وزارة التعليـــم العـــالـي والبـحـث العلمـــي

**BADJI MOKHTAR-ANNABA UNIVERSITY**
**UNIVERSITÉ BADJI MOKHTAR-ANNABA**

**جامعة باجي مختار– عنابـــــــــة**

**Faculté des Sciences de l'Ingéniorat** **Année: 2016**

**Département d'Informatique**

# THESE

**Présentée en vue de l'obtention**
**du diplôme de Doctorat 3ème Cycle**

---

# *Contribution à la Résolution des Processus Décisionnels de Markov Décentralisés*

---

Filière: **Informatique**

Option
**Sciences et Technologies de l'Information et de la Communication**

Par
**Melle. Hiba Abdelmoumène**

**Devant le Jury :**

| | | | |
|---|---|---|---|
| Directeur de thèse : | **Mme. Habiba BELLEILI-SOUICI** | MCA | Université Badji Mokhtar-Annaba |
| Président : | **Mme. Labiba SOUICI-MESLATI** | Prof | Université Badji Mokhtar-Annaba |
| Examinateurs : | **Mme. Yamina MOHAMED BEN ALI** | Prof | Université Badji Mokhtar-Annaba |
| | **Mr. Hamid SERIDI** | Prof | Université 8 Mai 1945-Guelma |

# *Remerciement*

*Mes remerciements s'adressent tout d'abord, à ma directrice de thèse, madame Habiba BELLEILI-SOUICI. Je vous remercie de m'avoir initiée à ce domaine de recherche passionnant, d'avoir cru en mes capacités, pour le temps et la patience que vous m'avez accordés tout au long de ces années. Vos encouragements, conseils et remarques m'ont permis de progresser et de mener à bon terme mon travail de thèse. Je tiens à vous exprimer à travers ces quelques mots ma profonde gratitude et mon très grand respect.*

*Je tiens à remercier également madame Labiba SOUICI-MESLATI, Professeur à l'université Badji Mokhtar – Annaba, de m'avoir fait l'honneur d'accepter de présider le jury de cette thèse. Mes remerciements vont également à madame Yamina MOHAMED BEN ALI, Professeur à l'université Badji Mokhtar – Annaba et à monsieur Hamid SERIDI, Professeur à l'université 8 Mai 1945 – Guelma qui ont accepté d'être examinateurs de ma thèse et d'évaluer ainsi mon travail.*

*Je tiens aussi à exprimer mes chaleureux remerciements à ma mère, mon père et mes sœurs qui m'ont constamment encouragée et soutenue tout au long de ces années.*

*Un grand merci à mes cousines et à mes amies de m'avoir rappelée de temps à autre qu'on peut faire de la recherche sans pour autant oublier de savourer le goût de la vie.*

*Enfin, je remercie toutes les personnes qui ont participé de près ou de loin au bon déroulement de cette thèse.*

*Hiba Abdelmoumène*

# *RÉSUMÉ*

De nombreuses applications du monde réel impliquent de multiples agents qui agissent ensembles d'une manière coopérative, sous pression de temps et sous incertitude. Les modèles les plus appropriés capables de traiter ces problèmes sont ceux issus de la planification de la théorie de la décision. Ces modèles sont très expressifs et sont capables de raisonner sur le gain des actions au fil du temps et en présence d'incertitude.

Dans cette thèse, nous nous intéressons particulièrement aux Processus Décisionnels de Markov Décentralisés (*Decentralized Markov Decision Processes, DEC-MDPs*), qui sont une extension du Processus Décisionnel de Markov (*Markov Decision Process, MDP*) au cas multi-agents. Cependant, l'utilisation de ces modèles pour la planification des actions dans le cadre des applications réelles reste limitée. Cette limitation est due au fait que le modèle de base des DEC-MDPs ne permet pas la prise en compte de contraintes d'exécution des actions. En outre, la complexité de leur résolution limite leur utilisation à de petits problèmes.

L'objectif du travail que nous présentons dans cette thèse est d'adapter le modèle DEC-MDP afin de prendre en considération les contraintes sur l'exécution des actions et proposer ainsi une modélisation adéquate du temps et des actions.

Nous nous sommes intéressées, dans un premier temps, à enrichir le modèle MDP encapsulé par un agent unique afin de prendre en considération les contraintes temporelles, les contraintes de précédences et des durées incertaines d'exécution des actions.

Dans un second temps, nous avons étudié les MDPs Décentralisés où chaque MDP est encapsulé par un agent. Particulièrement, nous nous sommes intéressées à une sous-classe des DEC-MDPs pour la gestion des contraintes temporelles et de ressources, OC-DEC-MDP (*Opportunity Cost Decentralized Markov Decision Process*). Malgré que ce modèle de l'état de l'art soit approprié pour la prise en compte de différents types de contraintes et permet le passage à l'échelle, il souffre de flexibilité ce qui limite son applicabilité. Notre objectif consistait, alors, à étendre l'OC-DEC-MDP ainsi que son algorithme de résolution afin de modéliser des contraintes d'ordre plus complexes entre les actions permettant ainsi une meilleure flexibilité.

La flexibilité n'est pas sans prix. En effet, des situations de mauvaise coordination (*mis-coordination)* se posent. Pour y remédier, nous avons proposé d'introduire la communication entre les agents. Comme l'introduction de la communication induit des coûts et une augmentation exponentielle de la taille du problème, nous avons construit des heuristiques qui évaluent l'impact des points de communication possibles. Les heuristiques permettent de guider le calcul de la politique jointe afin de trouver un compromis entre la qualité de la solution et la taille du problème. Par la suite, nous avons étudié la fiabilité de la communication en proposant un mécanisme qui permet la gestion des pertes de messages dans le cas de la communication stochastique.

# *MOTS CLÉS*

Planification sous incertitude, Processus Décisionnel de Markov Décentralisé, Contraintes d'exécution, Communication.

# *ABSTRACT*

Many real world applications involve multiple agents acting together as a team under time pressure and uncertainty. The more suitable models capable of handling such problems are those studied in decision theoretic planning. These models are very expressive and are able to reason about the gain of action over time in the presence of uncertainty.

In this thesis, we are particularly interested in Decentralized Markov Decision Processes (DEC-MDPs), which are an extension of Markov Decision Process (MDP) to multi-agent case. Even if DEC-MDPs describe a powerful framework for cooperative multi-agent decision, they fail to formalize constraints on task execution. Furthermore, the complexity of their resolution is such that it is difficult to determine an optimal solution except for small problems.

The work we present in this thesis aims to adapt the DEC-MDP model to provide adequate modeling of time and actions, and to allow the representation of real problems.

We are interested, at first, to enrich the single agent MDP model to consider temporal constraints, precedence constraints and uncertain actions' durations.

Secondly, we studied the Decentralized MDP where each MDP is encapsulated by an agent. Particularly, we are interested in a subclass of DEC-MDPs which is capable of addressing temporal and resource constraints, OC-DEC-MDP (Opportunity Cost Decentralized Markov Decision Process). Although this state of the art model is suitable for considering different types of constraints and allows scalability, it suffers from flexibility which limits its applicability. Our objective is, then, to extend the OC-DEC-MDP and its resolution algorithm to handle more complex order constraints between actions and hence allowing more flexibility.

The flexibility is not free. Indeed, situations of mis-coordination may occur. To remedy to that, we have proposed to introduce communication between agents. Since communication induces cost and an exponential increase in the problem size, we have developed heuristics that assess the impact of potential communication points. The reason behind adopting these heuristics is to guide the computation of the joint policy in order to find a tradeoff between the solution quality and problem size. Thereafter, we enhanced our proposition to deal with message loss in the case of stochastic communication.

# *KEYWORDS*

Planning under uncertainty, Decentralized Markov Decision Process, execution constraints, communication.

# ملخص

عدّة تطبيقات تتطلب مشاركة عدّة عملاء أذكياء يعملون معا، تحت ضغط الوقت و عدم اليقين (الشك). النماذج المناسبة للتعامل مع مثل هذه المشاكل هي التي يقدمها مجال التخطيط في ظل عدم اليقين لأخذ القرارات. هذه النماذج تسمح بالتفكير على الربح الناتج عن تنفيذ المهام مع مرور الوقت و بحضور عدم اليقين.

نهتم في هذه الأطروحة بنظام ماركوف للقرارات اللامركزية و هو امتداد لنظام ماركوف للقرارات في حالة نظام العملاء الأذكياء.

استخدام هذه الأنظمة لجدولة المهام في تطبيقات حقيقية يطرح بعض الصعوبات فليس هناك أخذ بعين الاعتبار القيود على تنفيذ المهام، كما أن حل هذه الأنظمة يعتبر صعب جدا مما يحدد تطبيقها للمشاكل الصغيرة.

العمل الذي نقدمه في هذا البحث يهدف إلى تطوير هذه الأنظمة: نظام ماركوف للقرارات في حالة وجود عميل واحد و نظام ماركوف للقرارات اللامركزية في حالة وجود عدة عملاء.

قمنا أيضا باقتراح طريقة للاتصال بين العملاء لتفادي حالات سوء التنسيق الناتجة عن الترتيب الجزئي لمهام العملاء و تبنينا بعض الاستدلالات لايجاد حل مشترك بين نوعية الحل و حجم المشكلة. تمت أيضا دراسة موثوقية الاتصالات من خلال توفير آلية لإدارة فقدان الرسائل.

# *كلمات مرشدة*

التخطيط في ظل عدم اليقين ، نظام ماركوف للقرارات اللامركزية، قيود التنفيذ ،الاتصال .

# *Table des Matières*

# *Liste des Figures*

# *Liste des Tableaux*

# Chapitre 1

# *Introduction Générale*

Une caractéristique fondamentale de tout système intelligent, naturel ou artificiel, est la capacité de prendre une décision rationnelle face à un ensemble de choix. Les êtres humains sont confrontés à des millions de décisions à prendre chaque jour. Certaines sont évidentes, d'autres peuvent être le résultat de plusieurs années de raisonnement. Un problème de décision couvre le processus de sélection de la bonne action compte tenu des informations disponibles.

Cette tâche, prise de décision, qui parait facile peut être très difficile à réaliser. En effet, les décisions ne doivent pas être prises isolément, les décisions d'aujourd'hui ont un impact sur les décisions de demain et celles de demain sur les décisions du lendemain. En ne tenant pas compte de la relation entre les décisions courantes et futures et les résultats actuels et les résultats futurs, nous ne pouvons pas parvenir à une bonne performance globale. Cet aspect de séquentialité rend le processus de décision plus complexe.

L'automatisation du processus de prise de décision est un domaine de recherche très actif. La motivation est que, en équipant le décideur (ou agent) avec des ressources de calcul et en développant des algorithmes de raisonnement efficaces pour sa décision, nous pouvons réaliser des systèmes extrêmement bénéfiques. Par exemple, les agents peuvent aider les médecins et les infirmières dans les unités de soin intensif de l'hôpital pour prendre des décisions rapides sur les options du traitement, les véhicules autonomes sous pilote peuvent livrer des fournitures de secours et de recherche de survivants à la suite d'une catastrophe naturelle, …

Dans chacun de ces domaines, différents niveaux d'incertitude émergent. Le premier niveau concerne les conséquences des décisions prises par un agent. En d'autres termes,

l'agent ne peut pas prévoir les issues de ses décisions de façon déterministe. C'est le cas d'actions dites stochastiques où les conséquences de chaque action sont représentées par une distribution de probabilités. Le deuxième niveau d'incertitude est du à l'observabilité partielle de l'état du problème nécessitant de maintenir un état interne de croyance pour la prise de décision. Une autre source d'incertitude est liée à la présence de plusieurs agents. Cela est d'autant plus vrai lorsque la performance de la décision d'un agent dépend des décisions des autres agents qui ne sont pas nécessairement prévisibles.

## 1.1 Domaine de recherche

Le premier but de l'Intelligence Artificielle (IA) consistait à développer un agent – que nous appellerons décideur – qui prend des décisions optimales en réponse à des changements dans un environnement qui ne contient pas d'autres agents. Cependant, les agents sont de plus en plus déployés dans des environnements contenant d'autres agents, que ce soit des agents logiciels ou physiques comme des robots.

Notre décideur affecte donc ces agents et est réciproquement affecté par eux. En conséquence, il ne peut plus raisonner sur ses décisions isolément; il doit tenir compte des autres agents et leurs influences sur l'environnement et sur chacun d'entre eux. Cette contrainte augmente le besoin de trouver des méthodes pour représenter et raisonner sur un tel problème difficile qui représente un grand challenge de l'IA et qui est la prise de décision multi-agents. De plus, l'aspect séquentiel du problème où la décision courante est susceptible d'influencer toutes les décisions futures complique encore le problème.

Dans cette thèse, nous nous intéressons aux problèmes de prise de décision séquentielle décentralisée (multi-agents) dans lesquels les agents opèrent dans des environnements incertains et prennent des décisions ayant des effets stochastiques à long et à court terme. Il est apparu que les décisions prises par une entité appartenant à un tel système (système multi-agents) sont complexifiées du fait des interactions. Il devient alors plus difficile, à partir des décisions individuelles, d'obtenir un comportement collectivement intelligent (et donc performant).

Ce problème va constituer le cœur de notre travail. Nous allons plus particulièrement chercher à élaborer un processus permettant à des entités de prendre, individuellement et de manière autonome, des décisions de façon à obtenir un comportement collectif rationnel. Nous développerons donc une approche permettant à chaque agent de prendre des décisions de manière autonome de façon à maximiser les performances de l'ensemble.

