi'}(s_0 = s, a_0 = a | s_0 = x)$$

En supposant le résultat vrai pour $t - 1$, on a :

$$\begin{aligned} P^{\pi'}(s_t = s, a_t = a | s_0 = x) &= P^{\pi'}(s_t = s | s_0 = x) P^{\pi'}(a_t = a | s_0 = x) \\ &= P^{\pi'}(s_t = s | s_0 = x) \cdot \\ &\qquad P^\pi(a_t = a | s_0 = x, s_t = s) \end{aligned} \tag{2.1}$$

Or la probabilité $P^{\pi'}(s_t = s|s_0 = x)$ d'arriver en s en appliquant π' et en sachant que $s_0 = x$ est égale à la somme sur tous les états de départ et sur toutes les actions possibles des probabilités d'arriver en s sachant que $s_0 = x$. Chacune de ces probabilités élémentaires correspond à la probabilité d'effectuer l'action a et en partant de r multipliée par la probabilité d'arriver en s partant de r et effectuant a. Cela se traduit par :

$$P^{\pi'}(s_t = s|s_0 = x) = \sum_{r \in S} \sum_{a \in A} P^{\pi'}(s_{t-1} = r, a_{t-1} = a|s_0 = x)p(s|r, a)$$

Et grace à l'hypothèse de récurrence :

$$\begin{aligned} P^{\pi'}(s_t = s|s_0 = x) &= \sum_{r \in S} \sum_{a \in A} P^{\pi}(s_{t-1} = r, a_{t-1} = a|s_0 = x)p(s|r, a) \\ &= P^{\pi}(s_t = s|s_0 = x) \end{aligned} \tag{2.2}$$

Les équations 2.1 et 2.2 fournissent :

$$\begin{aligned} P^{\pi'}(s_t = s, a_t = a|s_0 = x) &= P^{\pi}(s_t = s|s_0 = x)P^{\pi}(a_t = a|s_0 = x, s_t = s) \\ &= P^{\pi}(s_t = s, a_t = a|s_0 = x) \end{aligned}$$

Ce qui établit la récurrence et démontre donc le résultat. □

Forts de cette proposition, on peut affirmer qu'on ne perd rien en optimalité du problème en ne considérant que des politiques Markoviennes, ce que l'on fera à partir de ce point. On note $\mathcal{D}$ l'ensemble des politiques déterministes, stationnaires et Markoviennes.

On a donc défini l'environnement de l'agent et introduit le contexte et les outils associés aux MDP. On s'est doté d'un critère d'évaluation des politiques réaliste (le critère γ-pondéré). On sait également qu'on peut, sans risque de perdre la solution optimale (c'est-à-dire celle de plus grande valeur), considérer des politiques déterministes Markoviennes. On va donc s'attacher maintenant à caractériser et rechercher la politique optimale à notre problème.
On a :

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s)$$

Donc :

$$V^* = \max_{\pi \in \mathcal{D}} V^{\pi}$$

On s'intéresse donc plus précisément à la résolution du problème :

$$V^* = \max_{\pi \in \mathcal{D}} \sum_{t=0}^{\infty} \gamma^t r_t^{\pi}$$
$$\pi^* = \operatorname*{argmax}_{\pi \in \mathcal{D}} \sum_{t=0}^{\infty} \gamma^t r_t^{\pi}$$

On cherche à voir comment on peut améliorer la politique courante à chaque état, on cherche donc une caractérisation par récurrence de V^{π} et v^*. On introduit pour cela les deux opérateurs L_{π} et L :

Définition 11 (opérateur L_{π}). *Soit $\pi \in \mathcal{D}$ on définit l'opérateur L_{π} par :*

$$\forall V \in \mathcal{V} \quad \forall s \in S \quad LV(s) = r(\pi(s), s) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(s')$$

ou encore

$$\forall V \in \mathcal{V} \quad L_{\pi}V = r_{\pi} + \gamma P_{\pi}V$$

On peut alors montrer ([FFG01]) :

Théorème 1 (caractérisation de V^π). *Soient S et A finis, $\gamma < 1$ et $\pi \in \mathcal{D}$. V^π vérifie alors la propriété : V^π est l'unique solution de $L_\pi V = V$.*

De même, on définit l'opérateur de programmation dynamique L :

Définition 12 (opérateur de programmation dynamique). *On définit L, l'opérateur de programmation dynamique comme :*

$$\forall V \in \mathcal{V} \quad \forall s \in S \quad LV(s) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V(s') \right)$$

ou encore :

$$\forall V \in \mathcal{V} \quad LV = \max_{\pi \in \mathcal{D}} (r_\pi + \gamma P_\pi V)$$

On notera à titre de rappel que $P(s'|s,a) = T(s',a,s)$ et que cette distribution est donc entièrement définie dès que notre MDP est construit. Egalement, $r(s,a)$ est équivalent au $R(s,a,s')$ dans le cas où la récompense est affectée à la transition. Bellman [Bel57] a prouvé que la recherche de la politique optimale passe par la résolution du point fixe de l'opérateur de programmation dynamique :

Théorème 2 (equation de Bellman). *Soient S et A finis et $\gamma < 1$. V^* est l'unique solution de l'équation $V = LV$*

Démonstration. La preuve de ce théorème est donnée en détail dans [Put94, FFG01]. On en redonne ici les grandes lignes.
Au préalable, on montre que pour un même problème et une même fonction de valeur, la valeur maximale de la fonction de valeur est la même, que l'on considère des politiques déterministes ou aléatoires.
Puis dans un premier temps on montre que s'il existe une fonction de valeur V telle que $V \geq LV$ alors $V \geq V^*$:

$$\exists V \in \mathcal{V} \ / \ V \geq LV \Rightarrow V \geq V^*$$

Pour cela, on montre que si $V \geq LV$ alors pour tout π, $V \geq V^\pi$ ce qui implique que $V \geq V^*$.
De la même manière on montre que $V \leq LV \Rightarrow V \leq V^*$. On a donc démontré que s'il existe un $V \in \mathcal{V}$ tel que $V = LV$ alors $V = V^*$. Il reste à prouver son existence et son unicité.
On a :

- $\mathcal{V}$ muni de la norme max est un espace de Banach (espace vectoriel normé complet).
- L'opérateur L est une contraction sur $\mathcal{V}$ (avec la norme max).

D'après le théorème du point fixe de Banach, il existe un unique V^* tel que $V^* = LV^*$ (et la suite définie par $V_n = LV_{n-1}$ converge vers V^*). □

On peut tirer le sens de ce théorème de la façon suivante :
Supposons que l'état courant soit s. Le gain total qu'on peut espérer recevoir est la somme du gain associé à la transition qu'on va effectuer et des gains qu'on peut espérer des états futurs. Le gain qu'on va recevoir immédiatement en effectuant l'action a est $r(s,a)$. En effectuant a, on a une probabilité $P(s'|s,a)$ d'arriver en s', et donc de bénéficier du gain espéré $V(s')$. Le gain total qu'on peut espérer depuis l'état s en effectuant l'action a est donc bien $r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$. La valeur optimale (correspondant donc à l'action de la politique optimale) en s est donc le maximum de cette quantité. On peut donc définir une suite de fonctions $(V_n^\pi)_{n \in \mathbb{N}}$ telles que $V_{n+1}^\pi = LV_n^\pi$. On peut voir l'élement V_n^π comme un vecteur de $|S|$

valeurs et la suite V^π comme une suite croissante et convergente. La limite de V^π correspond alors au jeu de valeurs le plus élevé, c'est-à-dire à V^*.
On peut également adopter une approche (également récursive) plus propre à la programmation dynamique disant : si je n'ai qu'un coup à jouer, quelle est l'action optimale ? Si, à présent, je dois jouer un coup et je sais que toutes les actions qui suivront font partie de la politique optimale, quel coup dois-je jouer ? En « rétropropageant » ce raisonnement (c'est-à-dire en jouant un coup, puis deux sachant les résultats de un, puis trois sachant les résultats de deux, etc.) on construit une suite de fonctions de valeur et de politiques qui convergent également vers la fonction de valeur optimale et la politique optimale respectivement.
Ce sont sur ces idées que se basent les algorithmes de recherche d'une solution optimale.

Un dernier élément est encore nécessaire, à présent qu'on a défini l'optimalité et caractérisé la politique optimale, il s'agit de l'écart à l'optimum. Ce dernier est introduit par :

Définition 13 (erreur de Bellman). *Soit une politique $\pi \in \mathcal{D}$ et un état $s \in S$. L'*erreur de Bellman *pour la politique π en s est définie comme :*

$$BE(V^\pi(s)) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^\pi(s') \right) - V^\pi(s)$$

*L'*erreur de Bellman *pour la politique π s'écrit alors :*

$$BE(V^\pi) = \max_{s \in S} BE(V^\pi(s))$$

Cette définition se traduit simplement en :
L'erreur de Bellman est la quantité dont on peut améliorer la valeur de la politique π sur horizon infini en optimisant le premier coup.
Une relation de majoration de l'erreur de Bellman locale (en s) a été établie dans [WB93] :

$$V^*(s) - V^\pi(s) \leq \frac{BE(V^\pi)}{1-\gamma}$$

Cette quantification de l'écart à l'optimum permettra par la suite de définir et de différencier les notions d'optimalité et d'ϵ-optimalité.

2.2 Algorithmes de recherche d'une solution optimale

Le problème de recherche d'une solution optimale tel qu'il a été posé dans les paragraphes précédent se présente comme un problème de maximisation. On peut distinguer trois principaux algorithmes de résolution : la programmation linéaire, l'itération de la valeur et l'itération de la politique. Ces trois familles d'algorithmes ainsi que leurs raffinements et variantes sont décrites dans [Put94, FFG01].

2.2.1 Programmation linéaire

On cherche à formuler le problème défini par :

$$V^* = \max_{\pi \in \mathcal{D}} \sum_{t=0}^{\infty} \gamma^t r_t^\pi$$
$$\pi^* = \operatorname*{argmax}_{\pi \in \mathcal{D}} \sum_{t=0}^{\infty} \gamma^t r_t^\pi$$

sous la forme standard d'un problème de programmation linéaire.
On a :

$$V^* = \max_{\pi \in \mathcal{D}} V^\pi$$

Donc on sait que :

$$\forall V \in \mathcal{V} \quad \sum_{s \in S} V^* \geq \sum_{s \in S} V$$

D'autre part on a :

$$V \geq LV \Rightarrow V \geq V^*$$

Donc la minimisation de la quantité $\sum_{s \in S} V(s)$ sous la contrainte $V \geq LV$ permet de trouver V^*. Le problème de programmation linéaire s'écrit donc sous forme directe :

$$\min \sum_{s \in S} V(s)$$
$$\forall s \in S, \forall a \in A,\ V(s) - \gamma \cdot \sum_{s' \in S} p(s'|s,a)V(s') \geq R(s,a)$$

On met ce problème sous forme standard par deux opérations :

- On remplace $V(s)$ par deux variables positives distinctes $V^+(s)$ et $V^-(s)$ et on écrit une contrainte égalité supplémentaire : $V(s) = V^+(s) - V^-(s)$.
- On rajoute des variables d'écart dans les contraintes inégalité.

Le problème sous forme standard s'écrit :

$$\min \left(\sum_{s \in S} V^+(s) - \sum_{s \in S} V^-(s) \right)$$
$$\forall s \in S, \forall a \in A,\ V(s) - \gamma \cdot \sum_{s' \in S} p(s'|s,a)V(s') - e(s,a) = R(s,a)$$
$$V^+, V^-, e(s,a) \geq 0$$

On note que le passage à la forme standard a multiplé par $|A| + 2$ le nombre de variables, le faisant passer de $|S|$ à $|S| \cdot (|A| + 2)$. On sait toutefois que pour le sommet solution, $2|S|$ composantes sont nulles. On peut diminuer le nombre de variables total en considérant le problème dual et ramener la dimension de l'espace à $|S| \cdot |A|$ mais on n'a plus de garanties sur les composantes nulles des sommets ainsi définis.

La construction de la politique optimale s'effectue en deux temps. Tout d'abord la recherche de V^* puis l'extraction des actions appartenant à π^* selon :

$$\forall s \in S, \quad \pi(s) = \operatorname*{argmax}_{a \in A} [R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V^*(s)]$$

[FFG01] précise que si $p()$ et $R()$ sont codés sur b bits, alors l'algorithme de résolution du problème ainsi défini (recherche de V^*) et de construction de π^* est de complexité polynomiale en $|S| \cdot |A| \cdot b$ et les temps de résolution sont assez longs.
On cherche donc à se tourner vers des algorithmes de complexité moindre tels que ceux présentés dans les sections suivantes.

2.2.2 Itération de la valeur

L'algorithme d'itération de la valeur repose sur la propriété de convergence de la suite de fonctions de valeurs définie par $V_{n+1} = LV_n$, vers V^*, quel que soit V_0. [FFG01] précise qu'en

un nombre d'itérations polynomial en $\frac{|S|\cdot|A|\cdot b}{(1-\gamma)\log(1/(1-\gamma))}$, la politique converge vers π^*. La valeur continue alors un peu d'évoluer vers V^* mais la politique ne change plus. Chaque itération est de complexité $O(|A||S|^2)$ (Cf. l'algorithme présenté en Fig. 2.1). On décide d'arrêter l'algorithme lorsque $\| V_{n+1} - V_n \| \leq \epsilon$ avec ϵ un seuil d'arret fixé *a priori*.

FIG. 2.1 : Algorithme d'itération de la valeur

$V_0 \leftarrow 0$
$n \leftarrow 0$
répéter
  pour $s \in S$ **faire**
    $V_{n+1}(s) = \max_{a \in A} \left(r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a) \cdot V_n(s') \right)$
  $n \leftarrow n+1$
jusqu'à $\| V_{n+1} - V_n \| \leq \epsilon$
pour $s \in S$ **faire**
  $\pi(s) \leftarrow \operatorname{argmax}_{a \in A} \left(r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a) \cdot V_n(s') \right)$
retourner V_n,π

On remarque qu'une bonne initialisation de V_0 permet à l'algorithme de converger plus vite. Par ailleurs, on peut noter qu'une fois que la fonction de valeur est mise à jour pour un état s, on pourrait la réutiliser pour la mise à jour des autres états et donc d'utiliser $V_{n+1}(s)$ à la place de $V_n(s)$ pour calculer $V_{n+1}(s')$ si $V_{n+1}(s)$ est déjà calculée. On calcule donc la fonction de valeur en $n+1$ pour le premier état comme ci-dessus. Puis pour le second, on utilise la valeur fraîchement calculée en $n+1$ du premier et les valeurs en n des autres états. Puis pour le troisième, on utilise les valeurs des premier et second en $n+1$ et celles en n des autres états, et ainsi de suite. C'est l'idée qui mène à l'algorithme de Gauss-Seidel présenté en Fig. 2.2.

FIG. 2.2 : Algorithme d'itération de la valeur amélioré (Gauss-Seidel)

$V_0 \leftarrow 0$
$n \leftarrow 0$
répéter
  pour $i \leftarrow 1$ **à** $|S|$ **faire**
    $V_{n+1}(s_i) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{1 \leq j < i} p(s'|s,a) V_{n+1}(s_j) + \gamma \sum_{i \leq j \leq |S|} p(s'|s,a) V_n(s_j) \right)$
  $n \leftarrow n+1$
jusqu'à $\| V_{n+1} - V_n \| \leq \epsilon$
pour $s \in S$ **faire**
  $\pi(s) \leftarrow \operatorname{argmax}_{a \in A} \left(r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a) \cdot V(s') \right)$
retourner V_n,π

Une autre variante de l'algorithme d'itération de la valeur basée sur le même principe que l'algorithme de Gauss-Seidel est donnée par les algorithmes de programmation dynamique asynchrone qui mettent à jour les valeurs d'états choisis aléatoirement parmi S.

Une dernière classe d'algorithmes opère, non plus sur la fonction de valeur, mais sur la politique elle-même, ce sont les algorithmes d'itération de la politique.

2.2.3 Itération de la politique

L'algorithme d'itération de la politique repose sur la recherche à chaque itération d'un nouvelle politique améliorant la fonction de valeur. Plutôt que d'améliorer la valeur d'un état (et donc de considérer les états séparément), on considère la valeur de la politique et on l'améliore coup par coup. Cela repose sur la propriété suivante :

Proposition 2 (amélioration de la fonction de valeur). *Considérons $\pi \in \mathcal{D}$ et π^+ telle que :*

$$\pi^+ \in \operatorname*{argmax}_{d \in \mathcal{D}} \{r_d + \gamma \cdot P_d \cdot V^\pi\}$$

Alors on a :

$$V^{\pi^+} \geq V^\pi$$

Cette propriété nous informe du fait que modifier la politique courante en améliorant le premier coup joué permet d'augmenter la fonction de valeur. Or on sait que la suite des $(V_n)_{n \in \mathbb{N}}$ est majorée par V^*. On tient donc ici un moyen simple de converger vers la politique optimale. A terme, l'algorithme converge vers une politique constante et on peut vérifier qu'alors on a $V^\pi = V^*$. L'algorithme d'itération de la politique suit ce shéma en deux temps : d'abord il résout le système d'équations $V = L_\pi V$, puis il améliore la politique sur un coup. L'algorithme d'itération de la politique est représenté Fig. 2.3.

FIG. 2.3 : Algorithme d'itération de la politique

$\pi_0 \in \mathcal{D}$
$n \leftarrow 0$
répéter
  Résoudre le système de $|S|$ équations :
  $\forall s \in S \quad V_n(s) = r(s, \pi_n(s)) + \gamma! \sum_{s' \in S} p(s'|s, \pi_n(s)) V_n(s')$
  pour $s \in S$ **faire**
    $\pi_{n+1}(s) \in \operatorname*{argmax}_{a \in A} \left(r(s,a) + \gamma! \sum_{s' \in S} p(s'|s,a) V_n(s') \right)$
  $n \leftarrow n+1$
jusqu'à $\pi_n = \pi_{n+1}$
retourner V_n,π_n

[Put94] démontre que l'ordre du taux de convergence de l'algorithme est au moins linéaire, voire quadratique sous certaines conditions. [FFG01] précise que sa complexité est en $O(|S|^2 \cdot (|A| + |S|))$ avec un nombre maximum d'itérations polynomial en $|S| \cdot |A| \cdot b$ pour un γ donné.

Plutôt que de résoudre l'équation $V = L_\pi V$ à chaque itération, on préfère simplifier l'évaluation de la valeur de la politique courante en utilisant l'algorithme d'itération de la valeur. On obtient donc un algorithme hybride présentant, comme l'algorithme d'itération de la politique, deux phases. La première phase est celle de l'amélioration sur un coup de la politique courante, elle se déroule pareillement à l'algorithme d'itération de la politique. La seconde phase concerne l'évaluation de politique améliorée et plutôt que de résoudre le système linéaire, on effectue cette évaluation de façon itérative par une démarche similaire à celle employée pour l'itération de la valeur. Cet algorithme s'appelle algorithme d'itération de la politique modifié ; la version de [FFG01] est présentée en Fig. 2.4. La convergence de cet algorithme est assurée par le fait de prendre une fonction de valeur de départ telle que $LV_0 \geq V_0$ ([Put94] donne une preuve

complète de cette convergence) ; un choix possible de V_0 est :

$$V_0(s) = \frac{1}{1-\gamma} \min_{\substack{s' \in S \\ a \in A}} r(s', a)$$

FIG. 2.4 : Algorithme d'itération de la politique modifié

$V_0 \in \mathcal{V}$ et $LV_0 \geq V_0$
$flag \leftarrow 0$
$n \leftarrow 0$
répéter
  pour $s \in S$ **faire**
    $\pi_{n+1}(s) \in \operatorname{argmax}_{a \in A} \left(r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a) V_n(s') \right)$
    (si on a le choix entre plusieurs actions optimales pour π_{n+1} et que π_n en fait partie, on prend π_n en priorité)
    $V_n^0(s) = \max_{a \in A} \left(r(s,a) + \gamma \cdot \sum_{s' \in S} p(s'|s,a) V_n(s') \right)$
  $m \leftarrow 0$
  si $\| V_n^0 - V_n \| < \epsilon$ **alors** $flag \leftarrow 1$
  sinon
    répéter
      pour $s \in S$ **faire**
        $V_n^{m+1} = r(s, \pi_{n+1}(s)) + \gamma \cdot \sum_{s' \in S} p(s'|s, \pi_{n+1}(s) V_n^m(s')$
      $m \leftarrow m+1$
    jusqu'à $\| V_n^{m+1} - V_n^m \| < \delta$
    $V_{n+1} \leftarrow V_n^m$
    $n \leftarrow n+1$
jusqu'à $flag = 1$
retourner V_n,π_{n+1}

On retrouve bien sur la Fig. 2.4 les deux phases d'amélioration de la politique et d'approximation de la fonction de valeur. On note qu'on contrôle la complexité du calcul et la précision de l'algorithme par les paramètres ϵ et δ. On calcule dans un premier temps la politique améliorée puis, si l'erreur de Bellman pour la politique courante est inférieure à ϵ, on considère qu'on a trouvé une politique optimale et on s'arrête en renvoyant la politique améliorée dont la valeur est équivalente à ϵ près à la politique courante (et dont la valeur est équivalente à la valeur de la politque optimale à $\epsilon/(1-\gamma)$ près, d'après la propriété de majoration qui suit la définition 13). Sinon on approxime la valeur de la nouvelle politique grâce à l'algorithme type « itération de la valeur » et on recommence. On peut noter que l'algorithme peut être simplifié en plusieurs points lors de son implémentation. On peut notamment faire le calcul de l'erreur de Bellman avant de calculer la politique améliorée (ce qui, à la dernière itération, épargne éventuellement le calcul de la politque améliorée). Par ailleurs, on peut introduire un schéma de calcul type Gauss-Seidel pour l'approximation de la valeur de la politique. [Put94] propose une version légèrement différente de l'algorithme. Plutôt que de considérer un seuil d'arrêt sur la différence entre deux itérations successives pour l'approximation de la fonction de valeur de la politique courante, on peut donner un nombre d'itérations m_n limité, la valeur de m_n pouvant évoluer au fur et à mesure que l'algorithme se déroule.

En pratique, pour des PDM de taille « raisonnable », cet algorithme est le plus performant. L'algorithme d'itération de la politique devient vite très lourd à cause de la résolution du système linéaire qui, pour un nombre d'état grand, prend un temps trop long par rapport aux applications qu'on lui souhaiterait. Toutefois, l'algorithme d'itération de la politique, bien que ses itérations soient lourdes du point de vue du calcul converge généralement en moins d'itérations que l'algorithme d'itération de la valeur. Ainsi, l'algorithme d'itération de la politique modifié semble un bon compromis.

2.3 Introduire le temps comme paramètre du MDP

On a considéré jusqu'ici des MDP qui ne tiennent pas compte du facteur temps dans la résolution des actions et des transitions. On peut notamment noter que pour l'instant, chaque action commencée à t se termine à $t + 1$. Il apparaît donc utile d'introduire une notion de durée et de dépendance au temps dans le modèle des MDP. En effet, une grande majorité des phénomènes que l'on aborde sous forme de MDP comporte des aspects temporels : la durée des tâches, la consommation de carburant, certaines récompenses dépendant de la durée de l'action, ... Cette démarche conduit à l'utilisation de Semi-MDPs, ou SMDPs. Une grande partie des notions abordées ici sont détaillées dans [Put94] au chapitre « Continuous-time models ». Certaines approches qui seront abordées plus tard reprennent un certain nombre de ces notions, notamment dans [BM05] et dans [SPS99].

Pour adapter le modèle discret à un modèle dépendant du temps, il faut reconsidérer le problème à la base de la structure des MDP et caractériser de façon probabiliste le système en prenant le temps en compte.

2.3.1 Modèle probabiliste du processus

On cherche donc à redéfinir le modèle probabiliste. Dans un premier temps, on s'attache à faire la différence entre le processus que l'on étudie et le *processus réel* ou *naturel*. Dans le contexte des SMDP, le choix d'une action détermine la distribution de probabilité sur l'état conséquent à l'action et sur la durée de cette action. On introduit ainsi la notion de *date de décision*. Le processus décrit par le SMDP est donc un processus stochastique Markovien, remis à jour à chaque *date de décision* et qui fournit des probabilités d'occurence sur les états s et les durées d'actions t. Par opposition, le *processus réel* représente l'évolution des états du système comme si ce dernier était continument obervable (et observé) au cours du temps. On note donc qu'entre deux *dates de décision* successives, le *processus naturel* peut passer par plusieurs états et que les deux processus s'accordent aux instants de décision.
Dans l'absolu, en conservant la représentation d'état et l'espace d'action définis de manière générale plus haut, on cherche à connaître la grandeur $Q(t, j|s, a)$ qui décrit la probabilité que la prochaine date de décision arrive en t ou après (t étant la durée écoulée depuis la dernière décision), tandis que le système est dans l'état j, sachant qu'à la dernière décision on a réalisé l'action a partant de l'état s.
En général, on n'obtient pas $Q(t, j|s, a)$ directement. A la place, on dispose souvent des quantités suivantes :

- $F(t|s, a)$: Il s'agit de la probabilité que la prochaine date de décision arrive t unités de temps après la dernière decision prise, sachant que cette dernière décision était d'effectuer a en partant de s.

- $p(j|t,s,a)$: C'est la probabilité que le *processus réel* soit dans l'état j, t unités de temps après avoir pris la décision a en s. On peut de suite noter que cette probabilité n'est pas apparente du point de vue du SMDP et qu'elle sera utilisée de façon sous-jacente pour calculer des probabilités de transition et des récompenses entre les dates de décision.
- $P(j|s,a)$: C'est la probabilité que l'on soit dans l'état j lors de la prochaine date de décision sachant qu'on a effectué a quand on était en s à la dernière date de décision. Si θ est la date de décision qui suit l'action a prise en s, on a $P(j|s,a) = p(j|\theta,s,a)$.

On peut alors écrire aux dates de décision :

$$Q(t,j|s,a) = P(j|s,a) \cdot F(t|s,a) \tag{2.3}$$

On sera par ailleurs menés par la suite à considérer des quantités de temps infinitésimales. On peut définir la différentielle par rapport au temps de la fonction Q que l'on note $Q(dt,j|s,a)$ et qui correspond de façon intuitive à la variation de Q entre deux instants séparés de dt. Cette dernière est associée à la différentielle $F(dt|s,a)$.

On distingue parfois le Processus Décisionnel de Markov Inclus dans le processus réel (on le notera E-MDP pour Embedded-MDP). Dans la suite, on utilisera surtout la dénomination générique SMDP pour désigner le MDP étudié par opposition au *processus réel*. On note que $P(j|s,a)$ permet d'obtenir la fonction de transition du SMDP.

Enfin, l'historique h_n s'enrichit d'une variable temporelle de plus : On part de l'instant t_0, dans l'état s_0. On choisit d'effectuer l'action a_0 à l'instant t_1 ce qui nous mène en un état s_1, on effectue alors l'action a_2 et ainsi de suite, jusqu'à ce que qu'en t_n on arrive dans l'état s_n. De cette description de l'historique on peut tirer une représentation « naturelle » de chaque date de décision passée dans l'historique sous la forme d'un triplet $\{t_n, s_n, a_n\}$ et une écriture de l'historique :

$$h_n = (t_0, s_0, a_0, t_1, s_1, a_1, \ldots, t_n, s_n)$$

Et de façon récursive :

$$h_n = (h_{n-1}, a_{n-1}, t_n, s_n)$$

2.3.2 Modèle de récompense

Une fois le modèle de transition défini, il faut redéfinir les fonctions de gain. On distingue alors les gains et coûts immédiats associés à la décision (s,a) des gains obtenus pendant l'exécution. Les premiers sont notés $k(s,a)$. Pour les seconds on introduit un *taux de récompense* $c(j',s,a)$ défini pour la décision (s,a) et pour l'état j' du *processus réel*. On note qu'on pourrait définir ce taux comme une fonction du temps afin de laisser plus de liberté à la définition de la récompense en fonction du temps (non uniquement linéaire) mais pour l'instant, on se restreint à ce cadre que l'on pourra éventuellement adapter plus tard.

Une politique (histoire-dépendante, Markovienne, déterministe ou aléatoire) est toujours définie comme précédemment et on peut définir pour une politique déterministe Markovienne π les variables :
- $p_\pi(j|t,s) = p(j|t,s,\pi(s))$
- $F_\pi(t|s) = F(t|s,\pi(s))$
- $Q_\pi(t,j|s) = Q(t,j|s,\pi(s))$
- $k_\pi(s) = k(s,\pi(s))$

– $c_\pi(j', s) = c(j', s, \pi(s))$
Avec $j, j', s \in S$; $a \in A$; $t \in \mathbb{R}^+$.

Pour un processus quelconque défini le cadre précédent, on considère une variable d'historique $\omega = (t_0, s_0, a_0, t_1, s_1, a_1, \dots)$ et des variables aléatoires :

$$X_n(\omega) = s_n, \quad Y_n(\omega) = a_n, \quad \tau_n(\omega) = t_n$$

On définit également le temps écoulé comme :

$$\sigma_n(\omega) = \sum_{j=0}^{n} t_j \tag{2.4}$$

2.3.3 Fonction de valeur

A partir de la description probabiliste et du modèle de gains qu'on vient d'établir pour le processus réel, on peut construire le SMDP sur lequel on va raisonner.
On admettra que la proposition 1 reste vraie et qu'on peut donc sans risque considérer uniquement des politiques Markoviennes.

Il reste alors à reprendre l'expression du critère γ-pondéré afin de pouvoir évaluer les politiques et définir l'optimalité.
En désignant par $E^\pi()$ la fonction espérance d'une variable aléatoire par rapport à $P^\pi()$ — $P^\pi()$ elle-même désignant la distribution de probabilité sur les états sachant qu'on applique la politique π — on peut écrire la valeur d'une politique par le critère γ-pondéré en s comme :

$$V_\gamma^\pi(s) = E^\pi \left\{ \sum_{n=0}^{\infty} \gamma^{\sigma_n} \left[k(X_n, Y_n) + \int_{\sigma_n}^{\sigma_{n+1}} \gamma^{t-\sigma_n} \; c(W_t, X_n, Y_n) \; dt \right] \right\} \tag{2.5}$$

On retrouve bien comme prévu les deux termes séparés du gain immédiat et du taux de récompense. Comme précédemment, on ommetra l'indice γ sur le V_γ par la suite.
On définit la valeur optimale en un état s :

$$V^*(s) = \max_{\pi \in \mathcal{D}} V^\pi(s)$$

Cette définition étant faite, on peut chercher à retrouver les relations de récurrence dont on disposait avec les MDP classiques pour caractériser les politiques.
Afin de simplifier l'écriture précédente de V^π, on peut introduire la fonction de récompense associée à (s, a) et correspondant à la récompense attendue jusqu'à la prochaine date de décision si on considère un facteur de rabais γ :

$$r(s, a) = k(s, a) + E \left\{ \int_{t_0=0}^{t_1} \gamma^t \; c(W_t, s, a) \; dt \right\} \tag{2.6}$$

La fonction espérance $E()$ étant calculée sachant qu'on part de s en effectuant a et donc connaissant les distributions de probabilité $F(t|s, a)$ et $p(j|t, s, a)$. On établit ainsi une expression générale de $r(s, a)$:

$$r(s, a) = k(s, a) + \int_0^\infty \sum_{j \in S} \left[\int_0^u \gamma^t \; c(j, s, a) \; p(j|t, s, a) \; dt \right] F(du|s, a) \tag{2.7}$$

Le terme entre crochets représente la récompense que l'on peut espérer obtenir sachant que le *processus réel* reste dans l'état j pendant la durée u et change d'état en u. On somme donc toutes ces récompenses sur les différents états atteignables par le *processus réel*, puis on en calcule l'espérance vis-à-vis des probabilités que chaque état se termine effectivement en une date u. Si on réécrit cette équation avec les notations relatives à une politique donnée, on a :

$$r_\pi(s) = k_\pi(s) + \int_0^\infty \sum_{j \in S} \left[\int_0^u \gamma^t c_\pi(j,s) p_\pi(j|t,s) dt \right] F_\pi(du|s) \tag{2.8}$$

On peut alors réécrire la fonction de valeur d'une politique en s comme :

$$V^\pi(s) = E_s^\pi \left\{ \sum_{n=0}^\infty \gamma^{\sigma_n} r_\pi(X_n) \right\} \tag{2.9}$$

On cherche à établir une relation similaire au $V^\pi = r_\pi + \gamma P_\pi V^\pi$ qu'on avait précédemment. On considère une politique stationnaire π. On a :

$$V^\pi(s) = E_s^\pi \left(\sum_{n=0}^\infty \gamma^{\sigma_n} r_\pi(X_n) \right)$$

Donc on peut écrire :

$$V^\pi(s) = r_\pi(s) + E_s^\pi \left(\sum_{n=1}^\infty \gamma^{\sigma_n} r_\pi(X_n) \right)$$

car $X_0 = s$ et $\sigma_0 = t_0 = 0$ ou encore :

$$V^\pi(s) = r_\pi(s) + E_s^\pi \left(\gamma^{\sigma_1} \sum_{n=1}^\infty \gamma^{\sigma_n - \sigma_1} r_\pi(X_n) \right)$$

Le terme en $\sigma_n - \sigma_1$ revient à considérer l'application de la politique π à partir de l'instant 1 (et non plus 0) ; l'horizon étant infini l'espérance de ce terme correspond à $V^\pi(X_1)$, X_1 étant la variable aléatoire décrivant la valeur de l'état à la date de décision 1. On peut donc écrire :

$$V^\pi(s) = r_\pi(s) + E_s^\pi \left(\gamma^{\sigma_1} \cdot V^\pi(X_1) \right)$$

On développe le terme de droite en écrivant l'espérance comme la sommation pondérée par les probabilités sur les variables σ et X_1 :

$$V^\pi(s) = r_\pi(s) + \sum_{j \in S} \int_0^\infty \gamma^t \cdot V^\pi(j) \cdot Q_\pi(dt, j|s) \tag{2.10}$$

On réécrit cette égalité de façon vectorielle :

$$V^\pi = r_\pi + M_\pi \cdot V^\pi \tag{2.11}$$

Avec M_π la matrice $|S| \times |S|$ définie par :

$$m_\pi(j|s) = \int_0^\infty \gamma^t \cdot Q_\pi(dt, j|s) \tag{2.12}$$

j étant constant sur les colonnes et s sur les lignes pour un vecteur colonne V^π.
On a alors :

Théorème 3. *Soit $s \in S$ et $a \in A$. Supposons les récompenses associées à tous les couples (s,a) bornées. Alors pour toute politique π Markovienne, V^π est l'unique solution de l'équation en V :*

$$V = r_\pi + M_\pi \cdot V$$

et on a :

$$V^\pi = (I - M_\pi)^{-1} r_\pi.$$

2.3.4 Politique optimale

On vient de se ramener dans un cadre similaire aux MDP classiques, on peut donc écrire la valeur V^* de la politique optimale comme unique solution de :

$$V = \max_{\pi \in \mathcal{D}} \{r_\pi + M_\pi V\} = LV$$

ce qui revient, par état, à :

$$V^*(s) = \max_{a \in A} \left\{ r(s,a) + \sum_{j \in S} m(j|s,a) V^*(j) \right\}$$

On peut alors appliquer directement les algorithmes de recherche de π^* vus précédemment.