La planification de la théorie de la décision (*Decision Theoretic Planning*, DTP) offre des outils très puissants qui permettent de modéliser de tels problèmes. Les processus décisionnels de Markov (*Markov Decision Processes*, MDPs), les processus décisionnels de Markov partiellement observables (*Partially Observable Markov Decision Processes*, POMDPs) et les processus décisionnels de Markov partiellement observables décentralisés (*Decentralized Partially Observable Markov Decision Processes*, DEC-POMDPs) en sont des exemples.

Afin de gérer les incertitudes liées à la planification des tâches d'une équipe d'agents en vue de la prise de décisions optimales, nous ferons appel aux Processus Décisionnels de Markov Décentralisés (DEC-MDPs), une extension des Processus Décisionnels de Markov (MDPs) au cadre multi-agents.

## 1.2 Problématique

Comme nous venons de l'exposer, les recherches en IA se sont beaucoup orientées vers l'élaboration d'entités intelligentes en vue de la réalisation de tâches dangereuses, ou impossibles pour l'homme. Cependant, ces applications présentent, en plus de l'incertitude, plusieurs d'autres contraintes. Ces contraintes peuvent être liées au temps (des agents pompiers doivent éteindre le feu le plus rapidement que possible avant qu'il évolue), aux ressources (un robot doit bien gérer le niveau de sa batterie pour accomplir ses tâches), …

Bien que les DEC-MDPs permettent la modélisation de problèmes décisionnels décentralisés sous incertitude, leur utilisation pour la résolution de problèmes réels présentant des contraintes peut s'avérer difficile. Les DEC-MDPs, dans leur version originale, ne permettent pas, en effet, la prise en compte des contraintes sur l'exécution

des actions. Leur représentation des actions ne permet pas la modélisation du temps ni la prise en considération des durées des actions (exécution instantanée et d'une manière synchrone). La gestion des interactions et les contraintes de précédence entre les actions des agents n'est pas possible dans le modèle tel qu'il est défini. De plus, la complexité de leur résolution rend le calcul d'une solution optimale très difficile.

Nous chercherons donc tout d'abord à adapter ce modèle afin de prendre en compte des contraintes sur l'exécution des actions, à savoir, les contraintes temporelles et les durées incertaines d'exécution des actions. En outre, nous nous concentrons sur des situations où il existe un certain degré de dépendance entre les sous-problèmes des agents. Plus précisément, nous nous concentrons sur des situations où les interactions entre agents sont structurées et représentées par des relations de précédences.

Dans les applications du monde réel et en conséquence de la contrainte de temps et de la relation de précédence entre les actions, nous ne pouvons pas ignorer le besoin pour la communication. Dans un second temps, nous nous intéressons à l'implication de la communication lors de la prise de décision décentralisée afin d'alléger le problème en exploitant les interactions existantes entre les agents et tenant compte du coût de la communication.

## 1.3 Contributions

L'objectif principal de cette thèse consiste à adapter le modèle des DEC-MDPs afin de proposer une modélisation adéquate du temps et des actions et de permettre la représentation de problèmes réels.

Dans un premier temps, nous nous sommes intéressées à l'adaptation du modèle MDP mono-agent pour prendre en considération des contraintes temporelles sur l'exécution des actions, des contraintes de précédence et des durées des actions incertaines. La formalisation que nous avons proposée implique la construction explicite des composants du MDP [Abdelmoumène and Belleili, 2011, 2012]. Le but derrière cette modélisation est d'obtenir un plan de tâche robuste qui est capable de mener à bien la mission en dépit d'incertitude.

Comme ça était mentionné ci-dessus, la complexité des DEC-MDPs rend leur usage dans des applications réelles telles que l'exploration planétaire, la gestion des désastres, …, très limité. Afin de faire face à cette complexité, les chercheurs du domaine ont opté pour des hypothèses relaxantes et ainsi ils ont défini des sous-classes des DEC-MDPs où la structure du problème est simplifiée. Ces sous-classes sont principalement dérivées en se basant sur les dépendances entre les agents.

Parmi les sous-classes des DEC-MDPs, nous nous sommes intéressées principalement au modèle OC-DEC-MDP (*Opportunity Cost* DEC-MDP) qui est capable d'adresser les contraintes temporelles, les contraintes de précédence et les durées incertaines des actions. Cependant, ce modèle n'est adapté que pour le cas où les actions de chaque agent sont représentées par un plan linéaire. Notre seconde contribution a pour objectif l'obtention d'une forme de flexibilité en proposant une extension du modèle OC-DEC-MDP où les actions de chaque agent sont représentées par des plans partiels, sous forme de graphe ET/OU. De cette extension ont émergé d'autres problèmes de décision liés à la nouvelle modélisation des plans locaux des agents.

De plus, d'autres problèmes liés à la mis-coordination entre les agents résultant de l'ordre partiel des actions émergent. Pour pallier à ces problèmes nous avons proposé d'introduire la communication entre les agents. Comme la communication est coûteuse en terme de calcul et en terme de coût opérationnel, notre deuxième objectif était de proposer un riche modèle de communication entre agents afin de faire face aux problèmes de mis-coordination et en même temps limiter son usage. Afin de limiter la décision de la communication, nous avons élaboré des heuristiques sur le moment de communication (quand communiquer), la sémantique des messages partagés (quoi communiquer) et avec qui communiquer. Ces heuristiques ont été dérivées en exploitant la structure spécifique du problème considéré aidé par la relation de précédence [Abdelmoumène and Belleili, 2014b].

Par la suite nous avons étudié la résolution du DEC-MDP défini en procédant à la planification des actions de chaque agent en vue d'une prise de décision décentralisée et autonome en respectant les contraintes du problème. Notre objectif consistait à la

maximisation du gain cumulé en trouvant un compromis entre le coût de la communication, la perte en utilité locale et le gain en utilité globale.

Notre dernière contribution concerne la fiabilité de la communication. En effet, dans un premier temps, nous avons supposé dans un but de relaxation du problème, que les messages sont toujours reçus avec succès. Par la suite, nous avons injecté une nouvelle source d'incertitude relative à la communication en la considérant stochastique. Nous avons proposé un mécanisme qui permet de prendre en compte les messages perdus lors de la prise de décision [Abdelmoumène and Belleili, 2016].

## 1.4 Organisation de la thèse

Le présent document est organisé en six chapitres :

**Chapitre 2**, dans ce chapitre nous donnons les bases de la théorie de la décision, ainsi que les modèles de la théorie de la décision pour le cas mono-agent qui sont les processus décisionnels de Markov et les processus décisionnels de Markov partiellement observables et leurs algorithmes de résolution.

**Chapitre 3**, porte sur les modèles de la théorie de la décision quand plusieurs agents agissent sur l'environnement afin d'accomplir une mission d'une manière coopérative. Les processus décisionnels de Markov décentralisés partiellement observables ainsi que leurs différentes sous-classes et algorithmes de résolution sont présentés dans ce chapitre.

Dans le **chapitre 4** nous passons en revue les différents travaux existants dans la littérature et qui sont en liaison avec notre travail. Ces travaux portent sur les contraintes d'exécution dans les problèmes de contrôle décentralisé et la communication dans les modèles de la théorie de la décision**.**

**Chapitre 5**, dans ce chapitre nous présentons notre première contribution dont le but est la modélisation des contraintes temporelles, des contraintes de précédence et des durées incertaines des actions dans le modèle de base MDP.

**Chapitre 6**, dans ce chapitre nous proposons une extension d'un modèle de l'état de l'art afin de gérer des contraintes plus complexes ainsi que les décisions de la communication. Nous proposons un riche modèle de communication approprié aux types de problèmes que nous traitons dans cette thèse. Nous présenterons dans ce chapitre aussi les résultats expérimentaux destinés à mesurer l'impact de notre proposition sur les performances des agents construits selon le modèle proposé.

Nous conclurons enfin en tirant le bilan des travaux exposés dans cette thèse et présenterons différentes perspectives ouvertes par ce travail.

# Chapitre 2

# *Prise de décision séquentielle sous incertitude: cas mono-agent*

*La décision séquentielle a pour but de déterminer quelles actions les agents doivent entreprendre pour atteindre un but donné, en particulier lorsque les effets des actions sont stochastiques ou incertains. Son étude a des applications dans un grand nombre de disciplines telles que la robotique, l'automatique, la télécommunication, l'économie et l'éthologie (comportement des animaux et en particulier l'apprentissage), etc.*

*De manière générale, résoudre un problème de prise de décision séquentielle demande de prévoir les conséquences à long terme des actions et de planifier "un chemin", c'est-à-dire une séquence d'actions, vers une solution. A un instant donné $t$, le système est dans un état donné $s_t$. L'agent doit alors choisir une action $a_{t+1}$ parmi l'ensemble des actions qu'il peut effectuer. Le système réagit alors en passant dans un nouvel état $s_{t+1}$. Ce choix d'action influe alors sur l'évolution de l'environnement et donc sur les choix ultérieurs de l'agent: après avoir effectué son action, l'agent se trouve devant un nouveau choix. Le problème de décision séquentielle revient alors à déterminer quelle est la meilleure séquence de décision qui permette à un agent de réaliser au mieux la tâche qui lui est dévolue. La difficulté du problème est que ces choix ne sont pas indépendants.*

*Les processus décisionnels de Markov (MDPs: Markov Decision Processes) couplent l'aspect séquentiel et l'aspect stochastique (incertitude) et rendent la prise de décision séquentielle sous incertain plus adéquate. Ces modèles permettent de modéliser l'incertitude par des probabilités de transition et d'exprimer les performances par une fonction de récompense liée à la réalisation des actions.*

*Ce chapitre présente les bases de la planification stochastique. Nous présentons tout d'abord comment un agent peut prendre une décision simple par le principe de maximisation de l'utilité espérée. Puis, nous présentons une extension temporelle de cette prise de décision en introduisant le formalisme du MDP ainsi que les algorithmes de base pour trouver un comportement optimal.*

## 2.1 Prise de décision simple

La prise de décision simple s'intéresse aux problèmes épisodiques ou ponctuels, dans lesquels l'utilité du résultat de chaque action est bien connue. Dans ce type de prise de décision une seule action est suffisante pour atteindre un but.

Les préférences d'un agent entre deux états du monde sont représentées par une fonction d'utilité, qui attribue un nombre unique pour exprimer la « désirabilité » d'un état. Dans ce qui suit, nous emploierons $U(s)$ pour représenter l'utilité de l'état $s$ (selon l'agent qui prend les décisions). Une action non déterministe $a$ aura différents états résultants possible ; ainsi $Result_i(a) = s_i$ où l'indice $i$ parcourt les différents états. Avant d'exécuter $a$, l'agent affecte une probabilité $P(s_i|Faire(a), O)$ à chaque situation $s_i$, où $O$ représente les observations dont dispose l'agent sur le monde et $Faire(a)$ représente le fait d'exécuter $a$ dans l'état courant de l'agent. Dès lors l'utilité espérée de l'action $a$ connaissant $O$ se calcule en utilisant la formule [Russell and Norvig, 2003]:

$$EU(a|s) = \sum_i P(Effet_i(a)|a, s)U(Effet_i(s)) \tag{2.1}$$

Avec cette équation on ne peut malheureusement pas choisir une séquence d'actions car il nous faut énumérer toutes les séquences d'actions et choisir la meilleure ce qui, en pratique, n'est pas faisable. Dans les sections qui vont suivre nous verrons comment on pourrait traiter efficacement ce cas dans le cadre de la décision complexe du genre des MDPs. Dans cette partie, on se contentera de la prise de décision simple en s'inspirant de *Russell and Norvig* [Russell and Norvig, 2003].

La prise de décision simple est basée sur le principe de maximisation de l'utilité espérée telle qu'indiquée plus haut. Le principe de maximisation de l'utilité espérée indique qu'un agent rationnel doit choisir une action $a$ qui maximise l'utilité espérée :

$$a = \underset{a'}{\operatorname{argmax}}\, EU(a'|s) = \underset{a'}{\operatorname{argmax}} \sum_i P(Effet_i(a')|a,s)U(Effet_i(a')) \qquad (2.2)$$

La difficulté qui survient alors est de déterminer la fonction d'utilité $U(.)$. Pour cela, on utilise la notion de préférence qui peut être notée :

- $A \succ B$ l'agent préfère $A$ à $B$.
- $A \sim B$ l'agent est indifférent aux situations $A$ et $B$.
- $A \preccurlyeq B$ l'agent préfère $B$ à $A$ ou est indifférent.

Lorsque l'agent effectue une action, les situations rencontrées peuvent être soient des situations concrètes soit, dans les cas stochastiques, des *loteries*. Une loterie $L$ est représentée par une distribution de probabilité $p_1, \dots, p_n$ sur un ensemble de situations $S_1, \dots, S_n$. Elle est notée :

$$L = [p_1, S_1; \dots; p_n, S_n]$$

En général, chaque situation $S_i$ dans une loterie peut être soit une situation concrète soit une loterie.

Ce bref retour sur la prise de décisions simples permet d'introduire le principe de base de la maximisation de l'utilité espérée. Nous allons voir par la suite que ce principe est également applicable dans le cas de décisions séquentielles mais que la fonction d'utilité diffère légèrement pour prendre en compte le temps.

## 2.2 Prise de décision séquentielle

La prise de décision simple présentée plus haut, repose sur la maximisation de l'utilité espérée mais elle ne prend pas en compte l'aspect séquentiel des décisions. Le modèle formel utilisé pour décrire le problème de décisions séquentielles est le

processus de décision de Markov (*Markov Decision Process*, MDP) [Puterman, 1994]. Dans un MDP, le principe de maximisation de l'utilité espérée est également respecté. Cependant, comme nous le verrons, la définition de la fonction d'utilité dépend des décisions séquentielles et pas simplement de la situation courante.

Dans ce qui suit, nous introduisons tout d'abord le modèle MDP ainsi que les méthodes de bases qui servent à le résoudre.

### 2.2.1 Les Processus Décisionnels de Markov (MDPs)

Nous introduisons ici le fondement théorique de notre travail, qui est un formalisme mathématique pour la description et la résolution de problèmes de décision séquentielle, les processus de décision markovien ou MDPs, introduits dans les années 60 par *Richard Bellman* et *Ronald Howard* [Bellman, 1957; Howard, 1960]. D'un côté, le modèle MDP est très général et assez abstrait pour pouvoir être appliqué à des problématiques variées. D'un autre côté, il reste restreint à des environnements entièrement observables, et le contrôle se fait donc de manière centralisé.

Les processus décisionnels de Markov intègrent les concepts d'état qui résume la situation de l'agent à chaque instant, d'action (ou décision) qui influence la dynamique de l'état, de récompense qui est associée à chacune des transitions d'état. Les MDPs sont alors des chaînes de Markov visitant les états, contrôlées par les actions et valuées par les récompenses [Garcia, 2008]. Résoudre un MDP, c'est le contrôler de manière optimale. Toutefois, les solutions d'un MDP ne sont pas des décisions ou séquences de décisions, mais plutôt des politiques, ou stratégies, ou encore règles de décision, qui spécifient l'action à entreprendre en chacune des étapes pour toutes les situations futures possibles de l'agent. Du fait de l'incertitude, une même politique peut donner lieu à des séquences d'états/actions très variées selon les aléas. Un processus de décision Markovien modélise le problème de décision comme étant stochastique, séquentiel et totalement observable.

#### 2.2.1.1 Les composants d'un MDP

Un MDP est un modèle séquentiel pour la prise de décision dans des environnements totalement observables. Il est défini par un tuple $\langle S, A, P, R, T \rangle$ tel que, [Garcia, 2008] :

- $S$ est un ensemble fini d'états $s$.
- $A$ est un ensemble fini d'actions $a$.
- $P : S \times A \times S \rightarrow [0,1]$ est une fonction de transition qui donne la probabilité de passer de l'état $s$ à l'état $s'$ quand l'action $a$ est exécutée.
- $R : S \times A \rightarrow \mathbb{R}$ est une fonction de récompense. $R(s, a)$ est la récompense obtenue par l'agent lorsqu'il exécute l'action $a$ à partir de l'état $s$.
- $T$ est l'horizon ou le nombre d'étapes après lesquelles le problème se termine.

Les MDPs modélisent des problèmes de décision dans des environnements stochastiques. Dans chaque état, l'agent doit décider quelle action exécuter afin de passer à un nouvel état et il peut ainsi agir sur l'évolution du système.

- *Etats et actions:* les domaines $S$ et $A$ sont supposés finis, même si de nombreux résultats peuvent être étendus aux cas où $S$ et $A$ sont dénombrables ou continus [Ferns et al., 2012]. L'ensemble $S$ désigne les configurations possibles du monde. $A$ désigne l'ensemble des actions que l'agent peut choisir d'accomplir en vue de modifier l'état de son environnement. Dans le cas général, l'espace $A$ peut être dépendant de l'état courant ($A_s$ pour $s \in S$). De même, $S$ et $A$ peuvent être fonction de l'instant $t$ ($S_t$ et $A_t$).
- *Fonction de transition*: Les probabilités de transition caractérisent la dynamique de l'état du système. Elles caractérisent les réactions de l'environnement aux actions émises par l'agent. Pour une action $a$ fixée, $P(s'|s, a)$ représente la probabilité que le système passe dans l'état $s'$ après avoir exécuté l'action $a$ dans l'état $s$. En toute généralité, la matrice de transition permet de représenter des environnements stochastiques pour lesquels "les mêmes causes n'entraînent pas forcément les mêmes effets" : pour un état donné et une action donnée, l'état d'arrivée n'est pas forcément le même (figure 2.1). Cette représentation des lois

d'évolution du monde permet de modéliser des incertitudes quant au bon déroulement des actions. On impose classiquement que $\forall\, s, a\ \sum_{s'} P(s'|s,a) = 1$. Les distributions *P* vérifient la propriété fondamentale qui donne son nom aux processus décisionnels de Markov, propriété de *Markov (1856-1922).* Celle-ci stipule que la dynamique de l'état du système à un instant $t$ ne dépend pas de son historique, à savoir la suite des couples actions-états visités $h_t = (s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1}, s_t)$: la probabilité d'atteindre un état $s$ à la date $t+1$ dépend uniquement de l'état $s$ à la date $t$. En d'autres termes, il faut que l'état contienne suffisamment d'information pour en déduire l'état suivant connaissant l'action de l'agent. Plus formellement, nous pouvons résumer cette propriété par l'équation suivante:

$$P(s_{t+1} = s' | s_0, s_1, \dots, s_t, a_0, a_1, \dots, a_t) = P(s_{t+1} = s' | s_t, a_t)$$

Cette propriété peut paraître très restrictive. De nombreux systèmes à première vue non markoviens peuvent cependant être modélisés de façon à vérifier la propriété de Markov. En effet, tout dépend de la manière dont sont définis les états du système. Si ces derniers sont construits de façon à contenir toutes les informations nécessaires à la prédiction de l'évolution du système, le modèle est markovien. Cependant, dans les problèmes complexes, il est souvent difficile de garantir l'existence de cette hypothèse sans faire exploser le nombre d'états.

- *Fonction de récompense R:* permet d'exprimer les préférences sur les actions à réaliser en fonction de l'état courant. $R(s,a)$ retourne la récompense obtenue en exécutant l'action $a$ depuis l'état $s$, si $R(s,a)$ est positif, ou le coût d'exécuter $a$ en $s$, si $R(s,a)$ est négatif. La rationalité d'un agent s'exprime sous la forme d'un critère de performance individuel lié à cette fonction de récompense. Dans un MDP, à partir d'une fonction de récompense immédiate donnée, plusieurs critères de performance individuelle peuvent être définis. Dans ce qui suit, nous allons présenter les différents critères de performance utilisés par les agents.

La figure 2.1 représente un MDP. A chaque instant $t$ de *T*, l'action $a_t$ est appliquée dans l'état courant $s_t$, influençant le processus dans sa transition vers l'état $s_{t+1}$. La récompense $R_t$ est émise au cours de cette transition.

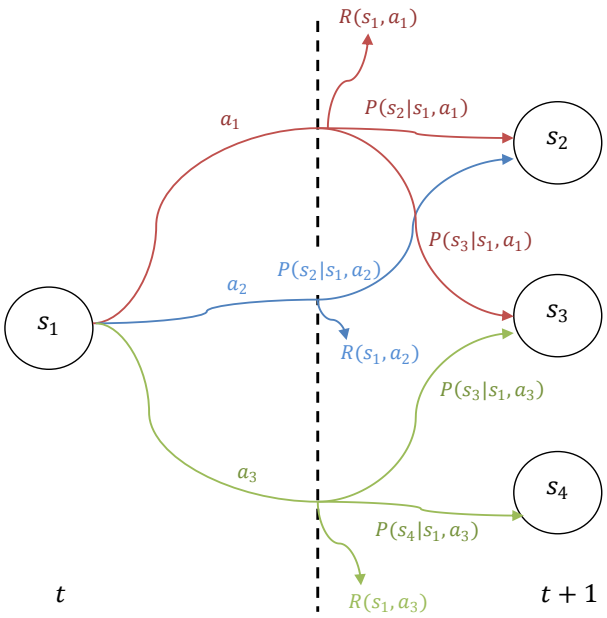


***Figure 2.1*** *Représentation d'un MDP. Lorsque l'agent se trouve dans l'état $s_1$ il peut accomplir trois actions $a_1$, $a_2$ ou $a_3$. Lorsqu'il effectue l'action $a_1$, il a une probabilité $P(s_2|s_1, a_1)$ d'arriver dans l'état $s_2$ et une probabilité $P(s_3|s_1, a_1)$ d'arriver dans l'état $s_3$, avec une récompense $R(s_1, a_1)$. Lorsqu'il effectue l'action $a_2$, il a une probabilité $P(s_2|s_1, a_2)$ d'arriver dans l'état $s_2$, avec une récompense $R(s_1, a_2)$. Lorsqu'il effectue l'action $a_3$, il une probabilité $P(s_3|s_1, a_3)$ d'arriver dans l'état $s_3$ et une probabilité $P(s_4|s_1, a_3)$ s'arriver dans l'état $s_4$, avec une récompense $R(s_1, a_3)$*

#### 2.2.1.2 Les critères de performance

On appelle problème de décision markovien un MDP avec un critère de performance associé. Le critère de performance précise comment on doit évaluer la qualité d'une politique pour un MDP donné. Une fois que le critère de performance est fixé, on peut s'intéresser à la politique qui optimise l'évaluation de ce critère. La recherche d'une telle politique constitue l'enjeu principal du problème de décision markovien.

Pour des problèmes à horizon fini $T$, un des critères de performance les plus utilisés est l'espérance de la somme des récompenses accumulées à partir de l'état initial $s_0$, ce critère est appelé le critère fini:

$$E\left[\sum_{t=0}^{T-1} R(s_t, a_t)|s_0\right] \tag{2.3}$$

Pour des problèmes à horizon infini, utiliser le même critère de performance pose le problème de l'accumulation non bornée de récompenses. C'est pour cela que l'on introduit en général un facteur d'escompte γ (*discount factor*) qui est un nombre compris entre 0 et 1. Ce facteur décrit dans quelle mesure un agent préfère les récompenses actuelles aux récompenses futures. Ce qui donne le critère suivant, appelé critère $\gamma$-pondéré:

$$E\left[\sum_{t=0}^{\infty} \gamma^t \, R(s_t, a_t)|s_0\right] \tag{2.4}$$

Les deux caractéristiques communes de ces deux critères sont en effet d'une part leur formule additive en $R$, qui est une manière simple de résumer l'ensemble des récompenses reçues le long d'une trajectoire et, d'autre part, l'espérance *E[.]* qui est retenue pour résumer la distribution des récompenses pouvant être reçues le long des trajectoires, pour une même politique et un même état de départ. Ce choix d'un cumul espéré est bien sûr important, car il permet d'établir le principe d'optimalité de Bellman [Bellman, 1957]: « *les sous-politiques de la politique optimale sont des sous-politiques optimales* ». Le principe de Bellman est détaillé dans la section 2.2.1.3.

Nous avons présenté le formalisme des processus décisionnels de Markov. Ce formalisme permet la modélisation des problèmes décisionnels sous incertitude. Ces composants *S* et *P* décrivent le système et sa dynamique. *A* fournit l'espace des actions à entreprendre par le décideur, et la fonction de récompense *R* permet de définir les objectifs de l'agent. Nous allons maintenant nous intéresser à la résolution d'un tel problème. Résoudre un MDP consiste à déterminer un comportement de l'agent (décideur) qui lui permette d'atteindre ses objectifs. Pour ce faire, il est nécessaire de déterminer une *politique* d'action de l'agent.

- **La notion de politique**

Une politique, ou stratégie, est la procédure qui permet à l'agent, étant donné les observations passées et présentes, de choisir à chaque instant $t$ l'action à exécuter dans l'environnement afin de tenter d'en contrôler la dynamique.

La politique $\pi$ d'un agent fait correspondre à chaque état $s$ du système, une action $a$ que l'agent doit entreprendre, d'où :

$$\pi : S \rightarrow A$$

Le système étant markovien, seul l'état courant est nécessaire pour décider quelle action réaliser.

Pour chaque problème de décision formalisé sous forme de MDP, il existe un ensemble $\Pi$ de politiques possibles $\pi \in \Pi$. Ces politiques sont très importantes car elles représentent des politiques optimales pour les principaux critères de performance présentés ci-dessus. Se poser un problème décisionnel de Markov, c'est rechercher parmi une famille de politiques celle qui optimise un critère de performance donné pour le MDP considéré. Cette solution est appelée politique optimale et est notée $\pi^*$ [Garcia, 2008].

Bellman [Bellman, 1957] a démontré que tout MDP possédait au moins une politique optimale déterministe et stationnaire. De ce fait, la recherche de la politique peut être réalisée dans l'espace des politiques déterministes, (une politique déterministe associe à chaque état une et une seule action. Elle s'oppose aux politiques stochastiques (ou stratégies mixtes) associant à chaque état différentes actions possibles suivant une certaine distribution de probabilités) [Beynier, 2006].

- **Evaluation d'une politique**

Dès qu'un critère de performance est associé à un MDP, on peut évaluer la valeur d'une politique en établissant sa fonction de valeur $V$, c'est-à-dire une fonction qui retourne pour chaque état la valeur du critère de performance.

En fonction de leurs valeurs $V^{\pi}$, les politiques pourront alors être classées. Nous avons précédemment montré (équations 2.3 et 2.4) que le critère de performance d'un agent consistait à maximiser ses récompenses à long terme. La valeur d'une politique interprétera donc l'utilité espérée de l'agent. Résoudre un problème de décision Markovien revient à trouver la politique qui maximise la fonction de valeur pour un état initial $s_0$ donné. On parlera alors de politique optimale que l'on notera $\pi^*$. La politique optimale $\pi^*$ est telle qu'elle maximise l'espérance de l'agent en tout état du système, soit [Beynier, 2006]:

$$\forall \pi \; \forall s \in S \qquad V^{\pi^*}(s) \geq V^{\pi}(s)$$

Autrement formulé:

$$V^{\pi^*} = \underset{\pi}{\operatorname{argmax}} V(s_0, \pi)$$

Dans les problèmes à horizon fini $T$, il faut considérer l'utilité espérée à partir d'un état $s$, et sur les $T$ prochaines étapes. Nous obtenons alors :

$$\forall s \in S \quad V^{\pi}(s) = E\left[\sum_{t=0}^{T-1} R(s_t, a_t) | s_0 = s\right] \tag{2.5}$$

Si la fonction, de récompense dépend seulement de l'état, alors $V^{\pi}(s) = Et{=}0T{-}1R(st)|s0{=}s$.