On pourrait également considérer le cas des MDP à temps continu (CTMDP) qui permettent de considérer des décisions à tout instant. Ces derniers sont toutefois un cas particulier de SMDPs dans lequel on considère des durées d'action suivant une distribution exponentielle et, dans le cas des problèmes à horizon infini, on considère des dates de décision à chaque transition entre états.

2.4 L'approche décomposée, une approche hiérarchique de l'espace d'état

Supposons un robot mobile agissant dans un monde grille comportant plusieurs pièces ; un robot mobile distributeur de café se déplaçant dans un ensemble de bureaux par exemple. On se rend intuitivement compte du fait qu'au sein d'une même zone géographique (un bureau), la politique optimale ne dépend que de la configuration de la zone elle-même (emplacement des récompenses, probabilités de transition au sein de la pièce) et de l'espérance de récompense associée à « la porte », c'est-à-dire, l'espérance de récompense associée à l'action de quitter la zone. On met ainsi en évidence une possible décomposition de l'espace d'état en zones faiblement couplées et faiblement communicantes : faiblement couplées car entre une zone et le reste de l'univers, on peut considérer des fonctions de valeurs globales distinctes et indépendantes, faiblement communicantes car pour entrer et sortir d'une zone, il n'y a qu'un nombre de transitions limité.

D'une manière plus générale, l'idée de séparer des zones de l'espace d'état faiblement couplées, qu'elles soient de nature géographique ou non représente une perspective intéressante puisque qu'elle permet de considérer d'une part des MDP plus petits donc plus facilement traitables, et d'autre part, elle permet de générer des macro-actions, des politiques locales réutilisales comme des actions sur les macro-états que sont les zones. Cette approche permet d'approcher les problèmes décisionnels à plusieurs échelles comme préconisé dans [Sut95] et autorise des replannifications rapides avec très peu de perte d'optimalité.

Plusieurs approches ont été entreprises du point de vue de la décomposition. On présente ici les aspects essentiels des approches d'Hauskrecht ([HMK+98]), de Parr ([Par98]), de Dean ([DL95]), et on aborde un modèle de décomposition automatique de l'espace d'état proposé par Sabbadin ([Sab02]).

2.4.1 La structure du MDP décomposé

Dans cette première approche, on définit une décomposition Π du MDP $M =< S, A, T, R >$ comme un partitionnement de l'espace d'état : $\Pi = \{S_1, \ldots, S_n\}$. On appelle les $S_i \in \Pi$ les régions de M et pour chaque S_i, on définit :

La périphérie de sortie : $XPer(S_i) = \{ \ t \in S \setminus S_i | \forall s \in S_i, \exists a \in A, T(s, a, t) > 0\}$
La périphérie d'entrée : $EPer(S_i) = \{t \in S_i | \forall s \in S \setminus S_i, \exists a \in A, T(s, a, t) > 0\}$

La périphérie de sortie rassemble les états hors de S_i accessibles directement depuis S_i tandis que la périphérie d'entrée rassemble les éléments de S_i accessibles directement depuis l'extérieur. On définit également l'ensemble de tous les états périphériques dans S :

$$Per_{\Pi}(S) = \bigcup_i \{EPer(S_i), \quad i \leq n\} = \bigcup_i \{XPer(S_i), \quad i \leq n\}$$

On définit une macro-action comme une politique π_i définie localement sur S_i. L'idée qui motive la définition des macro-actions est que l'on peut considérer une macro-action comme une action primitive du MDP d'origine pourvu qu'on ait les modèles de transition et de récompense adéquats.

Pour construire une politique locale, il faut construire les modèles de transition et de récompense sur l'espace d'état restreint à S_i et vers l'espace d'état constitué de $S_i \bigcup XPer(s_i)$. Pour les transitions vers l'« extérieur », on définit le modèle de transition T_i pour la macro-action π_i comme :

$$T_i : \left\{ \begin{array}{ccc} S_i \times XPer(S_i) & \rightarrow & [0; 1] \\ (s, s') & \mapsto & T_i(s, \pi_i, s') \end{array} \right.$$

La probabilité T_i correspond à la probabilité de la transition associée à la macro-action π_i. Il est important de bien noter qu'on considère les politiques locales π_i comme des actions dans le MDP global. On reste dans un cadre où le temps n'intervient pas de façon explicite, mais la définition de la fonction de transition T_i fait apparaître un aspect temporel : $T_i(s, \pi, s')$ est la probabilité qu'en suivant la politique π_i et en partant de s (en appliquant la macro-action π_i en s) on se retrouve en s' à un instant ou un autre.

$$T_i(s, \pi_i, s') = \int_0^{\infty} \gamma^t \cdot Pr(t, s' | s, \pi_i) dt$$

Avec $Pr(t, s' | s, \pi_i)$ la probabilité d'atteindre s' en t sachant qu'on est parti de s et qu'on applique la politique π_i.

La fonction de transition T_i vérifie l'égalité :

$$\forall s \in S_i, \forall s' \in XPer(S_i), T_i(s, \pi_i, s') = T(s, \pi_i(s) + \gamma \sum_{s'' \in S_i} T(s, \pi(s), s'') T_i(s'', \pi_i, s')$$

Cette égalité traduit le fait que la probabilité d'arriver en s' en partant de s et en appliquant la macro-action π_i est la somme de la probabilité d'arriver en s' en partant de s et en appliquant l'action $\pi_i(s)$ et de la somme (sur s'') des probabilités d'arriver en s' partant de s'' et appliquant

la macro-action π_i pondérées par la probabilité d'aller de s en s'' en effectuant l'action π_i.

L'équation ci-dessus conduit à $|XPer(S_i)|$ systèmes de $|S_i|$ équations pour calculer T_i.

On définit de même la fonction de récompense comme la somme des récompenses associées aux transitions franchies à un instant ou un autre lors de l'application de π_i, pondérées par leur éloignement temporel (en notant s^t l'état atteint à la date t) :

$$\forall s \in S_i, R_i(s, \pi_i) = \int_0^\infty \int_0^\tau \gamma^t R(s^t, \pi_i(s^t)|s, \pi_i) dt d\tau$$

Et cette fonction vérifie :

$$R_i(s, \pi_i) = R(s, \pi_i(s)) + \gamma \sum_{s' \in S_i} T(s, \pi(s), s') R_i(s', \pi_i)$$

Cette équation induit un système de $|S_i|$ équations pour calculer R_i.

2.4.2 MDP abstrait et macro-actions

Si on récapitule, on a une partition donnée de l'espace d'état $\Pi = \{S_1, \ldots, S_n\}$, sur chaque zone S_i on définit une politique π_i qu'on considère comme une action, une macro-action représentant une sorte de « tâche » mais se trouvant au même niveau hiérarchique qu'une action primitive et pour cette politique on définit un modèle de transition et de récompenses afin de l'intégrere dans le MDP global. On a alors deux possibilités. Supposons qu'à chaque région S_i, on ait défini un jeu de macro-actions $A_i = \{\pi_i^1, \ldots, \pi_i^{n_i}\}$:

- On peut considérer l'espace d'action augmenté $A \ \cup A_1 \cup \cdots \cup A_n$ en augmentant les modèles de transition et de récompense avec les T_i et les R_i, la solution est alors toujours optimale et peut être trouvée en un nombre d'itérations inférieur à celui nécessaire si on considère l'espace d'état seul. Toutefois, en fonction de l'initialisation de la fonction de valeur, ce résultat n'est pas toujours acquis.
- On peut considérer un espace d'action uniquement constitué des $\cup_i A_i$. Le problème ainsi posé est plus simple à résoudre grace à la simplification de l'espace d'action cependant on perd la garantie d'optimalité par rapport à un espace d'action complet car on limite le nombre de comportements que l'on peut différencier.

Ces deux approches présentent certains avantages mais l'objectif principal d'une approche de décomposition est de séparer l'espace d'état et surtout de le réduire afin de le rendre plus facile à traiter sans trop perdre en optimalité. L'approche d'Hauskrecht considère que, puisqu'on n'a pris que des macro-actions, les décisions ne doivent être prises qu'aux états périphériques et que l'on peut donc considérer un MDP abstrait ayant pour espace d'état l'ensemble des états périphériques et les macro-actions comme espace d'action. Considérons un MDP $M =< S, A, T, R >$ sur lequel on connait une décomposition Π et des jeux de macro-actions par régions A_i. On définit alors le MDP abstrait $M' =< S', A', T', R' >$ constitué de :

- $S' = Per_\Pi(S) = \cup\{EPer(S_i)|i \leq n\}$,
- $A' = \cup_i A_i$ avec les politiques π_i^k uniquement définies sur les $s \ \in EPer(S_i)$,

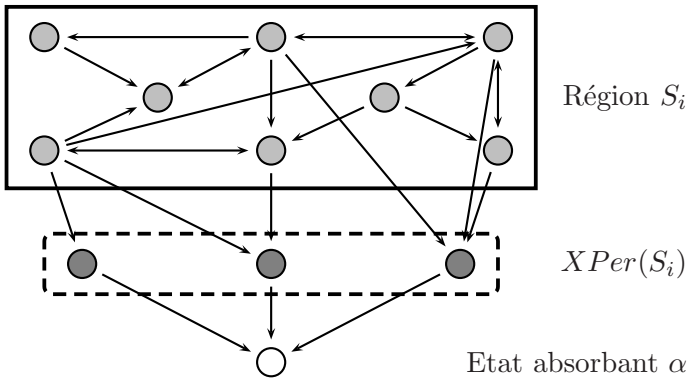

FIG. 2.5 – Illustration de l'espace d'état d'un MDP local

- $\forall s \in S', \exists S_i \in \Pi / s \in S_i$, on définit alors pour $s \in S$ et $s \in S_i$: $T'(s, \pi_i^k, s') = T_i(s, \pi_i^k, s')$. On note que $T'(s, \pi_i^k, s') = 0$ si $s' \notin XPer(S_i)$.
- $R'(s, \pi_i^k) = R_i(s, \pi_i^k)$

On définit ainsi une macro-politique, application associant aux états de S' des macro-actions de A'. On peut noter qu'une macro-politique, ramenée dans le contexte du MDP M d'origine n'est pas nécessairement Markovienne, en effet, l'action choisie en un état de $XPer(S_i)$ dépendra souvent de l'historique des actions du MDP M, en particulier de l'état par lequel on est entré dans la zone courante, cet état qui est le dernier état en date dans le MDP M'.

Cette approche qui implique une hiérarchisation des actions et macro-actions permet de considérer des espaces d'état et d'action nettement réduits et donc de résoudre le MDP global beaucoup plus rapidement. Par ailleurs, une telle approche permet de réutiliser les macro-actions calculées une fois pour un résoudre un second problème similaire pour lequel — par exemple — seules les récompenses changent. Le prix que l'on paie pour simplifier la résolution du MDP d'origine est le risque d'avoir une politique non optimale au final ; afin de limiter ce risque et la sous-optimalité de la solution trouvée, il faut s'assurer que les macro-actions générées couvrent bien un spectre d'action assez large et sont elles-même quasi-optimales.

2.4.3 L'approche « ϵ-grid »

On cherche donc une manière efficace de générer un jeu de macro-actions qui fassent partie ou s'approchent au plus près possible de morceaux de la politique optimale. On veut donc que la politique optimale calculée sur un MDP réduit à S_i corresponde à la restriction de la politique optimale à S_i. La méthode proposée par Hauskrecht procède de la façon suivante :

Soit un MDP $M =< S, A, T, R >$ partitionné en $\Pi = \{S_1, \ldots, S_n\}$, pour chaque S_i on veut construire les politiques qui partent des éléments de S_i. Les résultats des actions de ces politiques sont dans $S_i \cup XPer(S_i)$. On construit donc le MDP $M_i(\sigma)$ qui comporte :

Etats	:	les états de S_i	+	les états de $XPer(S_i)$	+	α
Récompenses	:	récompenses sur les états de S_i	+	$\sigma : XPer(S_i) \to \mathbb{R}$	+	0

L'idée est la suivante : on remplace toutes les régions externes à S_i par les éléments de

$XPer(S_i)$. Afin que ce masque soit équivalent au problème initial, il faut que les récompenses sur les états de $XPer(S_i)$ soient égales aux espérances totales de gain atteignable en partant de $XPer(S_i)$ (vers l'« extérieur » de S_i). C'est ce que l'on cherche à représenter en introduisant la fonction σ ; en effet, les transitions du MDP sont similaires à celles représentées à la figure 2.5 : Au sein de S_i, le modèle de transition demeure le même vers les éléments de S_i et de $XPer(S_i)$; à partir de chaque élement de $XPer(S_i)$, on construit une unique transition vers un état absorbant artificiel α représentant le monde extérieur absorbant. La récompense associée aux transitions à partir des états de $XPer(S_i)$ est celle qui doit avoir la valeur de l'élément de S_i en question dans le MDP M afin de correctement représenter le problème. C'est pourquoi on définit la fonction $\sigma : XPer(S_i) \to \mathbb{R}$ afin de définir la récompense associée à cette transition. Hauskrecht montre que si on a un MDP M de fonction de valeur optimale V, qu'on en extrait un MDP abstrait M' et qu'on le résout avec des MDP locaux $M_i(\sigma_i)$ eux-mêmes définis avec des σ_i tels que $|\sigma_i(s) - V(s)| < \epsilon$ pour $s \in XPer(S_i)$ alors la valeur optimale V' obtenue pour M' est telle que $|V'(s) - V(s)| \leq \frac{2\epsilon\gamma}{1-\gamma}$. Cette assertion indique que si on connait assez précisément la fonction de valeur optimale, alors on peut trouver une politique optimale pour le MDP abstrait et donc une politique quasi optimale pour le MDP d'origine. Grosso modo, cela revient à dire que si on connaît la fonction de valeur du MDP M alors on connait de façon assez précise la fonction de valeur optimale du MDP abstrait M'. Malheureusement, si on connaissait de façon précise la fonction de valeur optimale de M, alors on n'aurait plus de problème à résoudre. Afin de contourner cette difficulté, Hauskrecht propose de discrétiser l'intervalle des valeurs de sortie possibles du MDP local et de le résoudre autant de fois que nécessaire afin d'obtenir un jeu de macro actions correspondantes à chaque valeur supposée. C'est le rôle de la fonction σ dont on fait varier les valeurs dans l'intervalle des valeurs accessibles depuis les états de sortie. On peut approcher de façon grossière cet intervalle en le minorant par le gain minimum présent dans l'univers et en le majorant par la somme des gains qu'on peut obtenir, il y a donc un nombre fini de points de discrétisation de cet intervalle pour un pas de discrétisation δ donné. On calcule donc la politique optimale π_i pour chaque σ_i et une fois qu'on a acquis ainsi un jeu complet de macro-actions on sait que parmi elles il y a des politiques locales qui sont à moins de $\delta/2$ de la politique optimale.

A la fin de cet algorithme, on a, par région S_i et par jeu de valeurs σ_i, une politique optimale qui mène nécessairement vers un des états de $XPer(S_i)$. On a donc un jeu de macro-actions définies sur S_i. Dans le MDP abstrait composé uniquement des éléments de $Per_{\Pi}(S)$ on connait les modèles de transition et de récompense associés à ces macro-actions. On peut donc résoudre le MDP abstrait et cette solution nous donne une solution quasi-optimale au MDP initial. C'est ainsi qu'Hauskrecht construit une politique quasi-optimale au problème en en décomposant l'espace d'état.

Il est important de noter que la discrétisation de l'intervalle des valeurs de sortie d'une région et le calcul conséquent des politiques optimales est un processus très gourmand en ressources qui ne se justifie que par la réutilisation que l'on peut faire des politiques ainsi calculées. Un tel modèle est très utile par sa robustesse quand les récompenses sont sujettes à changements, par exemple lorsque l'objectif du robot change de pièce, celui-ci recalcule simplement quelques politiques locales puis il réétablit son modèle de transition et récompense et il résout de nouveau son MDP abstrait sans avoir à calculer la totalité des macro-actions ou à re-considérer le problème dans son intégralité. Un autre aspect important que ce modèle ne prend pas en compte regroupe les macro-actions qui reviennent sur le même macro-état. Ces dernières sont en fait prises en compte dans le traitement des MDP locaux comme faisant partie d'une politique qui amène l'agent vers « la sortie », dans un modèle de MDP abstrait, il peut toutefois être souhaitable

de voir apparaître comme macro-actions des actions qui bouclent sur un état.

Dans la section suivante, on aborde un modèle proposé par Parr [Par98] qui permet d'éviter le problème de la multiplication des calculs lors de la discrétisation de l'intervalle des valeurs de sortie des régions.

2.4.4 Recherche dans l'espace des valeurs des états de sortie

Afin d'éviter les redondances avec la section précédente, on réutilisera les mêmes notations. Soit, donc, un MDP décomposé en régions S_i. Soit G une de ces régions. On pose V^0 le vecteur des valeurs des états de $XPer(G)$. A chaque politique π définie sur G, on associe la fonction :

$$f_\pi(\cdot, V^0) : \begin{cases} S & \rightarrow & \mathbb{R} \\ s & \mapsto & V^\pi(s) \end{cases}$$

L'objectif ici est de générer des jeux (ou caches) de politiques pour chaque région comme dans l'approche précédente ; par contre, on cherche à définir uniquement les politiques dont on va avoir besoin. On définit dans ce cadre les notions suivantes :

Une politique π définie sur G, est *optimale pour* V^0 si elle est solution du MDP constitué de G, $XPer(G)$ et l'état absorbant α et défini à la section précédente et à la figure 2.5.

Une politique π définie sur G, est *ϵ-optimale pour* V^0 si dans le MDP local précédemment défini :
$\forall s \in G, BE(V^\pi(s)) \leq \epsilon$
ou encore :
$BE(V^\pi) \leq \epsilon$
(avec $BE()$ l'erreur de Bellman)

Une politique π_i est dominante en s pour V^0 si et seulement si :
$\forall j \neq i, f_{\pi_j}(s, V^0) \leq f_{\pi_i}(s, V^0)$

Un jeu (ou cache) de politiques définies sur G est ϵ-optimal en s si pour tout V^0, la politique dominante en s est ϵ-optimale.

Un cache de politiques est ϵ-optimal s'il est ϵ-optimal en tout $s \in G$.

On commmence par montrer ([Par98]) que :

Théorème 4 (optimalité de la solution du MDP abstrait). *Soit un MDP décomposé en m régions $G_1, \ldots, G_m$. Soit, pour chaque région G_i un cache ϵ-optimal de politiques π_i. Alors on peut combiner ces politiques en une politique globale $\frac{\epsilon}{1-\gamma}$-optimale en résolvant le PDM ayant pour espace d'état $\cup_{i=1}^{n} XPer(Gi)$ et comme espace d'action les macro-actions constituées des politiques contenues dans les caches.*

Ce résultat est similaire à celui établi plus haut par Hauskrecht. On est donc rendu au point où il faut calculer les caches de politiques afin de les combiner en une politique globale.

L'approche de discrétisation à pas constant de l'espace d'état induisant beaucoup trop de calculs, on va se concentrer sur une recherche optimale dans l'espace des valeurs de sortie. Pour celà on introduit le lemme :

Proposition 3. *Pour tout $s \in G$ la valeur de la politique dominante en s est une fonction de V^0 convexe, linéaire par morceaux.*

Afin de percevoir l'importance de ce lemme, on peut condisérer un exemple simple. Supposons qu'on ait un agent dans un monde grille de faibles dimensions (quelques cases de côté). Ce monde grille est en fait un MDP local d'un problème plus grand. Considérons un unique état de sortie à ce MDP affecté de la valeur V^0 représentant l'espérance de gain du « monde extérieur ». Considérons enfin que dans ce monde grille, il y a une récompense positive associée à un état donné qu'on ne peut prendre qu'une fois (il suffit de se déplacer sur la case de l'état pour percevoir la récompense). Quand V^0 est petit (supposons qu'on le fasse commencer à 0), la politique dominante dans la région converge vers les plus grands gains possibles qui se trouvent alors *dans* la région. Avec cette première politique π_1, la fonction de valeur de π_1 en s est affine en V^0 (affine et non constante car comme on est dans un cadre incertain, on a des probabilités de transition constantes mais non nulles vers l'état de sortie). Puis, à force d'augmenter V^0, on dépasse un certain seuil et la politique optimale change en π_2, de fonction de valeur également affine en V^0. Cette fonction de valeur est supérieur à ce que la politique π_1 aurait donné puisque par définition, π_2 est la nouvelle politique optimale. Par ailleurs, cette croissance est plus rapide que précédemment car plus V^0 est grand, plus les actions vont droit vers la sortie et plus les probabilités de transition vers la sortie sont grande.

La figure 2.6 illustre cet exemple simple.

On peut effectuer la même opération en prenant plusieurs états de sortie : on fait alors varier les élements de V^0 individuellement et de la même manière que précédemment, la valeur de la politique dominante en s est affine en les éléments de V^0. Si on cherche à le représenter graphiquement, à un état de sortie on peut simplement représenter les segments de droite de pentes croissantes qui forment une enveloppe convexe à la fonction $(V^\pi(s))(V^0)$. Avec n états de sortie, cette fonction se représente dans un espace à $n+1$ dimensions comme un hyperplan convexe.

Ainsi de suite entre V_{min} et V_{max} ; on définit ainsi des morceaux d'hyperplans qui définissent un intérieur convexe.

On cherche à contruire des jeux minimaux de politiques optimales qui « sautent » d'un état d'entrée d'une région à un état de sortie. Pour construire ces caches minimaux, l'idée générale est de rechercher, par rapport au cache courant, où, dans l'espace des V^0, l'erreur de Bellman sur les états d'entrée pour la politique dominante du cache courant est la plus grande et de construire une meilleure politique afin de compléter le cache courant. On construit ainsi un cache avec un nombre de politiques minimal puisqu'à chaque fois, on rajoute une unique politique et on sait qu'on couvre le pire cas. Afin de trouver ce pire cas, on introduit le théorème :

Théorème 5 (recherche de l'erreur de Bellman maximale). *Soit une région G d'un MDP décomposé. Sur la région G, on a :*

$$\begin{cases} n \text{ états dans l'espace d'état de } G \\ a \text{ actions disponibles sur } G \\ m \text{ politiques dans le cache de politiques } \Pi \\ d \text{ états de sortie} \end{cases}$$

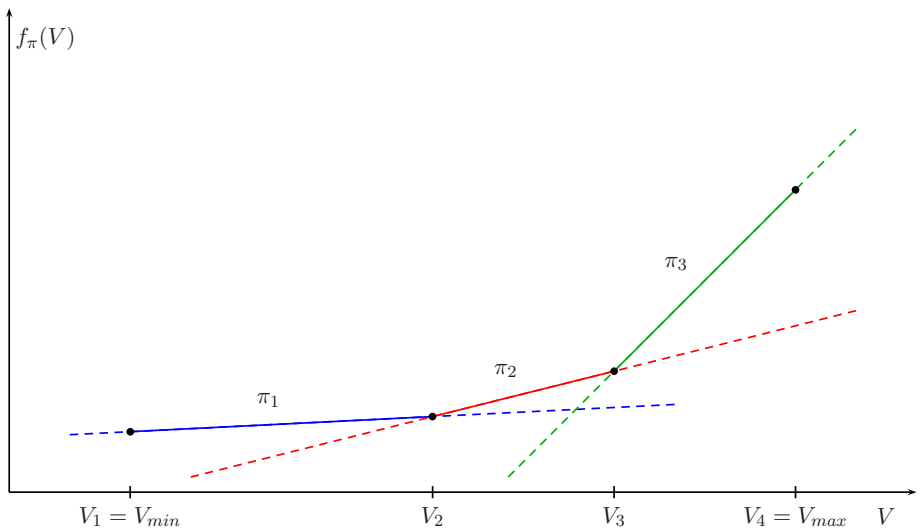

FIG. 2.6 – Illustration de l'évolution de la fonction de valeur de la politique dominante en fonction des valeurs de V^0. π_1 représente une politique qui reste dans la zone, le gain espéré à l'extérieur V étant trop faible pour que la politique optimale sorte de la zone. En V_2, le gain associé à π_2 dépasse celui de π_1 et π_2 devient la politique dominante. De même en V_3, c'est π_3 qui devient dominante. Comme les V augmentent, les politiques tendent à aller droit vers la sortie et donc les probabilités de transition vers les gains de la sortie augmentent. Cela fait que la pente indiquant la dépendance de f_π en V augmente et cela entraîne la convexité de la courbe.

Le point dans l'espace des V^0 pour lequel l'erreur de Bellman de la politique dominante est la plus grande est trouvable en un temps de calcul polynomial en n, a, m et d.

(on note qu'on en a profité pour nommer Π le cache de politiques, en remplacement du partitionnement de l'espace d'état) La preuve de ce théorème utilise un résultat de programmation linéaire, il s'agit de résoudre :

$$\forall a \in A, s \in G, t \in EPer(G), \pi \in \Pi$$
$$\max_{V^0} \left(r(s,a) + \gamma \sum_{s' \in G \cup Xper(G)} T(s,a,s') f_\pi(s', V^0) - f_\pi(s, V^0) \right)$$
$$\text{avec les contraintes :}$$
$$\forall \pi_i \in \Pi, f_{\pi_i}(t, V^0) \leq f_\pi(t, V^0)$$
$$\forall i \in [1, d], V^0[i] \leq V_{max}$$

La complexité d'un algorithme de programmation linéaire est polynomial en le nombre d'inconnues (ici d puisque l'inconnue est V^0, on notera cette complexité $O(d^p)$) et on effectue cette recherche pour tout état de G (n états), toute action de A (a actions), tout état de $EPer(G)$ (le nombre d'état de $EPer(G)$ est majoré par n) et pour toutes les politiques de Π (m politiques). On a donc un algorithme total de complexité $O(d^p \times n^2 \times a \times m)$ ce qui est bien polynomial en n, a, m et d.

L'algorithme précédent effectue bien la recherche voulue car le premier jeu de contraintes assure que la politique considérée est dominante et qu'on se trouve donc bien sur la fonction convexe évoquée dans le lemme. Par ailleurs, le deuxième jeu de contraintes majore V^0 et le

force à être dans les limites des valeurs possibles.

Cet algorithme permet de construire des chaches de politiques ϵ-optimaux : si au vecteur V_1^0 renvoyé par l'algorithme correspond une erreur de Bellman plus grande que ϵ, alors on rajoute au cache de politiques la politique optimale calculée pour V_1^0. A l'inverse, si V_1^0 est plus petit que ϵ alors on a l'assurance que dans notre cache de politiques, aucune n'aura d'erreur de Bellman en les états d'entrée de G plus grande que ϵ. Si on note $[V_{min}; V_{max}]$ l'intervalle de toutes les valeurs de sortie possibles, alors le théorème précédent permet de construire l'algorithme qui produit des caches ϵ-optimaux minimaux. Notons, comme dans [Par98], find-worst la fonction qui implémente l'alorithme de recherche du « pire point » présenté au théorème 5 qui renvoie la valeur de ce point et l'erreur de Bellman associée. Notons pol-opt la fonction qui pour un V^0 donné renvoie la politique optimale calculée sur G pour des valeurs de sortie valant V^0 (cette fonction peut être par exemple basée sur un algorithme d'itération de la valeur). Alors l'algorithme de construction des caches minimaux ϵ-optimaux sur G s'écrit comme à la figure 2.7.

FIG. 2.7 : Algorithme de construction d'un cache ϵ-optimal minimal

```
Π ← {pol-opt ((V_min, ..., V_min))}
quit ← false
répéter
    (pire-erreur, pire-point) = find-worst(Π)
    si pire-erreur > ε alors
        Π ← Π ∪ {pol-opt(pire-point)}
    sinon
        quit=true
jusqu'à quit == true
```

2.4.5 L'encadrement par enveloppe convexe

Une troisième approche possible du problème de la recherche d'un jeu minimal de politiques sur une région est apportée par l'encadrement par enveloppe convexe. Cette approche se base sur le lemme :

Proposition 4 (Encadrement de la fonction de valeur). *Soit un point V^0 de l'espace des valeurs des états de sortie. d est la dimension de V^0. Soit un jeu de points $V_1, \ldots, V_{d+1}$ tels que les segments reliant les (V_i, V_{i+1}) forment une enveloppe convexe autour de V^0 dans l'espace des valeurs des états de sortie. Soient les politiques $\pi_1, \ldots, \pi_{d+1}$, optimales respectivement par rapport à $V_1, \ldots, V_{d+1}$.*
Quel que soit $s \in S$, la valeur en s de la politique optimale par rapport à V^0 est minorée par $\max_i f_{\pi_i}(s, V^0)$ et majorée par l'hyperplan contenant les points $(V_i, f_{\pi_i}(s, V_i)$.
On peut réécrire ce résultat comme :

$$\forall s \in S, \pi \text{ optimale par rapport à } V^0 \Rightarrow \max_i f_{\pi_i}(s, V^0) \leq f_\pi(s, V^0) \leq H(s, V^0)$$

avec $H(s, V^0)$ l'équation de l'hyperlan contenant les $(V_i, f_{\pi_i}(s, V_i)$ évaluée en V^0.

Pour démontrer la minoration il suffit de faire le même raisonnement que dans le théorème précédent : la politique optimale au point V^0 a une valeur supérieure à toutes les autres politiques quel que soit l'état considéré donc elle a une valeur supérieure au max des valeurs de

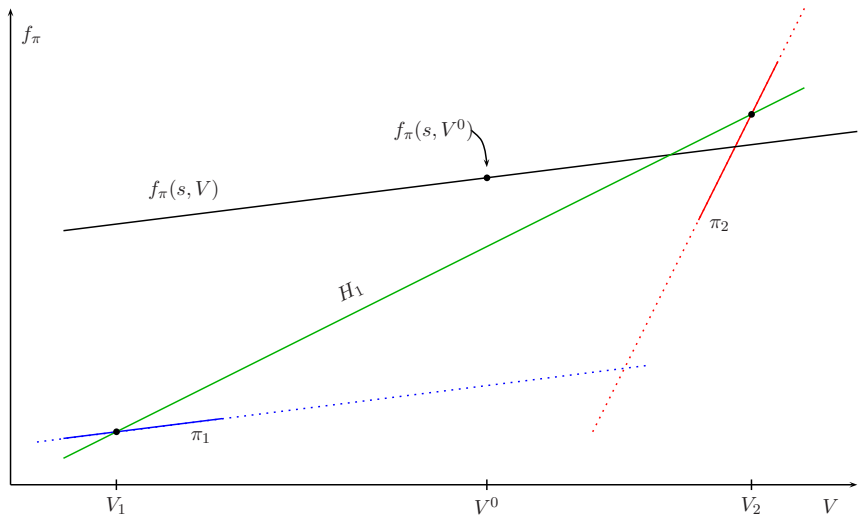

FIG. 2.8 – Illustration du raisonnement pour la démonstration du lemme dans le cas $d = 1$. Il apparait immédiatement que $f_\pi(s, V^0)$ doit être au dessus de $f_{\pi_1}(s, V^0)$ et de $f_{\pi_2}(s, V^0)$ sinon π n'est pas optimale en s par rapport à V^0. On en déduit la minoration. Par ailleurs, si $f_\pi(s, V^0)$ est plus grand que $H_1(V^0)$ alors en au moins un des deux points V_1 ou V_2 la fonction $f_\pi(s, V)$ est plus grande que $f_{\pi_i}(s, V_i)$. Supposons (comme sur le schéma) que ce soit en V_1 : on a $f\pi(s, V_1) > f_{\pi_1}(s, V_1)$ ce qui est absurde puisque π_1 est optimale en V_1. On en déduit la majoration.

toutes les autres politiques quel que soit l'état.

Pour démontrer la majoration, on raisonne par l'absurde :
On appelle H_1 l'hyperplan contenant les $(V_i, f_{\pi_i}(s, V_i))$.
On appelle H_2 l'hyperplan correspondant à la politique π (optimale pour V^0) en s.
H_1 est issu des « coins » de l'enveloppe convexe (les $(V_i, f_{\pi_i}(s, V_i))$), si $f_\pi(s, V^0) > H_1(V^0)$ alors il existe un coin (un V_i) où $f_\pi(s, V_i) > f_{\pi_i}(s, V_i)$ ce qui est impossible puisque π_i est optimale en s par rapport à V_i. On a donc bien $f_\pi(s, V^0) \leq H_1(V^0)$.

La démonstration du lemme est illustrée figure 2.8.