Dans le cas des problèmes à horizon infini, le gain espéré est pondérée par un facteur d'escompte:

$$\forall s \in S \quad V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s\right] \tag{2.6}$$

La pondération par un facteur d'escompte assure la convergence de l'espérance. Il faut noter que le choix de la valeur du facteur d'escompte influence le comportement de l'agent.

Une autre façon de déterminer la valeur d'une politique sur un horizon infini, consiste à calculer le gain moyen par étape de décision [Garcia, 2008]:

$$\forall s \in S \quad V^{\pi}(s) = \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{t=0}^{n} R(s_t, a_t)|s_0 = s\right] \tag{2.7}$$

#### 2.2.1.3 Principe d'optimalité de Bellman

*"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."* [Bellman, 1957]

Intuitivement, le principe d'optimalité de Bellman [Bellman, 1957] stipule que pour une politique optimale $\pi^*$ en horizon $T$, toutes les sous-politiques $\pi^*_{t:T}$ pour les instants $t$ à $T$ sont des politiques optimales du sous-problème constitué de ces instants. Autrement dit, n'importe quelle politique optimale peut être calculée d'une manière incrémentale, horizon par horizon. Cette technique de recherche de politique optimale est appelée programmation dynamique et décompose le problème en deux étapes successives:

1) La recherche d'une sous-politique optimale $\pi^*_{1:T-1}$ (sous-problème);
2) La construction d'une politique optimale $\pi^*$ à partir de la sous-politique optimale $\pi^*_{1:T-1}$.

Bellman a montré que la fonction de valeur d'une politique peut être calculée par récurrence, grâce à l'équation de *Bellman* suivante :

$$V(s) = R\big(s, \pi(s)\big) + \sum_{s' \in S}\big[\gamma\, P\big(s'|s, \pi(s)\big)\, V(s')\big] \tag{2.8}$$

La valeur d'un état $V(s)$ est alors définie comme une somme pondérée de sa récompense immédiate après réalisation de l'action $\pi(s)$ et des valeurs des états atteignables proportionnellement aux probabilités de les atteindre.

De plus, la fonction de valeur de la politique optimale d'un MDP satisfait l'équation suivante :

$$V^*(s) = \max_{a \in A}[R(s,a) + \sum_{s' \in S} \gamma\, P(s'|s,a)\, V^*(s')] \tag{2.9}$$

Connue comme l'équation de Bellman. Résoudre ce système non-linéaire de ces équations pour chaque état $s$ donne la fonction de valeur optimale, et à partir de laquelle nous obtiendrons la politique optimale en utilisant l'équation suivante:

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \left[ R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s') \right] \tag{2.10}$$

Cependant, à cause de l'opérateur non-linéaire *max*, résoudre le système pour chaque état simultanément n'est pas faisable pour les MDPs larges [Puterman, 1994]. Pour pallier à ce problème, Bellman a introduit une technique d'approximation successive qui est à la base de plusieurs algorithmes de résolution des MDPs, en particulier *Value Iteration* et *Policy Iteration* que nous présenterons par la suite. La fonction de valeur optimale $V_0^*$, quand l'agent peut seulement prendre une action, est définie par le modèle de récompense:

$$V_0^*(s) = \max_{a \in A} R(s,a)$$

Afin de considérer une étape plus profonde dans le futur, c'est-à-dire de calculer $V_{t+1}^*$ à partir de $V_t^*$, l'équation (2.9) devient une mise à jour:

$$V_{t+1}^{*}(s) = \max_{a \in A} \left[ R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)\, V_t^{*}(s') \right] \tag{2.11}$$

Qui est un mapping qui va converger au point fixe $V^*$.

Autrement dit, la politique optimale à l'étape $t+1$ est une sous-partie de la politique optimale à partir de l'étape $t$.

Appliquer l'équation de *Bellman* à chaque état mène à une seule solution $V^*$ qui correspond à la politique optimale. Plusieurs politiques optimales peuvent exister pour le même problème de décision, mais ces dernières auront toutes pour valeur $V^*$.

#### 2.2.1.4 Complexité

Dans cette section, nous présentons la complexité de la résolution optimale de ces problèmes.

Papadimitriou et Tsitsiklis [Papadimitriou and Tsitsiklis, 1987] ont étudié la complexité de la résolution optimale des MDPs. Il a été démontré que la résolution d'un MDP à horizon fini ou infini est un problème *P-complet*. De ce fait, plusieurs algorithmes efficaces ont été développés pour leur résolution.

#### 2.2.1.5 Les algorithmes de résolution

Cette partie traite la résolution des modèles markoviens représentant des problèmes de prise de décision mono-agent (à savoir les MDPs). Cette section nous semble importante pour deux raisons. D'une part, ces techniques constituent le fondement des approches utilisées pour résoudre les DEC-POMDPs et DEC-MDPs comme nous le verrons dans le prochain chapitre. D'autre part, nous souhaitons utiliser des techniques de construction de comportement individuel pour construire des comportements collectifs. Nous serons amenés à réutiliser ces techniques dans nos contributions.

Déterminer la politique optimale d'un MDP revient à calculer la valeur de la politique optimale en se basant sur l'équation de Bellman présentée à la section 2.2.1.3.

L'exécution d'une politique $\pi$ correspond à la boucle perception-action d'un agent [Russell and Norvig, 2003]. Pour un tuple $\langle S, A, P, R \rangle$ donné elle est constituée de cycle de 4 étapes (figure 2.2) [Thomas, 2005]:

- L'agent observe l'état $s_0$ de l'environnement (figure 2.2 (a))
- Il détermine l'action $a$ qu'il souhaite entreprendre en fonction de sa politique $\pi, a \leftarrow \pi(s_0)$ (figure 2.2 (b))
- Il effectue cette action et l'état du monde s'en trouve modifié $s_1 \leftarrow P(s_1|s_0, a)$ (cette dernière expression cache en réalité une réalisation selon la densité de probabilité $P(s_0, a)$) (figure 2.2 (c))
- Il reçoit une récompense $r \leftarrow R(s_0, a)$ (figure 2.2 (d))

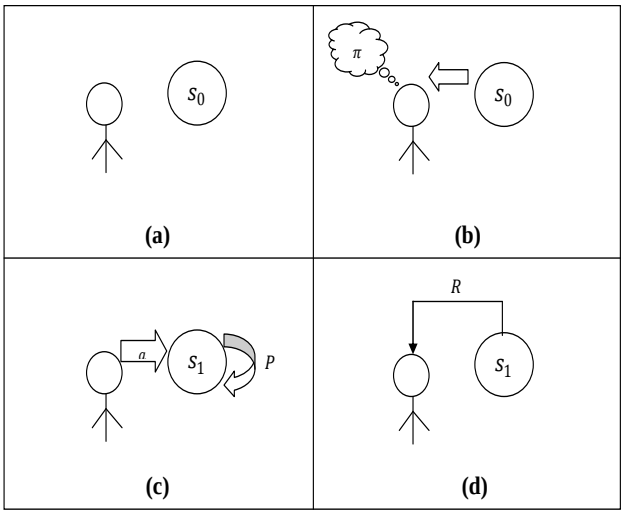


***Figure 2.2*** *Exécution d'une politique d'un MDP constituée de plusieurs étapes: a) état initial, b) étape de décision, c) étape d'exécution de l'action, d) étape de modification de l'état, une récompense est reçue*

Nous allons à présent décrire deux algorithmes pour la résolution de ces problèmes décisionnels.

Il faut noter que ces algorithmes supposent que le modèle du système soit connu. En particulier, la fonction de transition $P$ et la fonction de récompense $R$ doivent être connues à l'avance.

- **L'algorithme Value Iteration (VI)**

L'équation de *Bellman* constitue la base de l'algorithme *Value Iteration*, l'un des algorithmes les plus utilisés pour la résolution des MDPs à horizons finis ou infinis.

Le principe de base de cet algorithme de programmation dynamique décrit par *Bellman* [Bellman, 1957] consiste en une amélioration itérative de la valeur de chaque état du MDP. Le principe consiste en des réévaluations approximatives successives de chaque état. A partir d'une valeur initiale arbitraire de chaque état, *Value Iteration* améliore itérativement l'évaluation des états en utilisant l'équation de *Bellman*. La valeur d'un état à l'itération $t$ est calculée à partir de sa valeur à l'itération $t-1$. En pratique plusieurs conditions d'arrêt de l'itération peuvent être envisagées. La plus classique consiste à stopper l'itération lorsque la différence entre deux fonctions de valeurs successives $V_t - V_{t-1}$ est inférieur à $\epsilon$ où $\epsilon$ est un seuil d'erreur fixé *a priori* [Kaelbling et al., 1998].

La convergence de l'algorithme vers une fonction de valeur optimale est garantie étant donné que l'équation de *Bellman* est une équation de point fixe.

A partir de la valeur $V_t$ obtenue par cet algorithme, nous pouvons déduire une politique $\pi_t$ optimale à $\epsilon$ près. En pratique, cette politique s'avère généralement être la politique optimale du MDP qui est souvent obtenue bien avant que l'algorithme ne s'arrête. Les dernières itérations sont consacrées à une amélioration de la fonction de valeur.

**Algorithme 1 :** Value Iteration

**Entrées :** un MDP $\langle S, A, P, R \rangle$

un paramètre de précision $\epsilon$

un facteur d'escompte $\gamma$

une valeur initiale $V_0$ arbitraire

$t \leftarrow 0$
**Pour tous les** $s \in S$ **faire**
  $V_0(s) \leftarrow V_0$
**fin**
**répéter**
$t \leftarrow t + 1$
  **Pour tous les** $s \in S$ **faire**

$$V_t(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} P(s'|a,s) V_{t-1}(s')]$$

  **fin**
**jusqu'à** $\max_{s \in S} |V_t(s) - V_{t-1}(s)| < \epsilon$ ;

**Sorties :** $V_n$ l'approximation de $V^*$ à $\frac{2\epsilon\gamma}{\gamma - 1}$ près

*Complexité*

La complexité en temps de l'algorithme dépend de la complexité de chaque itération et du nombre d'itérations. Une itération nécessite $|A||S|^2$ opérations. Par ailleurs, le nombre d'itérations nécessaires pour converger est, au pire cas, polynomial en $|S|$, $|A|$, $\gamma$ et $B$ où $B$ est le nombre total de bits requis pour représenter les données du problème [Beynier, 2006].

Sutton et Barto [Sutton and Barto, 1998] ont montré que des MDPs d'un million d'états pouvaient facilement être résolus par l'algorithme *Value Iteration*. De plus, cet algorithme possède des propriétés *anytime* c'est-à-dire que la qualité de la solution augmente de façon monotone à chaque itération et l'amélioration est décroissante au cours du temps. L'algorithme est également interruptible : si nous l'arrêtons avant qu'il ait atteint son point de convergence, nous disposerons quand même d'une solution.

*Value Iteration* calcule une politique $\epsilon$-optimale pour n'importe quel état initial. Ceci peut constituer un avantage si nous souhaitons calculer une politique sans connaître *a priori* la situation initiale de l'agent. Cependant, lorsque les états initiaux sont connus, l'algorithme n'en tire pas avantage ce qui conduit à l'évaluation de trajectoires non atteignables. En ce qui concerne la complexité en espace, *Value Iteration* est linéaire en $|S||A|$ [Beynier, 2006].

- **L'algorithme Policy Iteration (PI)**

A l'instar de *Value Iteration*, l'algorithme *Policy Iteration* utilise le principe d'optimalité de Bellman afin de résoudre des MDPs à horizons finis ou infinis. La différence entre ces deux algorithmes réside dans la manière dont la politique optimale est trouvée. *VI* calcule une valeur puis déduit une politique. *PI* construit directement la politique optimale. Cet algorithme décrit par Howard en 1960 [Howard, 1960] propose d'actualiser, à $\epsilon$ près, les valeurs de l'équation de Bellman $V(s)$, entre deux itérations sur la politique contruite.

L'algorithme part d'une politique initiale arbitraire $\pi_0$, et est divisé en deux phases. Une phase d'évaluation de la politique courante qui utilise l'équation de Bellman et une phase d'amélioration qui consiste à mettre à jour la politique courante en utilisant les valeurs calculées lors de la phase d'évaluation.

L'algorithme converge lorsqu'aucune modification de la politique n'est plus possible, soit $\pi_{t+1}(s) = \pi_t(s)\ \forall\ s \in S$ et fournir ainsi une politique optimale.

Howard a montré que chaque itération améliorait la politique : la valeur de chaque état à l'itération $t+1$ est supérieure ou égale à celle de l'itération $t$ et elle est strictement supérieure pour au moins un état. Chaque itération dans *policy iteration* demande alors plus de ressources calculatoires qu'une itération dans *value iteration* [Putterman, 1994]. Cependant, *policy iteration* a statistiquement besoin de moins d'itérations pour converger vers la politique optimale.

**Algorithme 2 :** Policy Iteration

**Entrées :** un MDP $\langle S, A, P, R \rangle$

un facteur d'escompte $\gamma$

une politique initiale $\pi_0$arbitraire

**1** $t \leftarrow 0$
**2** $\pi_{t+1} \leftarrow \pi_0$
**3** **répéter**
**4** $\pi_t \leftarrow \pi_{t+1}$
**5** $t \leftarrow t + 1$
// Phase d'évaluation
**6** **Pour tous les $s \in S$ faire**
**7** Calculer $V_{\pi_t}(s)$
**8** **Fin**
// Phase d'amélioration
**9** **Pour tous les $s \in S$ faire**
**10** **Si** $\exists a \in A$ tel que $R(s,a) + \gamma \sum_{s' \in S} P(s', a, s) V_{\pi_t}(s') > V_{\pi_t}(s)$ **alors**
**11** $\pi_{t+1}(s) = a$
**12** **sinon**
**13** $\pi_{t+1}(s) = \pi_t(s)$
**14** **fin**
**15** **fin**
**16** **jusqu'à** $\pi_t(s) = \pi_{t+1}(s) \quad \forall s \in S$ ;

**Sorties :** $\pi_{t+1}$la politique optimale

*Complexité*

La complexité de l'algorithme $PI$ est en $O(|A||S|^2) + O(|S|^3)$ par itération avec un nombre maximum d'itération polynomial en $|S|, |A|$ à $\gamma$ constant [Papadimitriou and Tsitsiklis, 1987; Garcia, 2008].

Le nombre d'itérations de l'algorithme dépend fortement de la politique initiale $\pi_0$. Il existe $|A|^{|S|}$politiques possibles. Étant donné qu'à chaque itération la politique mise à jour domine strictement la politique précédente, le nombre d'itérations sera, au pire cas, exponentiel en nombre d'états. Toutefois, il a été prouvé que la borne supérieure concernant le nombre d'itérations était bien inférieure [Beynier, 2006]. En effet,

Puterman [Puterman, 1994] a montré que l'algorithme ne converge pas moins vite que *Value Iteration*, pour les problèmes à horizon infini avec facteur d'escompte.

*Policy Iteration* a besoin de moins d'itérations pour converger vers la politique optimale que *Value Iteration*. Ceci s'explique par le fait que *Value Iteration* continue à itérer tant que la valeur de la politique peut être améliorée, et ce même si la politique ne change pas. En revanche, une itération de *Policy Iteration* consomme plus de temps qu'une itération de *Value Iteration*.

#### 2.2.1.6 D'autres méthodes de résolution des MDPs

Les algorithmes classiques de programmation dynamique comme *VI* et *PI,* résolvent un MDP d'une manière optimale en mettant à jour la valeur de chaque état d'une manière itérative, dans un ordre fixe, un état à chaque itération. Cela peut être inefficace, puisque la structure graphique du problème est négligée, ce qui peut fournir une information importante sur les dépendances entre les états. Récemment, les chercheurs ont développé des algorithmes de recherche heuristique qui utilisent les informations d'accessibilité et les fonctions heuristiques pour éviter certaines mises à jour (*backups*) qui ne sont pas nécessaires. Ces approches, telles que, ILAO* (Improved LAO*) [Hansen and Zilberstein, 2001], LRTDP [Bonnet and Geffner, 2003b], HDP [Bonnet and Geffner, 2003a], BRTDP [McMahan et al., 2005] et Bayesian RTDP [Sanner et al., 2009], dépassent fréquemment *VI.* Cependant, sur certains problèmes, les algorithmes de recherche heuristiques offrent un bénéfice minime et il est difficile de prédire quand est ce qu'ils sont excellents.

[Dai et al., 2011] proposent deux algorithmes qui résolvent les MDPs d'une manière optimale et accélèrent la convergence de *VI: topological Value Iteration* (TVI) et *Focused Topological Value Iteration* (FTVI). *TVI* utilise la structure graphique du MDP. Il applique l'équation de mise à jour de Bellman dans un ordre plus intelligent, après avoir appliqué une analyse additionnelle de la topologie de l'espace d'états du MDP. Il a été prouvé que cet algorithme donne de bons résultats par rapport à *VI* et même les algorithmes de recherche heuristique dans certains domaines. *FTVI* adresse les points faibles de *TVI.* Il applique, en premier, une phase de recherche heuristique afin d'éliminer les actions probablement sous-optimales. Par la suite, il construit une

structure graphique informative basée sur les actions qui restent. Il a été prouvé que *FTVI* donne des résultats meilleurs que *TVI* et les algorithmes de recherche heuristique dans certains domaines.

D'autres travaux ont abordé le problème des MDPs larges, qui ne peuvent pas être résolus d'une manière optimale mais plutôt approximative. Pour le faire, des relaxations déterministes du MDP et/ou les fonctions basiques ont été utilisés [Guestrin et al., 2003; Poupart et al., 2002; Patrascu et al., 2002; Yoon et al., 2007; Kolobov et al., 2009, 2010a, 2010b].

L'approche de planification en ligne est très différente des méthodes de résolution des MDPs standards considérées dans la programmation dynamique et l'apprentissage par renforcement. Ces dernières trouvent souvent une solution globale, contrairement à la planification en ligne qui trouve des actions sur demande, localement pour chaque état quand c'est nécessaire. Cependant, la planification en ligne est beaucoup moins dépendante de la taille de l'espace des états. Elle est généralement sous-optimale, mais des limites (*bounds*) utiles peuvent être posées sur sa sous-optimalité [Busoniu et al., 2012].

Un algorithme récent de la famille de la planification optimiste est UCB (*Upper Confidence Bound*) appliqué aux arbres (UCT, *Upper Confidence Bound for Tree*) [Kocsis and Szepesvari, 2006], qui a eu un grand impact dans le domaine de la planification grâce au résultat compétitif qu'il a donné dans les jeux tels que Go [Gelly et al., 2006]. UCT applique UCB à chaque nœud dans l'arbre de planification. Il parcourt un chemin optimiste tout au long l'arbre en choisissant des actions avec un UCB maximal sur les récompenses obtenues à partir du nœud courant, et échantillonne les états indépendamment.

#### 2.2.1.7 Les domaines d'application des MDPs

Les MDPs ont été premièrement utilisés pour la gestion de route de l'état d'Arizona en 1978 [Golabi et al., 1982; Puterman, 1994; Mundhenk et al., 2000]. Depuis ces formalismes puissants ont attiré l'attention de plusieurs chercheurs dans différents domaines.

Dans [Puterman, 1994], la diversité des applications des MDPs a été présentée. En effet, ces modèles peuvent être utilisés pour la gestion de l'inventaire, remplacement de pièces et d'équipement, entretien des chaussées routières, les modèles de communication et le routage, ainsi que pour l'étude des phénomènes biologiques.

Plus récemment ces modèles ont été appliqués dans la robotique [Kaelbling et al., 1998; Roy et al. 2000]. Les MDPs et POMDPs ont également été utilisés afin de résoudre des problèmes de navigation. Les chercheurs de La NASA utilisent les MDPs pour modéliser les problèmes de planification des robots envoyés sur Mars [Bresina et al., 2002; Feng and Zilberstein, 2004; Mausam et al., 2005; Meuleau et al., 2009]. Les MDPs sont également utilisés pour formuler les problèmes de planification des opérations militaires [Aberdeen et al., 2004] et dans le cadre des réseaux de capteurs sans fil [Abu Alsheikh et al., 2015].

### 2.2.2 L'observabilité partielle dans les MDPs : POMDP (Partially Oservable Markov Decision Process)

Le modèle MDP fait l'hypothèse forte de l'observabilité totale: l'agent connait à chaque instant et avec exactitude l'état du système qui constitue donc un signal markovien et lui permet de prendre la décision optimale. Cette propriété n'est, cependant, pas vérifiée dans de nombreux problèmes: citons les exemples d'applications robotiques où l'agent n'a qu'une visibilité de son entourage immédiat, généralement fortement bruitée, plutôt qu'une connaissance exacte du système dans sa totalité. En effet, l'agent ne connait plus forcément à chaque instant $t$ l'état $s_t$ du système mais reçoit une observation $O_t$ qui dépend de manière stochastique de l'état du système. L'observation peut être une information partielle, une mesure bruitée, etc.

#### 2.2.2.1 Le formalise du POMDP

Le formalisme du POMDP ajoute au MDP la notion des observations. Ces observations sont stochastiques, et elles sont générées durant chaque transition d'état selon une fonction d'observation. Dans ce qui suit, on va introduire le formalisme de POMDP pour la prise de décision optimale des environnements partiellement observables.

Un POMDP peut être défini par un tuple $M = \langle S, A, P, R, \Omega, O, T \rangle$, avec

- $S$ est un ensemble fini des états $s \in S$
- $A$ est un ensemble fini des actions $a \in A$
- $P$ est la fonction de transition, $P(s'|s, a)$ est la probabilité de transition du système de l'état $s$ vers l'état $s'$ après exécution de l'action $a$
- $R$ est la fonction de récompense, $R(s, a)$ est la récompense immédiate produite par le système quand l'action $a$ est exécutée dans l'état $s$
- Ω est un ensemble fini d'observations
- O est le modèle d'observation, $O(o|s, a, s')$ est la probabilité d'observer $o$ si l'action $a$ est exécutée et le système passe de l'état $s$ à l'état $s'$
- $T$ est l'horizon ou le nombre d'étapes après lesquelles le problème se termine.

A la différence de l'état courant, qui est une information suffisante pour le contrôle optimal d'un MDP, l'observation courante ne vérifie pas nécessairement la propriété de Markov dans le cas d'un POMDP. Ceci veut dire que les méthodes de résolution et les équations d'optimalité qui sont valables pour les états d'un MDP ne peuvent pas être appliquées directement aux observations d'un POMDP.

Afin de vérifier la propriété de Markov, il a été considéré que la probabilité du système de se trouver dans un état $s$ au moment $t$ de l'exécution dépend uniquement de la distribution de probabilités sur les états au moment précédent. Cet état est appelé état de croyance.

Il a été montré par *Astrom* [Astrom, 1965] que l'état de croyance constitue une information suffisante pour représenter le passé du système. Il est donc possible de considérer un POMDP comme un MDP à états continus, le *belief-state* MDP. L'obstacle principal à sa résolution réside dans la continuité de son espace d'états. Un opérateur de type itération de valeur devrait donc être appliqué une infinité de fois pour établir une fonction de valeur. Il a cependant été montré par *Smallwood* et *Sondik* [Smallwood and Sondik, 1973] que la fonction de valeur à l'itération $i$ peut toujours être représentée par un ensemble fini de paramètres. En effet, si $V_i$ est une fonction de valeur convexe et linéaire par morceaux, alors la fonction $V_{i+1}$ est aussi convexe et linéaire par

morceaux. On peut montrer en particulier que la fonction de valeur à l'horizon 1 est forcément convexe et linéaire par morceaux. Ceci garanti, avec la propriété précédente, que la fonction de valeur pour un horizon fini est toujours représentable par des moyens finis. Le théorème de *Smallwood* et *Sondik* représente le fondement essentiel pour la résolution d'un POMDP.

#### 2.2.2.2 Complexité

Les POMDPS sont significativement plus difficile à résoudre que les MDPs: la complexité monte de polynomiale à exponentielle.

La résolution d'un POMDP à horizon fini est un problème PSPACE. En revanche, un POMDP à horizon infini est indécidable.

En général, résoudre un POMDP à horizon fini d'une manière optimale nécessite un temps exponentiel avec respect de la taille du problème. Quand on passe aux horizons infinis, le problème change sa classe de complexité et devient, en général, insoluble. C'est pour cette raison, résoudre un POMDP à horizon infini est usuellement restreint à une certaine classe de politiques, tel que les automates finis. En outre, Littman [Littman, 1996] s'est intéressé à un cas particulier de POMDP à horizon infini dans lequel les récompenses sont égales à 0 ou 1 (des valeurs booléennes). Il a été montré que, dans ce cas, le problème est EXP.

## 2.3 Conclusion

Dans ce chapitre, nous avons abordés les problèmes de décision séquentielle sous incertitude dans lesquels l'agent connait à chaque instant l'état actuel du système. Nous avons présenté le modèle approprié à ce type de problèmes, à savoir, les processus décisionnels de Markov (MDP), les équations de Bellman qui régissent la sélection des politiques, ainsi que les principaux algorithmes de résolution. Nous avons, en particulier, décrit les algorithmes d'itérations de valeurs et de politiques. L'optimalité et la complexité de ces algorithmes ont enfin été discutés.

Tout au long de ce chapitre, nous ne nous sommes pas préoccupés de l'aspect décentralisé des problèmes de décision, ce qui constitue le cœur de notre travail. Les

MDPs ont, en effet, été introduits comme des formalismes permettant de représenter des problèmes décisionnels à agent unique. La question qui se pose à ce moment, concerne l'extension de ces modèles aux problèmes de décision décentralisée. En effet, les ébauches de la théorie de la planification dans ce cadre ont consisté pour l'essentiel à adapter au cadre décentralisé plusieurs des techniques rudimentaires décrites dans ce chapitre. Nous détaillerons cette adaptation dans le chapitre suivant.