On utilise alors ce lemme pour démontrer le théorème :

Théorème 6. *Soit une région G d'un MDP décomposé. Soient les politiques locales $(\pi_i)_{i=1..m}$ optimales aux points $(V_i)_{i=1..m}$. Alors pour tout $s \in G$ et pour tout jeu de valeurs de sortie V^0, la valeur de la politique optimale en V est minorée par la surface convexe définie par les f_{π_i} et majorée par les facettes de la surface convexe reliant les $(V_i, f_{\pi_i}(V_i)$.*

La minoration est immédiate après avoir vu le lemme sur l'encadrement de la fonction de valeur.

La majoration se fait par morceaux : on prend le plus « bas » localement des hyperplans du lemme précédent reliant les $(V_i, f_{\pi_i}(s, V_i)$ et on applique le lemme localement. On dit donc que V^0 est dans l'enveloppe convexe des V_i définissant cet hyperplan ; la fonction de valeur optimale est donc majorée par cet hyperplan. Docn sur tout le domaine, la fonction de valeur optimale est majorée par le collage bout à bout de toutes ces petites portions d'hyperplans.

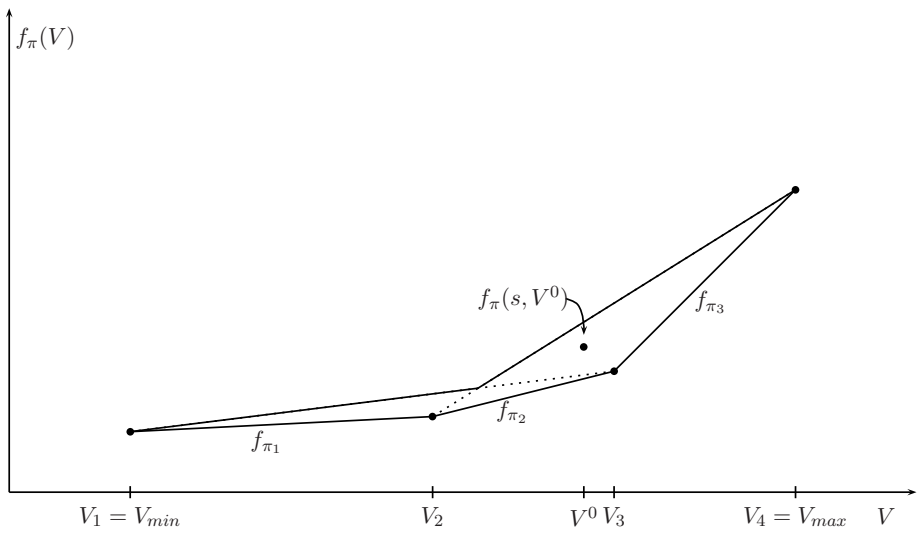

FIG. 2.9 – Illustration de l'encadrement de la fonction de valeur optimale par les enveloppes convexes dans le cas $d = 1$. On a repris l'exemple de la figure 2.6 afin de bien mettre en évidence les deux coquilles convexes entre lesquelles est comprise la fonction de valeur optimale. Selon l'algorithme qui construit les jeux minimaux de macro-actions, on va chercher la politique optimale en V_3 car c'est là que l'écart entre les deux coquilles est le plus grand, on ajoutera alors cette politique π_4 au cache courant de politiques et on rajoutera les points d'intersection de $f_{\pi_4}(V_4)$ avec la coquille inférieure afin de reconstruire deux nouvelles coquilles plus proches. On itère ainsi jusqu'à avoir un écart entre les coquilles inférieur à ϵ.

Ce collage regroupe toutes les plus basses portions des hyperplans, c'est donc bien la surface convexe reliant les $(V_i, f_{\pi_i}(V_i)$.

On encadre donc la fonction de valeur optimale sur tout le domaine entre deux coquilles, deux enveloppes convexes. Dans le cas $d = 1$, ce théorème est illustré à la figure 2.9.

Pour tenter de construire un jeu de politiques optimales, on observe l'écart entre les deux coquilles et là où il est le plus grand, s'il est plus grand que ϵ, on calcule la politique optimale et on l'ajoute au cache courant de politiques pour la région G. On définit alors deux nouvelles coquilles avec le nouveau jeu de politiques et on recommence.

Avant d'aborder un autre type d'algorithmes de décomposition, il est important de noter que la compexité des algorithmes de décomposition limitent leur efficacité. En effet, même si l'approche type « diviser pour régner » permet un traitement simple, fin et compact du problème, les diverses itérations rendent vite la simplification trop ... complexe. L'objet de cette étude n'est pas d'approfondir ce sujet mais une réflexion y a été menée et la complexité des algorithmes de décomposition rend leur utilisation dans un cadre systématique relativement dangereuse.

Les deux sections suivantes sont volontairement raccourcies afin de centrer le propos sur les idées qui importent à notre problématique sans faire de redite des auteurs cités.

2.4.6 La décomposition en étoile

Dans [DL95], Dean et Lin présentent une autre approche décomposée où on définit une région U qui contient tous les états périphériques issus de la partition de l'espace d'état. On définit alors les macro-états comme les régions elles-mêmes, ces dernières communiquant via U. On affecte à chaque état de U un paramètre λ_i et on note λ le vecteur de tous les λ_i.

Pour un λ donné, le MDP originel peut être décomposé en petits MDPs séparés, chacun déterminant une politique locale sur chaque région R. Cette politique est définie sur les états de $R \cup Per(R)$, avec comme modèle de transition, celui de R qu'on complète par la définition des états de $Per(R)$ comme des états absorbants. De même, les récompenses sont définies comme dans R et les transitions vers $Per(R)$ se voient affectées de leur valeur d'origine à laquelle on ajoute le λ_i correspondant. Les transitions des états absorbants de $Per(R)$ vers eux même sont à récompense (ou coût) nulle. Chacune des politiques locales optimales pour un λ donné sont définies comme des macro-actions qui assurent qu'on agira de façon optimale dans la région concernée.
[DL95]

2.4.7 Un algorithme de décomposition automatique

Dans tous les modèles précédents, on part d'une partition pré-existante de l'espace d'état pour construire des jeux de politiques hiérarchisées approximant la politique optimale. Un algorithme de construction automatique de partitions basé sur la recherche de zones découplées de l'espace d'état (ie. ayant peu de transitions entre elles) est donnée dans [Sab02] et pourra servir de base à une automatisation du traitement d'un MDP de manière décomposée.

2.5 La décomposition MAXQ, une représentation hiérarchisée de la mission

2.5.1 Le graphe MAXQ

L'approche MAXQ est une méthode de hiérarchisation de l'espace d'action. Elle consiste principalement en la construction d'une hiérarchie de tâches (et donc d'une interprétation de la dépendance d'une tâche ou d'une action sur une autre) et en la représentation associée de la fonction de valeur. Les travaux de Dietterich de développement de la méthode MAXQ ([Die98], [Die00]) et d'utilisation de cette dernière pour l'apprentissage par renforcement constituent la base de cette section et de l'utilisation qui en est faite plus loin.

L'idée principale qui motive l'approche MAXQ repose dans la remarque suivante : lors d'une mission, l'ensemble des actions peut être intuitivement factorisé en tâches et sous-tâches. L'exemple du problème érarchisation des actions. Pour un taxi occupant un monde-grille, devant prendre un passager en un point donné, le déposer en un autre, ayant la contrainte de son carburant limité et disposant d'une station service en un dernier point donné, les actions de l'agent au cours de la mission sont décomposables en trois grandes tâches : « prendre le passager », « déposer le passager », « faire le plein ». Chacune de ces tâches fait appel à des sous-tâches (comme la sous tâche « naviguer ») ou des actions élémentaires comme « faire monter le passager ».
La structure hiérarchique de l'espace d'action apparait donc naturellement lors de l'interprétation de la mission.

On représente cette hiérarchie dans un graphe MAXQ comme celui de la figure 2.10. Un tel graphe contient deux types de noeuds :

- Les « Max nodes ». Ceux-ci représentent les tâches elles-mêmes. Ils sont représentés dans le graphe par des triangles. Un Max node n'ayant pas de noeud enfant représente une action élémentaire de l'agent. Un Max node ayant des noeuds enfants est une tâche. La racine de l'arbre du graphe MAXQ est le Max node « Mission », c'est la tâche principale qui va faire appel à ses sous-tâches. Dans l'exemple du taxi de [Die98] les Max nodes sont donc les tâches « mission », « prendre le passager », « déposer le passager », « faire le plein », « naviguer » mais aussi les actions élémentaires « faire monter », « faire descendre », « remplir le réservoir », « Nord », « Sud », « Est », « Ouest ». Chaque Max node a pour descendants des Q nodes.
- Les « Q nodes ». Ils sont représentés dans des ovales. Un Q node représente une action executée exclusivement dans le cadre d'une tâche donnée (sa tâche parent). Dans l'exemple précédent, on peut noter par exemple que la tâche « déposer le passager » va faire appel entre autres à la tâche « naviguer » pour amener le passager à destination. Il y a donc un Q node « naviguer pour déposer le passager » qui représente l'action de navigation issue de la tâche « naviguer » et à laquelle la tâche « déposer le passager » fait appel.

On construit ainsi un graphe complet hiérarchisant les tâches et actions de l'agent. Ce graphe commence à la racine par la tâche la plus générale : "mission" et se propage en alternant Max nodes et Q nodes jusqu'aux feuilles qui sont les Max nodes des actions élémentaires de l'agent.

L'intérêt immédiat d'une telle décompostion est la réduction des variables nécessaires à la décision pour chaque tâche. Le fait de combiner des actions de base pour former des tâches permet de considérer chaque tâche comme un MDP de dimension plus petite que le MDP global. En effet, au sein d'une même tâche, seulement une partie de l'état est cruciale à la prise de décision (la tâche naviguer(emplacement) n'a pas besoin de connaître la variable d'état "passager à bord" pour décider de sa meilleure stratégie). On définit donc, pour chaque tâche, un MDP réduit car on ne considère alors qu'un espace d'action et d'état réduit.

Il faut noter sur le graphe de la figure 2.10 que la tâche « Naviguer(but) » représente en fait autant de MDP (et donc de politiques optimales) qu'il y a d'états buts à la navigation.

Au sein d'une même tâche, on distingue particulièrement les états terminaux. Les états terminaux sont définis comme les états qui font se terminer la tâche courante quand on y rentre. Parmi ces états terminaux, on retrouve les états buts de la tâche en question.

2.5.2 La résolution du problème sous forme de graphe MAXQ

La solution d'un problème posé sous forme d'un graphe MAXQ est un jeu de politiques hiérarchisé — une par Max node — $\{\pi_0, \pi_1, \ldots, \pi_n\}$. Chaque politique est définie sur un Max node donné et s'applique donc sur l'espace d'état réduit du Max node en question et renvoie comme actions les Q nodes associés. De ce point de vue, l'organisation hiérarchique de l'approche MAXQ est similaire au concept de programmes informatiques appelant des sous-programmes. Une politique (comme un sous-programme) s'exécute jusqu'à parvenir dans un état terminal. Quand c'est le cas, c'est l'action du Q node de niveau immédiatement supérieur qui a fait appel à la politique qui s'achève.

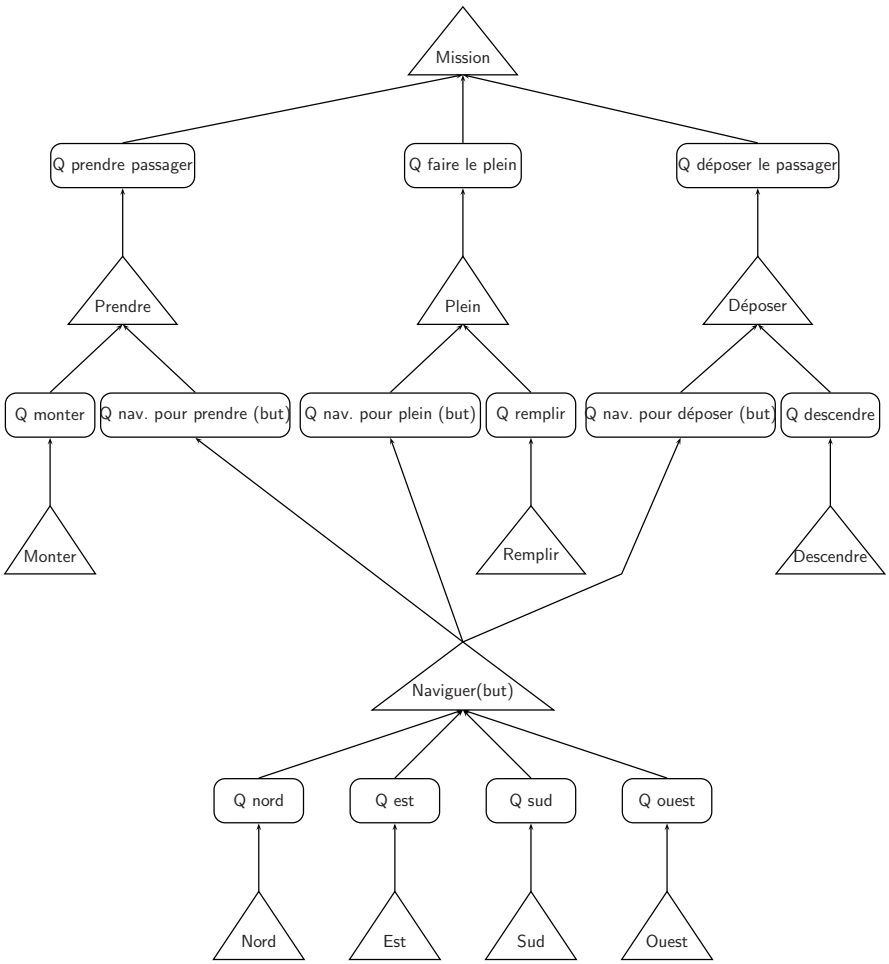

FIG. 2.10 – Un graphe MAXQ pour le problème du taxi

La résolution des politiques des Max nodes et donc l'obtention de la politique optimale pour le problème utilise une représentation hiérarchique de la fonction de valeur, de la même manière qu'on a créé une représentation hiérarchique de la politique. Considérons un Max node M_i et posons $V_i^\pi(s)$ la valeur de l'état s lorqu'on execute la politique π (contenant la politique π_i définie au Max node M_i). Cette politique π va s'exécuter jusqu'à parvenir en un état terminal de π_i. Par ailleurs, π_i associe aux états du problème des actions qui sont les Q nodes issus de M_i en les considérant comme des actions élémentaires. On peut donc écrire la probabilité de transition de s à s' en effectuant l'action a (associée au Q node Q_a et par là, au Max node M_a) comme $P_i(s'|s,a)$. Cette probabilité est bien définie si a est une action élémentaire. Dans le cas où M_a ne représente pas une action élémentaire, a représente une tâche M_a. La politique sur M_a définie par π est π_a. Si on connaît toutes les probabilités de transition des actions entreprises au sein de la tâche M_a alors on peut calculer la probabilité de transition de la tâche a elle-même. De même, tous les descendants de M_a appliquent une politique fixée jusqu'à arriver aux actions élémentaires. On peut donc définir sans ambugüité la valeur de $P_i^\pi(s'|s,a)$. On peut définir ainsi la récompense immédiate associée à l'exécution de a en s dans M_i comme :

$$V_i^a(s) = \begin{cases} r(s,a) & \text{si } a \text{ est une action élémentaire} \\ V_a^\pi(s) & \text{si } a \text{ est une tâche} \end{cases} \tag{2.13}$$

Dans le second cas $V_a^\pi(s)$ est la l'espérance de récompense associée à l'application de la politique π_a jusqu'à ce qu'on parvienne à un état terminal de M_a.

On peut alors écrire la fonction de valeur associée à un état s lors de l'exécution de la politique π dans le Max node M_i comme :

$$V_i^\pi(s) = V_a^\pi(s) + \sum_{s'} P_i(s'|s,a) \cdot V_i^\pi(s') \quad \text{avec } a = \pi_i(s) \tag{2.14}$$

Cette formulation présente l'intérêt de donner une définition récursive de la fonction de valeur. Elle montre par ailleurs que la fonction de valeur associée à un état dans la tâche racine "mission" représente la fonction de valeur de l'état en question selon la politique globale courante sur le MDP global.

On peut reformuler l'expression précédente en utilisant la représentation sous forme de Q-fonctions. La Q-fonction $Q_V(s,a)$ représente la valeur associée à l'action a dans l'état s sachant que les états d'arrivée sont décrits par la fonction de valeur V. Dans le cas d'un MDP classique, Q_V s'écrit :

$$Q_V(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s',a,s)V(s') \tag{2.15}$$

Et la recherche de politique optimale devient une maximisation itérative de la fonction Q_V. Ici, en posant :

$$C_i^\pi(s,a) = \sum_{s'} P_i(s'|s,a) \cdot V_i^\pi(s') \tag{2.16}$$

Et en écrivant que Q_i^π est la Q-fonction définie sur le MDP M_i et pour la fonction de valeur associée à la politique π, on peut écrire :

$$Q_i^\pi(s,a) = V_i^a(s) + C_i^\pi(s,a) \tag{2.17}$$

On peut alors écrire pour généraliser l'expression de la valeur d'un état dans une politique donnée :

$$V_i^\pi(s) = \begin{cases} Q_i^\pi(s, \pi_i(s)) & \text{si } M_i \text{ a des descendants} \\ \sum\limits_{s' \in S_i} P(s'|s, \pi(s)) R(s'|s, \pi(s)) & \text{si } M_i \text{ est une action élémentaire} \end{cases} \tag{2.18}$$

Le calcul de la fonction de valeur associée à une politique MAXQ (donc à un jeu de politiques - une par MDP M_i) se fait alors par propagation des valeurs depuis les feuilles de l'arbre vers la racine. Ainsi, pour une politique $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$, la recherche de la fonction de valeur globale V^π, emprunte les étapes :

$V^\pi(s) = V_0^\pi(s)$	D'après (2.18) on a :
$V^\pi(s) = Q_0^\pi(s, \pi_0(s))$	Posons $\pi_0(s) = a_0$. Avec (2.17) on a :
$V^\pi(s) = V_0^{a_0}(s) + C_0^\pi(s, a_0)$	Or d'après (2.13) on a :
$V_0^{a_0}(s) = V_{a_0}^\pi(s)$	Et avec (2.18) :
$V_{a_0}^\pi = Q_{a_0}^\pi(s, \pi_{a_0}(s))$	On pose $\pi_{a_0}(s) = a_1$
$V^\pi(s) = Q_{a_0}^\pi(s, a_1) + C_0^\pi(s, a_0)$	On a donc (avec (2.17)) :
$V^\pi(s) = V_{a_0}^{a_1}(s) + C_{a_0}^\pi(s, a_1) + C_0^\pi(s, a_0)$	Soit (avec (2.13)) :
$V^\pi(s) = V_{a_1}^\pi(s) + C_{a_0}^\pi(s, a_1) + C_0^\pi(s, a_0)$	

De même, le chemin des tâches parcourues dans le graphe MAXQ est noté $(a_0, a_1, \ldots, a_n)$, on obtient au final (en itérant les étapes précédentes) :

$$V^\pi(s) = V_{a_n}^\pi + C_{a_{n-1}}(s, a_n) + \cdots + C_{a_0}^\pi(s, a_1) + C_0^\pi(s, a_0) \tag{2.19}$$

Pour chaque feuille a de l'arbre MAXQ (donc pour chaque action élémentaire), on connait V_a^π grâce à (2.18). On peut donc calculer les $C_i^\pi(s, a)$ de proche en proche selon (2.16), (2.18) et (2.17) et donc déterminer la valeur de V^π en propageant la fonction de valeur des feuilles de l'arbre jusqu'à sa racine.

2.5.3 Le critère γ-pondéré et l'introduction du temps dans la représentation MAXQ

La formulation précédente correspond à l'expression du critère moyen. On peut aisément se ramener au critère γ-pondéré et à un cadre SMDP comme dans [GMM03].

La fonction de valeur s'exprime :

$$V_i^\pi(s) = E_s^\pi\{\sum_{n=0}^{\infty} \gamma^{\sigma_n} r(s_n, a_n)\} \tag{2.20}$$

avec $r(s_n, a_n)$ défini comme en (2.7) et σ_n défini comme en (2.4).

L'équation (2.14) devient :

$$V_i^\pi(s) = V_a^\pi(s) + \sum_{s'} P_i(s'|s, a) \int_0^\infty \gamma^t V_i^\pi(s') F_i(dt|s, a) \quad \text{avec } a = \pi(s) \tag{2.21}$$

avec $F_i(dt|s, a)$ défini comme précédemment comme la probabilité que la porchaine date de décision survienne dans l'intervalle dt alors qu'on a effectué l'action a en s dans le cadre de la

tâche i.

De même on peut réécrire la définition de la Q-fonction (2.15) :

$$Q_i^\pi(s,a) = V_i^a(s) + \sum_{s'} P_i(s'|s,a) \int_0^\infty \gamma^t V_i^\pi(s') F_i(dt|s,a) \tag{2.22}$$

Ce qui redéfinit $C_i^\pi(s,a)$ comme :

$$C_i^\pi(s,a) = \sum_{s'} P_i(s'|s,a) \int_0^\infty \gamma^t V_i^\pi(s') F_i(dt|s,a) \tag{2.23}$$

On a donc bien toujours l'équation (2.17) :

$$Q_i^\pi(s,a) = V_i^a + C_i^\pi(s,a)$$

La représentation MAXQ est principalement utilisée dans le cadre de l'apprentissage par renforcement, sujet qui ne sera pas abordé ici. Afin de résoudre le problème posé sous forme de graphe MAXQ, on résout de façon standard les MDP hiérarchisés dans le graphe MAXQ jusqu'à disposer d'une politique hiérarchisée $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$ optimale.

2.6 Discussion sur le lien entre décomposition et graphe MAXQ

Dans le cadre le la présente recherche, on s'est intéressé aux problèmes de réduction de l'espace d'état, de factorisation des variables déterminantes et de simplification de l'algorithme de prise de décision. Sans aborder la thématique multi-agents qui nous occupera plus loin ni rentrer dans le détail du problème particulier considéré, on peut tirer quelques conclusions quant aux relations entre l'approche décomposée et la décomposition MAXQ.

On a cherché à combiner ces deux approches ou, du moins, à mettre en évidence leurs similarités et leurs liens, afin de tenter d'exploiter leurs intérêts combinés. La question posée peut-être formulée ainsi : « Quelle est la place du MAXQ dans l'analyse et la décomposition du MDP de l'agent et quelle est la place de l'approche décomposée ? ». Cela a mis en lumière plusieurs aspects antagonistes des deux approches.

L'approche décomposée :
- s'appuie sur S (l'espace d'état) et T (la fonction de transition) et isole des zones ayant peu de liens (transitions) entre elles,
- peut-être automatisé,
- traduit la structure intrinsèque de l'univers dans lequel évolue l'agent (les couplages entre zones),
- permet de générer des macro-actions pouvant être vues comme des tâches locales et de changement de région.

La représentation MAXQ :
- s'appuie sur une *interprétation a priori* par l'utilisateur des tâches de la mission et des capacités du robot,
- n'est pas automatisable comme on va le voir dans la suite.

– traduit la structure de la mission courante et en dépend entièrement.

Le principal souci lorsqu'on cherche à combiner les approches MAXQ et décomposition repose sur le fait que l'une des approches est ascendante et repose sur une construction simple (la décomposition part des états et construit de façon systématique des zones) tandis que l'autre est descendante et se base sur une interprétation par l'utilisateur (l'approche MAXQ décompose la mission en grande tâches pour arriver aux actions élémentaires et il n'y a pas de processus d'organisation systématique de la mission en tâches).

Afin d'harmoniser les deux approches, on a cherché à donner un cadre mathématique à la décomposition de l'espace d'action afin d'envisager un processus de décomposition automatique ou, du moins, afin de mettre en évidence les propriétés et les relations des actions entre elles.

Dans cette optique, on peut commencer par remarquer que les actions peuvent être regroupées en tâches de la façon suivante : On peut regrouper en une tâche les actions qui affectent exactement les mêmes variables. Un exemple de ce cas est donné par la tâche « Naviguer » du problème du taxi qui regroupe les actions « Nord », « Sud », « Est » et « Ouest ».

On considère ainsi l'ensemble $\widetilde{S}$, ensemble des variables d'état. Avec S l'espace d'état, considérons $X \in \widetilde{S}$, alors X prend ses valeurs dans S_X et S est le produit cartésien de tous les S_X. En particulier : posons S_X l'ensemble des valeurs que peut prendre la variable X, S_Y constitue l'ensemble des valeurs que peut prendre la variable Y. Si X et Y sont les deux variables d'état du système, alors $S = S_X \times S_Y$ et $\widetilde{S} = \{X, Y\}$.

Avec A l'espace d'action du système et $\mathcal{P}(\widetilde{S})$ l'ensemble des parties de $\widetilde{S}$, on définit l'application :

$$affect : \left\{ \begin{array}{lcl} A & \rightarrow & \mathcal{P}(\widetilde{S}) \\ a & \mapsto & \{\text{variables sur lesquelles agit } a\} \end{array} \right.$$

De même on définit :

$$dep : \left\{ \begin{array}{lcl} A & \rightarrow & \mathcal{P}(\widetilde{S}) \\ a & \mapsto & \{\text{variables dont dépend le résultat de } a\} \end{array} \right.$$

On peut voir ces deux fonctions de la façon suivante : lorsqu'on considère une transition lors de l'exécution de l'action a, on considère la probabilité de transition vers j sachant qu'on exécute a en s : $P(j|a, s)$. $affect(a)$ représente alors les variables importantes dans j (celles qui vont vraiment changer) et $dep(a)$ représente les varaibles importantes dans s (celles dont dépend la valeur de la probabilité de transition).

On peut alors définir des classes :

$$\begin{array}{ll} \forall X \in \mathcal{P}(\widetilde{S}), & Aff_X = \{a \in A / X \subset affect(a)\} \\ \forall X \in \mathcal{P}(\widetilde{S}), & Aff'_X = \{a \in A / X = affect(a)\} \end{array}$$

Aff_X (et Aff'_X) représente l'ensemble des actions qui affectent entre autres X (uniquement X).

En effectuant un petit saut d'anticipation sur la suite du document et en considérant la formalisation du problème du taxi présentée à la section 3.2 on peut appliquer la fonction $affect$

à l'exemple du taxi.

Action a	$affect(a)$
Nord	P
Sud	P
Est	P
Ouest	P
Remplir	Fuel
Monter	Pass_in, Pass_att
Descendre	Pass_in, Pass_att

A partir des définitions de ces classes, on peut immédiatement effectuer un regroupement des actions en tâches qui affectent la même variable. Cependant on ne traduit pas l'aspect hiérarchique des tâches que l'on cherche pour se rapprocher de la représentation MAXQ.

On peut alors construire un processus automatique d'analyse de mission. Supposons que, du point de vue global, trois récompenses soient disponibles : R_1, R_2 et R_3 avec $R_i = r(s_i)$. Si on note l'état initial s_0 alors l'objectif de l'analyse de mission est de trouver les grandes étapes afin de passer de s_0 à s_i. Prenons le cas de s_1.

s_1 est un vecteur de taille $|\widetilde{S}|$ qui peut s'écrire :

$$s_1 = \{X_1 = x_{11}, X_2 = x_{12}, \ldots, X_{|\widetilde{S}|} = x_{1|\widetilde{S}|}\}$$

On cherche dans un premier temps à passer de x_{01} à x_{11}. Pour cela, on prend les éléments de Aff'_{X_1} et de Aff_{X_1} et on les regroupe en une tâche « agir sur X_1 ». En détaillant ce processus, on arrive à une hiérarchisation des actions dans un selon une optique qui peut être résumée par : « comment décomposer l'espace d'action en actions élémentaires puis les regrouper en tâches avec comme objectif que chaque tâche principale corresponde à l'obtention d'une récompense et que chaque sous-tâche corresponde à la transformation de l'état courant vers l'état de la récompense ? ». En effectuant une telle décomposition, on se retrouve avec des tâches « orthogonales » (ie. indépendantes les unes des autres) que l'on peut combiner à loisir afin de trouver la politique optimale permettant de rallier s_0 à s_1 par exemple.

Cette approche si elle semble intéressante au premier abord ne présente que la similarité de la hiérarchisation avec l'approche MAXQ. En effet, les tâches obtenues ne correspondent pas à une organisation hiérarchique de la mission mais plutôt à un découplage des actions.

A la suite de cette approche, on a été mené à considérer la question de savoir si la hiérarchisation doit dépendre de la mission elle-même. En effet, si elle dépend de la mission, il y a une forte contrainte de précédence sur les tâches et la hiérarchisation doit alors se baser sur cette contrainte. A l'opposé, si le processus de hiérarchisation ne dépend pas de la mission, alors il doit traduire les capacités propres de l'engin.

Il est assez aisé de se rendre compte que la hiérarchisation dépend obligatoirement de la mission. Si, par exemple, on reprend l'exemple du taxi et que l'on change très légèrement la mission en rajoutant une récompense pour aller faire un passage en un point donné (aller chercher les places gratuites pour le concert de ce soir par exemple) — la récompense étant attribuée juste sur l'action de navigation par le point de passage — alors la hiérarchisation présentée à la figure

2.10 n'est plus valable puisque'on n'a pas accès directement à la tâche « Navigate(but) » depuis la tâche racine « Mission ».

Enfin, et d'un point de vue complètement différent, il est important que préciser que la résolution d'un problème sous forme MAXQ n'est pas toujours possible sans faire appel à d'autres arguments de type empirique. En effet, si on prend une des tâches « Naviguer(but) » du problème du taxi initial, il n'y a *a priori* pas de récompense associée à un état en particulier et toutes les politiques se valent puisqu'aucune récompense n'est reçue pour une action de navigation. Il faut donc garder à l'esprit que l'on rajoute artificiellement des récompense sur les états buts et les états terminaux d'une tâche afin de pouvoir calculer la politique optimale qui y correspond. Cette approche empirique s'éloigne d'une représentation réaliste de la mission. De fait, cette représentation, comme mentionné plus haut, est particulièrement utilisée dans le cas de l'apprentissage par renforcement. Chaque tâche est initialisée avec une politique donnée et les fonctions de valeur associées aux états sont mis à jour au fur et à mesure de l'apprentissage. Dans le cas d'un traitement direct du MDP, cet aspect est limitant puisque si on n'impose pas une politique initiale ou des récompenses arbitraires et non-réalistes, on ne peut pas calculer une politique optimale en particulier.

Cette brève étude, si elle ouvre des pistes intéressantes sur une possible analyse de mission via la construction d'un arbre des tâches, souligne très fortement les antagonismes entre approche décomposée et représentation MAXQ. La première traduit la structure probabiliste et les couplages présents dans l'univers de jeu tandis que l'autre exprime une organisation possible *a priori* de la hiérarchisation des tâches à effectuer. Cet aspect souligné, il apparaît assez difficile de faire cohabiter les deux représentations. On verra sur le problème concret considéré plus loin que l'on conserve l'aspect de décomposition automatique afin de construire des zones et des macro-actions que l'on considère alors comme des tâches. Cependant, le parallèle avec l'approche MAXQ s'arrête pour l'instant à ce formalisme de tâches puisque celles considérées plus loin sont abordées sous l'angle de tâches chronologiques et utilisées en tant que telles.

2.7 L'approche factorisée

Un approche possible pour simplifier le traitement des MDP est la factorisation des espaces d'état et d'action. La factorisation de l'espace d'action peut être vue de façon similaire à la hiérarchisation vue avec le graphe MAXQ. On peut réduire les modèles probabilistes en ordonnant l'espace d'état sous forme d'arbres : une transition n'étant alors plus représentée par une distribution sur tout l'espace d'état mais par un arbre comportant quelques feuilles et dont les noeuds correspondent aux paramètres cruciaux. On peut également faire dépendre certains états d'autres (comme par exemple factoriser des zones de l'espace d'état par des variables indiquant qu'on est dans telle ou telle phase de la mission). On représente ainsi le problème sous forme de réseaux Bayésiens qui présentent l'avantage d'être nettement plus compacts qu'une énumération complète de l'espace d'état. L'approche factorisée, si elle a été envisagée dans cette étude, n'y a pas une place prépondérante. Cependant, dans la mesure où certains traitements présentés plus loin s'en inspirent et parce qu'elle reste d'actualité pour des travaux futurs (car de nombreux développements ont montré son efficacité), on renvoie le lecteur à [BDG99] et [HSAHB00] pour retrouver certains points qui ont servi pour cette étude.

2.8 Les systèmes multi-agents et l'approche décentralisée

De même que pour l'approche factorisée, les travaux dans le cadre multi-agents dont s'est inspirée cette étude sont présentés dans [BM05], [Mou04], [GKP01], [Bou96], [GMM03], et en particulier dans [BM04]. La dernière référence est particulièrement importante pour l'introduction d'un modèle riche traitant le temps du point de vue des durées et pour le système de coopération qui y est présenté. On y retrouvera certains points similaires avec le modèle discret développé plus loin. Les présenter ici n'aurait que valeur de recopie puisque la réflexion et la discussion sur ces modèles a débouché sur la contribution présentée plus loin. Ils doivent pourtant être mentionnés comme des bases sur lesquels se sont construits les algorithmes développés en section 3.

Chapitre 3

Le problème multi-agents considéré

3.1 Présentation de la problématique et ébauche de solution

Le cadre du problème dans lequel on se place concerne l'interaction de deux agents. En l'occurence, il s'agit de deux drones hélicoptères ou d'un drone hélicoptère et d'un robot mobile progressant au sol. La mission globale du duo d'agents est une mission d'exploration en terrain inconnu. Afin de parvenir à une modélisation adaptée du problème et à une solution réalisable, on part de problèmes voisins de même type.

Chaque agent doit être globalement indépendant au sens où il n'attend pas ses ordres d'une entité centrale qui recueillerait les informations de tous les agents pour résoudre le problème et leur communiquer la solution optimale. La recherche de la stratégie optimale se fait donc de façon décentralisée et, plus précisément, distribuée sur l'ensemble des agents. Par ailleurs, les aléas de la mission peuvent conduire à une possible replanification, qu'elle soit dûe à la communication des agents entre eux ou à un évènement extérieur ; la solution développée est donc orientée vers des algorithmes suffisamment légers pour pouvoir être utilisés en ligne.

Dans un premier temps, on a cherché à modéliser le problème de la prise de décision des robots et de la recherche de la politique optimale dans un cadre où les deux agents sont identiques, c'est-à-dire qu'ils bénéficient des mêmes espaces d'action. Néanmoins, le problème avec deux agents différents est en tous points similaire car la méthode de résolution ne change pas comme on le verra plus loin.

L'approche adoptée consiste à considérer que chaque robot dispose de son propre MDP. Cette approche est motivée par l'idée que les agents sont indépendants et ont, d'une part, une vision de l'environnement et des récompenses qui leur est propre (en particulier si les agents sont de nature différente), d'autre part une information différente liée à l'approche décentralisée. Par ailleurs, la recherche d'une stratégie optimale de façon distribuée implique une communication et une prise en compte d'espaces d'état et d'action augmentés comprenant les états et action conjoints des deux agents. La taille des espaces d'état et d'action croit alors exponentiellement avec le nombre d'agents. L'énumération de l'espace d'état est alors très longue. On cherche à éviter une telle lourdeur pénalisante pour la mission en développant des algorithmes adaptés et en conservant un MDP individuel par agent.

Sur son MDP individuel, chaque agent effectue une décomposition automatique de l'espace d'état en régions découplées. Il calcule également les politiques optimales associées à chaque région et aux transitions entre régions. Il définit enfin pour chaque politique optimale une macro-

action qui est une action élémentaire d'un MDP abstrait où chaque région est un macro-état et où les macro-actions représentent les transitions entre macro-états. Pour chaque macro-action il calcule alors la distribution de problabilité sur la durée de la macro-action grace aux distributions de probabilité sur les actions élémentaires de chaque macro-action. Il définit ainsi des tâches (des couples macro-action - durée). On suppose que chaque agent connaît les capacités de l'autre agent, c'est à dire que les ensembles de macro-actions sont connus. Cette hypothèse se justifie facilement par une brève communication entre les agents au début de la mission. Par ailleurs on verra que cette hypothèse n'est pas toujours nécessaire. L'intérêt de la décomposition et de la constitution des tâches, au delà de la simplification des problèmes à résoudre, est de disposer de la possibilité de communiquer en utilisant les tâches plutôt que les actions élémentaires.

Afin de parvenir à la stratégie conjointe optimale, l'on se heurte à une limitation dimensionnante : puisqu'on a supposé qu'il n'y avait pas d'entité omnisciente (en particulier sur les états des agents) il y a une nécessaire communication entre agents pour transformer des politiques individuelles optimales compte tenu des MDP individuels, en politiques conjointes optimales ou pseudo-optimales dans un MDP augmenté que l'on souhaite ne pas avoir à considérer (et formuler explicitement) à cause de sa taille. L'élément clé de ce dilemme est donc le protocole de communication et la manière qu'ont les agents de s'échanger de l'information et de construire leurs plan. Etant donné le coût (en temps et en récompenses) de toute communication, on cherche à les minimiser au maximum. On cherche donc un procédé de communication qui permette aux agents d'améliorer respectivement leurs politiques tout en limitant au maximum l'information échangée.

L'approche proposée pour la communication repose sur les idées suivantes :
- L'agent n'a pas besoin de connaître en détail l'état de son homologue pour améliorer la politique courante,
- Pour pouvoir améliorer la politique et parvenir à une coopération ou une coordination, l'agent a besoin de communiquer à son homologue la disponibilité des récompenses et des rendez-vous.

Ces deux conditions posent des bornes supérieures et inférieures à la quantité d'information à échanger et précisent sa nature. L'idée du processus de communication développé est similaire à un processus d'enchères. Chaque agent résout son MDP local afin de trouver la politique optimale pour lui seul. Puis il émet une chronologie de ce qu'il compte faire connaissant sa politique optimale. Cette chronologie comprend des tâches, des états initiaux et des dates de fin de tâche. A partir de cette chronologie, le second agent met à jour son MDP afin de tenir compte de la possible prise des récompenses par le premier agent, il y ajoute également les récompenses associées aux rendez-vous et aux actions conjointes et il résout enfin son MDP mis à jour. A partir de sa nouvelle politique optimale, il communique sa chronologie au premier agent qui procède de la même manière. L'échange se poursuit jusqu'à convergence vers une pseudo-optimalité. A chaque itération de ce processus, chaque agent ne fait qu'améliorer sa politique courante et la fonction de valeur associée, on ne peut donc perdre une bonne politique dans l'échange.

Au final, ce protocole semble utilisable car il permet aux agents d'« argumenter » et d'améliorer la politique courante conjointe (qui n'est jamais explicitement formulée) tout en limitant au maximum la quantité d'information émise (cette dernière se limite à une nombre de messages de type $(\textit{tâche}, \textit{état}, \textit{date})$ négligeable devant la taille de l'espace d'état ou de l'espace d'action - ces tailles étant les tailles caractéristiques d'un messages impliquant l'émission de la politique complète). On peut noter que d'autres formes équivalentes de l'information échangée peuvent

être encore plus compactes ; on pourrait notamment envisager un message du type (« numéro de récompense », « date »). Ce type de message supprime la donnée de la tâche effectuée dans le message mais présuppose une numérotation commune des récompenses. L'intérêt est multiple : d'une part les agents n'ont plus besoin de connaître respectivement les ensembles de tâches, d'autre part le second agent n'a que la mise à jour de son MDP à faire à partir des récompenses qu'envoie le premier ; en effet, pour une récompense connue, la date correspond à la date de fin de disponibilité, pour une récompense nouvelle (comme par exemple l'apparition d'une récompense de rendez-vous), la date correspond à la date de début de disponibilité. On le voit assez simplement, il existe de nombreux modes de communication possibles selon que les agents disposent d'un language commun plus ou moins développé ; l'élément essentiel réside toutefois dans l'émission d'un message contenant le minimum d'information nécessaire à une amélioration de la politique de l'autre agent dans le cadre de la coopération.

Afin d'évaluer la validité du processus bi-agents ainsi développé, on cherche à être aussi exhaustif et précis que possible dans les situations d'évaluation. On considère donc, en partant des modèles qui semblent les plus simples et en allant vers les plus complexes, les cas de test suivants. Ces cas de test ne sont pas simplement des illustrations des problèmes traitables par notre approche, ils recouvrent des types de problèmes plus généraux dans le cadre de la coopération bi-agents (extensibles à la coopération multi-agents).

- Les problèmes à une unique récompense, illustré par le problème des deux taxis. Le problème des deux taxis est une extension du problème du taxi vu précédemment et détaillé dans [Die98]. L'idée est de démontrer que dans un problème sovable par un agent seul, la paire d'agents peut faire au moins aussi bien. De façon plus générale, cet exemple montre comment l'algorithme procède à l'échelle d'une unique récompense, en particulier en ce qui concerne le processus de « négociation » entre les deux agents.
- Les problèmes à plusieurs récompenses sans coopération, illustré par le problème de la coopérative agricole. Cet exemple met en jeu deux agents devant collecter de nombreuses récompenses (des fruits) et les apporter en un lieu de collecte. Chaque récompense ne peut être perçue qu'une fois. Cette classe de problèmes recouvre tous les problèmes où les actions des agents ne sont pas couplées. Il n'y a donc pas de récompense de rendez-vous. L'aspect mis en évidence ici est plus focalisé sur l'optimisation de la coopération pour éviter que les agents ne visent la même récompense et pour que le gain global soit maximal.
- Les problèmes à une unique récompense avec coopération et coordination nécessaires. Parmi ceux-ci on distingue deux sous-classes de problèmes :
  - Les problèmes du type « terrains différents », illustrés par le problème de l'île au trésor. Ce type de problème met en jeu deux robots aux capacités différentes et nécessairement complémentaires. Ici en particulier, on a un robot flottant et un robot roulant. Les deux robots partent du port et doivent aller chercher une récompense sur une île. Le robot flottant peut transporter le robot roulant et seul ce dernier peut se déplacer sur l'île. Les aspects mis en valeur par cette classe de problèmes couvrent le spectre des problèmes de rendez-vous.
  - Les problèmes de rendez-vous généralisé de type « labyrinthe à portes ». Dans cet exemple, les deux robots se trouvent dans un labyrinthe ayant des portes fermées et dont les mécanismes d'ouvertures se trouvent trop loin de la porte pour qu'un robot seul puisse ouvrir la porte et la franchir avant qu'elle ne se referme. Il s'agit d'un problème de rendez-vous plus complexe dans sa formulation que précédemment qui implique une coordination et une coopération plus contraignante.
- Les problèmes à plusieurs récompenses avec ou sans coopération, ce qui, au final, représente

le cas général.

Dans la partie suivante, on aborde plus en détail le problème des deux taxis.

3.2 Détail d'un cas de test : le problème des deux taxis

3.2.1 Présentation du problème à deux agents

Le problème du taxi dans le cadre mono-agent tel que présenté par Diettrich dans [Die98] a été introduit à la section 2.5. On a un peu simplifié le problème afin de se recentrer sur l'essentiel de la coopération multi-agents et de rester dans le cadre du problème à une unique récompense. On y revient plus en détail afin de le détailler dans le cadre à deux agents.

Dans le cas mono-agent, le problème du taxi se définit de la manière suivante. Un agent — le taxi — habite un monde-grille de taille 5x5. Dans ce monde-grille, il y a une station d'essence (case F) et quatre points de passage notés B(leu), V(ert), J(aune) et R(ouge). Il y a également un certain nombre de murs entre certaines cases. Un passager apparaît aléatoirement sur un des points de passage et désire être amené à un autre point de passage (éventuellement son point de départ) également déterminé aléatoirement. Le taxi démarre dans une case aléatoire et avec une quantité de carburant aléatoire (entre 5 et 12 unités). Il doit se rendre au lieu ou le passager attend, le faire monter, l'emmener à sa destination et le faire descendre.

On s'intéresse à présent au cas où deux taxis sont en présence. Dans un premier temps, afin de simplifier le problème, on modifie l'énoncé afin de considérer un problème plus orienté « navettes » : il y a un unique point de départ et un unique point d'arrivée. Les taxis démarrent toujours dans des cases choisies aléatoirement et avec des quantités de carburant aléatoires (on contraint cependant la quantité de carburant initiale afin de permettre aux taxis d'aller refaire le plein).

Les agents sont capables d'effectuer les actions suivantes :

- Nord
- Sud
- Est
- Ouest
- Embarquer
- Débarquer
- Plein
- No-Op

L'état d'un agent est caractérisé par les variables suivantes :

- Position : les coordonnées du taxi qu'on considère totalement observables.
- Pass_in : Y a-t-il un passager dans le taxi ?
- Fuel : Carburant restant
- Pass_att : Y a-t-il un passager en attente sur le trottoir ?

Les variables Pass_in et Pass_att semblent redondantes, cependant on choisit de les conserver

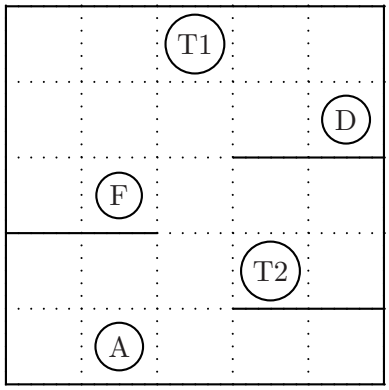

FIG. 3.1 – Un exemple de grille pour le problème du taxi

par souci de clarté. Par ailleurs, dès que l'on envisage la possibilité de plusieurs récompenses, ou de plus de deux agents, la redondance disparait.

Les récompenses sont définies par le coût réel (financier) que le chauffeur doit payer ou recevoir pour ses différentes actions :

– Prix de la course : Avec A la case d'arrivée on a :

$$r(\mathsf{Position} = \mathrm{A}, \mathsf{Débarquer}) = +40$$

– Dépannage en cas de panne sèche :

$$r(\mathsf{Fuel} = 0) = -60$$

– Collision avec un mur, réparation de la carosserie et de la mécanique :

$$r(\mathsf{Position} =< \text{A côté du mur} >, < \text{Mouvement vers le mur} >) = -300$$

– Faire le plein :

$$r(\mathsf{Plein}) = -20$$

Afin de simplifier les notations, on introduit une variable artificielle « Crash » qui traduit le fait que dans la position courante, l'action de déplacement entreprise vise un mur. Par ailleurs on note P la position courante, Pv la position correspondant au résultat « naturel » de l'action de déplacement et Pa une des trois autres positions accessibles (on considère que les seules positions accessibles sont les 4 positions en croix autour de la position courante). La position de départ du passager est notée D, celle d'arrivée est notée A, les positions initiales des taxis sont notées T1 et T2 celle de la station d'essence est notée F. Un exemple de situation de départ pour un problème à deux taxis est représenté figure 3.1.

Les probabilités de transition entre états sont représentées sur les figures 3.2 à 3.5. La flèche de gauche partant d'une case indique que la valeur de la case est vraie. La flèche de droite indique que la valeur est fausse. Chaque noeud représente un jeu de variables d'état. Les feuilles mentionnent la modification de l'état initial ainsi que la probabilité associée à cette modification (implicitement, les variables non mentionnées ne changent pas). L'action No-Op ne change rien à l'état courant. Afin de simplifier le traitement du problème chaque action a une durée unitaire.

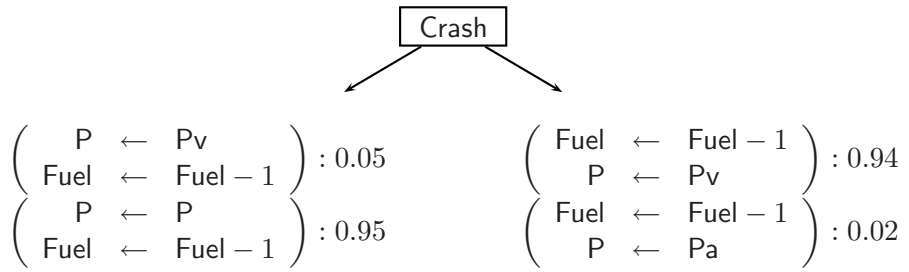

FIG. 3.2 – Probabilités de transition pour les actions de déplacement — Variables concernées : P, Fuel.

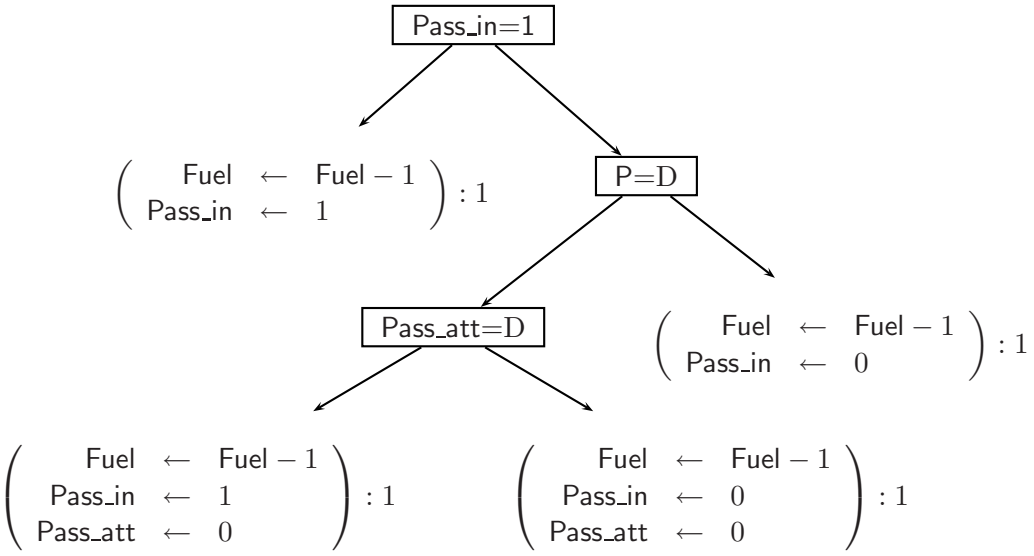

FIG. 3.3 – Probabilités de transition pour l'action Embarquer — Variables concernées : Pass_in, Pass_att, P, Fuel

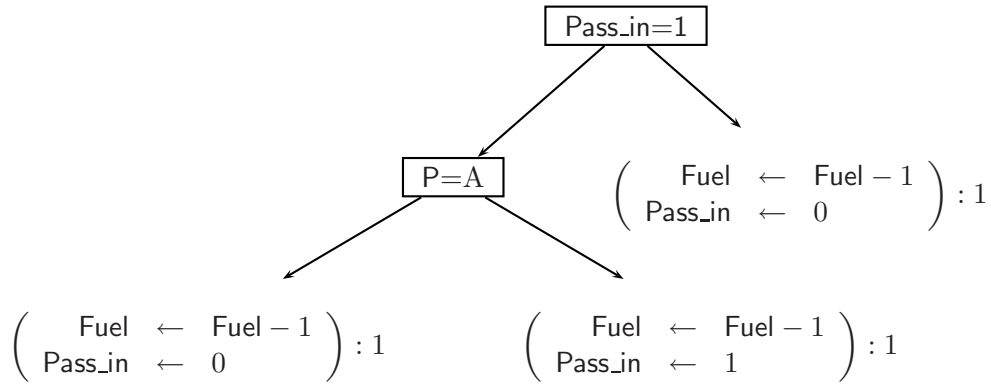

FIG. 3.4 – Probabilités de transition pour l'action Débarquer — Variables concernées : Pass_in, P, Fuel

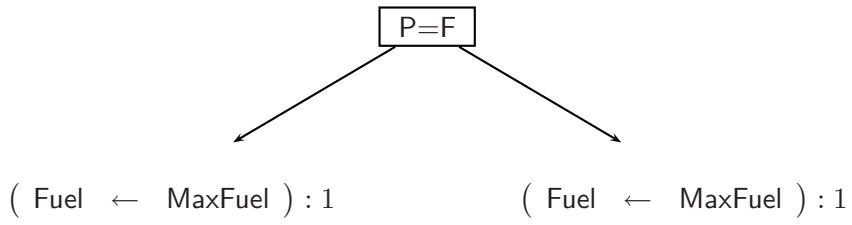

FIG. 3.5 – Probabilités de transition pour l'action Plein — Variables concernées : P, Fuel

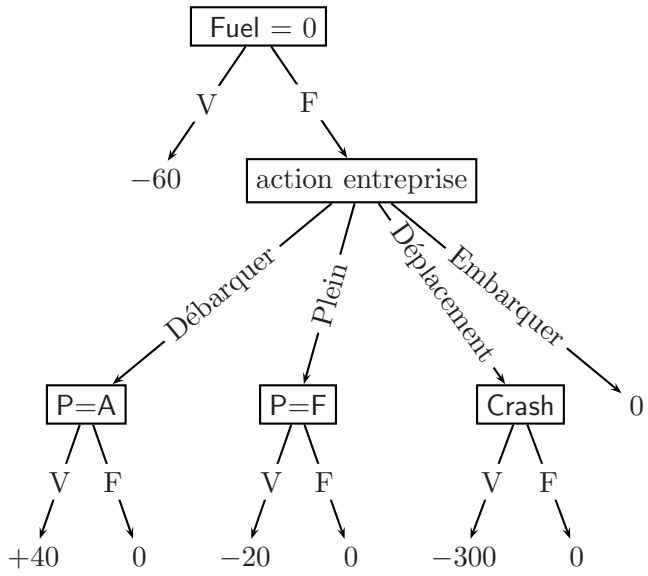

FIG. 3.6 – Représentation compacte des récompenses pour le problème du taxi

L'arbre des récompenses est représenté de façon compacte à la figure 3.6.

L'univers étant défini comme précédemment on y introduit deux agents, les deux taxis. L'objectif est de mettre en place un algorithme permettant de trouver une politique optimale ou pseudo-optimale pour le binôme d'agents. On s'impose par ailleurs comme contrainte qualitative la limitation au maximum des communications entre agents. Cette contrainte n'est pas chiffrée. La section suivante présente la modélisation de l'univers sous forme de MDP adaptée au problème des deux taxis et plus généralement aux problèmes similaires à deux agents.

3.2.2 Modèle de l'univers

De la même manière que cela a été introduit dans le cas particulier de la formulation du problème des deux taxis, chaque agent dispose d'une représentation de l'univers sous forme MDP.

Dans le cas général, on a donc, pour chaque agent :

- Un espace d'état S comprenant, entre autres :
  - des variables géographiques (la position du robot P),
  - des variables d'état du robot (la réserve de carburant Fuel),
  - des variables d'« historique » (la variable Pass_in),
  - des variables sur l'état du problème et du monde extérieur (Pass_att)
- Un espace d'action A propre à chaque agent contenant la liste exhaustive des actions élémentaires dont est capable l'agent (dans l'exemple des deux taxis, l'espace d'action contient les huit actions élémentaires décrites plus haut),
- des probabilités de transition $P(j|s, a)$,
- des récompenses et des coûts associées aux états/actions $R(j, a, s)$.

Pour le problème du taxi, toutes ces variables ont été définies à la section précédente.

L'introduction du temps sous forme SMDP se fait en parallèle de la décomposition de l'espace d'état de chaque agent. On ajoute l'aspect de durée en associant à chaque action élémentaire une distribution de probabilité sur la durée de l'action $F(t|s, a)$. On propage alors ces distributions de probabilité dans les macro-actions en calculant la probabilité sur la durée d'une macro-action en fonction des probabilités sur la durée des actions élémentaires mises en jeu.

Une fois le MDP décomposé et les politiques optimales explicitées région par région, l'agent définit les macro-actions et leur associe une probabilité de durée, créant ainsi des tâches. Chaque tâche est en fait un MDP réduit sur lequel on a calculé une politique optimale, on peut ici établir explicitement le parallèle avec un Max node de la représentation MAXQ.

3.3 Une solution considérant le temps comme variable continue

L'intérêt de cette section est de présenter un modèle qui pousse au plus loin possible l'introduction du temps comme une variable continue au sein du MDP. On verra ainsi que dès que l'on considère un système multi-agents, on se ramène à un cas de MDP proche d'un MDP instationnaire et que l'on peut résoudre ce type de problèmes en introduisant la notion de date qui vient compléter celle de durée. On abordera ensuite les algorithmes possibles de résolution en observant leur faisabilité en terme d'implémentation et d'utilisation. On cherchera enfin à le replacer dans le cadre de notre algorithme de résolution distribué pour la recherche d'une

politique conjointe optimale.

3.3.1 Les dates, conséquences des durées

Considérons deux agents pour lesquels le modèle d'univers est un SMDP. Ces agents représentent leurs actions comme affectés d'une distribution sur leur durée. Afin de simplifier le raisonnement, plaçons nous dans une optique où ces durées sont déterministes : à chaque action correspond une durée fixée. On peut effectuer, pour un agent en particulier, la recherche de la politique optimale, comme on l'a présenté plus haut à la section 2.3. Supposons à présent qu'un second agent arrive dans cette situation et cherche à améliorer la politique du premier en cherchant à prendre les mêmes récompenses mais en dépensant moins de temps (moins de carburant). Si l'on souhaite que le second agent ne résolve que son MDP propre, il faut introduire une variable de disponibilité de la récompense. Cette variable dépend du temps puisque le problème courant revient à chercher s'il est possible de prendre la récompense avant qu'elle ne « disparaisse », prise par l'autre agent. Cet exemple simple met l'accent sur le lien direct qu'il y a entre durées et dates dans le cadre de la coordination entre deux agents. Les durées induites par le premier SMDP se sont retrouvées sous forme d'échéances dans le second.

Par ailleurs, considérons que les actions du premier agent sont toujours affectées d'une durée fixe et cherchons à résoudre son SMDP dans lequel on introduit une échéance (une date de fin de disponibilité d'une récompense). La résolution de l'équation de Bellman implique la propagation dans tout l'espace d'état de la variable de date. Illustrons ce fait par un exemple simple : notre premier agent est seul dans son univers et la seule récompense disponible sera périmée (donc disparaitra) dans t unités de temps. On pourrait imaginer décomposer ce problème en deux MDPs séparés, l'un à horizon fini jusqu'à la date de péremption, l'autre à horizon infini correspondant à la situation postérieure. De la même manière, on peut imaginer *a fortiori* décomposer tout problème à plusieurs récompenses ayant des échéances en autant de petits MDP à horizon fini que l'on résoudrait individuellement. Cependant, il y a un obstacle qui empêche une telle décomposition : en fonction de l'état de départ, la durée totale des actions pour rejoindre la récompense varie. Il suffit pour s'en convaincre de considérer simplement des états de départ plus ou moins éloignés de la récompense, ou des suites d'actions très différentes qui induisent des durées différentes. Ainsi, on ne peut pas définir de date générale à partir de laquelle une récompense n'est plus disponible. De plus, si on considère le cas simple de deux récompenses ayant des dates de « péremption » ou de disponibilité différentes et que l'on adopte une approche par MDPs à horizon fini séparés, alors on est sûrs de perdre en optimalité. En effet, pendant les derniers instants d'un MDP, alors que la récompense est trop éloignée pour pouvoir être prise, l'action optimale calculée est de ne rien faire alors que l'action optimale réelle est de se mettre en quête de la seconde récompense.

Ces quelques considérations illustrent le fait que l'introduction de durées dans un MDP à deux agents implique l'utilisation de dates. Ces dates correspondent aux dates de décision introduites avec les SMDP mais ici, les durées des actions influent sur la disponibilité des récompenses et la notion de durée ne suffit plus : il faut introduire des dates. Par ailleurs, le fait que les durées soient différentes implique de considérer — sur un horizon infini — des dates de décision infiniment proches ce qui revient à considérer un axe du temps continu.

L'enrichissement du modèle SMDP avec des dates permet, par ailleurs, de considérer des récompenses qui dépendent du temps (de la date courante) et donc de mettre en oeuvre des pro-

cessus décisionnels plus riches. Cela permet en fait de considérer des processus décisionnels qui se rapprochent d'un cadre instationnaire puisque leurs récompenses peuvent dépendre du temps.

Le fait de rajouter la date comme variable revient au final à considérer un espace d'état augmenté d'une variable continue ayant une comportement un peu particulier (elle est uniquement croissante) : le temps, sous forme de dates et non plus de durées.

On cherche donc, à la section suivante, à construire un modèle basé sur les SMDP qui prennent en compte un temps continu.

3.3.2 Un modèle de SMDP avec dates

Nous laissons de côté la préoccupation du multi-agents pour un temps afin de considérer le problème de l'introduction d'un temps continu et absolu (c'est-à-dire qui a une origine, donc qui ne se définit plus en relatif avec des durées mais en absolu avec des dates). On construit ainsi le modèle suivant :

- Un état augmenté σ qui se décompose en :
  - Un espace d'état discret S
  - Un axe du temps $t \in \mathbb{R}$
- Un espace d'action discret A
- Une fonction de transition $P(\sigma'|\sigma, a)$ décomposable en $P(\sigma'|\sigma, a) = P(s'|\sigma, a)F(t'|\sigma, a)$
- Une fonction de récompense $r(\sigma, a, \sigma')$

Les politiques sont à présent définies sur Σ, l'ensemble des σ.

On définit la fonction de valeur avec un critère γ-pondéré de façon récursive :

$$V^\pi(\sigma) = \sum_{s' \in S} \int_t^\infty \Big(r(\sigma, \pi(\sigma), \sigma') + \gamma^{t'-t} V^\pi(\sigma') \Big) F(t'|\sigma, \pi(\sigma)) P(s'|\sigma, \pi(\sigma)) dt'$$

L'algorithme d'itération de la valeur se base alors sur la suite des $V_n(\sigma)$:

$$V_0(\sigma) = 0$$
$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_{s' \in S} \int_t^\infty \Big(r(\sigma, a, \sigma') + \gamma^{t'-t} V_n(\sigma') \Big) F(t'|\sigma, a) P(s'|\sigma, a) dt' \right\}$$

Ce modèle est la généralisation de la notion de durée à un axe des temps ayant une origine. On peut noter dès maintenant que d'un point de vue calculatoire, les difficultés vont se poser au niveau du calcul de l'intégrale et de la manière d'effectuer le calcul de cette dernière sans que la taille de l'espace d'état n'explose. Une solution envisagée par la suite consiste à séparer l'espace des variables continues et discrètes et de considérer une expression littérale pour chaque expression de variables continues. Comme on le verra, cette expression littérale peut être très simple à calculer alors que sa discrétisation sur un grand nombre de points risque de rendre le calcul infaisable. D'autres options sont également envisagées dans la suite de cette étude.

3.3.3 Le cas particulier des récompenses linéaires par morceaux - un exemple

Afin de simplifier la présentation on considérera une unique récompense. On considère cette dernière comme prenable en un état donné, constante jusqu'à un instant t_α et nulle ensuite, la fonction de récompense s'écrit :

$$\begin{aligned} r(\sigma', a, \sigma) = r(\sigma') &= \begin{cases} 0 & \text{si } s' \neq s_\alpha \\ r_\alpha \times r_\alpha(t') & \text{si } s' = s_\alpha \end{cases} \\ r_\alpha(t') &= \begin{cases} 1 & \text{pour } t' \leq t_\alpha \\ 0 & \text{pour } t' > t_\alpha \end{cases} \end{aligned}$$

En posant :

$$r_\alpha(s') = \begin{cases} 0 & \text{si } s' \neq s_\alpha \\ 1 & \text{si } s' = s_\alpha \end{cases}$$

On a $r(\sigma', a, \sigma) = r_\alpha \times r_\alpha(s') \times r_\alpha(t')$.

Les probabilités de transition de toutes les actions de notre exemple sont des probabilités portant sur les états, les actions et les durées. On notera $\tau = t' - t$. Toujours pour des raisons de simplicité de l'exposé, on considérera des actions déterministes sur les états et des transitions équiprobables dans un intervalle de temps donné $I_{s,a}$ (de longueur $l(I_{s,a})$) :

$$\begin{aligned} P(s'|\sigma, a) &= \begin{cases} 1 & \text{pour l'état visé} \\ 0 & \text{pour les autres} \end{cases} \\ F(t'|\sigma, a) &= F(\tau|s, a) = \begin{cases} \dfrac{1}{l(I_{s,a})} & \text{si } \tau \in I_{s,a} \\ 0 & \text{pour les autres durées} \end{cases} \end{aligned}$$

Le problème ainsi formulé peut être une modélisation du problème des deux taxis. En effet, il permet de prendre en compte la « disparition » de la récompense une fois que l'autre agent l'a prise. Les actions étant des actions de déplacement, le problème est encore simplifiable car tous les $I_{s,a}$ peuvent dans un premier temps être considérés comme identiques.

On peut expliciter les premières étapes d'une itération de la valeur :
On initialise $V_0(\sigma) = 0, \quad \forall \sigma \in \Sigma$.

$$
\begin{aligned}
V_1(\sigma) &= \max_{a\in A}\left\{\sum_{s'\in S}\int_t^{\infty}\Big(r(\sigma,a,\sigma')+\gamma^{t'-t}V_0(\sigma')\Big)\,F(t'|\sigma,a)P(s'|\sigma,a)dt'\right\}\\
&= \max_{a\in A}\left\{\sum_{s'\in S}\int_t^{\infty}\big(r(s',t')\big)\,F(t'|\sigma,a)P(s'|\sigma,a)dt'\right\}\\
&= \max_{a\in A}\left\{\sum_{s'\in S}r_\alpha\cdot r_\alpha(s')\int_t^{\infty}\big(r_\alpha(t')\big)\,F(t'|\sigma,a)dt'P(s'|\sigma,a)\right\}\\
&= \max_{a\in A}\left\{\sum_{s'\in S}r_\alpha\cdot r_\alpha(s')\cdot P(s'|\sigma,a)\int_{I_{s,a}}\left(\frac{1}{l(I_{s,a})}r_\alpha(t+\tau)\right)d\tau\right\}\\
&= \max_{a\in A}\left\{r_\alpha\cdot P(s_\alpha|\sigma,a)\int_{I_{s,a}}\left(\frac{1}{l(I_{s,a})}r_\alpha(t+\tau)\right)d\tau\right\}
\end{aligned}
$$

L'intégrale $V_1'(s,a,t)=\int_{I_{s,a}}\left(\frac{1}{l(I_{s,a})}r_\alpha(t+\tau)\right)d\tau$ est l'intégrale par rapport à τ d'une fonction linéaire par morceaux par rapport à t et τ, c'est donc une fonction linéaire par morceaux par rapport à t. Ici, elle s'écrit :

on pose :
$I^+_{s,a}=\max_{t\in I_{s,a}}\{t\}$
$I^-_{s,a}=\min_{t\in I_{s,a}}\{t\}$

$$
V_1'(s,a,t)=\begin{cases}
\text{si} & & t > t_\alpha - I^-_{s,a} & : & 0\\
\text{si} & t_\alpha - I^-_{s,a}\geq & t > t_\alpha - I^+_{s,a} & : & \dfrac{1}{l(I_{s,a})}\cdot(t_\alpha - I^-_{s,a} - t)\\
\text{si} & t_\alpha - I^+_{s,a}\geq & t & : & \dfrac{1}{l(I_{s,a})}
\end{cases}
$$

Dans le cas du problème des deux taxis, les durées des actions de déplacement peuvent toutes être considérées comme ayant la même distribution de probabilité et donc considérer un intervalle de durée unique : $\forall(a,s)\in A\times S,\quad I=I_{s,a}$. On a alors $V_1'(s,a,t)=V_1'(t)$.