# Chapitre 3

# *Prise de décision séquentielle décentralisée sous incertitude*

*Les recherches en intelligence artificielle se sont tout d'abord orientées vers l'élaboration d'entités agissant seules (systèmes constitués d'un seul robot par exemple). Ces travaux ont démontré la faisabilité et les potentialités du développement de telles entités intelligentes.*

*Plus récemment les chercheurs ont commencé à s'intéresser à la mise en interaction de telles entités et à la complexité des comportements qui en découle. Il est alors apparu que les décisions prises par une entité appartenant à un tel système sont complexifiées du fait des interactions. Il devient alors plus difficile, à partir des décisions individuelles, d'obtenir un comportement collectivement intelligent.*

*Quand plusieurs entités (agents) rentrent en interaction et constituent un seul système, la prise de décision est généralement décentralisée. Ces systèmes ne sont pas contrôlés par une entité unique ayant une vue globale du problème mais par une équipe d'agents indépendants ayant chacun une vue local et partielle du problème. Dans ce chapitre et tout au long de cette thèse, nous nous intéressons aux systèmes coopératifs dans lesquels les agents ne sont pas en compétition mais coopèrent pour réaliser une mission commune de façon coordonnée.*

*Ce chapitre va porter sur les problèmes de décision séquentielle décentralisée. Plus particulièrement, nous aborderons l'extension des modèles présentés au chapitre précédent (MDPs et POMDPs) au cas multi-agents. Nous commencerons par illustrer la problématique de la prise de décision décentralisée sous incertitude en décrivant*

*trois exemples couramment utilisés dans ce domaine. Par la suite nous présenterons un modèle mathématique de prise de décision séquentielle décentralisée sous incertitude, à savoir le processus décisionnel de Markov décentralisé et partiellement observable (DEC-POMDP: Decentralized Partially Observable Markov Decision Process) une extension du POMDP au cas multi-agents. Nous discuterons ensuite la complexité de ce modèle et nous donnerons un aperçu des algorithmes de sa résolution optimale et approchée. Enfin, nous présenterons les différentes extensions de ce modèle qui permettent la réduction de sa complexité ainsi que l'introduction de la communication.*

## 3.1 Le contrôle centralisé : MMDP (Multi-agent Markov Decision Process)

Tout d'abord, nous allons présenter l'extension des MDPs pour le cas multi-agents en gardant l'aspect centralisé.

Dans les systèmes à contrôle centralisé, la politique optimale jointe est calculée par un contrôleur central qui est chargé de prendre les décisions pour tous les agents. Cette politique peut être facilement calculée à raison de l'observabilité totale de l'état du système par le contrôleur central. Cette observabilité totale n'est généralement pas satisfaite, car dans un monde réel, les agents sont souvent répartis dans l'espace.

Afin d'adapter le formalisme des MDPs aux systèmes multi-agent coopératifs, Boutilier [Boutilier, 1996; Boutilier et al., 1999] a défini les Processus Décisionnels de Markov Multi-agents ou MMDP (*Multi-agent Markov Decision Processes*). Dans ces modèles la fonction de récompense est définie communément à tous les joueurs et ainsi ils modélisent uniquement des systèmes coopératifs.

Un MMDP est défini par les mêmes composants qu'un MDP, à savoir un tuple $\langle S, A, P, R \rangle$. Cependant, chaque action est appelée action jointe et est décrite par l'ensemble des actions individuelles des agents. De plus, une variable $\alpha$ décrivant le nombre d'agents est ajoutée à l'ensemble des composants.

Un MMDP est donc défini par :

- $\alpha$, le nombre d'agents du système.

- $S$ correspond à l'ensemble des états $s$ du système.
- $A = A_1 \times ... \times A_n$, définit l'ensemble des actions jointes des agents, $A_i$est l'ensemble des actions locales de l'agent $i$.
- $P$ est une fonction de transition, elle donne la probabilité $P(s'|s,a)$ que le système passe dans l'état $s'$ quand les agents exécutent l'action jointe $a \in A$ à partir de l'état $s$.
- $R$, définit la fonction de récompense. $R(s,a,s')$ est la récompense obtenue par le système lorsqu'il passe d'un état $s$ à un état $s'$ en exécutant l'action $a$.

Résoudre un MMDP consiste à calculer une politique jointe $\pi = \langle \pi_1, \dots, \pi_n \rangle$ où $\pi_i$ correspond à la politique locale de l'agent $i$. Elle définit une fonction $\pi_i : S \to A_i$ qui fait correspondre à tout état du système, une action $a_i$ de l'agent $i$. La combinaison de toutes ces actions individuelles constitue une action jointe. La politique jointe peut alors être calculée par un algorithme classique comme *Value Iteration* ou *Policy Iteration*. Par conséquent, un MMDP peut être considérée comme un MDP à grand espace d'états et d'actions

Puisque la politique d'un MMDP est une politique centralisée, chaque agent doit avoir un accès à l'état global du système. En effet, une telle hypothèse n'est généralement pas vérifiée dans les systèmes multi-agents ce qui implique que les agents doivent communiquer à tout moment.

## 3.2 Le contrôle décentralisé

Appliquer la politique d'un MMDP de manière décentralisée est cependant possible si les agents peuvent communiquer à volonté et gratuitement, c'est-à-dire si la communication n'est pas limitée par des contraintes physiques telles que la taille de la bande passante, et qu'elle n'a aucune influence sur l'utilité des agents. Ces derniers peuvent alors échanger leurs informations sur leurs états afin de déterminer l'état global du système. Il est malgré tout nécessaire que l'état du système soit collectivement totalement observable. Sinon, les agents ne pourront pas déduire l'état du système même s'ils ont échangé toutes leurs informations.

Dans le contrôle décentralisé, la politique peut être calculée de deux manières. Centralisée où un contrôleur central est chargé de calculer la politique optimale jointe $\pi = \langle \pi_1, \dots, \pi_n \rangle$. Les politiques locales $\pi_i$ constituant cette politique optimale jointe sont ensuite envoyées aux agents chargés de les exécuter.

Décentralisée où chaque agent calcule individuellement la politique optimale jointe. Les agents obtiendront la même politique optimale jointe, s'ils suivent tous le même raisonnement. Cependant, s'il existe plusieurs politiques optimales, une phase de coordination est nécessaire afin d'éviter les problèmes liés à l'adoption de différents comportements (mauvaise coordination).

Les problèmes de contrôle décentralisé sont omniprésents dans le monde réel. Récemment, plusieurs domaines où ces problèmes apparaissent ont été étudiés. On pourrait citer par exemple, l'exploration spatiale avec la coordination de plusieurs robots [Washington et al., 1999], l'équilibrage de charge pour les files d'attente décentralisées [Cogill et al., 2004], vols coordonnés en hélicoptère [Pynadath and Tambe, 2002; Tambe, 1997], la gestion de réseaux de capteurs [Nair et al., 2005], etc. Tous ces problèmes se caractérisent par la présence de plusieurs décideurs qui ont un contrôle joint sur l'environnement mais qui ne peuvent pas partager toutes leurs informations à chaque étape de temps. Ainsi, des solutions reposant sur une centralisation des données ne sont pratiquement pas envisageables.

Afin d'illustrer ceci, nous présenterons ci-dessous trois problèmes qui ont été largement utilisés par les chercheurs de ce domaine.

### 3.2.1 Le problème du tigre multi-agents

Le problème le plus utilisé dans ce domaine est le problème du tigre multi-agents. Il a été d'abord introduit par [Kaelbling et al., 1998] pour les problèmes mono-agent modélisés par les POMDPs. La version multi-agents de ce benchmark a été introduite par [Nair et al., 2003]. Il s'agit de deux agents qui se trouvent face à deux portes fermées et doivent choisir laquelle ouvrir. Derrière une des portes un tigre sanguinaire et derrière l'autre un trésor (figure 3.1). La difficulté réside dans le fait que les agents, d'une part ne

savent pas où est initialement le tigre, et d'autre part n'ont qu'une observation partielle et bruitée de l'environnement.

Les agents peuvent gagner de l'information sur la position du tigre en écoutant derrière les portes plutôt que les ouvrir. Cependant, l'écoute a un coût et le résultat n'est pas nécessairement fiable (l'information sur la position du tigre n'est révélée qu'avec une certaine probabilité < 1). En outre, les agents ne peuvent pas communiquer leurs observations. A chaque instant, un agent décide d'écouter ou d'ouvrir une des portes. Si un des agents ouvre la porte avec le trésor derrière, les deux agents reçoivent la récompense. Si l'un d'eux ouvre la porte avec le tigre derrière, les deux agents sont pénalisés. Dans ces deux cas, si les agents coordonnent leurs actions, la récompense est alors augmentée et la pénalité est réduite. Les agents doivent donc trouver une politique jointe qui consiste à écouter jusqu'à finalement ouvrir une porte. Dès qu'un des agents ouvre une porte, le problème est réinitialisé.

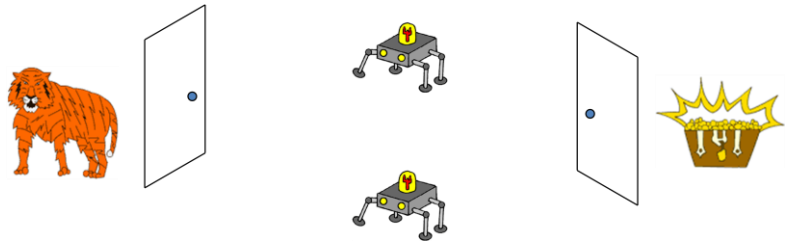

***Figure 3.1*** *Le problème du tigre multi-agents*

### 3.2.2 Rencontre sous incertitude

Ce problème a été initialement présenté par [Bernstein et al., 2000 ]. C'est une version simplifiée du problème réel de la planification multi-robot [Suzuki and Yamashita, 1999]. Dans ce problème, deux agents doivent se rencontrer le plutôt possible sur une grille 2D où des obstacles bloquent certaines parties de l'environnement. Chaque agent a 5 actions: nord, sud, est, ouest et rester dans la même position. A chaque étape du temps, un agent $i$ atteint l'emplacement désiré avec une probabilité $P_i$ et avec une probabilité de $1 - P_i$ reste sur la même position. Après avoir effectué une transition, l'agent peut percevoir certaine information. Cette information peut être simplement sa position comme elle peut être une information sur la topologie du terrain. Dans les deux cas, l'information partielle de l'agent n'est pas suffisante pour

déterminer l'état global du système. A cause des transitions incertaines des agents, la solution optimale est difficile à calculer puisque la stratégie de chaque agent dépend seulement de quelques croyances sur la position de l'autre agent. La solution optimale de ce problème consiste à une séquence de déplacements pour chaque agent tel que le temps attendu pour la rencontre est le plus petit temps possible. La figure 3.2 montre un exemple d'une grille de $8 \times 8$.

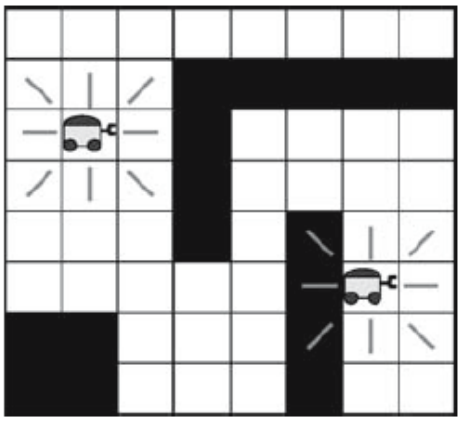

***Figure 3.2*** *Le problème de rencontre dans l'incertain [Seuken and Zilberstein, 2008]*

### 3.2.3 Partage de canal de communication

Le problème de partage de communication modélise deux agents communicants par un canal à bande passante limitée [Hansen et al., 2004]. Chacun des agents doit envoyer un ensemble de messages à l'autre agent via ce canal mais un seul message peut passer à la fois sur le canal, autrement une collision se produit. Les agents ont pour but de maximiser le nombre de messages échangés.

Comme l'illustre la figure 3.3, à chaque étape de temps, les agents doivent décider de communiquer ou non un message. Les deux agents reçoivent une récompense de 1 lorsqu'un message est transmis, et 0 si une collision est produite ou si aucun message n'est transmis. À la fin de chaque étape de temps, les agents reçoivent une observation parfaite sur le contenu de leur propre *tampon (buffer)*, et une observation bruitée s'il y a eu collision ou non en cas d'envoi de message. Le défi de ce problème réside dans le fait que les observations sont bruitées et que donc les agents ne peuvent construire qu'une croyance sur les résultats de leurs actions.

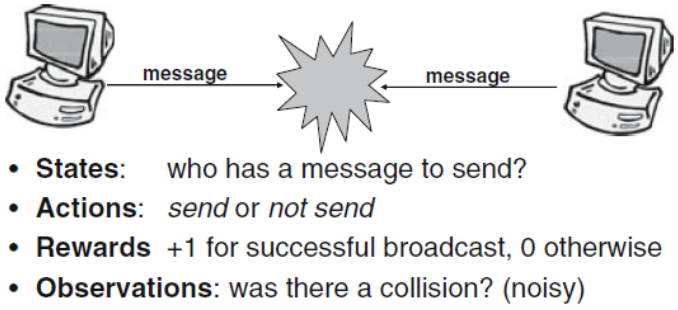


***Figure 3.3*** *Le problème de partage de canal de communication [Seuken and Zilberstein, 2008]*

## 3.3 Contrôle décentralisé et processus décisionnels de Markov

### 3.3.1 Formalisme des DEC-POMDPs

Les Processus Décisionnels de Markov Décentralisés Partiellement Observables (DEC-POMDPs) et les Processus Décisionnels de Markov Décentralisés (DEC-MDPs) constituent des extensions naturelles des formalismes POMDPs et MDPs à des problèmes où la prise de décision est distribuée. Ceci veut dire que plusieurs entités influencent le système en même temps. Chaque agent choisit son action en fonction d'une politique locale et possède ses propres actions et observations qui ne peuvent pas être partagées avec les autres agents, mais l'évolution du système (fonctions de transition, observation et de récompense) dépend de l'action jointe de tous les agents. Le modèle DEC-POMDP est donc prédestiné à la description de systèmes multi-agents. Il a été introduit par Bernstein et al. dans [Bernstein et al., 2000].