De façon générale on peut écrire :
$$V_1(\sigma) = \max_{a\in A}\{r_\alpha\cdot P(s_\alpha|\sigma,a)V_1'(s,a,t)\}$$

Et dans le cas du problème des deux taxis avec $I=I_{s,a}$, on a même :
$$V_1(\sigma) = \max_{a\in A}\{r_\alpha\cdot P(s_\alpha|\sigma,a)V_1'(t)\}$$
soit :
$$V_1(\sigma) = \max_{a\in A}\{P(s_\alpha|\sigma,a)\}\,V_1'(t)\cdot r_\alpha$$

Dans le cas général, on a stocké $|S|\times|A|$ fonctions linéaires pour pouvoir faire une itération de Bellman et au final on a une fonction de valeur définie de façon exacte. Cette fonction est définie linéaire par morceaux puisque à s donné, elle est égale à une succession de morceaux

proportionels aux $V_1'(s,a,t)$, ces morceaux étant déterminés en fonction des actions a optimales de cette itération. Afin de représenter simplement la fonction $V_1(\sigma)$ on introduit les fonctions $V_{1s}(t)$. Pour un s donné, $V_{1s}(t)$ est la fonction définie linéaire par morceaux égale à $V_1(s,t)$. On peut ainsi enchaîner sur la seconde itération de Bellman en écrivant de façon symbolique que :

$$V_1(\sigma) = V_{1s}(t)$$

Pour cette itération, on a :

$$\begin{aligned} V_2(\sigma) &= \max_{a\in A}\left\{\sum_{s'\in S}\int_t^\infty \left(r(\sigma,a,\sigma') + \gamma^{t'-t}V_1(\sigma')\right)F(t'|\sigma,a)P(s'|\sigma,a)dt'\right\} \\ &= \max_{a\in A}\left\{\sum_{s'\in S}\int_t^\infty \left(r_\alpha\cdot r_\alpha(s')\cdot r_\alpha(t') + \gamma^{t'-t}V_{1s'}(t')\right)F(t'|\sigma,a)P(s'|\sigma,a)dt'\right\} \end{aligned}$$

Ici, on peut séparer le terme intégral en deux et l'écrire comme :

$$\begin{aligned} V_2'(s',s,a,t) &= \int_t^\infty \left(r_\alpha\cdot r_\alpha(s')\cdot r_\alpha(t') + \gamma^{t'-t}V_{1s'}(t')\right)F(t'|\sigma,a)dt' \\ &= J_1 + J_2 \end{aligned}$$

avec :

$$\begin{aligned} J_1 &= \int_t^\infty \left(r_\alpha\cdot r_\alpha(s')\cdot r_\alpha(t')\right)F(t'|\sigma,a)dt' \\ J_2 &= \int_t^\infty \left(\gamma^{t'-t}V_{1s'}(t')\right)F(t'|\sigma,a)dt' \end{aligned}$$

on a :

$$J_1 = r_\alpha\cdot r_\alpha(s')\cdot\int_t^\infty \left(r_\alpha(t')\right)F(t'|s,a)dt'$$

donc :

$$J_1 = r_\alpha\cdot r_\alpha(s')\cdot\int_{I_{s,a}} \left(r_\alpha(t+\tau)\right)F(\tau|\sigma,a)d\tau$$

et :

$$J_1 = r_\alpha\cdot r_\alpha(s')\cdot V_1'(s,a,t)$$

Par ailleurs :

$$
\begin{aligned}
J_2 &= \int_t^{\infty} \left(\gamma^{t'-t} V_{1s'}(t')\right) F(t'|\sigma, a) dt' \\
&= \int_{I_{s,a}} \left(\gamma^{\tau} V_{1s'}(t+\tau)\right) F(\tau|s, a) d\tau
\end{aligned}
$$

$V_{1s'}$ est définie par morceaux donc l'intégrale ci-dessus se calcule également par morceaux. Il s'agit donc d'intégrales séparées de fonctions de type $\gamma^x \times (ax+b)$. On note ici que :
$\int_t [\gamma^x \times (ax+b)]\, dx = \dfrac{\gamma^t}{\ln(\gamma)} \left[at + b - \dfrac{a}{\ln \gamma}\right]$

Comme F est une distribution uniforme, on peut poser :
$J_2 = V'_{2s'}(s, a, t)$

On sait calculer simplement la fonction $V'_{2s'}(s, a, t)$ grâce à l'expression de la primitive précédente. On a donc :
$V'_2(s', s, a, t) = r_\alpha \cdot r_\alpha(s') \cdot V'_1(s, a, t) + V'_{2s'}(s, a, t)$

Et donc :

$$
\begin{aligned}
V_2(\sigma) &= \max_{a \in A} \left\{ \sum_{s' \in S} \left[r_\alpha \cdot r_\alpha(s') \cdot V'_1(s, a, t) + V'_{2s'}(s, a, t)\right] P(s'|\sigma, a) \right\} \\
&= \max_{a \in A} \left\{ r_\alpha \cdot V'_1(s_\alpha, a, t) \cdot P(s_\alpha|\sigma, a) + \sum_{s' \in S} V'_{2s'}(s, a, t) \cdot P(s'|\sigma, a) \right\}
\end{aligned}
$$

On peu alors créer, pour chaque s, les intervalles sur lesquels la définition de V_2 est unique (où la politique optimale ne dépend pas de la date). Pour un s donné on construit ainsi des fonctions par morceaux qu'on note $V_{2s}(t)$ comme pour V_1. Chaque morceau d'une fonction $V_{2s}(t)$ est une fonction de type $\gamma^t \times (at+b)$ (par la suite, on notera ce type « exp-aff » pour « exponentielle fois affine »). On obtient ainsi une nouvelle fonction de valeur exacte à l'issue de la seconde itération de Bellman que l'on note symboliquement :

$V_2(\sigma) = V_{2s}(t)$

On donne ici les grandes étapes de la troisième itération de Bellman. On a :

$$
\begin{aligned}
V_3(\sigma) &= \max_{a \in A} \left\{ \sum_{s' \in S} \int_t^{\infty} \left(r(\sigma, a, \sigma') + \gamma^{t'-t} V_2(\sigma')\right) F(t'|\sigma, a) P(s'|\sigma, a) dt' \right\} \\
&= \max_{a \in A} \left\{ \sum_{s' \in S} \int_t^{\infty} \left(r_\alpha \cdot r_\alpha(s') \cdot r_\alpha(t') + \gamma^{t'-t} V_{2s'}(t')\right) F(t'|\sigma, a) P(s'|\sigma, a) dt' \right\}
\end{aligned}
$$

Comme précédemment, on pose :

$$\begin{aligned} V_2'(s',s,a,t) &= \int_t^{\infty} \left(r_\alpha \cdot r_\alpha(s') \cdot r_\alpha(t') + \gamma^{t'-t} V_{2s'}(t') \right) F(t'|\sigma,a) dt' \\ &= J_1 + J_2 \end{aligned}$$

On a toujours :

$$J_1 = r_\alpha \cdot r_\alpha(s') \cdot V_1'(s,a,t)$$

Et :

$$\begin{aligned} J_2 &= \int_t^{\infty} \left(\gamma^{t'-t} V_{2s'}(t') \right) F(t'|\sigma,a) dt' \\ &= \int_{I_{s,a}} \left(\gamma^{\tau} V_{2s'}(t+\tau) \right) F(\tau|s,a) d\tau \end{aligned}$$

La fonction à intégrer est maintenant de la forme $\gamma^{2x} \times (ax+b)$. On a :

$$\int_t \left[\gamma^{2x} \times (ax+b) \right] dx = \frac{\gamma^{2t}}{2\ln(\gamma)} \left[at + b - \frac{a}{2\ln\gamma} \right]$$

Et de façon plus générale, on a :

$$\int_t \left[\gamma^{nx} \times (ax+b) \right] dx = \frac{\gamma^{nt}}{n\ln(\gamma)} \left[at + b - \frac{a}{n\ln\gamma} \right]$$

Le terme J_2 est donc toujours un terme exp-aff par morceaux et on l'écrit :

$$J_2 = V_{3s'}'(s,a,t)$$

On a donc :

$$V_3'(s',s,a,t) = r_\alpha \cdot r_\alpha(s') \cdot V_1'(s,a,t) + V_{3s'}'(s,a,t)$$

Et donc :

$$\begin{aligned} V_3(\sigma) &= \max_{a \in A} \left\{ \sum_{s' \in S} \left[r_\alpha \cdot r_\alpha(s') \cdot V_1'(s,a,t) + V_{3s'}'(s,a,t) \right] P(s'|\sigma,a) \right\} \\ &= \max_{a \in A} \left\{ r_\alpha \cdot V_1'(s_\alpha,a,t) \cdot P(s_\alpha|\sigma,a) + \sum_{s' \in S} V_{3s'}'(s,a,t) \cdot P(s'|\sigma,a) \right\} \end{aligned}$$

On a évité de mentionner comment à chaque étape précédente on trouvait les intervalles pour la constitution des morceaux de la fonction de valeur. Il s'agit d'une recherche de maximum parmi de nombreux segments exp-aff. On considérera cette opération comme une boîte noire pour simplifier l'exposé. Une fois de plus, on obtient une fonction de valeur exacte, définie par morceaux de fonctions exp-aff. A chaque itération, le nombre de fonctions exp-aff que l'on stocke augmente jusqu'à parvenir à approcher d'assez près la fonction de valeur réelle pour que la politique optimale calculée ne change plus et que l'algorithme s'arrête.

On peut noter que le cas des fonctions de récompense linéaires par morceaux est un cas

particulier du modèle qu'on a illustré ici : toute fonction de récompense de type exp-aff permet de mettre en oeuvre un algorithme similaire.

Il est important de noter que la construction du cas particulier des fonctions de récompenses exp-aff repose sur une autre hypothèse sur laquelle on n'a pas insisté et qui est d'une importance capitale. Il s'agit de l'uniformité de la distribution de probabilité sur les durées. En effet, sans cette hypothèse, la résolution de l'intégrale J_2 ne permet pas la simplification du calcul direct de la primitive d'une fonction exp-aff.

Le développement et l'implémentation d'un système utilisant ce modèle continu est laissé pour l'instant dans la catégories des travaux futurs. Une piste d'amélioration du modèle est présentée à la section suivante et l'intégration de ce modèle dans le contexte bi-agents est abordée ensuite.

3.3.4 Une tentative de simplification par séparation des variables - illustration de la complexité du problème

L'idée à la base de cette section est de s'interroger sur la nature de la fonction de valeur d'un système à temps continu et plus particulièrement de trouver sous quelles conditions la fonction de valeur est à variables séparables entre les variables continues et discrètes. L'intérêt d'une telle décomposition repose dans le temps de calcul : en effet, si les variables sont séparables, alors on peut calculer la partie intégrale (ou le jeu de fonctions intégrales) indépendemment des itérations de la valeur sur les états et ainsi on économise un temps de calcul substantiel puisque dans l'absolu, il faudrait recalculer l'intégrale à chaque état.

Les conditions mises en évidence lors de cette étude sont suffisantes mais il n'a pas été démontré qu'elles soient nécessaires, c'est pourquoi cette section s'intitule « une tentative » : il reste certainement des moyens plus simples et plus efficaces de simplifier le problème, notamment par séparation de variables.

L'expérience a montré que plutôt que de chercher l'absolue séparation des variables, c'est-à-dire d'écrire la fonction de valeur $V(s,t) = V(s)V(t)$ il est préférable de chercher une variante que l'on a nommée Σ-séparation des variables : on cherche à considérer des fonctions de valeur $V(s,t) = \sum_i V_i(s)V_i(t)$.

On se replace dans le contexte précédent mais l'on note à présent la fonction de récompense comme :

$$r(\sigma, a, \sigma') = \sum_k r_k(s'|\sigma, a) r_k(t'|\sigma, a)$$

Afin de simplifier l'exposé, on note toujours la probabilité de transition comme à variables séparables, mais l'analyse avec une probabilité de transition à variables Σ-séparables est en tous points similaires :

$$P(\sigma'|\sigma, a) = P(s'|\sigma, a) F(t'|\sigma, a)$$

En fait, pour être rigoureux et strictement séparer variables continues et discrètes, on devrait écrire :

$$P(\sigma'|\sigma, a) = P(s'|s, a)F(t'|t)$$

On va voir qu'on va effectivement être menés à considérer un système à variables séparables (ou Σ-séparables) mais afin d'illustrer le fait que cette condition est nécessaire, on conserve la notation précédente jusqu'à ce que « ça coince » pour mettre en valeur la séparation des variables.

On définit la fonction de valeur avec un critère γ-pondéré de la même façon que précédemment. On cherche à savoir si on peut écrire une fonction de valeurs à valeurs Σ-séparables à partir d'une telle situation de départ et en effectuant des itérations sur l'équation de Bellman.

On a :

$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_{s' \in S} \int_t^\infty \left[\sum_k r_k(s'|\sigma, a) r_k(t'|\sigma, a) + \gamma^{t'-t} V_n(\sigma') \right] \right.$$
$$\left. \cdot F(t'|\sigma, a) P(s'|\sigma, a) dt' \right\}$$

Supposons que la fonction de valeur soit à variables séparables, on a :

$$V_n(s, t) = \sum_i V_{n_i}(s) V_{n_i}(t)$$

Donc :

$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_k \sum_{s' \in S} r_k(s'|\sigma, a) P(s'|\sigma, a) \int_t^\infty r_k(t'|\sigma, a) F(t'|\sigma, a) dt' \right.$$
$$\left. + \sum_i \sum_{s' \in S} V_{n_i}(s') P(s'|\sigma, a) \int_t^\infty \gamma^{(t'-t)} V_{n_i}(t') F(t'|\sigma, a) dt' \right\}$$

On veut éviter de multiplier les intégrales sur une itération de Bellman, il s'agit donc d'éviter toute dépendance de la partie intégrale en s et a, on écrit donc :

$$F(t'|\sigma, a) = F(t', t)$$

Cette écriture peut paraître contraignante mais si l'on fait le lien avec la Σ-séparabilité des variables, on s'aperçoit qu'il s'agit d'une contrainte beaucoup plus souple en écrivant le modèle de transition sous la forme :

$$P(\sigma'|\sigma, a) = \sum_i P_i(s'|\sigma, a) F_i(t'|t)$$

On procède de même pour les r_k :

$$r_k(t'|\sigma, a) = r_k(t', t)$$

On a donc :

$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_k \sum_{s' \in S} r_k(s'|\sigma, a) P(s'|\sigma, a) \int_t^\infty r_k(t', t) F(t', t) dt' \right.$$
$$\left. + \sum_i \sum_{s' \in S} V_{n_i}(s') P(s'|\sigma, a) \int_t^\infty \gamma^{(t'-t)} V_{n_i}(t') F(t'|t) dt' \right\}$$

On aimerait pouvoir écrire V_{n+1} comme une fonction à variables Σ-séparables ce qui permettrait de conclure (par récurrence) que sous les conditions qu'on a énoncées jusqu'ici, la fonction de valeur est à varaibles séparables et son calcul est donc simplifié.

Pour pouvoir écrirce V_{n+1} comme à variables Σ-séparables, il faut pouvoir isoler dans le max des morceaux à variables Σ-séparables. On impose donc les conditions :

$$\begin{aligned} r_k(s'|\sigma, a) &= r_k(s', s, a) \\ P(s'|\sigma, a) &= P(s', s, a) \end{aligned}$$

On retrouve bien le résultat annoncé plus haut comme quoi au final les modèles de récompenses et de transition doivent être à variables Σ-séparables pour valider notre hypothèse. Les mêmes remarques que pour $P(t'|\sigma, a)$ s'appliquent concernant la contrainte sur le modèle de transition $P(s'|\sigma, a) = P(s', s, a)$. On a alors :

$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_k \sum_{s' \in S} r_k(s'|s, a) P(s'|s, a) \int_t^\infty r_k(t', t) F(t', t) dt' \right.$$
$$\left. + \sum_i \sum_{s' \in S} V_{n_i}(s') P(s'|s, a) \int_t^\infty \gamma^{(t'-t)} V_{n_i}(t') F(t'|t) dt' \right\}$$

Nous allons généraliser ce résultat de suite aux cas d'un modèle de transition Σ-séparable (et non juste séparable comme le cas qu'on vient de voir) et nous allons introduire le résultat supplémentaire de la suite des V_i sous la forme du théorème suivant :

Proposition 5. *Soit un SMDP incluant le temps comme variable continue* $< \Sigma, A, P, r >$. *Si les modèles de transition et de récompenses sont à variables Σ-séparables, c'est-à-dire :*

$$P(\sigma'|\sigma, a) = \textstyle\sum_j P_j(s', s, a) \cdot F_j(t', t)$$
$$r(\sigma'|\sigma, a) = \textstyle\sum_k r_k(s', s, a) \cdot r_k(t', t)$$

Alors la fonction de valeur est à variables Σ-séparables à chaque étape de son calcul itératif :

$$V_n(\sigma) = \sum_i V_{n_i}(s) V_{n_i}(t)$$

Et son calcul itératif s'écrit :

$$V_{n+1}(\sigma) = \max_{a \in A} \left\{ \sum_{s' \in S, k, j} r_k(s', s, a) P_j(s', s, a) \int_t^\infty r_k(t', t) F_j(t', t) dt' \right.$$
$$\left. + \sum_{s' \in S, i, j} V_{n_i}(s') P_j(s', s, a) \int_t^\infty \gamma^{(t'-t)} V_{n_i}(t') F_j(t', t) dt' \right\}$$

$V_0(\sigma) = 0$

Cette proposition se démontre en suivant le même cheminement que celui suivi plus haut pour le cas réduit des variables séparables. L'intérêt de cette proposition, outre le fait de fournir une belle formule de deux lignes (!), est d'illustrer la complexité inhérente à un algorithme qui considère le temps comme une variable continue et absolue. On a cherché, dans le cadre qu'on a défini, à expliciter et à simplifier le calcul de la fonction de valeur optimale et ce calcul - ainsi que les restrictions qu'on a du ajouter en chemin - apparait très complexe (et fort peu simplifié par l'introduction de nos contraintes). On peut, pour achever la démarche entreprise, chercher à calculer le nombre d'intégrales que l'on calcule par itération puisque l'objectif initial de la Σ-séparation des variables était justement d'éviter un nombre énorme d'intégrales à calculer.

A la première itération, on calcule $|k||j| = K \cdot J$ intégrales (une par fonction $r_k(t', t)$ et par fonction $F_j(t', t)$). La fonction de valeur est alors définie par morceaux de ces $K \cdot J$ fonctions. A la seconde itération, on retrouve les $K \cdot J$ premières, on ne les recalcule donc pas, par contre, on a le terme en $\int_t^\infty \gamma^{(t'-t)} V_{n_i}(t') F_j(t', t) dt'$ qui implique de recalculer $K \cdot J^2$ intégrales (car il y a $K \cdot J$ expressions différentes des V_i et qu'on les multiplie par les J fonctions F_j). La fonction de valeur est alors définie par morceaux des $K \cdot J$ premières intégrales et des $K \cdot J^2$ secondes. A la troisème itération, on va retrouver les $K \cdot J$ premières, les J intégrales des $K \cdot J$ premières qu'on a déjà calculées à la seconde itération (les $K \cdot J^2$ intégrales qu'on y a calculées) et on va devoir calculer $K \cdot J^3$ nouvelles intégrales. Ainsi de suite, on aboutit au résultat qu'à la $n^{\text{ième}}$ itération, on calcule $K \cdot J^n$ intégrales et que jusque là, on a calculé $\sum_{i=1}^n K \cdot J^i = \frac{KJ(1-J^n)}{1-J}$ intégrales. La fonction de valeur V_n s'écrit alors avec des morceaux de combinaisons linéaires de ces $\frac{KJ(1-J^n)}{1-J}$ fonction intégrales.

Le nombre K peut correspondre entre autres au nombre de récompenses disponibles dans l'univers qu'on considère. Si on prend des problèmes tels que ceux présentés en introduction, K vaut généralement 1, parfois on peut définir plusieurs récompenses, mais leur nombre est toujours faible devant la taille de l'espace d'état. Par contre, le nombre J définit le nombre de *transitions* différentes que l'on peut rencontrer. Pour être plus précis, il définit le nombre de distributions sur les durées des transitions qu'on peut rencontrer. Si on se trouve dans un modèle simple où toutes les actions ont la même durée, alors le problème est très simple à résoudre puisqu'on n'a qu'une intégrale à calculer par itération. La taille du problème que l'on peut traiter limite toutefois très vite le nombre de distributions sur les durées de transition différentes que l'on peut définir ; si l'on pose T le nombre d'intégrales que l'on estime pouvoir résoudre par itération et si l'on estime à N le nombre d'itérations dont on a besoin pour trouver la politique optimale, alors (en approximant $\sum_{i=1}^n K \cdot J^i$ à $K \cdot J^n$) on a au mieux $J = \sqrt[N]{\frac{T}{K}}$.

On peut prendre le problème dans l'autre sens : on sait qu'un majorant du nombre d'itérations effectuées dans une itération de la valeur standard où l'on cherche la valeur optimale à ϵ près est $\frac{\ln[\epsilon(1-\gamma)]}{\ln(\gamma)}$ (pour la démonstration voir l'annexe B). On sait donc qu'on calculera au plus $\frac{KJ(1-J^{\frac{\ln(\epsilon(1-\gamma))}{\ln(\gamma)}})}{1-J}$ intégrales lors de la recherche de la fonction de valeur optimale.

Dans le cas d'un problème de coopération comme celui entre le robot terrestre et le drone aérien, on peut estimer que sur des problèmes simples, avec des récompenses de l'ordre de 1 et un ϵ de 0.001 (un ϵ de 10^{-3} fois l'ordre de grandeur du modèle de récompense est un choix pertinent pour avoir une fonction de valeur relativement proche de l'optimalité), avec une unique récompense et deux comportements temporels différents (si on considère le drone hélicoptère, on peut distinguer un comportement temporel correspondant aux déplacements et un autre

correspondant à l'émission des données), avec, enfin, un facteur d'oubli de 0.9, alors on a :

$$K = 1, \quad J = 2, \quad \epsilon = 10^{-3}, \quad \gamma = 0.9$$
$$\frac{KJ(1-J^{\frac{\ln(\epsilon(1-\gamma))}{\ln(\gamma)}})}{1-J} \approx 4.13 \cdot 10^{26}$$

On doit donc calculer $4.13 \cdot 10^{26}$ fonctions intégrales pour résoudre ce problème. Ce n'est donc pas réalisable dans un temps raisonnable et cela illustre la complexité du modèle présenté. Cette complexité se retrouverait dans n'importe quel algorithme de complexité exponentielle en le nombre de boucles dans un algorithme d'itération de la valeur.

3.3.5 L'intégration du modèle à temps continu dans la recherche d'une politique conjointe optimale

Dans l'optique du traitement des variables continues et du temps dans les problèmes de recherche d'une politique optimale, on peut citer les approches différentes de [FDMW04] et [BDM+02]. Celle présentée ici diffère par le fait qu'elle s'adapte au cadre multi-agents présenté plus haut et qu'elle dispose d'un développement peut-être plus général que les modèles dédiés présentés dans les références sus-cités. En revanche, il faut mentionner que cette approche n'a pas été implémentée et que son efficacité n'a pas été prouvée.

Le problème de la recherche d'un politique optimale dans le cadre bi-agents est abordé plus en détail à la section suivante et l'algorithme de résolution qui a été développé est détaillé à la section 3.4.3. Pour éviter les redites, on renvoie le lecteur au paragraphe 3.4.3 pour le principe de l'algorithme d'itération de la chronologie qui permet de construire des politiques conjointes pour les agents qui coopèrent.

3.4 Résoudre les problèmes d'amélioration d'une politique commune sans introduire de variable temporelle continue

3.4.1 Contexte et modèle

On suppose deux agents 1 et 2, ayant des espaces d'action distincts. Chaque agent dispose d'un modèle propre de leur environnement commun. Ce modèle est sous forme d'un MDP.

Pour 1 par exemple, le MDP en question est constitué de :

- un espace d'état S comportant :
  - des variables d'environnement communes aux deux agents (variables géographiques par exemple),
  - des variables d'état de l'agent propre à 1 (comme le carburant restant),
  - des variables de mission ou d'historique concernant des grandeurs communes aux deux agents (comme la variable de fin de mission ou une variable de disponibilité de récompense),
  - des variables de mission propres à 1 (comme le disponibilité d'une récompense qui concerne 1 seul).
- un espace d'action A constitué de l'espace d'action de 1 uniquement,
- des probilités de transition entre états de S suivant les actions de A,

– des récompenses associées aux triplets $(s, a, s') \in S \times A \times S$.

On note donc que dans sa représentation de l'univers, chaque agent se croit seul. L'idée intuitive plus générale derrière cette modélisation est la suivante : on cherche à générer, pour le couple d'agents, une politique conjointe qui soit optimale (ou quasi-optimale). On pourrait considérer le binôme d'agents comme un seul agent, construire un MDP avec des espaces d'action et d'état conjoints et le résoudre. Cependant, la taille desdits espaces d'action et d'état explose littéralement : en supposant des espaces d'état individuels de taille de l'ordre de N, on a un espace d'état global de taille de l'ordre de N^2 pour deux agents et de N^n pour n agents. De même pour l'espace d'action. Cette première limitation incite fortement à chercher des solutions alternatives. Par ailleurs, une des problématiques abordées par l'approche courante est de savoir s'il est nécessaire de formuler explicitement les espaces d'action et d'état conjoints pour parvenir à une politique optimale. L'idée intuitive qui guide notre approche est que l'on peut calculer pour chaque agent individuellement une politique et que ces deux politiques, lors de leur exécution, correspondent à la politique optimale conjointe. On peut donc parvenir à une politique commune quasi-optimale sans jamais la formuler explicitement. Afin de parvenir à calculer ces politiques individuelles, il s'agit de mettre en place un processus de communication minimal, c'est-à-dire qui envoie un volume d'information réduit à son maximum.

En conservant des MDP individuels aux agents, on limite le temps de calcul nécessaire à la génération des politiques individuelles et on se garde la possibilité d'embarquer les algorithmes de résolution pour un traitement en ligne.

Le modèle présenté dans cette partie est un modèle qu'on a voulu simplifier au maximum afin de pouvoir le tester. On a donc cherché à s'éloigner le moins possible du cadre MDP. On introduit toutefois un paramètre supplémentaire à notre modèle. Comme on l'a vu, la gestion du paramètre temps est cruciale à toute tentative d'amélioration de la politique conjointe, on va ici gérer un temps discret et on introduit pour cela un pas de discrétisation temporelle que l'on note T.

3.4.2 Introduction du facteur temps

Avec ce paramètre temporel, on construit un modèle inspiré du modèle SMDP. A chaque transition, on associe une durée. Cette durée est représentée par une distribution de probabilités sur une variable discrète τ. τ prend ses valeurs dans $T\mathbb{N}$ (les multiples de T). Ainsi, on enrichit le modèle d'une fonction $F(t|s, a)$ décrivant la probabilité que l'action a commencée en s se termine à la date t, et on écrit $Q(t, s'|s, a) = P(s'|s, a) \times F(t|s, a)$.

La définition d'une politique ne change pas et comme dans le modèle SMDP, on définit le coût ou le gain d'une transition comme la somme d'une récompense immédiate et du résultat d'un taux de récompense qui s'applique durant l'exécution. En reprenant les notations du modèle SMDP, on redéfinit la fonction de récompense :

$$r(s, a, s') = k(s, a, s') + \sum_{u=0}^{\infty} \sum_{s' \in S} \left[\sum_{t=0}^{u} \gamma^t c(s', s, a) p(s'|t, s, a) \right] F(u|s, a)$$

On définit alors la valeur d'une politique de façon récurrente comme :

$$V^\pi(s) = \sum_{s' \in S} \sum_{t=0}^{\infty} \left[r(s, \pi(s), s') + \gamma^t V^\pi(s') \right] Q(t, s'|s, \pi(s))$$

ou, en adoptant la notation qui associe à une transition l'espérance des récompenses sur les états d'arrivée :

$$V^\pi(s) = r(s, \pi(s)) + \sum_{s' \in S} \sum_{t=0}^{\infty} \gamma^t V^\pi(s') Q(t, s'|s, \pi(s))$$

On peut définir une notation matricielle en posant :

$$m_\pi(s'|s) = \sum_{t=0}^{\infty} \gamma^t Q(t, s'|s, \pi(s))$$

Avec M_π la matrice des $m_\pi(s'|s)$ (s' étant constant sur les colonnes et s sur les lignes), on peut écrire :

$$V^\pi = r_\pi + M_\pi \cdot V^\pi$$

On est ainsi ramenés au cas de la résolution simple d'un MDP, on peut appliquer les algorithmes éprouvés dont on dispose pour les résoudre.

3.4.3 Algorithme de résolution dans le cadre des deux agents

Dans le cadre de nos deux agents, la construction des plans individuels se déroule de la façon suivante. Par souci de clarté d'exposé, on suivra un seul agent dans les grandes étapes de la résolution avant de formuler plus formellement l'algorithme de résolution.

L'agent 1 commence par résoudre son MDP défini plus haut. Pour cette résolution, on ne rentre pas dans les détails des outils dont l'agent dispose, notamment dans les possibles procédures de décomposition de l'espace d'état, afin de simplifier la résolution.
Une fois la politique optimale obtenue, l'agent construit une chronologie des récompenses communes qu'il prend. L'originalité de notre approche consiste en l'introduction de cette chronologie : de la même manière qu'un plan commun peut se contruire pour deux partenaires au cartes en annonçant des contrats associés à des actions, chaque agent annonce à son partenaire ce qu'il compte faire. On note ici la distinction entre la ligne de conduite d'un agent (sa politique) et ce qu'il compte faire (sa chronologie) : le partenaire n'a aucunement besoin de connaître dans un premier temps la ligne de conduite de l'agent 1, la déclaration d'intentions suffit. De même, il n'est pas nécessaire d'envoyer la politique de l'agent 1 en entier à l'agent 2 : la chronologie suffit. Cette manière de procéder permet de limiter au maximum le volume de données échangées : en effet l'émission d'une politique implique de communiquer les $|S|$ actions associées à chacun des $|S|$ états ; à l'opposé, la chronologie ne comporte qu'un nombre de messages au plus égal au nombre de récompenses. Ainsi, l'agent 1 établit des couples (récompense, date) qui constituent sa chronologie. La date que l'agent 1 émet est la date de plus forte probabilité pour la fin de la suite d'actions menant à la récompense.
L'agent 2 a procédé de même de son côté et émet donc sa chronologie que l'agent 1 reçoit.
C'est ici que le processus de communication prend tout son sens et permet de construire une politique individuelle faisant partie d'une politique conjointe sous-optimale sans jamais formuler cette dernière. En effet, à partir de la chronologie de l'agent 2, l'agent 1 modifie son MDP de la façon suivante :

- Il prend la plus grande date T_f émise par l'agent 2 et crée une variable d'état t prenant ses valeurs dans $[0; T_f] \cap T\mathbb{N}$. La variable t est une variable discrète qui agit comme un compteur : elle compte les pas de temps entre l'instant initial et le dernier instant où les récompenses présentes dans l'univers de 1 changent. Chaque date émise par 2 est un multiple de T. Cette nouvelle variable t représente en fait l'état "la date discrète courante est t", elle indique qu'on se trouve dans l'intervalle de temps $[t_i; t_{i+1}]$. Les dates discrètes sont donc les dates d'évolution des récompenses communes aux agents approximées à T près, la dernière date discrète de t représente l'intervalle $[T_f; +\infty]$.
- Avec cet espace d'état augmenté, l'agent 1 modifie la fonction de récompense de son MDP de la façon suivante : considérons une récompense R_1 associée au triplet (s, a, s') dont l'agent 2 indique dans sa chronologie qu'il la prend à l'échéance t_e. Supposons que la fonction de valeur soit décomposée en somme de fonctions de n récompenses individuelles et que R_1 soit associée à r_1 ; on peut écrire $r(s, a, s') = r_1(s, a, s') + \sum_{i=2}^{n} r_i(s, a, s')$. La fonction de récompense devient :

$$r(s,a,s') = \begin{cases} \text{si } t < t_e \text{ la fonction de récompense ne change pas :} \\ \quad \forall (s,a,s') \in S \times A \times S \quad r(s,a,s') \leftarrow r(s,a,s') \\ \text{si } t \geq t_e \text{ la récompense est automatiquement acquise :} \\ \quad \forall (s,a,s') \in S \times A \times S \quad r(s,a,s') \leftarrow \sum_{i=2}^{n} r_i(s,a,s') + R_1 \end{cases}$$

  Cette représentation permet de rendre compte du comportement « la récompense va être prise dans tous les cas, mais si elle peut être prise à moindres frais, alors il est intéressant d'aller la prendre ».
- Ensuite, l'agent 1 modifie sa fonction de transition afin de tenir compte de la nouvelle variable t. Cette opération se fait comme on s'y attend de façon intuitive à savoir — en notant j l'état sans la variable t et s l'état augmenté :

$$P(s,a,s') = P(j,a,j') \times P(t,j,a,t')$$

  avec, en posant d la durée associée à $F(d|s,a)$:

$$\begin{cases} \text{si } t + d \leq T_f : P(t,j,a,t' = t+d) = F(d|j,a) \\ \text{si } t + d > T_f : P(t,j,a,t' = t+d) = \sum_{d/t+d>T_f} F(d|j,a) \end{cases}$$

  On traduit ainsi la durée des actions sur le temps discrétisé. On peut noter entre autres que $P(Tf, Tf) = 1$ ce qui est bien cohérent.

- Enfin, pour chaque état j de l'espace d'état initial, on effectue l'opération :

$$\pi(j,t) \leftarrow \pi(j)$$

On effectue alors sur ce MDP augmenté une série d'itérations afin de trouver la politique optimale en utilisant un des algorithmes connus pour la résolution des MDP classiques. Afin de limiter les calculs et de partir d'une politique partiellement optimale, on commence les itérations en utilisant la politique π qui a été redéfinie à la dernière étape.

Une fois le MDP augmenté résolu, l'agent 1 crée sa chronologie et l'envoie à l'agent 2.

A la réception de la chronologie de l'agent 2, l'agent 1 modifie sa variable t et son MDP de la façon suivante :

- On retire ou on rajoute des valeurs à t selon que le nouveau T_f est plus ou moins élevé que l'ancien.
- On modifie la fonction de récompense comme à l'étape précédente
- On modifie la fonction de transition de même (on peut notamment récupérer les fonctions de récompense et de transition de l'étape précédente pour limiter le nombre d'opérations)
- On modifie la dernière politique calculée à l'étape précédente pour la prolonger sur le nouvel espace d'état. Dans le cas où le nouveau T_{f_new} est supérieur à l'ancien T_{f_old}, les $\pi(j,t)$ pour $t > T_{f_old}$ sont égaux à $\pi(j, T_{f_old})$. Sinon, on « tronque » la politique précédente sur le nouvel espace des t.

De nouveau, on résout le MDP mis à jour, on calcule la chronologie et on l'émet.

Les deux agents améliorent ainsi itérativement la politique commune. On peut noter qu'il y a deux politiques principales qui s'échangent car il y a eu deux initialisations séparées. Le critère de fin des itérations est un critère de convergence, il peut s'appliquer sur la fonction de valeur, sur la politique ou sur la chronologie selon le cas que l'on choisit.

A chaque itération, chaque agent stocke la politique et la valeur de son état de départ selon cette politique dans des variables π_1, V_1, π_2, V_2. Lorsque l'algorithme s'arrête, il choisit la politique de valeur la plus élevée pour son état de départ et l'annonce à l'autre agent. Si les agents sont d'accord sur la politique à appliquer (1 ou 2), alors le calcul de la politique s'arrête là et l'exécution peut commencer. Sinon, on construit un MDP où les récompenses sont définies comme dans le cas précédent avec échéances sauf qu'à la date de l'échéance, plutôt que d'être automatiquement acquises, elles deviennent nulles. On calcule la valeur de l'état de départ, dans ce MDP, selon la dernière politique établie. Cette valeur constitue la contribution de l'agent à la valeur globale. On procède de même pour l'autre politique. Chaque agent émet les deux valeurs ainsi calculées et, à la réception, somme respectivement les valeurs correspondant aux politiques globales 1 et 2. La politique choisie est celle de plus grande valeur globale.

3.4.4 Replanification en cours de mission

Si, pour une raison ou une autre, le MDP d'un agent devait changer en cours de mission, il faut prévoir la possibilité d'une replanification efficace. Les causes d'une modification du MDP peuvent être nombreuses, on en retiendra deux principales :

- Un écart trop grand à la chronologie : Si un des agents s'écarte trop de la chronologie qu'il a annoncée, il peut rendre non-optimale la stratégie de l'autre agent par rapport à la réalité. Il s'agit alors de corriger le modèle du second agent au niveau des échéances et de lui donner l'occasion de replanifier.
- La modification des récompenses ou des probablilités de transition suite à l'acquisition d'une information par un des deux agents : si, par exemple, un des agents découvre qu'un terrain correspond à un coût en carburant plus élevé que prévu pour l'autre agent, il va émettre cette information et le modèle de récompense du second agent va changer. Il faut donc laisser une possibilité pour des replanifications rapides en cours d'exécution.

Une replanification est déclenchée sur un message d'un des deux agents accompagné de sa nouvelle chronologie. Cette chronologie est construite sur le MDP initial auquel ont été retirées toutes les récompenses déjà prises (la donnée de la disponibilité des récompenses est contenue dans l'état de chaque agent à l'instant de la replanification). La variable t est remise à zéro à l'instant de la replanification. Cependant, la politique courante est conservée (en effectuant une translation des valeurs de t afin de mettre la date courante à zéro) et les itérations reprennent à partir de cette politique. L'intérêt de procéder de cette façon est que si la modification du MDP

est mineure, la politique ne va pas changer et l'algorithme va s'arrêter après une itération sans avoir eu à recalculer intégralement une politique identique.

Un agent peut émettre un message de replanification en se basant sur un critère donné : si le critère est satisfait, l'agent recalcule seul sa politique optimale puis crée sa chronologie et l'émet avec le message de replanification à l'intention de l'autre agent. Les itération sur la chronologie démarrent alors comme décrit ci-dessus sur la base de la chronologie proposée par l'agent qui a initié la replanification.

3.5 Le problème de navigation d'un drone hélicoptère et d'un robot terrestre

On s'intéresse plus particulièrement au problème précis d'un drone hélicoptère et d'un robot terrestre ayant une mission commune d'amener le robot terrestre jusqu'à une récompense dont la localisation n'est pas initialement connue. Par ailleurs, la carte est mal connue et les coûts de traversée d'une zone connus initialement peuvent se révéler différents en cours de mission. Il s'agit donc d'un problème d'exploration doublé d'une recherche de chemin optimal dans un cadre à deux agents et dans un environnement incertain.

3.5.1 Le drone hélicoptère

Ce drone est un robot aérien doté d'instruments de prise de vues. Son rôle dans le binôme est de repérer des passages praticables pour le robot terrestre. Pour cela, il explore la carte à une certaine altitude et envoie les informations de texture de l'image — correspondant à des types de terrain et donc à des coûts de traversée — au robot terrestre. Les informations fournies par le robot aérien sont toutefois encore entachées d'une certaine incertitude traduisant la résolution de ses capteurs : par exemple, il pourra déclarer praticable une zone de l'espace parce qu'il voit qu'elle est goudronnée mais n'aura pas distingué le grillage de fil de fer qui la coupe en deux et en empêche la traversée. On modélise ce phénomène en écrivant que les informations du robot aérien sont fiables à 80% et en introduisant une erreur aléatoire dans le résultat de ses mesures.

L'univers des deux agents est un monde-grille. Afin de simplifier la représentation du problème, on supposera que l'hélicoptère vole à une altitude constante. Par ailleurs, la connaissance des coûts de traversée sur le monde-grille est représentée par une seconde grille où le coût en question connu est associé à chaque case. Le champs de vision du drone hélicoptère couvre un carré de 3x3 cases centré sur le drone.

Le drone hélicoptère est un de nos deux agents ; selon la modélisation de la section précédente, son contexte de décision est représenté par le MDP comprenant :

- Un espace d'état discret et dénombrable S_h composé des variables
  - pos_h, la position sur la grille, variable entière (ou couple de variables entières),
  - f_h, le carburant restant, variable entière,
  - sol, variable binaire indiquant si le drone est posé,
  - ev, une liste, variable globale de mission indiquant quelles cases ont déjà été explorées (en particulier pour le robot aérien, on note qu'à chaque émission d'information, on ajoute entre 0 et 9 cases à ev),

  - em, une sous-liste incluse dans ev indiquant sur quelles cases le drone a effectué une communication vers le robot terrestre,
  - obs, une variable entière comprise entre 0 et 9 indiquant si l'observation que fait le drone aérien est conforme à la connaissance qu'il a du terrain : 0 correspond à « aucune différence », 9 correspond à « toutes les cases sont différentes »,
  - rt, une variable binaire, commune aux deux robots, indiquant si la récompense a été trouvée. En fait, il s'agit d'une variable d'historique, ou de phase de mission permettant de définir des récompenses (et donc des comportements) très différents en fonction de sa valeur.
- Un espace d'action A_h discret et dénombrable également constitué des actions :
  - N, S, E, O, les actions de déplacement élémentaire dans les quatre directions cardinales,
  - $emettre$, l'action d'envoyer le résultat de la prise de vues au robot terrestre,
  - $poser$, l'action de se poser,
  - $null$, l'action de ... ne rien faire pendant une unité de temps.
- une fonction de transition $P_h(s'|s,a)$ décrite plus bas,
- une fonction donnant les distributions de probabilité sur les durées des transitions $F_h(t|s,a)$ décrite plus en détail plus loin,
- enfin une fonction de récompense $r_h(s,a,s')$ elle aussi décrite plus bas.

Il est important de noter que, potentiellement, les variables ev et em représentées de façon binaire comportent autant de sous-variables qu'il y a de cases dans la grille et donc que la taille de l'espace d'état est, de façon symbolique : $|pos_h| \cdot |f_h| \cdot |sol| \cdot |ev| \cdot |em| \cdot |obs| \cdot |rt|$. En posant a le nombre de cases de la grille et b le nombre d'états discrets de la réserve de carburant, l'espace d'état initial contient $40 \cdot b \cdot a^3$ états. On peut par contre remarquer également que cette valeur est un majorant puisqu'une partie de l'espace d'état peut être écarté dès le début en considérant les dépendances des variables d'état entre elles.

Afin de simplifier la lecture, on note tout de suite et on omettra de mentionner plus loin que si $sol = 1$ toutes les actions ont un effet nul sur l'état. On remarque donc que le fait de se poser est la seule manière pour le drone hélicoptère d'éviter le crash (!) quand son niveau d'essence devient bas car — on va le voir — toutes les actions consomment du carburant sauf lorsque le drone est posé. Par contre, le fait que toutes les actions aient un effet nul indique qu'une fois posé, le drone ne peut plus redécoller. On omettra donc dans la suite tous les cas où $sol = 1$.

La figure 3.7 représente de façon schématique les probabilités d'arrivée dans les différentes cases accessibles lors d'une action de déplacement (ici l'action E).

Pour l'action de déplacement m (indifféremment N, S, E ou O), on peut écrire :

$$\begin{aligned} P_h(s'|s,m) &= P_x(pos'_h|s,m) \cdot P_f(s|f_h,m) \cdot P_{sol}(sol'|s,f'_h,m) \cdot P_{ev}(ev'|s,pos'_h,m) \cdot \\ &\quad P_{em}(em'|s,m) \cdot P_{obs}(obs'|s,pos'_h,m) \cdot P_{rt}(rt'|s,pos'_h,m) \end{aligned}$$

avec :

$$P_x(pos'_h|s,m) = \begin{cases} 0.94 & \text{si } pos'_h \text{ est la case visée} \\ 0.02 & \text{si } pos'_h \text{ est une des 3 cases sur la perpendiculaire} \\ & \quad \text{au « vecteur » } m \\ 0 & \text{sinon (voir la figure 3.7)} \end{cases}$$

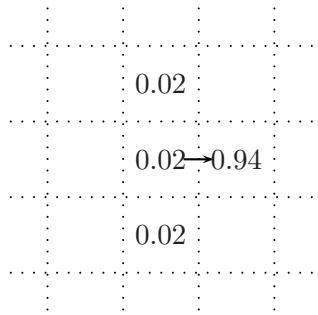

FIG. 3.7 – Probabilités sur les états d'arrivée d'une action E

$$P_f(f'_h|s,m) = \begin{cases} \text{si } f_h \leq 1 & p(f'_h = -1) = 1 \\ \text{si } f_h = 2 & p(f'_h = -1) = 0.9 \\ & p(f'_h = 0) = 0.1 \\ \text{si } f_h \geq 3 & p(f'_h = f_h - 4) = 0.1 \\ & p(f'_h = f_h - 3) = 0.8 \\ & p(f'_h = f_h - 2) = 0.1 \end{cases}$$

Le niveau -1 de carburant correspond à la panne sèche en cours d'action : c'est le marqueur d'accident, le drone tombe en panne avant d'avoir fini son action et s'écrase au sol.

$$P_{sol}(sol'|s, f'_h, m) = \begin{cases} 1 & \text{si } sol' = 0 \text{ et } f'_h \neq -1 \\ 1 & \text{si } sol' = 1 \text{ et } f'_h = -1 \\ 0 & \text{sinon} \end{cases}$$

On considère que si on a suffisamment de carburant, il n'y a aucun risque de se poser (ou de s'écraser) involontairement. Par contre, en cas de panne sèche, on se retrouve assez violemment au sol et ce, de façon déterministe.

$$P_{ev}(ev'|s, pos'_h, m) = \begin{cases} 1 & \text{si } ev' = ev \cup square(pos'_h) \\ 0 & \text{sinon} \end{cases}$$

L'opérateur $square()$ associe à chaque case x l'ensemble de ses huit cases adjacentes et elle-même (le carré centré en x) ; ces cases correspondent au champ de vision de l'hélicoptère.

$$P_{em}(em'|s,m) = \begin{cases} 1 & \text{si } em' = em \\ 0 & \text{sinon} \end{cases}$$

$$P_{obs}(obs'|s,m) = \begin{cases} 0.40 & \text{si } obs' = 0 \\ 0.27 & \text{si } obs' = 1 \\ 0.18 & \text{si } obs' = 2 \\ 0.15 & \text{si } obs' = 3 \\ 0.00 & \text{si } obs' = 4,5,6,7,8 \text{ ou } 9 \end{cases}$$

On peut noter que la probabilité d'avoir obs' dans $\{4,5,6,7,8,9\}$ est nulle car quel que soit le résultat du mouvement, seulement 3 cases du champ de vision du drone changent entre l'état de départ et celui d'arrivée. A ce stade, on va simplifier le problème en transformant la variable *obs* en une variable binaire : si *obs* vaut zéro, les cases vues sont conformes à la connaissance qu'on en avait, si *obs* vaut 1, l'observation diffère de la connaissance. La taille de l'espace d'état

est ainsi ramenée à $8 \cdot b \cdot a^2$. La fonction $P_{obs}(obs'|s,m)$ s'écrit :

$$P_{obs}(obs'|s,m) = \begin{cases} 0.35 & \text{si } obs' = 0 \\ 0.65 & \text{si } obs' = 1 \end{cases}$$

Cependant, il convient de noter qu'aucune de ces deux représentations n'est fidèle à la réalité. En effet, afin d'établir une distribution exacte sur les états, il faudrait prendre en compte la configuration géométrique de ev. Une fonction de transition exacte possible pour obs serait : (avec $|E|$ le cardinal de l'ensemble E)

$$P_{obs}(obs'|s,pos'_h,m) = \begin{cases} \text{si } |\,square(pos'_h) \cap ev| = 6 & p(obs' = 0) = 0.40 \\ & p(obs' = 1) = 0.27 \\ & p(obs' = 2) = 0.18 \\ & p(obs' = 3) = 0.15 \\ \text{si } |\,square(pos'_h) \cap ev| = 7 & p(obs' = 0) = 0.49 \\ & p(obs' = 1) = 0.30 \\ & p(obs' = 2) = 0.21 \\ \text{si } |\,square(pos'_h) \cap ev| = 8 & p(obs' = 0) = 0.64 \\ & p(obs' = 1) = 0.36 \\ \text{si } |\,square(pos'_h) \cap ev| = 9 & p(obs' = 0) = 1.00 \end{cases}$$

Afin de retrouver le calcul menant à cette distribution équivalente, il faut considérer le problème :

Considérons une grille 3x3 *composée de cases pouvant recevoir au maximum une bille. Sachant qu'on connaît la distribution de probabilité P_9 sur le nombre de billes présentes dans la grille* 3x3*, quelle est la distribution P_8 sur le nombre de billes présentes dans la même grille à laquelle on enlève une case a ?*

Ce problème se résout en (la démonstration est donnée en annexe) :

$$P_8(k) = \frac{k+1}{9} P_9(k+1) + \frac{k-9}{9} P_9(k)$$

et se généralise en :
Si on connait la distribution de probabilités P_n sur le remplissage d'une grille de n cases alors la distribution sur une grille de $n-1$ cases s'écrit :

$$P_{n-1}(k) = \frac{k+1}{n} P_n(k+1) + \frac{n-k}{n} P_n(k)$$

On peut ainsi obtenir une formule littérale ou une version itérative afin de calculer les distributions comme on l'a fait pour le résultat ci-dessus (à partir de la distribution connue pour n=3). Pour l'instant toutefois, on se limitera à considérer obs comme une variable binaire et on se réserve la possibilité d'étendre le modèle plus tard.

Enfin la distribution sur rt' s'écrit (en posant a le nombre de cases dans la grille) :

$$P_{rt}(rt'|s,pos'_h,m) = \begin{cases} \frac{9-|square(pos'_h)\cap ev|}{a-|ev|} & \text{si } rt = 0 \text{ et } rt' = 1 \\ 1 & \text{si } rt = 1 \text{ et } rt' = 1 \\ 0 & \text{sinon} \end{cases}$$

En effet, on a une probabilité de $\frac{1}{a-|ev|}$ de trouver la récompense finale, par case et on a $9 - |square(pos'_h) \cap ev|$ cases nouvelles.

L'action *null* n'a pas d'autre effet sur l'état courant que de diminuer la quantité de carburant :

$$\forall s \in (S_h \setminus \{f_h\}), P((s, f_h - 1)|null, (s, f_h)) = 1.$$

L'action *emettre* affecte les variables *obs*, *em* et f_h en remettant la première à zéro, en mettant à jour la seconde et en consommant une unité de carburant de la troisième. On note que dans ce premier modèle on ne prend pas en compte les possibles erreurs de transmission ou les cas où la transmission est impossible.

$$P(s'|s, emettre) = P_f(f'_h|s, emettre) \cdot P_{em}(em'|s, emettre) \cdot P_{obs}(obs'|s, emettre)$$

$$P_h(f'_h|emettre) = \begin{cases} 1 & \text{si } f'_h = f_h - 1 \\ 0 & \text{sinon} \end{cases}$$
$$P_{em}(em'|s, emettre) = \begin{cases} 1 & \text{si } em' = 0 \\ 0 & \text{sinon} \end{cases}$$
$$P_{obs}(obs'|s, emettre) = \begin{cases} 1 & \text{si } obs' = 0 \\ 0 & \text{sinon} \end{cases}$$

L'action *poser* a comme effet de tenter de poser l'appareil à la verticale du point courant. La fonction de transition pour l'action *poser* s'écrit :

$$P(s'|s, poser) = P_f(f'_h|s, poser) \cdot P_{sol}(sol'|s, f'_h, poser)$$
avec :
$$P_f(f'_h|s, poser) = \begin{cases} \text{si } f_h - 2 \leq 0 & p(f'_h = -1) = 1 \\ \text{si } f_h = 3 & p(f'_h = -1) = 0.9 \\ & p(f'_h = 0) = 0.1 \\ \text{si } f_h \geq 4 & p(f'_h = f_h - 5) = 0.1 \\ & p(f'_h = f_h - 4) = 0.8 \\ & p(f'_h = f_h - 3) = 0.1 \end{cases}$$
$$P(sol'|s, f'_h, poser) = \begin{cases} \text{si } f'_h = -1 & p(sol' = 1) = 1 \\ \text{si } f'_h \geq 0 & p(sol' = 1) = 0.97 \\ & p(sol' = 0) = 0.03 \end{cases}$$

Avec les fonctions précédentes la fonction de transition est entièrement définie. Il s'agit à présent de définir les probabilités sur la durée des actions. On a :

Pour une action de déplacement :

$$F(t|s,m) = \begin{cases} 0.1 & \text{pour } t = 2 \\ 0.8 & \text{pour } t = 3 \\ 0.1 & \text{pour } t = 4 \\ 0 & \text{sinon} \end{cases}$$

Pour l'action *emettre*

$$F(t|s,emettre) = \begin{cases} 1 & \text{pour } t = 1 \\ 0 & \text{sinon} \end{cases}$$

Pour l'action *null*

$$F(t|s,null) = \begin{cases} 1 & \text{pour } t = 1 \\ 0 & \text{sinon} \end{cases}$$

Pour l'action *poser*

$$F(t|s,poser) = \begin{cases} 0.1 & \text{pour } t = 3 \\ 0.8 & \text{pour } t = 4 \\ 0.1 & \text{pour } t = 5 \\ 0 & \text{sinon} \end{cases}$$

Il reste enfin à définir les fonctions de récompense et de coût. Afin de définir les notions de récompenses et de coût, il faut déterminer sur quelles grandeurs portent ces valeurs et donc anticiper sur la mission de l'agent. Les questions à se poser sont donc : « Qu'est-ce que l'agent cherche à économiser ? » (coût) et « Qu'est-ce que l'agent cherche à obtenir ? » (récompense). Ici, on est dans un cas où les réponses à ces deux questions sont, respectivement, le carburant et l'information du terrain ; ces deux grandeurs n'étant pas homogènes, la fonction de récompense traduit l'importance relative que l'on apporte à chacune de ces grandeurs. Par ailleurs, on note qu'une des variables d'état se retrouve dans les récompenses, l'espérance de la variable en question (le carburant ici) associée à une transition donnée doit être cohérente avec la distribution de probabilité sur la durée de l'action et avec le taux de récompense défini. On définit ainsi :

$$c(s',s,a) = \begin{cases} -1 & \text{si } sol = 0 \\ 0 & \text{si } sol = 1 \end{cases}$$

On note que la fonction de coût est bien définie ce façon cohérente avec les fonction de transition et de durée : pour une action durant en moyenne 3 unités de temps et qui a une forte probabilité de consommer 6 unités de carburant, la consommation doit être définie à -2.

Les récompenses portent ensuite sur l'exploration de la carte quand l'objectif n'a pas encore été trouvé. La récompense concerne en fait le fait de tenir le robot terrestre au courant de l'évolution des récompenses et de la connaissance de la carte. La récompense est donc attribuée à l'émission de l'information d'observation si elle est différente de la connaissance antérieure. La fonction de récompense notée r_h s'exprime :

$$\forall s \in S, \forall s' \in S, (pos'_h \notin em) \text{ et } (rt = 0) \Rightarrow r_h(s, emettre, s') = 1$$

Dans un modèle plus fin, on pourrait choisir une fonction non-binaire pour définir la fonction de transition de *obs* (comme dans les exemples plus haut) et utiliser ainsi la valeur de obs' pour moduler la récompense associée à une émission.

Une fois la récompense trouvée, les récompenses portent sur l'aide à la navigation du robot terrestre. Via la chronologie communiquée par le robot terrestre, on établit deux listes : $traj$ contient les cases appartenant à la trajectoire prévue par le robot terrestre, adj contient $traj$ et les cases adjacentes à $traj$. La fonction de récompense vaut alors :
(on note $\overline{em} = \{pos_h | pos_h \notin em\}$)

$$r_h(s,a,s') = \begin{cases} \text{si } \overline{em} = \emptyset & & r(poser) = 30 \\ \text{si } \overline{em} \neq \emptyset & \text{si } pos'_h \in \overline{em} \cap adj \text{ et } obs' \neq 0 & r(s', emettre) = 4 \\ \text{si } \overline{em} \neq \emptyset & \text{si } pos'_h \in \overline{em} \cap \overline{adj} \text{ et } obs' \neq 0 & r(s', emettre) = 1 \end{cases}$$

Les listes $traj$ et adj ne font pas partie de l'état du drone hélicoptère car elles ne sont pas représentatives de son état courant. Elles ne servent qu'à définir les valeurs de r_h et ne sont donc créées et utilisées que lors d'une communication afin de remettre la fonction de récompense à jour.

Il faut également définir le coût d'un crash. C'est fait assez simplement avec :

$$r_h(s,a,s') = -200 \text{ si } sol = 0, sol' = 1 \text{ et } f'_h = -1$$

Avec ces données, le MDP du drone hélicoptère est entièrement défini. Il faut donc s'attacher maintenant à définir celui du robot terrestre.

3.5.2 Le robot terrestre

Le robot terrestre est une robot mobile autonome progressant sur le sol avec plus ou moins de succès en fonction du terrain. Malgré son handicap de déplacement, il « voit » de façon très fine à faible distance ce qui lui permet de corriger les éventuelles erreurs d'estimation du drone hélicoptère. Par ailleurs, il est le seul des deux robots à pouvoir prendre la récompense finale. La recherche d'une politique optimale pour le robot terrestre peut se voir intuitivement comme la recherche d'une manière de trouver et de rallier la récompense en dépensant un minumum de carburant.

L'univers du robot terrestre est le même monde-grille que pour le robot aérien. La vision du robot terrestre est limitée à la case où il se trouve. La portée de la vision des robots est particulièrement importante car — comme on le verra plus loin — le coût du déplacement porte sur les transitions et l'information sur ce coût est connue de façon incertaine au début de la mission (c'est la méconnaissance de la carte). Le fait de « voir » une case revient à estimer les quatre coûts de sortie de cette case, ainsi, le robot terrestre ne connaît précisément que les coûts de sortie des cases qu'il a déjà visitées ; il connait avec une certaine incertitude les coûts de sortie des cases que le drone hélicoptère a visitées et communiqués et il a une estimation initiale des coûts de sortie de toutes les autres cases.

On peut écrire le MDP du robot terrestre comme :
- Un espace d'état discret et dénombrable S_r comprenant les variables :
  - pos_r, la position sur la grille, variable entière (ou couple de variables entières),
  - f_r, le carburant restant, variable entière,
  - ev, une liste, variable globale de mission indiquant quelles cases ont déjà été explorées,
  - rt, une variable binaire, commune aux deux robots, indiquant si la récompense a été trouvée. Il s'agit, de fait, d'une variable d'historique permettant de séparer les phases de mission et des comportements différents.

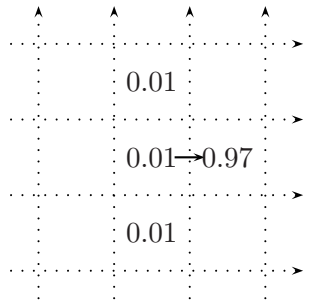

FIG. 3.8 – Probabilités sur les états d'arrivée d'une action E

  - ra, une variable binaire indiquant la disponibilité de la récompense finale.
- Un espace d'action A_r discret et dénombrable constitué des actions :
  - N, S, E, O, les actions de déplacement élémentaire dans les quatre direction cardinales,
  - *null*, l'action de ne rien faire pendant une unité de temps.
  - *prendre*, l'action de prendre la récompense (déterrer la mine trouvée, prendre la photo détaillée de l'objet trouvé, ...)
- Une fonction de transition $P_r(s'|s,a)$ décrite plus bas,
- une fonction donnant les distributions de probabilité sur les durées des transitions $F_r(s'|s,a)$ décrite en détail plus loin,
- enfin une fonction de récompense $r_r(s,a,s')$ elle aussi décrite plus bas.

La taille de l'espace d'état du robot terrestre est significativement moins élevée que celle du robot aérien. En effet, en adoptant les même notations de a pour le nombre de cases du monde-grille et de b pour le nombre d'états discrets de la variable f_r, on a un espace d'état de taille $4 \cdot a \cdot b$. C'est particulièrement intéressant, en particulier si on met cette donnée en relief avec le fait que le robot au sol est celui qui effectue le plus de replannifications.

Comme précisé plus haut, tous les états tels que $ra = 1$ sont des états terminaux, c'est à dire que :

$$\forall (s,a) \in (S_r, A_r), \qquad ra = 1 \Rightarrow P_r(s,a,s) = 1$$

Dans la suite, on ne considérera donc que des cas où $ra = 0$.

La figure 3.8 représente de façon schématique les probabilités d'arrivée dans les différentes cases accessibles lors d'une action de déplacement (ici l'action E).

Pour une action de déplacement m (indifféremment N, S, E ou O) on peut écrire :

$$\begin{aligned} P_r(s'|s,m) &= P_x(pos'_r|s,m) \cdot P_f(f'_r|s,m) \cdot P_{ev}(ev'|s,pos'_r,m) \cdot P_{rt}(rt'|s,m) \cdot \\ &\quad P_{ra}(ra'|s,m) \end{aligned}$$

avec :

$$P_x(pos'_r|s,m) = \begin{cases} 0.97 & \text{si } pos'_r \text{ est la case visée} \\ 0.01 & \text{si } pos'_r \text{ est une des 3 cases sur la perpendiculaire} \\ & \quad \text{au « vecteur » } m \\ 0 & \text{sinon (voir la figure 3.8)} \end{cases}$$

On définit une fonction $c(pos_r, m)$ indiquant la consommation de carburant par unité de temps pour la transition (pos_r, m). Cette fonction est à valeurs entières. Cette information est équivalente à la connaissance du terrain. En fait, lorsqu'une information de « connaissance du terrain » est échangée entre les deux agents, ce sont les valeurs de cette fonction qui sont échangées. Ce sont donc les valeurs de cette fonction qui sont initialisés de façon identique pour les deux robots, c'est sur la comparaison de ces valeurs avec l'observation que le drone hélicoptère trouve la valeur de *obs* et c'est la mise à jour de cette valeur qui peut provoquer une replanification pour le robot terrestre. En effet, à chaque état traversé, le robot terrestre compare la valeur réelle de $c(pos_r, m)$ à celle qu'il avait précédemment stockée ; si la valeur est identique, il continue son exécution, sinon, il replannifie (on verra plus loin dans quel cas cela entraine une replannification du drone hélicoptère également).

$$P_f(f'_r|s,m) = \begin{cases} \text{si } f_r \leq 7 \cdot c(pos_r, m) - 1 & p(f'_r = -1) = 1 \\ \text{si } f_r = 8 \cdot c(pos_r, m) - 1 & p(f'_r = -1) = 0.9 \\ & p(f'_r = 0) = 0.1 \\ \text{si } f_r \geq 9 \cdot c(pos_r, m) - 1 & p(f'_r = f_r - 7 \cdot c(pos_r, m)) = 0.1 \\ & p(f'_r = f_r - 8 \cdot c(pos_r, m)) = 0.8 \\ & p(f'_r = f_r - 9 \cdot c(pos_r, m)) = 0.1 \end{cases}$$

Le niveau -1 de carburant correspond à la panne sèche en cours d'action : c'est le marqueur d'accident, le robot tombe en panne avant d'avoir fini son action et la mission s'arrête sur un échec du robot terrestre.

$$P_{ev}(ev'|s, pos'_r, m) = \begin{cases} 1 & \text{si } ev' = ev \cup \{pos'_r\} \\ 0 & \text{sinon} \end{cases}$$

Enfin la distribution sur rt' s'écrit (en posant a le nombre de cases dans la grille) :

$$P_{rt}(rt'|s, pos'_r, m) = \begin{cases} \frac{1}{a-|ev|} & \text{si } rt = 0 \text{ et } rt' = 1 \text{ et } pos'_r \in \overline{ev} \\ 1 & \text{si } rt = 1 \text{ et } rt' = 1 \\ 0 & \text{sinon} \end{cases}$$

On ne peut pas avoir à la fois $rt = 0$ et $ra = 1$; il n'existe aucune transition menant dans cet état et, logiquement, on ne peut commencer dans cet état de départ. Les phases de mission sont caractérisées par :
$rt = 0, ra = 0$ pendant la recherche de la récompense,
$rt = 1, ra = 0$ lorsque la récompense est trouvée et qu'on va la chercher,
$rt = 1, ra = 1$ lorsque la récompense est acquise.
On note que l'on pourrait choisir une unique variable globale rt qui prendrait des valeurs entières différenciant les phases de la mission. Pour l'instant on conserve les deux variables ra et rt séparées et on se réserve cette amélioration pour plus tard (on ne perd pas en optimalité puisqu'au final le fait de coder les phases de mission sur deux variables binaires est équivalent au fait de les coder sur une variable allant de 1 à 3).

$$P_{ra}(ra' = ra|s, m) = 1$$

L'action *null* n'a aucun effet sur l'état, le robot ne consomme même pas de carburant :

$$\forall s \in S_r, \qquad P_r(s|null, s) = 1$$

Lorsque la récompense est découverte — que ce soit par le robot terrestre ou par le drone hélicoptère — suffisamment d'information est disponible pour définir la fonction $r_t(pos_r)$ qui vaut 1 en à l'emplacement de la récompense et 0 ailleurs. Avec cette fonction, on peut définir la fonction de transition de l'action *prendre* qui n'affecte que les variables ra et f_h :

$$P_r(ra', f_h - 1|prendre, s) = \begin{cases} 1 & \text{si } ra = 0 \text{ et } ra' = 0 \\ 1 & \text{si } ra = 0 \text{ et } ra' = 1 \text{ et } r_t(pos_r) = 1 \\ 1 & \text{si } ra = 1 \text{ et } ra' = 1 \\ 0 & \text{sinon} \end{cases}$$

Il faut également écrire les distributions sur les durées des actions :

Pour une action de déplacement :

$$F(t|s, m) = \begin{cases} 0.1 & \text{pour } t = 7 \\ 0.8 & \text{pour } t = 8 \\ 0.1 & \text{pour } t = 9 \\ 0 & \text{sinon} \end{cases}$$

Pour l'action *null*

$$F(t|s, null) = \begin{cases} 1 & \text{pour } t = 1 \\ 0 & \text{sinon} \end{cases}$$

Pour l'action *prendre*

$$F(t|s, prendre) = \begin{cases} 1 & \text{pour } t = 1 \\ 0 & \text{sinon} \end{cases}$$

Il faut enfin définir les fonctions de récompense et de coût. Le coût d'un déplacement est une fonction de l'action entreprise : pour un déplacement, elle dépend du terrain et de la fonction $c(pos_r, m)$, pour l'action *null*, on a vu plus haut que la consommation était nulle. La fonction de coût s'écrit donc :

$$c(s', s, a) = \begin{cases} c(pos_r, m) & \text{si } a = m \\ 1 & \text{si } a = prendre \\ 0 & \text{si } a = null \end{cases}$$

Les récompenses portent ensuite sur l'exploration de la carte. Dans le cas où rt vaut zéro, même si l'information de localisation de la récompense est cruciale au robot terrestre, il ne cherche pas réellement à l'acquérir mais à acquérir directement la récompense, l'information n'est donc associée à aucune récompense, ou plutôt, la récompense sur l'information découle naturellement de la récompense finale. Dans le cas où la récompense a été trouvée, le drone hélicoptère a fourni les données nécessaires à la définition de la fonction $r_t(pos_r)$ (ou le robot mobile l'a trouvée tout seul) et la récompense vaut 200 là où $r_t(pos_r)$ vaut 1.

$$r_r(ra' = 1, prendre, ra = 0) = 200 \cdot r_t(pos_r)$$

On définit également un malus pour l'endommagement mécanique du robot terrestre qui tombe en panne sèche et abandonne la mission :

$$r_r(s, a, s') = -300 \text{ si } f_r \neq -1 \text{ et } f'_r = -1$$

Avec ces données, le MDP du robot terrestre est entièrement défini et on commence à avoir une idée du fonctionnement du binôme d'agents. On détaille ce fonctionnement dans la section suivante.