Les figures 3.4 et 3.5 ci-dessous décrivent un DEC-POMDP à horizon fini.

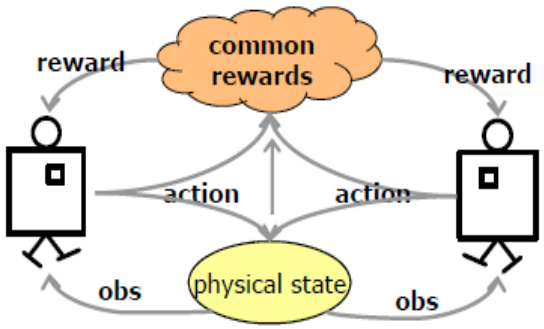


***Figure 3.4*** *Schématisation d'un DEC-POMDP[Doshi and Rabinovich, 2011]*

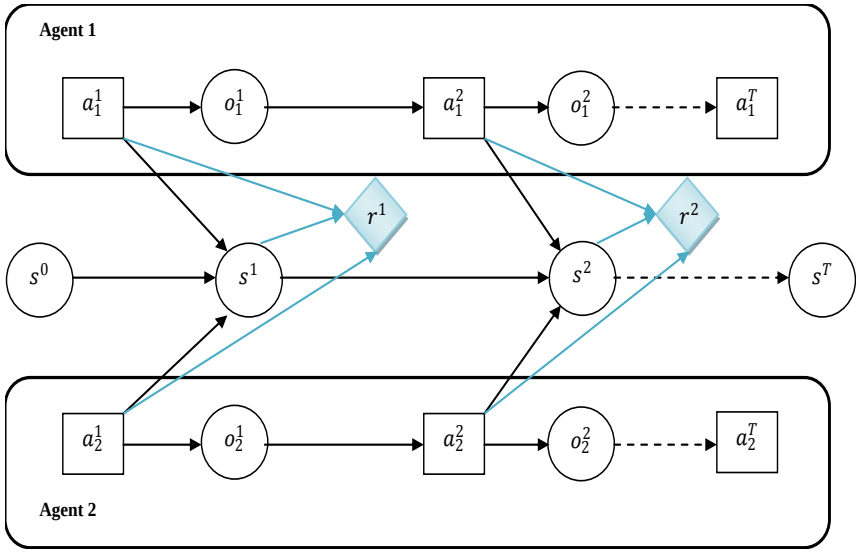


***Figure 3.5*** *Le formalisme du DEC-POMDP à horizon fini*

Un DEC-POMDP est défini par un tuple $\langle S, A, P, \Omega, O, R, T \rangle$, où

- $S$ est un ensemble fini d'états du système $s \in S$,
- $A = A_1 \times ... \times A_n$ est un ensemble fini d'actions jointes. $A_i$ dénote l'ensemble des actions $a_i$ pour l'agent $i$,
- $P: S \times A \times S \rightarrow [0,1]$ définit la fonction de transition. $P(s'|s,a)$est la probabilité de transition du système de l'état $s$ vers l'état $s'$après exécution de l'action jointe $a$,
- $\Omega = \Omega_1 \times ... \times \Omega_n$ est un ensemble fini d'observations jointes des agents. $\Omega_i$ dénote l'ensemble d'observations $o_i$ pour l'agent $i$,
- $O = S \times A \times S \times \Omega \rightarrow [0,1]$définit la fonction d'observation. $O(s,a,o,s')$ est la probabilité de l'observation jointe $o = \langle o_1, o_2, ..., o_n \rangle$ lors de la transition du système de l'état $s$ vers l'état $s'$après exécution de l'action joint $a$.
- $R(s,a)$ est la récompense produite par le système lorsque l'action jointe $a$ est exécutée dans l'état $s$,
- $T$ est l'horizon ou le nombre d'étapes après lesquelles le problème se termine.

Tout comme les POMDPs et les MDPs, les DEC-POMDPs sont définis sur un horizon $T$ fini ou infini. La fonction de transition ainsi que la fonction de récompense dépendent de l'action jointe exécutée par tous les agents: après exécution d'une action

jointe, une récompense est attribuée à l'état du système. Le but des agents étant de maximiser la somme des récompenses obtenues. Le processus qui considère les actions jointes comme actions atomiques, et qui ne permet donc pas une exécution décentralisée, est appelé POMDP sous-jacent au DEC-POMDP.

### 3.3.2 DEC-POMDP collectivement totalement observable

Un DEC-POMDP est dit collectivement totalement observable (DEC-MDP) quand l'état global du système peut être déduit à partir des observations locales cumulées des agents. L'état global du système est alors collectivement totalement observable. Formellement:

$$si\ \ O(s,a,s^{'},o=\langle o_1,o_2,\dots,o_n\rangle)>0\ \ alors\ P(s^{'}|\langle o_1,o_2,\dots,o_n\rangle)=1$$

Il faut noter que cette propriété n'implique pas nécessairement que chaque agent observe totalement son état local. De plus l'état global du système n'est forcément pas individuellement observable. Un DEC-MDP présente donc les mêmes composants qu'un DEC-POMDP. Dans le cas où les états des agents sont localement totalement observables, c'est-à-dire que $\forall o_i \in \Omega_i\ \exists s_i\ P(s_i = o_i) = 1$, l'ensemble des observations $\Omega$ et la fonction d'observation $O$ peuvent être omis afin d'éviter les redondances d'information. Le DEC-MDP est alors défini par l'ensemble $\langle S, A, P, R\rangle$ [Goldman and Zilberstein, 2004].

### 3.3.3 Le formalisme du MTDP

Le problème du contrôle décentralisé des processus décisionnels de Markov a également été formalisé via le modèle MTDP (*Multi-agent Team Decision Problem*) et introduit par [Pynadath and Tambe, 2002]. Ce formalisme alternatif a été prouvé équivalent au modèle DEC-POMDP [Seuken and Zilberstein, 2008].

L'une des différences fondamentales avec le modèle DEC-POMDP réside dans l'ajout de l'ensemble des croyances jointes $B$. L'état de croyances $b_i^t$ d'un agent $i$ décrit son état mental à l'instant $t$. Il est important de noter que l'état de croyance est différent de l'observation.

En effet, dans le formalisme des DEC-POMDPs, l'état de croyances d'un agent $i$ correspond à ses observations et la politique locale est une fonction $\pi_i : \Omega_i \rightarrow A_i$ qui fait correspondre une action à chaque observation de l'agent $i$. L'introduction des états de croyances dans la formalisation du problème, permet une modélisation plus riche de l'état mental de l'agent qui n'est plus seulement limité aux observations. La politique d'un agent devient alors une fonction $\pi_i: B_i \rightarrow A_i$ qui fait correspondre une action à chaque état de croyance de l'agent $i$.

Pour cette raison, il est possible que les solutions retournées par les deux formalismes soient différentes. En effet, il est possible que l'ensemble de croyances jointes considéré par le MTDP ne soit pas suffisant pour déterminer une politique jointe optimale pour l'ensemble des agents. Néanmoins, il est toujours possible de déterminer une politique jointe optimale pour l'ensemble des agents sur l'ensemble des croyances jointes considéré.

Un MTDP est défini par un tuple $\langle S, A, P, B_1, \dots B_n, O, \Omega, R \rangle$ tel que :

- $S$ définit l'ensemble des états du système.
- $A = \langle A_1, \cdots, A_n \rangle$ est l'ensemble des actions jointes et $A_i$ définit l'ensemble des actions $a_i$ de l'agent $i$.
- $P = S \times A \times S \rightarrow [0,1]$ dénote la fonction de transition. $P(s'|s,a)$ correspond à la probabilité que le système passe d'un état $s$ à un état $s'$ lorsque l'action jointe $a$ est exécutée.
- $B_i$ décrit l'ensemble des états de croyances de l'agent $i$. Chaque agent détient un ensemble d'états de croyances. Un agent $i$ construit son état de croyances $b_i^t$ à l'instant $t$ à partir de ses observations jusqu'à l'instant $t$.
- $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n$ est l'ensemble des observations des agents tel que $\Omega_i$ est l'ensemble des observations de l'agent $i$.
- $O = S \times A \times S \times \Omega \rightarrow [0,1]$ définit la fonction d'observation. $O(s, a, s', \langle o = o_1, \dots o_n \rangle)$ correspond à la probabilité que chaque agent $i$ observe $o_i$ lorsque les agents exécutent l'action jointe $a$ à partir de l'état $s$ et que le système arrive dans l'état $s'$.

- $R$ définit la fonction de récompense. $R(s, \langle a_1, \dots, a_n \rangle, s')$ est la récompense obtenue par le système lorsque les agents exécutent l'action $\langle a_1, \dots, a_n \rangle$ à partir de l'état $s$ et arrivent dans l'état $s'$.

Le modèle MTDP devient un DEC-POMDP quand l'état de croyances d'un agent se limite à ses observations.

Les relations entre les MTDPs, DEC-POMDPs, DEC-MDPs, POMDPs et MDPs peuvent être résumées par le diagramme de la figure 3.6 et le tableau 3.1.

| **Type de contrôle** | **Contrôle centralisé** | | **Contrôle décentralisé** | |
|---|---|---|---|---|
| **Etat global collectivement totalement observable** | **Oui** | **Non** | **Oui** | **Non** |
| **Formalise** | MDP | POMDP | DEC-MDP | DEC-POMDP<br>MTDP |

***Tableau 3.1.*** *Relation entre les DEC-POMDPs, DEC-MDPs, MTDPs, POMDPs et MDPs [Beynier, 2006]*

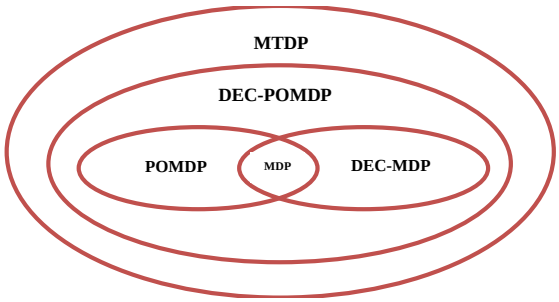


***Figure 3.6*** *Classification des différents types de processus décisionnels de Markov*

### 3.3.4 Complexité

Dans un DEC-POMDP, les décisions de chaque agent affectent tous les agents du domaine, mais dû à la nature décentralisée du modèle chaque agent doit choisir des actions basées seulement sur l'information locale. Résoudre un DEC-POMDP d'une

manière optimale est très difficile, car chaque agent reçoit une observation séparée qui ne fournit pas une information suffisante pour raisonner efficacement concernant les autres agents. Par exemple, chaque agent peut recevoir une information différente qui ne permet pas d'estimer l'état commun ou d'estimer les décisions des autres agents. Ces estimations sont cruciales dans les problèmes mono-agent, elles permettent de résumer l'historique des agents (états de croyances), mais généralement elles ne sont pas disponibles dans les DEC-POMDPs. Ce manque d'états d'estimation (et ainsi le manque des statistiques suffisantes) exigent des agents de se souvenir des historiques des actions et observations pour pouvoir agir d'une manière optimale. Cela veut dire aussi qu'on ne peut pas transformer les DEC-POMDPs en des *MDPs état de croyance* et on doit utiliser de différents outils pour les résoudre. Une autre manière pour considérer ce problème est qu'il existe un troisième type d'incertitude dans les DEC-POMDPs. Plus à l'incertitude sur les sorties des actions et l'incertitude sur les états, il existe aussi une incertitude sur les choix des autres agents et l'information qu'ils ont.

Cette différence peut être vue dans la complexité du problème à horizon fini avec au moins deux agents, qui est NEXP-complet [Bernstein et al., 2002] et ainsi dans la pratique peut nécessiter un temps doublement exponentiel. Ceci s'oppose avec les MDPs qui sont P-complet et les POMDPs qui sont PSPACE-complet. Comme les POMDPs à horizon infini, résoudre d'une manière optimale un DEC-POMDP est indécidable, mais les solutions $\epsilon$-optimales peuvent être trouvées avec temps et mémoire finis. Néanmoins, l'introduction de multiples agents décentralisés rendent le DEC-POMDP significativement plus difficile qu'un POMDP mono-agent.

La relation entre l'observabilité de l'état du système et la complexité en temps des modèles DEC-POMDP et MTDP est donnée dans le tableau 3.2.

| Observabilité de l'état du système | Individuellement observable | Collectivement observable | Collectivement partialement observable | Non observable |
|---|---|---|---|---|
| **Complexité** | P-complet | NEXP-complet | NEXP-complet | NP-complet |

***Tableau 3.2*** *Observabilité et Complexité en temps des DEC-POMDPs et MTDPs [Beynier, 2006]*

### 3.3.5 Gestion de la communication

Afin de mieux se synchroniser dans les systèmes à contrôle décentralisé, la communication semble une bonne décision. Cette décision va permettre aux agents de partager des informations

Il existe trois types de communication, *communication directe, communication indirecte et communication par observation commune de caractéristiques de l'environnement* [Goldman et Zilberstein, 2004].

1- *Communication indirecte :* Les actions de l'agent $i$ peuvent affecter les observations de l'agent $j$, ces observations peuvent servir comme des messages transmis par l'agent $i$. Ainsi, même si les agents ne communiquent pas directement, les dépendances entre les observations peuvent rapportées une information qui est partagée entre ces agents.