3.5.3 Remarques sur le déroulement d'une mission

Au début de la mission, comme prévu par l'algorithme indiqué plus haut, chaque agent résout seul son MDP puis établit sa chronologie et l'envoie à son homologue. Prenons l'exemple du robot terrestre pour lequel l'action optimale dans l'état de départ est l'action de déplacement N. L'état atteignable le plus probable est s_{r1} qui contient la case x_{r1} et le temps le plus probable d'exécution est 8. Le premier élément de la chronologie du robot terrestre est donc $(s_{r1}, 8)$. Puis on effectue la même opération à partir de s_{r1} : on cherche l'état d'arrivée s_{r2} le plus probable et le temps moyen d'exécution de l'action optimale t_{r2}. La chronologie s'enrichit ainsi d'un élément supplémentaire et devient : $(s_{r1}, 8, sr2, 8 + t_{r2})$. On continue ainsi à enrichir la chronologie jusqu'à l'échec du robot ou la découverte de la récompense.

De son côté, le drone hélicoptère procède de même : il calcule sa politique optimale et construit sa chronologie.

Dans notre cas particulier, la mission a plusieurs phases qui s'enchaînent dans un ordre logique. Cet ordre logique est imposé lors de transitions par l'évolution des variables de mission rt et ra. Or dans la recherche des politiques optimales, certains paramètres ne sont pas connus avant la fin d'une phase de mission. On ne peut se satisfaire de l'approximation qui consiste à considérer les différentes phases de mission comme des MDPs séparés que les agents résoudraient l'un après l'autre en utilisant les résultats du précédent pour construire et résoudre le suivant, pour la simple raison qu'un ne peut autoriser un agent à dépenser tout son carburant pour une unique phase de mission. Ce genre de considération est primordial pour notre exemple : au cas où le robot terrestre n'a que peu de carburant dans son état initial, il est plus intéressant pour lui de laisser le drone hélicoptère trouver la récompense de façon certaine et ensuite seulement, de se mettre en mouvement pour aller la chercher. Dans une modélisation où on considère des MDP séparés, le robot terrestre perd cette « contrainte d'économie » qui est due à la structure globale du problème. Il est donc important, pour conserver la politique optimale parmi toutes les politiques générables, de conserver la globalité du problème. On peut toutefois appliquer ici les algorithmes de décomposition vus aux sections 2.4.4, 2.4.5 et 2.4.6 afin de séparer le problème en morceaux plus petits sans perdre l'optimalité globale. En conservant la globalité du problème, on génère directement des politiques qui sont optimales globalement, en particulier qui respectent la contrainte d'économie. D'un point de vue naïf, on peut voir cet état de fait comme le fait que le coût d'un déplacement du robot au sol dans les premières phases de la mission est trop élevé comparé au gain espéré qui découle, certes, du gain final, mais qui est pondéré par une probabilité très faible d'accéder à cette récompense finale. Toujours dans cette optique naïve, on peut s'imaginer que la première politique envisagée par le robot au sol le fasse nécessairement avancer malgré le coût des déplacements puisqu'il se croit seul. A la deuxième itération de l'échange des chronologies, l'hélicoptère a déclaré ses intentions via sa chronologie et cela change le modèle de transition du robot au sol puisque - notamment - la probabilité de passer de $rt = 0$ à $rt = 1$ devient significative au fur et à mesure de la chronologie du robot aérien. Ainsi la politique optimale du robot terrestre peut changer et dépendre du

facteur temps nouvellement introduit et - par exemple - commander de ne rien faire pendant les premiers instants. L'exemple précédent, s'il est, une fois de plus, fort naïf, illustre comment le fait de conserver la globalité du problème permet, tout comme dans le cadre mono-agent, de conserver l'optimalité, en particulier dans la recherche de stratégie coopératives optimales.

On peut noter qu'une intervention d'interprétation humaine peut faciliter le calcul des chronologies : en effet, limiter les chronologies dans le temps jusqu'au passage à $rt = 1$ permet d'éviter aux robots de « tirer des plans sur la comète » à propos de la prise d'une récompense dont ils ne savent pas où elle est. Dans le même ordre d'idées, afin d'éviter de construire des chronologies trop longues, on peut limiter leur taille en nombre d'états visités ou en durée maximale et convenir que quel que soit la situation à la fin de l'exécution de la chronologie, les agents démarrent une replannification. Ce type de limitation par l'utilisateur semble très intéressant pour optimiser le temps de calcul, compte tenu de l'expérience *a priori* de l'utilisateur. Il existe cependant un moyen plus automatique et plus simple de limiter la taille des chronologies générées par les agents. Il suffit de calculer, au fur et à mesure de la création de la chronologie, sa « fiabilité », c'est-à-dire la probabilité que l'on arrive effectivment dans l'état donné par la chronologie à l'instant précisé dans la chronologie. Ce calcul est simple à réaliser puisque pour ajouter une action à la chronologie, on considère l'état courant, la durée et l'état d'arrivée. On a alors les probabilités de transition associées à chaque transition de la chronologie. La fiabilité d'une chronologie est le produit de toutes ces probabilités. On peut ainsi définir un critère qui arrête la génération de la chronologie lors que sa fiabilité passe en dessous d'un certain seuil, ce qui évite de considérer des transitions trop improbables.

Une extension possible du modèle actuel qui suivrait cette idée est la suivante : on pourrait insérer à chaque état de la chronologie, une valeur de fiabilité. Cette valeur pourrait alors être utilisée par le second agent dans la redéfinition de son propre MDP : l'acquisition automatique d'une récompense à une date donnée parce qu'elle a été déclarée comme prise par le premier agent pourrait être pondérée, comme toute autre transition, par la fiabilité de la prévision du rpemier agent. Cela permettrait de renforcer le caractère incertain des éléments de la chronologie et d'affiner l'optimalité des politiques générées par les agents. Cette idée est gardée pour une extension future.

3.5.4 Représentation factorisée d'un problème

Afin de préparer une possible implémentation d'un cas de test, on représente le problème sous forme d'arbres comme on l'a fait pour le problème des deux taxis.

On considère un monde grille de 5x5 cases afin d'avoir un cas simple qu'on n'a pas à décomposer en sous-zones et on définit dessus les arbres des figures 3.9 à 3.28 qui décrivent l'effet de chaque action sur chaque variable.

On essaie au maximum d'avoir des variables de taille limitée afin de limiter le nombre de tests à faire dans le parcours des arbres. En particulier, on sépare les deux coordonnées de l'agent sur sa grille, on écrit donc $pos_h = (x_h, y_h)$.

Par ailleurs, on adopte une représentation factorisée (voir [BDG99]) de l'historique des états visités, on représente donc la variable ev comme autant de variables d'état ev_i visités qu'il y a de cases dans la grille (à savoir 25 dans notre cas) et on ajoute artificiellement une variable

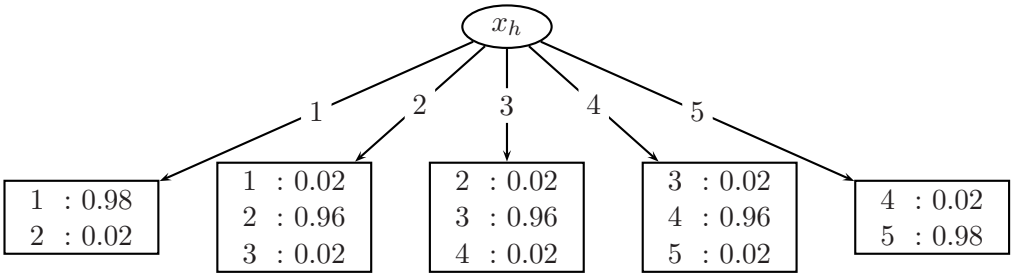

FIG. 3.9 – Arbre de transition de la variable x'_h pour l'action N du drone hélicoptère

de cardinal de ev que l'on note simplement $|ev|$. On numérote les ev_i le long d'une par abscisse croissante et on revient au début de chaque ligne pour continuer la numérotation par ordonnée croissante (le numéro d'une case (x, y) est donc obtenu par l'opération $x + 5 \times floor(y/5)$). Lorsque ev_i vaut 1, cela signifie que la case numéro i a déjà été explorée. Afin d'éviter de reproduire des schémas identiques, sur la figure 3.11 on représente le cas particulier d'une case en bord de grille et sur la figure 3.12 on représente le cas général d'une case dans la grille.

La variable f_h représentant le carburant restant est discrétisée sur 32 unités de carburant allant de -1 (panne sèche) à 30 (plein).

La convention de représentation de l'approche factorisée se lit de la façon suivante : pour l'action a et la variable résultat s', chaque variable s_i noeud de l'arbre traduit une dépendance de s' en s_i. Chaque banche partant du noeud de s_i représente une instanciation de s_i. Les feuilles de l'arbre indiquent les valeurs que peut prendre la variable s' et les probabilités associées.

Les variables que l'on considère donc pour l'hélicoptère sont : x_h, y_h, les 16 ev_i, f_h, *sol* et *rt* et une variable *em* indiquant s'il est nécessaire d'effectuer une transmission vers le robot terrestre. Les actions sont : N, S, E, O, *emettre* et *poser*.

Pour l'hélicoptère, l'action N est décrite de la figure 3.9 à la figure 3.17. Les autres actions de déplacement sont décrites de façon similaire. On représente également les récompenses et coûts associés à N sur les figures 3.18 et 3.19. Enfin il faut représenter les durées des transitions, ce qu'on fait sur la figure 3.20. Pour l'action *emettre*, les probabalités sont représentées de la figure 3.21 à la figure 3.22, les récompenses sont présentés sur la figure 3.23, les coûts sont définis comme pour l'action N à la figure 3.18 et les temps de transition sont représentés à la figure 3.24. De même pour l'action *poser* de la figure 3.25 à la figure 3.28. L'action *null* n'est pas représentée, cependant la construction de ses arbres est triviale.

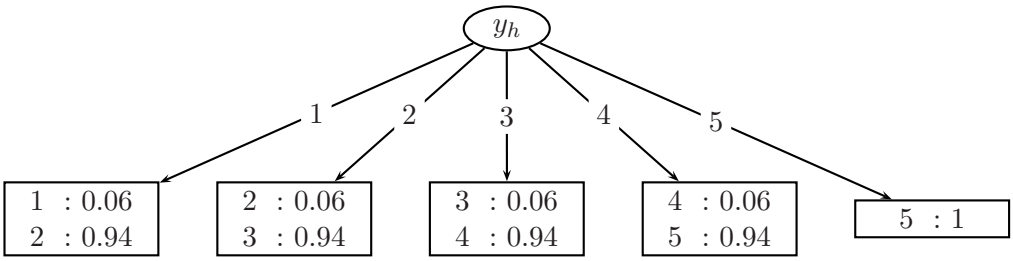

FIG. 3.10 – Arbre de transition de la variable y'_h pour l'action N du drone hélicoptère

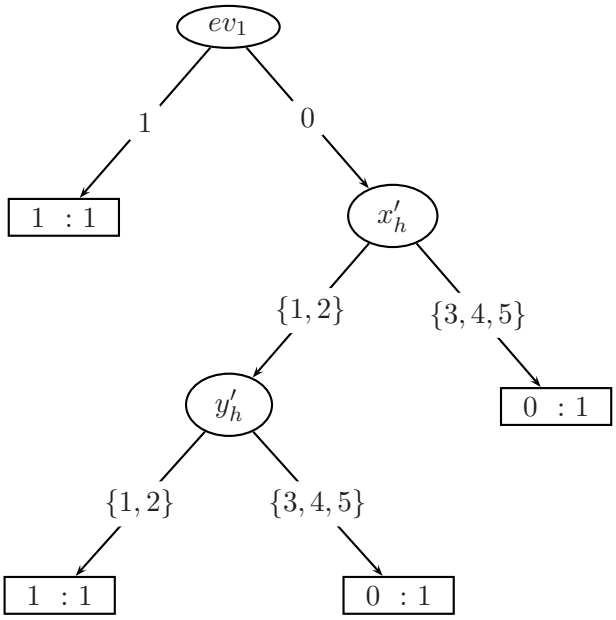

FIG. 3.11 – Arbre de transition de la variable ev'_1 pour l'action N du drone hélicoptère

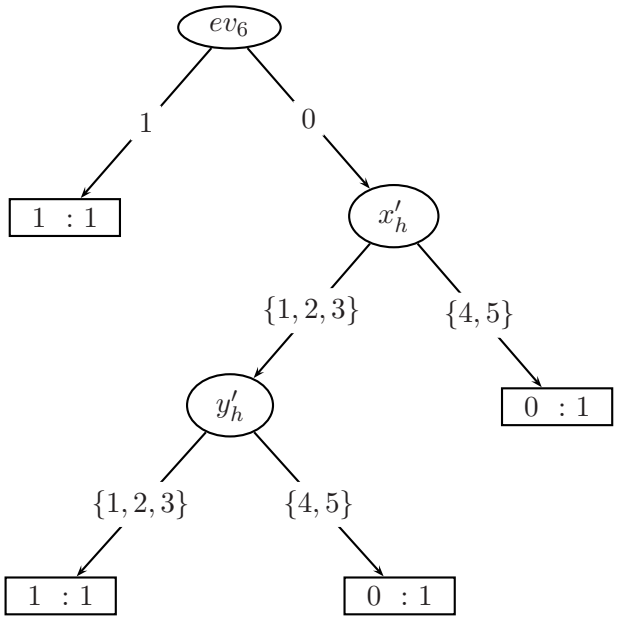

FIG. 3.12 – Arbre de transition de la variable ev'_6 pour l'action N du drone hélicoptère

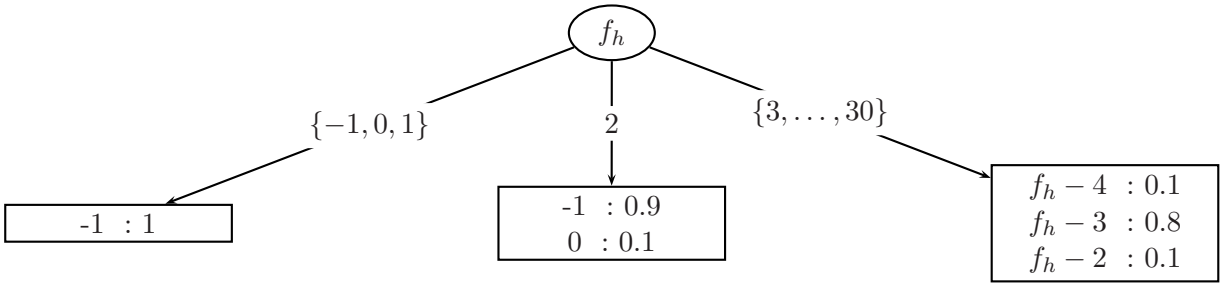

FIG. 3.13 – Arbre de transition de la variable f'_h pour l'action N du drone hélicoptère

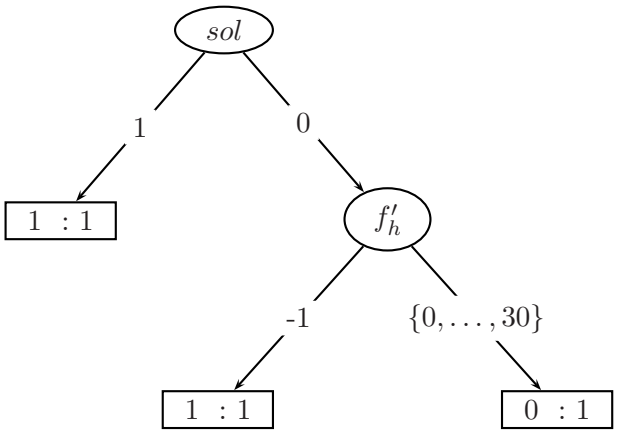

FIG. 3.14 – Arbre de transition de la variable sol' pour l'action N du drone hélicoptère

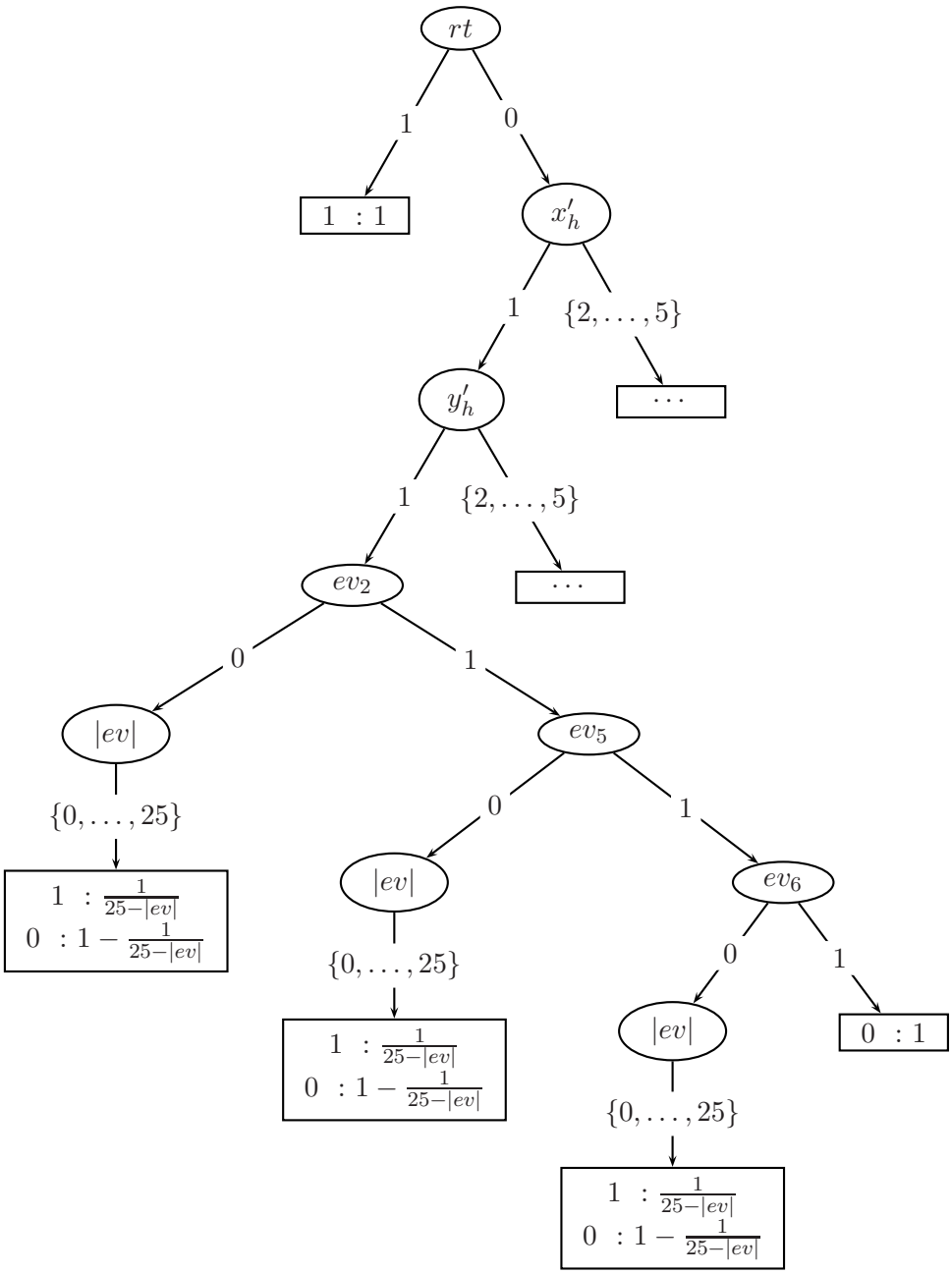

FIG. 3.15 – Arbre de transition de la variable rt' pour l'action N du drone hélicoptère, les "..." désignent les tests équivalents à ceux présentés pour $x'_h = y'_h = 1$: on regarde si au moins une des cases adjacentes à la case d'arrivée n'a jamais été visitée, s'il n'y en a aucune, alors on est sûrs de ne pas trouver la récompense à ce tour ci. On notera qu'on a pris le cas $x'_h = y'_h = 1$ pour sa simplicité de représentation (il n'y a que trois tests sur les cases 2, 5 et 6), dans le cas général, l'arbre est plus profond puisqu'il y a 5 tests à faire (deux pour déterminer la case d'origine (x_h, y_h) qui détermine les trois nouvelles cases explorées et trois pour tester si ces nouvelles cases ont déjà été visitées). Le cas général est donné dans l'arbre de la variable $|ev|'$.

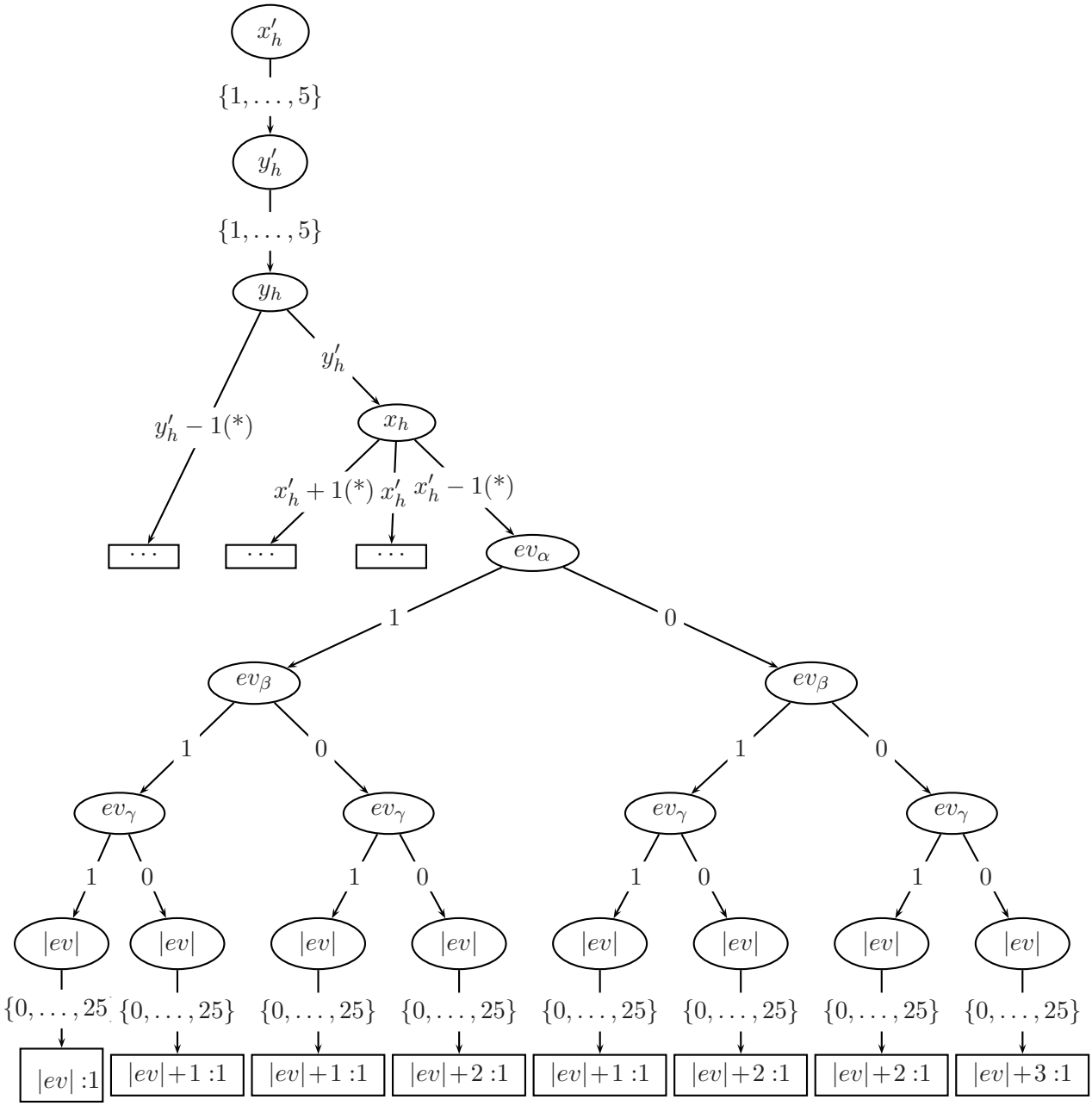

FIG. 3.16 – Arbre de transition de la variable $|ev|'$ pour l'action N du drone hélicoptère. Les "..." indiquent que l'arbre se développe de façon similaire à l'exemple explicité. α, β et γ représentent respectivement les points correspondant (x'_h+1, y'_h+1), (x'_h+1, y'_h), (x'_h+1, y'_h-1) c'est-à-dire les trois nouveaux points explorés par le drone. Cet arbre se parcourt de la façon suivante : on détermine la position actuelle, à partir de la position précédente on en déduit les trois nouvelles cases explorées, on les teste pour savoir si elles avaient déjà été explorées, si ce n'est pas le cas on augmente la valeur de $|ev|$. (*) s'il existe.

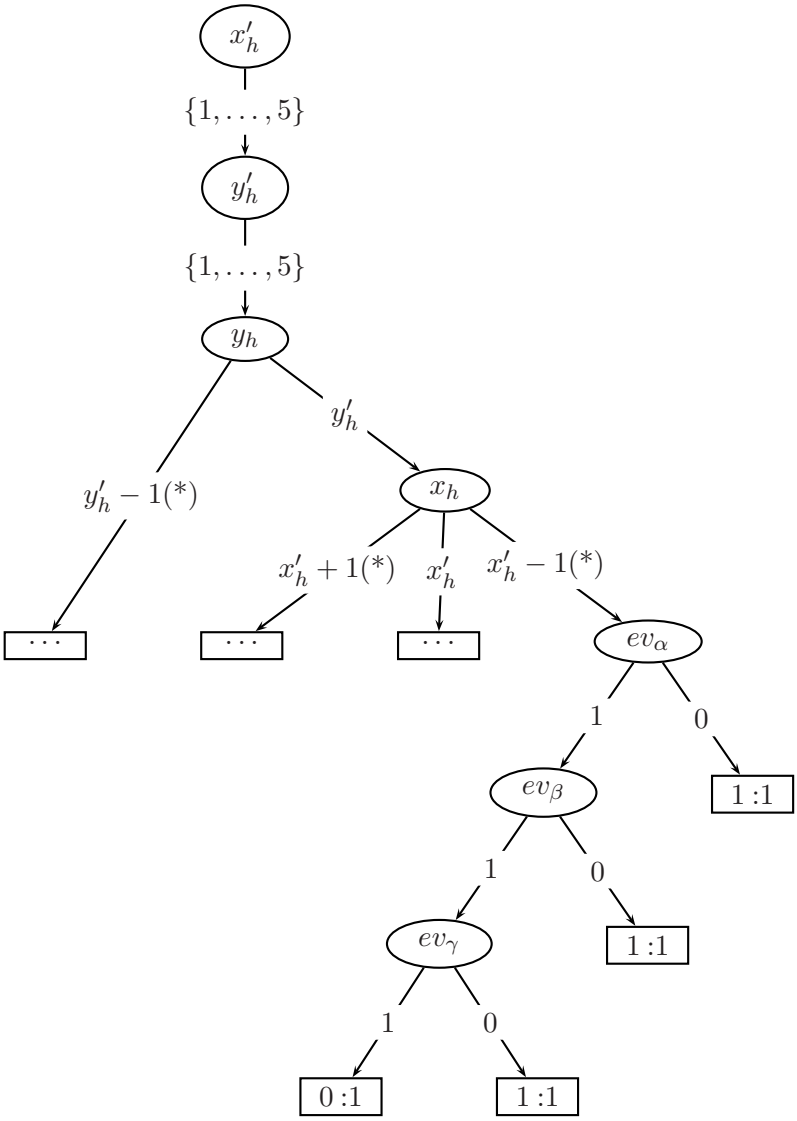

FIG. 3.17 – Arbre de transition de la variable $|em|'$ pour l'action N du drone hélicoptère. Les "..." indiquent que l'arbre se développe de façon similaire à l'exemple explicité. α, β et γ représentent respectivement les points correspondant (x'_h+1, y'_h+1), (x'_h+1, y'_h), (x'_h+1, y'_h-1) c'est-à-dire les trois nouveaux points explorés par le drone. Cet arbre se parcourt de la façon suivante : on détermine la position actuelle, à partir de la position précédente on en déduit les trois nouvelles cases explorées, on les teste pour savoir si elles avaient déjà été explorées, si ce n'est pas le cas il faut effectuer une emission et em passe à 1. (*) s'il existe.

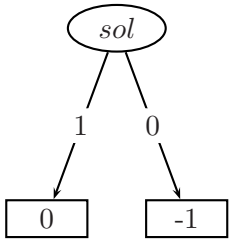

FIG. 3.18 – Arbre de coût par unité de temps pour l'action N du drone hélicoptère

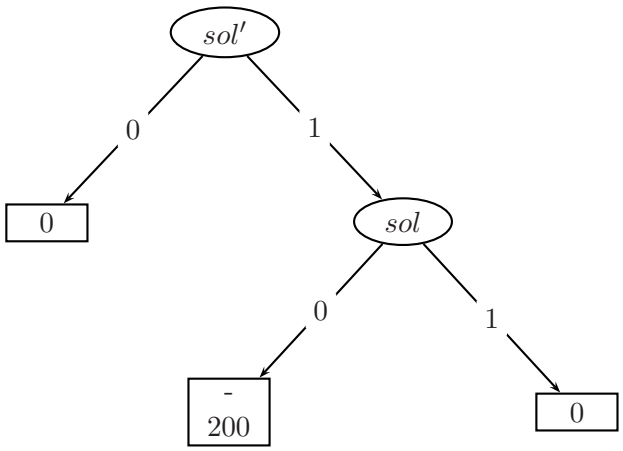

FIG. 3.19 – Arbre de récompense instantanée pour l'action N du drone hélicoptère

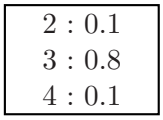

FIG. 3.20 – Arbre des durées de transition pour l'action N du drone hélicoptère

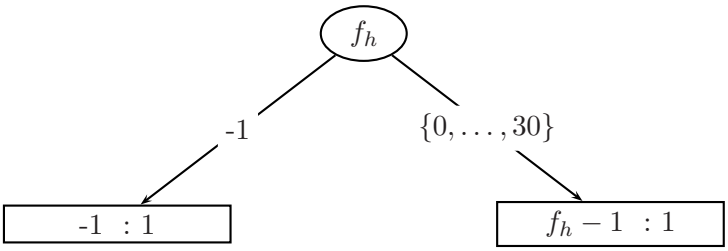

FIG. 3.21 – Arbre de transition de la variable f'_h pour l'action *emettre* du drone hélicoptère

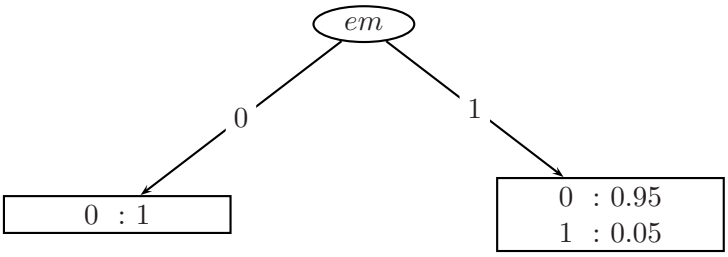

FIG. 3.22 – Arbre de transition de la variable em' pour l'action *emettre* du drone hélicoptère. La probabilité que em reste à 1 correspond à la probabilité d'avoir une erreur de transmission et donc que la transmission soit à refaire. Si on veut être exactement fidèle au modèle décrit plus haut alors il faut considérer que toutes les communications réussissent cependant une telle implémentation est un premier pas vers une prise en compte d'un coût de la communication.

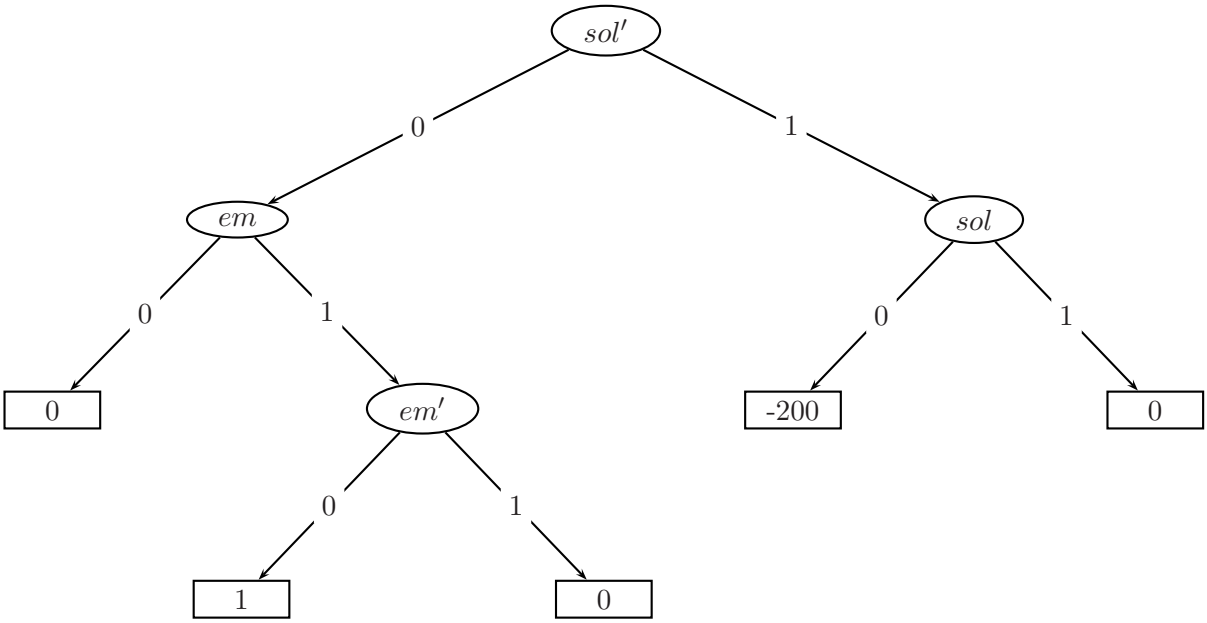

FIG. 3.23 – Arbre de récompense pour l'action *emettre* du drone hélicoptère. Il n'y a de récompense que pour une communication réussie.

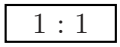

FIG. 3.24 – Arbre des durées de transition pour l'action *emettre* du drone hélicoptère

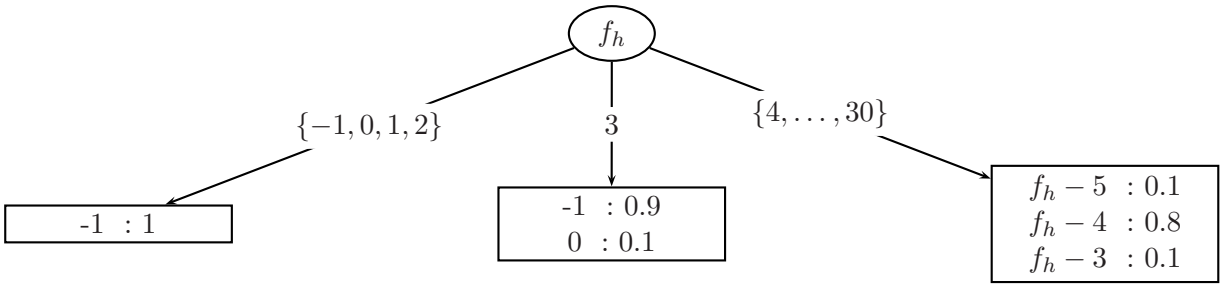

FIG. 3.25 – Arbre de transition de la variable f'_h pour l'action *poser* du drone hélicoptère

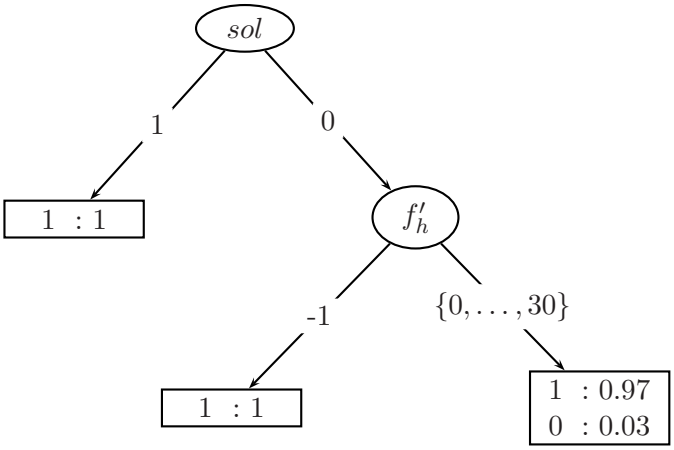

FIG. 3.26 – Arbre de transition de la variable sol' pour l'action *poser* du drone hélicoptère

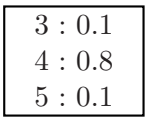

FIG. 3.27 – Arbre des durées de transition pour l'action *poser* du drone hélicoptère

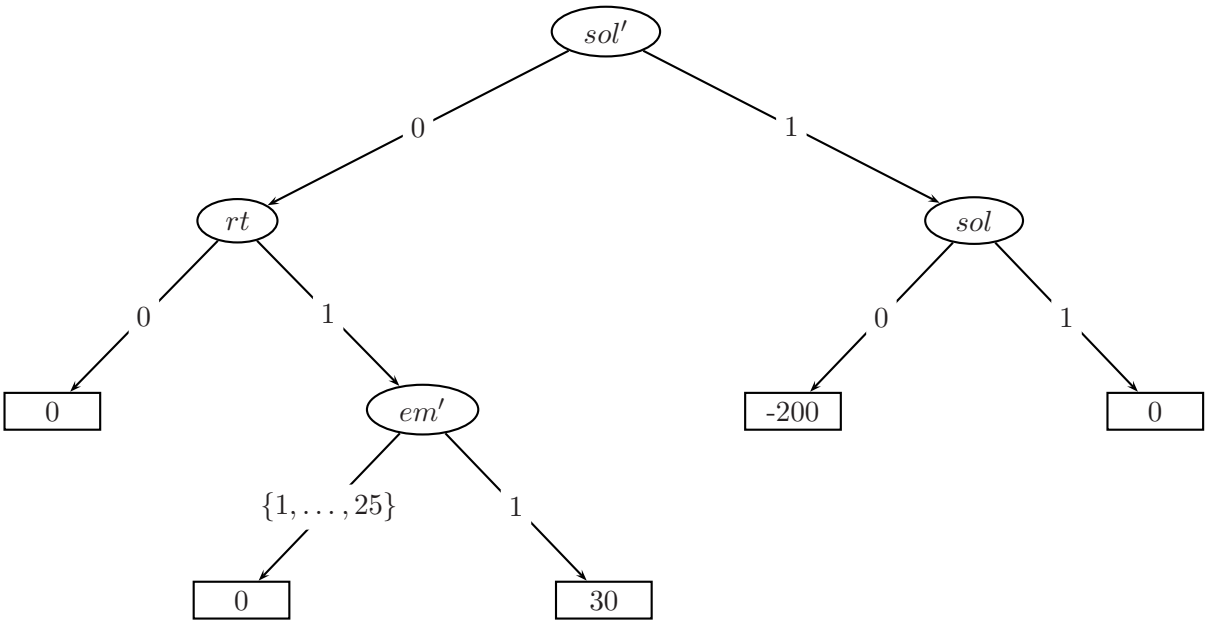

FIG. 3.28 – Arbre de récompense pour l'action *poser* du drone hélicoptère.

En effectuant le même travail de formalisation sur les fonctions de transition, de récompense, de coût et de durée du robot terrestre, on parvient à une forme compacte du problème traitable dans le cadre des MDP factorisés ([BDG99]). On récapitule de façon abstraite une exécution de l'algorithme à la section suivante.

3.5.5 Récapitulatif d'exécution de l'algorithme

Un premier plan

Lorsque la résolution du problème commence, chaque agent commence par résoudre le MDP du problème comme s'il était seul. Comme on l'a vu au 3.4.3 et avec les outils de décomposition (introduits au 2.4) et de factorisation de l'espace d'état, cette première politique est utilisée pour construire une chronologie par agent. Cette chronologie est, ou une chronologie complète des intentions de l'agent, ou un enchaînement d'actions et de résultats qui s'arrête lorsque le dernier résultat devient trop peu probable. Cette chronologie comporte l'information nécessaire afin que le second agent puisse mettre son MDP à jour afin de tenir compte des déclarations du premier agent.

L'échange des chronologies

Le message d'une chronologie, beaucoup plus léger qu'un message comportant une politique complète, est alors émis par chaque agent en direction de son homologue. La politique actuelle est stockée dans une première variable π_1. A la réception de la chronologie de l'autre agent, chaque agent met à jour son MDP comme indiqué au 3.4.3 ; il introduit une variable décrivant un temps discret, modifie ses fonctions de transition et de récompense afin d'intégrer les intentions de son collègue, et adapte enfin sa politique π_2 précédente (lors de la première itération, il prend π_1 ou une politique vide) au nouvel espace d'état enrichi de la variable temporelle.

L'amélioration des politiques individuelles

Chaque agent résout le MDP nouvellement défini. Cette résolution peut se faire en un temps plus court que la génération de la politique initiale car on part d'une solution (la dernière politique π_1 ou π_2) qui est déjà proche de l'optimalité. L'agent stocke alors cette nouvelle politique dans π_1 ou dans π_2 selon l'itération et reconstruit une chronologie. Il émet cette chronologie, reçoit celle de l'autre agent et recommence le processus d'amélioration d'une politique conjointe qui n'est jamais explicitement formulée.

La politique finale

Lorsque les agents parviennent à convergence pour les deux politiques π_1 et π_2, si c'est nécessaire, ils déterminent la valeur conjointe des deux politiques afin de choisir la meilleure des deux. On peut noter que pour un problème ayant une solution, on peut s'attendre à ce que l'algorithme converge puisque les fonctions de valeur des politiques π_1 et π_2 pour les deux agents ne font qu'augmenter et ne peuvent dépasser la fonction de valeur optimale. Par contre il est important de noter qu'il est possible que l'on ne converge pas vers la politique conjointe optimale, c'est le contre-coup de la simplification grace à laquelle on ne formule jamais la politique conjointe : comme on ne formule jamais non plus le problème avec deux agents, on n'a pas la garantie de converger vers la politique optimale conjointe. L'analyse de l'optimalité et de la ro-

bustesse de la solution trouvée par notre méthode est laissée dans la catégorie des travaux futurs.

L'exécution

Une fois que la meilleure politique a été choisie, chaque agent commence à en appliquer sa composante. Il conserve toutefois sa chronologie relative à la politique qu'il applique et à chaque étape de cette chronologie il calcule l'écart entre la situation réelle et la chronologie annoncée. Si cet écart est trop grand il envoie un message de replannification à son homologue.

Les causes de replannification

Comme on vient de le voir, une cause de replannification possible est un écart à la chronologie personnelle. En fait, c'est la cause essentielle mais un autre cas peut se présenter : si dans le cas présent c'est l'exécution du plan qui s'écartait ded la chronologie, on peut également rencontrer le cas d'un changement de chronologie en cours de mission. Cette possibilité repose dans une communication entre les agents dont la nature est liée à la mission. Dans le cas présent du drone hélicoptère et du robot terrestre, la communication d'information sur la nature du terrain, du drone vers le robot, implique une constante replannification du robot terrestre. Si, à un moment donné, le plan du robot terrestre change et que sa chronologie change aussi, ce dernier envoie alors au drone aérien un message de replannification afin de retrouver la politique optimale conjointe. Une dernière cause de replannification est l'aboutissement d'une chronologie : si l'exécution des politiques suit exactement la chronologie annoncée, alors quand les agents arrivent au bout de la chronologie, ils relancent un processus de plannification ensemble afin de retrouver la politique optimale et les chronologies associées. Il est ici important de noter que si les chronologies sont définies sur un horizon fini (plus précisément, elles définissent elles-mêmes une limite d'horizon), les politiques, elles, sont définies comme solutions optimales ou quasi-optimales du problème à horizon infini et que les agents pourraient continuer l'exécution de leur mission avec les politique finales sans que ça ne pose de réel problème (mais on perdrait l'avantage et la souplesse des replannifications dans le traitement du problème).

Le processus de replannification

Lorsqu'un agent reçoit un message de replannification de son homologue, celui-ci est accompagné d'une nouvelle chronologie. L'agent met alors à jour son MDP avec les informations de la chronologie, puis il le résout et génère une nouvelle politique optimale. Ce processus, comme dans le processus d'itération de la chronologie précédent, est plus rapide qu'une génération de la politique optimale en partant de zéro car l'agent initialise la politique avec sa politique courante qui est déjà proche de l'optimalité. Il crée alors sa chronologie et les itérations de la chronologie reprennent comme lors de la première phase de la mission. Une fois que les agents convergent sur une politique, l'exécution reprend.

La fin de la mission

La mission prend fin lorsque les agents ont pris toutes les récompenses possibles et effectuent la transition dans la dernière valeur de leur variable temporelle (et que l'action optimale est l'action *null*) ou alors quand les agents échouent définitivement. Pour un problème ayant des récompenses persistantes, le processus peut cependant continuer indéfiniement si on introduit bien la notion de fiabilité d'une chronologie (voir 3.5.3) afin de limiter la taille de la variable temporelle générée.

Chapitre 4

Conclusion

Les deux modèles développés dans cette étude, ainsi que l'algorithme d'itération de la chronologie, sont le résultat de plusieurs influences différentes. D'une part, l'acquisition de la « culture MDP », par la recherche dans la littérature des différentes approches, des bases, des extensions et des limites des modèles MDP. Cette recherche a débouché sur des approches plus hybrides, comme le graphe MAXQ, la décomposition automatique, les SMDP, et certaines manières de traiter des problèmes non-Markoviens. Par ailleurs, la problématique globale de l'étude sur laquelle était centré le stage - entre les problèmes de coopération en général, les problématiqeus multi-agents, et le cas appliqué du binôme aéro-terrestre - a amorcé une réflexion sur le sens d'une coopération dans le cadre de la décision en robotique.

La rencontre de ces différents aspects a donné naissance aux différents points de vue qui ont été décoloppés sur le sujet : les deux modèles présentés. La réflexion sur une manière de coopérer en échangeant le moins d'information possible s'est basée sur des exemple simples et il est juste de rendre aux jeux de cartes avec annonces le mérite d'avoir inspiré l'algorithme d'itération de la chronologie en soulignant les aspects essentiels d'une communication visant à effectuer une amélioration itérative d'une politique commune divisée en politiques individuelles et jamais fomulée explicitement.

Etant arrivé dans le domaine des MDPs sans expérience particulière des outils, le recul que m'a fourni le stage permet aujourd'hui d'entrevoir les améliorations, recentrages, affinements et évolutions que peuvent prendre les pistes que j'ai suivi dans une recherche - un peu naïve au début, et de plus en plus fine au fur et à mesure du stage - des solutions au problème théorique et pratique posé.

Dans les différentes parties de ce rapport, il est régulièrement fait allusion aux possibles améliorations du modèle, depuis l'intégration d'une décomposition automatique à l'utilisation d'une hiérarchisation à plus de deux niveaux des politiques et des états, en passant par la notion de fiabilité d'une chronologie ou de robustesse de l'algorithme développé. On peut donc conclure sur cette étude en définissant les grands axes envisageables pour la poursuivre :

- La synthèse, l'intégration et la maturation des techniques de résolution des MDP adaptées au cadre de notre problème,
- La preuve et, le cas échéant, les conditions d'optimalité et de robustesse de l'algorithme d'itération de la chronologie,
- L'implémentation des cas de tests,
- L'amélioration des modèles des systèmes coopératifs,

– L'amélioration de l'algorithme de résolution.

C'est sur ces cinq grands axes, auxquels vient se rajouter l'exploration de nouvelles perspectives proches de la problématique actuelle, que je souhaite poursuivre cette étude. Parmi les domaines particulièrement prometteurs dans le cadre multi-agents coopératifs, les modèles POMDPs (partially-observable MDPS) ont une place toute particulière en raison de leur traitement de la connaissance du système. En effet, grace à une approche POMDP, on peut représenter l'ignorance que l'on a, par exemple, de la position exacte de la récompense dans la grille que le drone aérien et le robot terrestre doivent explorer. Une autre approche qui a brièvement été abordée mais qui demande à être approfondie concerne les modèles d'apprentissage par renforcement. Un système hybride où les politiques locales sont apprises par renforcement (par essai-erreur) et sont ensuite combinées comme dans le graphe MAXQ ou les approches décomposées, pour former une politique globale que l'on souhaite optimale, un tel système hybride permettrait notamment un traitement plus simple dans l'algorithme de la chronologie et une automatisation de la construction de telles politiques permettrait d'avoir un système de résolution plus complet et plus efficace.

Pour conclure sur le travail effectué, il s'agit autant d'un travail de synthèse que d'une réflexion entamée sur les implications de la recherche de politiques optimales dans un cadre coopératif. Beaucoup de pistes ont été ouvertes et attendent d'être explorées. En attendant, la construction des modèles et algorithmes de résolution en contexte coopératif se sont révélés une recherche passionnante et particulièrement motivante pour la continuer en thèse !

Annexe A

Démonstration de la solution au problème du passage d'une distribution sur une grille de p éléments à une grille à k éléments

Problème vu page 74.

Un cas appliqué du problème se formule de la façon suivante :

Considérons une grille 3x3 *composée de cases pouvant recevoir au maximum une bille. Sachant qu'on connaît la distribution de probabilité* P_9 *sur le nombre de billes présentes dans la grille* 3x3*, quelle est la distribution* P_8 *sur le nombre de billes présentes dans la même grille à laquelle on enlève une case* a *?*

On veut donc connaître $P_8(n)$ la probabilité d'avoir n billes dans la grille à 8 cases que l'on note G_8. De même, on note G_9 la grille à 9 cases. On peut répartir les cas « n billes en G_8 » en deux catégories :

- 1 bille en a sachant qu'il y en a $n+1$ en G_9
- 0 bille en a sachant qu'il y en a n en G_9

On a donc :

$$P_8(n) = P(1 \text{ en } a | n+1 \text{ en } G_9)P_9(n+1) + P(0 \text{ en } a | n \text{ en } G_9)P_9(n) \tag{A.1}$$

Or :

$$P(1 \text{ en } a | n+1 \text{ en } G_9) = \frac{\begin{array}{c}\text{nombre de manières de mettre}\\ n \text{ billes dans 8 cases}\end{array}}{\begin{array}{c}\text{nombre de manières de mettre}\\ n+1 \text{ billes dans 9 cases}\end{array}} = \frac{\binom{8}{n}}{\binom{9}{n+1}} = \frac{n+1}{9} \tag{A.2}$$

Et :

$$P(0 \text{ en } a|n \text{ en } G_9) = \frac{\begin{array}{c}\text{nombre de manières de mettre}\\ n \text{ billes dans 8 cases}\end{array}}{\begin{array}{c}\text{nombre de manières de mettre}\\ n \text{ billes dans 9 cases}\end{array}} = \frac{\binom{8}{n}}{\binom{9}{n}} = \frac{9-n}{9} \tag{A.3}$$

Donc :

$$P_8(n) = \frac{n+1}{9}P_9(n+1) + \frac{9-n}{9}P_9(n) \tag{A.4}$$

L'équation A.4 établit le résultat utilisé plus haut à la page 74. On peut vérifier que la somme de $P_8(n)$ fait bien 1 :

$$\begin{aligned}
\sum_{n=0}^{8} P_8(n) &= \sum_{n=0}^{8} \frac{n+1}{9}P_9(n+1) + \frac{9-n}{9}P_9(n) \\
&= \frac{1}{9}\sum_{n=0}^{8} P_9(n+1) + \frac{1}{9}\sum_{n=0}^{8} n \cdot P_9(n+1) + \sum_{n=0}^{8} P_9(n) + \frac{1}{9}\sum_{n=0}^{8} n \cdot P_9(n) \\
&= \frac{1}{9}(1 - P_9(0)) + 1 - P_9(9) + \frac{1}{9}\left[\sum_{n=1}^{9}(n-1)\cdot P_9(n) - \sum_{n=0}^{8} n \cdot P_9(n)\right] \\
&= \frac{1}{9}(1 - P_9(0)) + 1 - P_9(9) + \frac{1}{9}\left[-\sum_{n=1}^{8} P_9(n) + 8P_9(9)\right] \\
&= \frac{1}{9}(1 - P_9(0)) + 1 - P_9(9) + \frac{1}{9}\left[P_9(0) + P_9(9) - 1 + 8P_9(9)\right] \\
&= 1
\end{aligned}$$

On peut généraliser ce résultat au cas d'une grille de n cases que l'on note G_n :

On cherche à calculer $P_{n-1}(k)$, comme précédemment, on a les deux catégories :
- 1 bille en a sachant qu'il y en a $k+1$ en G_n
- 0 bille en a sachant qu'il y en a k en G_n

On a donc :

$$P_{n-1}(k) = P(1 \text{ en } a|k+1 \text{ en } G_n)P_n(k+1) + P(0 \text{ en } a|k \text{ en } G_n)P_n(k) \tag{A.5}$$

Or :

$$P(1 \text{ en } a | k+1 \text{ en } G_n) = \frac{\begin{array}{c}\text{nombre de manières de mettre} \\ k \text{ billes dans } n-1 \text{ cases}\end{array}}{\begin{array}{c}\text{nombre de manières de mettre} \\ k+1 \text{ billes dans } n \text{ cases}\end{array}} = \frac{\binom{n-1}{k}}{\binom{n}{k+1}} = \frac{k+1}{n} \tag{A.6}$$

Et :

$$P(0 \text{ en } a | k \text{ en } G_n) = \frac{\begin{array}{c}\text{nombre de manières de mettre} \\ k \text{ billes dans } n-1 \text{ cases}\end{array}}{\begin{array}{c}\text{nombre de manières de mettre} \\ k \text{ billes dans } n \text{ cases}\end{array}} = \frac{\binom{n-1}{k}}{\binom{n}{k}} = \frac{n-k}{n} \tag{A.7}$$

Donc :

$$P_{n-1}(k) = \frac{k+1}{n} P_n(k+1) + \frac{n-k}{n} P_n(k) \tag{A.8}$$

On peut enfin chercher à obtenir une formule explicite à partir de cette définition récursive. On va démontrer par récurrence que :

$$\begin{gathered} \forall (n,p,k) \in \mathbb{N}^3, \quad p < n, \quad k \leq n-p \\ P_{n-p}(k) = \binom{n-p}{k} \sum_{i=0}^{p} \frac{\binom{p}{i}}{\binom{n}{k+p-i}} P_n(k+p-i) \end{gathered} \tag{A.9}$$

Pour $p = 0$ on a :

$$\begin{aligned} P_{n-0}(k) &= \binom{n-0}{k} \sum_{i=0}^{0} \frac{\binom{0}{i}}{\binom{n}{k+0-i}} P_n(k+0-i) \\ P_{n-0}(k) &= \frac{\binom{n}{k}}{\binom{n}{k}} P_n(k) \\ P_{n-0}(k) &= P_n(k) \end{aligned}$$

La relation est encore vérifiée pour $p = 1$, c'est la relation de récurrence démontrée plus haut :

$$P_{n-1}(k) = \binom{n-1}{k} \sum_{i=0}^{1} \frac{\binom{1}{i}}{\binom{n}{k+1-i}} P_n(k+1-i)$$

$$P_{n-1}(k) = \binom{n-1}{k} \left[\frac{\binom{1}{0}}{\binom{n}{k+1}} P_n(k+1) + \frac{\binom{1}{1}}{\binom{n}{k}} P_n(k) \right]$$

$$P_{n-1}(k) = \binom{n-1}{k} \left[\frac{1}{\binom{n}{k+1}} P_n(k+1) + \frac{\binom{1}{1}}{\binom{n}{k}} P_n(k) \right]$$

$$P_{n-1}(k) = \frac{\binom{n-1}{k}}{\binom{n}{k+1}} P_n(k+1) + \frac{\binom{n-1}{k}}{\binom{n}{k}} P_n(k)$$

$$P_{n-1}(k) = \frac{k+1}{n} P_n(k+1) + \frac{n-k}{n} P_n(k)$$

On suppose le résultat vrai pour $p-1$, soit :

$$P_{n-p+1}(k) = \binom{n-p+1}{k} \sum_{i=0}^{p-1} \frac{\binom{p-1}{i}}{\binom{n}{k+p-1-i}} P_n(k+p-1-i)$$

D'après l'équation A.8, on a :

$$P_{n-p}(k) = \frac{k+1}{n-p+1} P_{n-p+1}(k+1) + \frac{n-p+1-k}{n-p+1} P_{n-p+1}(k)$$

En remplaçant, on obtient :

$$\begin{aligned} P_{n-p}(k) = {} & \frac{k+1}{n-p+1} \binom{n-p+1}{k+1} \sum_{i=0}^{p-1} \frac{\binom{p-1}{i}}{\binom{n}{k+p-i}} P_n(k+p-i) \\ & + \frac{n-p+1-k}{n-p+1} \binom{n-p+1}{k} \sum_{i=0}^{p-1} \frac{\binom{p-1}{i}}{\binom{n}{k+p-1-i}} P_n(k+p-1-i) \end{aligned}$$

Soit :

$$P_{n-p}(k) = \frac{k+1}{n-p+1}\binom{n-p+1}{k+1}\sum_{i=0}^{p-1}\frac{\binom{p-1}{i}}{\binom{n}{k+p-i}}P_n(k+p-i)$$
$$+\frac{n-p+1-k}{n-p+1}\binom{n-p+1}{k}\sum_{i=1}^{p}\frac{\binom{p-1}{i-1}}{\binom{n}{k+p-i}}P_n(k+p-i)$$

Donc :

$$P_{n-p}(k) = \binom{n-p}{k}\sum_{i=0}^{p-1}\frac{\binom{p-1}{i}}{\binom{n}{k+p-i}}P_n(k+p-i)$$
$$+\frac{n-p+1-k}{n-p-k+1}\binom{n-p}{k}\sum_{i=1}^{p}\frac{\binom{p-1}{i-1}}{\binom{n}{k+p-i}}P_n(k+p-i)$$

Que l'on factorise en :

$$P_{n-p}(k) = \binom{n-p}{k}\left[\frac{\binom{p-1}{0}}{\binom{n}{k+p}}P_n(k+p)+\frac{\binom{p-1}{p-1}}{\binom{n}{k}}P_n(k)\right.$$
$$\left.+\sum_{i=1}^{p-1}\left(\frac{\binom{p-1}{i}}{\binom{n}{k+p-i}}+\frac{\binom{p-1}{i-1}}{\binom{n}{k+p-i}}\right)P_n(k+p-i)\right]$$

On a :

$$\binom{p-1}{i}+\binom{p-1}{i-1}=\binom{p}{i}$$
$$\frac{\binom{p-1}{0}}{\binom{n}{k+p}}=\frac{\binom{p}{0}}{\binom{n}{k+p}}$$
$$\frac{\binom{p-1}{p-1}}{\binom{n}{k}}=\frac{\binom{p}{p}}{\binom{n}{k+p-p}}$$

Donc :

$$P_{n-p}(k) = \binom{n-p}{k} \sum_{i=0}^{p} \frac{\binom{p}{i}}{\binom{n}{k+p-i}} P_n(k+p-i)$$

Ce qui achève la récurrence et démontre le résultat.

Annexe B

Complexité de l'algorithme d'itération de la valeur

D'après [Put94] (page 163, théorème 6.3.3.d), lors d'une execution d'un algorithme d'itération de la valeur, si le facteur d'oubli est noté γ, si V^n désigne le résultat de la $n^{\text{ième}}$ itération et V^* désigne la fonction de valeur, alors on a :

$$\forall n \in \mathbb{N}, \quad \| V^n - V^* \| \leq \frac{\gamma^n}{1-\gamma} \| V^1 - V^0 \|$$

On veut trouver une approximation ϵ-optimale de la fonction de valeur optimale. On aura la garantie d'être ϵ-optimal si on a :

$$\frac{\gamma^n}{1-\gamma} \| V^1 - V^0 \| < \epsilon$$

En posant $\| V^1 - V^0 \| = K$, cela signifie que :

$$n < \frac{\ln(\epsilon(1-\gamma)) - \ln(K)}{\ln(\gamma)}$$

Cela nous fournit un majorant du nombre d'itérations qu'il faut effectuer afin de trouver une fonction de valeur qui soit ϵ. De l'expression précédente, on ne retient que la partie qui ne dépend pas de l'initialisation et on considère que l'ordre de grandeur du nombre d'itérations à effectuer pour trouver une solution ϵ-optimale est $\frac{\ln(\epsilon(1-\gamma))}{\ln(\gamma)}$.

Enfin, dans chaque boucle de l'algorithme d'itération de la valeur, on effectue une recherche pour tout $s \in S$, on y calcule un maximum sur les $a \in A$, et l'expression que l'on cherche à maximiser comporte une somme sur les $s \in S$. Le test afin de savoir si l'algorithme doit s'arrêter ne prend qu'une opération. La complexité d'une boucle est donc en $O(|S|^2|A|)$.

La complexité globale de l'algorithme est donc en $O(|S|^2|A|\frac{\ln(\epsilon(1-\gamma))}{\ln(\gamma)})$.

Table des figures

1.1 Le drone hélicoptère Yamaha R-MAX utilisé à l'ONERA 9

2.1 Algorithme d'itération de la valeur 19
2.2 Algorithme d'itération de la valeur amélioré (Gauss-Seidel) 19
2.3 Algorithme d'itération de la politique 20
2.4 Algorithme d'itération de la politique modifié 21
2.5 Illustration de l'espace d'état d'un MDP local 29
2.6 Evolution de la fonction de valeur de la politique dominante 33
2.7 Algorithme de construction d'un cache ϵ-optimal minimal 34
2.8 Illustration de la démonstration du lemme d'encadrement de la fonction de valeur 35
2.9 Illustration du théorème de l'enveloppe convexe 36
2.10 Un graphe MAXQ pour le problème du taxi 39

3.1 Un exemple de grille pour le problème du taxi 51
3.2 Probabilités de transition pour les actions de déplacement — Variables concernées : P, Fuel. 52
3.3 Probabilités de transition pour l'action Embarquer — Variables concernées : Pass_in, Pass_att, P, Fuel 52
3.4 Probabilités de transition pour l'action Débarquer — Variables concernées : Pass_in, P, Fuel 52
3.5 Probabilités de transition pour l'action Plein — Variables concernées : P, Fuel . . 53
3.6 Représentation compacte des récompenses pour le problème du taxi 53
3.7 Probabilités sur les états d'arrivée d'une action E 73
3.8 Probabilités sur les états d'arrivée d'une action E 78
3.9 Arbre de transition de la variable x'_h pour l'action N du drone hélicoptère 83
3.10 Arbre de transition de la variable y'_h pour l'action N du drone hélicoptère 84
3.11 Arbre de transition de la variable ev'_1 pour l'action N du drone hélicoptère . . . 84
3.12 Arbre de transition de la variable ev'_6 pour l'action N du drone hélicoptère . . . 85
3.13 Arbre de transition de la variable f'_h pour l'action N du drone hélicoptère 85
3.14 Arbre de transition de la variable sol' pour l'action N du drone hélicoptère . . . 85
3.15 Arbre de transition de la variable rt' pour l'action N du drone hélicoptère 86
3.16 Arbre de transition de la variable $|ev|'$ pour l'action N du drone hélicoptère . . . 87
3.17 Arbre de transition de la variable $|em|'$ pour l'action N du drone hélicoptère . . 88
3.18 Arbre de coût par unité de temps pour l'action N du drone hélicoptère 88
3.19 Arbre de récompense instantanée pour l'action N du drone hélicoptère 89
3.20 Arbre des durées de transition pour l'action N du drone hélicoptère 89
3.21 Arbre de transition de la variable f'_h pour l'action *emettre* du drone hélicoptère . 89
3.22 Arbre de transition de la variable em' pour l'action *emettre* du drone hélicoptère 89
3.23 Arbre de récompense pour l'action *emettre* du drone hélicoptère 90
3.24 Arbre des durées de transition pour l'action *emettre* du drone hélicoptère 90

3.25 Arbre de transition de la variable f'_h pour l'action *poser* du drone hélicoptère . . 91
3.26 Arbre de transition de la variable sol' pour l'action *poser* du drone hélicoptère . 91
3.27 Arbre des durées de transition pour l'action *poser* du drone hélicoptère 91
3.28 Arbre de récompense pour l'action *poser* du drone hélicoptère. 92

Bibliographie

[BDG99] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations, 1999.

[BDM+02] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty : A challenge for ai. In *18th Conference on Uncertainty in Artificial Intelligence*, 2002.

[Bel57] R.E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[BM04] Aurélie Beynier and Abdel-Illah Mouaddib. Decentralized markov decision processes for handling temporal and resource constraints in a multiple robot system. In *7th International Symposium on Distributed Autonomous Robotic System*, 2004.

[BM05] Aurélie Beynier and Abdel-Illah Mouaddib. Processus décisionnels de markov décentralisés et interactifs avec des contraintes temporelles. *Journal électronique d'intelligence artificielle*, 4(22), 2005.

[Bou96] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

[Die98] Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 118–126. Morgan Kaufmann, San Francisco, CA, 1998.

[Die00] Thomas G. Dietterich. State abstraction in maxq hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 12*, pages 994–1000, 2000.

[DL95] T. Dean and S. Lin. Decomposition techniques for planning in stochastic domains, 1995.

[FDMW04] Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic programming for structured continuous markov decision problems. In *20th Conference on Uncertainty in Artificial Intelligence*, pages 154–161, 2004.

[FFG01] Patrick Fabiani, Jean-Loup Farges, and Frédérick Garcia. *Decision dans l'incertain.* Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, 2001.

[GKP01] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. In *In 14th Neural Information Processing Systems (NIPS-14)*, 2001.

[GMM03] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. Extending hierarchical reinforcement learning to continuous-time, average-reward, and multi-agent models. Technical Report UM-CS-2003-23, Department of Computer Science, University of Massachusetts Amherst, 2003.

[HMK+98] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Uncertainty in Artificial Intelligence*, pages 220–229, 1998.

[HSAHB00] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutillier. Optimal and approximate stochastic planning using decision diagrams, 2000.

[Mou04] Abdel-Illah Mouaddib. Co-operative scheduling for a resource bounded multiagent planning system. *Journal of Experimental & Theoretical Artificial Intelligence*, 16(2) :57–71, 2004.

[Par98] R. Parr. Flexible decomposition algorithms for weakly coupled markov decision problems, 1998.

[Put94] Martin L. Puterman. *Markov Decision Processes*. John Wiley & Sons, Inc, 1994.

[Sab02] Régis Sabbadin. Graph partitioning techniques for markov decision processes decomposition. In *ECAI*, pages 670–674, 2002.

[SPS99] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs : A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2) :181–211, 1999.

[Sut95] Richard S. Sutton. TD models : Modeling the world at a mixture of time scales. In *International Conference on Machine Learning*, pages 531–539, 1995.

[WB93] Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, College of Computer Science, Northeastern University, Boston, Massachussetts, 1993.