Il faut noter que quand on suppose qu'il existe des dépendances entre les observations qui résultent dans le partage d'information par la communication indirecte, le problème de contrôle décentralisé général inclut le problème de quoi communiquer et quand. On peut dire alors que la communication indirecte peut être considérée comme une conséquence d'une action qui est exécutée par un agent et les observations faite par les autres agents comme un résultat de cette action. Indépendamment de la politique, ce type de communication est limité au transfert de l'information concernant les caractéristiques de l'état.

2- *Communication directe* : L'information peut être partagée entre les agents s'ils peuvent envoyer directement des messages à un ou plusieurs agents. Dans ce cas, les observations peuvent être dépendantes ou indépendantes.
3- *Observation commune de caractéristiques de l'environnement :* ce sont les connaissances qui caractérisent l'environnement. Ces caractéristiques peuvent être observées par tous les agents mais qui ne sont pas affectées par leurs actions. En se basant sur ces observations, les agents peuvent décider de la manière de se coordonner et d'agir. Ce processus établit une communication implicite entre les agents. Dans ce type de communication, aucune décision sur la communication ou le contenu du message n'est prise. En effet, les agents ne sont que récepteurs de messages.

Nous nous intéresserons, dans ce chapitre (et dans notre travail), à l'influence de la communication sur l'observabilité des agents. Nous ne traiterons donc que de la communication directe. En effet, cette dernière constitue le seul type de communication qui permet d'atteindre l'observabilité totale lorsqu'il n'existe pas de caractéristiques incontrôlables communément observables par tous les agents.

#### 3.3.5.1 Le contrôle décentralisé avec communication

La communication directe peut être bénéfique dans le contrôle décentralisé car les agents ne peuvent pas observer totalement l'état global du système. Ceci veut dire que la valeur de la politique jointe optimale qui permet la communication peut être plus importante que la valeur de la politique jointe optimale sans communication. On ne s'intéresse ici qu'à la résolution hors ligne du problème de contrôle décentralisé. Les agents considèrent alors l'information attendue pendant le calcul de leurs politiques jointes optimales, ainsi dériver une politique pour quand et quoi communiquer.

Si on suppose que la communication directe mène à une observabilité totale de l'état du système, qu'elle est gratuite et que les observations sont indépendantes alors c'est évident que les agents vont bénéficier plus en communicant constamment. Cela nous mène à un processus décentralisé totalement observable, qui est équivalent à un MMDP. Ce problème est connu à être P-complet.

Dans les scénarios du monde réel, il est raisonnable de supposer que la communication directe a, en effet, un coût additionnel qui lui est associé, ce coût peut refléter le risque de révéler une information à des agents compétitifs, la bande passante nécessaire à la transmission ou même la complexité de calculer l'information à transférer. En conséquence, la communication peut ne pas être possible ou encore désirée.

Intégrer la communication entre les agents revient à ajouter un noveau type d'action: action de communication. A chaque étape de décision, un agent peut décider de communiquer ou non. Ensuite, il détermine quelle action réaliser. Comme le montre la figure 3.7, chaque étape de décision est donc décomposée en deux phases : la phase de communication et la phase d'exécution d'une action niveau domaine. L'action niveau domaine est toute action différente de la communication.

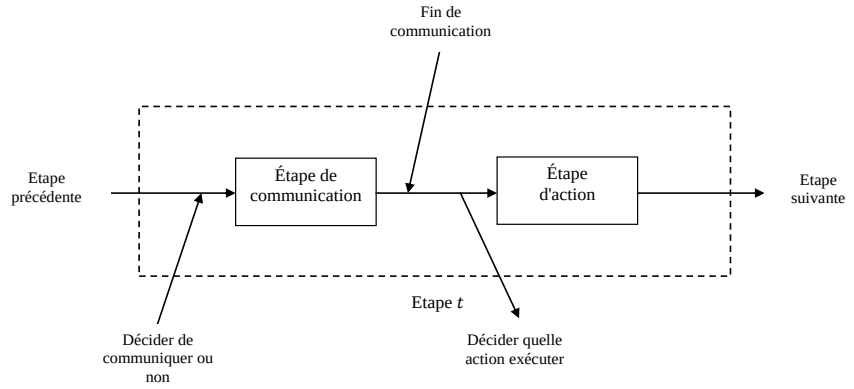


***Figure 3.7*** *Différentes phases d'une étape de décision*

Les DEC-POMDPs-COM (Processus Décisionnels de Markov Décentralisé Partiellement Observables avec Communication) proposés par Goldman et Zilberstein [Goldman and Zilberstein, 2003] sont définis de la même manière que les DEC-POMDPs. Deux nouvelles composantes sont cependant ajoutées au formalisme : un langage de communication Σ et une fonction de coût d'envoi des messages $C_\Sigma$.

- *DEC-POMDP avec communication*

Un DEC-POMDP-COM est défini par un tuple $\langle S, A, \Sigma, C_\Sigma, P, O, \Omega, R \rangle$ tel que :

- $S$ est l'ensemble des états du système.
- $\langle A = A_1, \dots, A_n \rangle$ est l'ensemble des actions jointes et $A_i$ définit l'ensemble des actions $a_i$ de l'agent $i$.
- $\Sigma$ désigne l'alphabet des messages. $\sigma_i \in \Sigma$ correspond à un message de l'agent $i$.
- $C_\Sigma$ associe à tout message un coût. Cette fonction est définie ainsi: $C_\Sigma: \Sigma \to \mathbb{R}$.
- $P = S \times A \times S \to [0, 1]$ définit la fonction de transition. $P(s'|s, a)$ correspond à la probabilité que le système passe d'un état $s$ à un état $s'$ lorsque l'action jointe $a$ est exécutée.
- $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n$ est l'ensemble des observations des agents tel que $\Omega_i$ est l'ensemble des observations de l'agent $i$.
- $O = S \times A \times S \times \Omega \to [0, 1]$ définit la fonction d'observation. $O(s, a, s', o = \langle o_1, o_2, \dots, o_n \rangle)$ correspond à la probabilité que chaque agent $i$ observe $o_i$ lorsque les agents exécutent l'action jointe $a$ à partir de l'état $s$ et que le système arrive dans l'état $s'$.
- $R$ dénote la fonction de récompense. $R(s, \langle a_1, \dots, a_n \rangle, s')$ est la récompense obtenue par le système lorsque les agents exécutent l'action $\langle a_1, \dots a_n \rangle$ à partir de l'état $\langle s_1, \dots, s_n \rangle$ et arrivent dans l'état $\langle s'_1, \dots, s'_2 \rangle$.

Selon [Goldman and Zilberstein, 2003], résoudre un DEC-POMDP-COM revient à calculer deux politiques locales par agent, une politique pour l'action niveau domain et une politique pour la communication, tel que leur exécution maximise la somme des récompenses attendues accumulées par le système.

La politique $\pi_i$ de chaque agent est alors composée de deux politiques: une politique de communication $\pi_i^\Sigma$ et une politique d'exécution des actions niveau domaine $\pi_i^A$. Son exécution est une sélection alternée d'une action à exécuter, et un message à envoyer. Cela veut dire que le système est toujours dans une phase d'action ou de communication.

Xuan et al. [Xuan et al., 2001] ont étendu le modèle DEC-MDP afin d'intégrer la communication.

Le modèle MTDP a aussi été étendu par Pynadath et Tambe [Pynadath et Tambe, 2002] afin de modéliser la communication. Un COM-MTDP (*Communicative Multi-agent Team Decision Problem*) est défini par un tuple $\langle S, A, \Sigma, P, \Omega, O, B, R \rangle$. La fonction de récompense $R$ est étendue afin de modéliser le coût des envois de messages. Ce formalisme est plus général que les DEC-POMDP-COM car il permet, tout comme les MTDPs, de représenter l'état mental des agents à l'aide des états de croyances.

Une question évidente est comment la communication peut affecter la complexité de génération des politiques. On peut distinguer entre trois types de communication [Pynadath and Tambe 2002; Goldman and Zilberstein, 2004]:

- Pas de communication, quand les agents ne disposent pas de la capacité de partager des messages ou la communication est très coûteuse de telle sorte qu'elle est impossible dans tous les cas.
- Communication ubiquitaire, où les agents peuvent échanger des messages sans coûts.
- Communication générale, où la capacité d'envoyer des messages existe, mais la communication induit un coût.

Dans l'absence de communication, la complexité des DEC-POMDPs et DEC-MDPs est NEXP-complet [Bernstein et al., 2002]. Il a été démontré que la communication gratuite transforme le DEC-POMDP en un POMDP (et le DEC-MDP en un MDP) avec complexité P-SPACE. En effet, si la communication est ubiquitaire, il est toujours préférable que les agents communiquent leurs observations à chaque étape de temps que de suivre une politique de communication qui choisit de ne pas partager certaines observations.

Cependant, la communication générale (avec coût) ne réduit pas la complexité du problème qui est NEXP-complet [Pynadath and Tambe 2002; Goldman and Zilberstein, 2004]. Ce résultat a renforcé la motivation des chercheurs d'explorer les approches heuristiques pour générer des politiques de communication, comme nous allons le discuter au chapitre 4. Le tableau 3.3 résume l'impact de la communication sur la complexité de génération de politiques optimales pour des équipes décentralisées.

| | **MMDP** | **DEC-MDP** | **DEC-POMDP** |
|---|---|---|---|
| **Pas de communication** | P-complet | NEXP-complet | NEXP-complet |
| **Communication générale** | P-complet | NEXP-complet | NEXP-complet |
| **Communication ubiquitaire** | P-complet | P-complet | PSPACE-complet |

***Tableau 3.3*** *L'impact de la communication sur la complexité des modèles décentralisés*

## 3.3.6 Propriétés et sous-classes des DEC-POMDPs

### 3.3.6.1 Motivation

Le contrôle décentralisé de plusieurs agents est NEXP-complet même dans le cas de deux agents avec une observabilité totale jointe. Ainsi, tous ces problèmes sont insolubles. En outre, il a été montré par *Rabinovich et al.* [Rabinovich et al., 2003] que même trouver des solutions $\epsilon$-optimales pour un DEC-MDP reste NEXP-complet. Pour faire face à la barrière de complexité, les chercheurs ont identifié des sous-classes spécifiques du DEC-POMDP qui leur complexité diffère de P à NEXP-complet. Certaines sous-classes ont un grand intérêt car elles décrivent parfaitement les problèmes du monde réel.

Quoique tous les problèmes introduits dans la section 3.2 représentent des processus de contrôle décentralisé et d'une complexité NEXP-complet dans le pire cas, ils restent différents dans le niveau de leur décentralisation. Dans certains problèmes décentralisés, les agents sont dépendants les uns des autres, leurs actions ont une grande influence sur les états futurs de chacun (exemple, partage de canal de communication). Dans d'autres classes, les agents résolvent des problèmes localement indépendants et interagissent entre eux rarement ou leurs actions ont seulement une petite influence sur les états des autres (exemple, rencontre sous incertitude, navigation des robots sur Mars).

Si un problème présente ce niveau de décentralisation plus formellement, une sous classe intéressante des DEC-MDPs émerge, appelée DEC-MDP à transition et observation indépendante [Becker et al., 2004a]. Dans ce modèle, les actions d'agents n'affectent pas les observations et les états locaux des autres agents. De plus, les agents ne peuvent ni communiquer ni observer les états et observations des autres agents. La seule façon pour que les agents interagissent entre eux c'est à travers la fonction de valeur globale qui rend le problème décentralisé, car il ne s'agit pas seulement de la somme des récompenses obtenues par chacun des agents mais d'une certaine combinaison non linéaire.

Un exemple du monde réel est le contrôle des opérations de plusieurs robots d'exploration planétaire, comme ceux utilisés par la NASA pour explorer la surface de Mars [Becker et al., 2004a]. Chaque robot explore et collecte des informations sur une région particulière. Périodiquement, les robots communiquent avec le centre de contrôle terrestre mais la communication continue n'est pas possible. Certains sites d'exploration peuvent s'imbriquer et si deux robots collectent de l'information sur ces sites (prendre des photos par exemple) la récompense est sub-additive. Pour d'autre partie de la planète, la récompense peut être super-additive, par exemple si les deux robots travaillent ensemble pour construire un modèle 3D d'une certaine zone.

#### 3.3.6.2 Les modèles formels

Comme mentionné ci-dessus, le niveau d'interaction entre les agents peut être capturé formellement. Dans ce qui suit, on va introduire ces sous-classes.

- *DEC-MDP à n-agents factorisé* est un DEC-MDP tel que l'état du monde peut être factorisé en $n + 1$ composantes, $S = S_0 \times S_1 \times \ldots \times S_n$.

Dans ce modèle, les caractéristiques qui appartiennent à un agent sont séparées de celles des autres agents et des caractéristiques externes. $S_0$ indique les caractéristiques externes que tous les agents observent et qui les affectent aussi, mais qui ne changent pas par les actions des agents. Dans l'exemple des robots explorateurs, ces caractéristiques peuvent être le temps ou le climat.

A partir de cette définition on doit indiquer les notions d'état local, action locale et observation locale. $s_i \in S_i \times S_0$ décrit l'état local, $a_i \in A_i$ est l'action locale et $o_i \in \Omega_i$ montre l'observation locale pour l'agent $i$.

- *DEC-MDP à transitions indépendantes*, un DEC-MDP factorisé à n-agents est dit à transitions indépendantes s'il existe $P_0$ jusqu'à $P_n$ tel que :

$$P\left(s_i^{'} \middle| (s_0, \dots, s_n), a, (s_1^{'}, \dots, s_{i-1}^{'}, s_{i+1}^{'}, \dots, s_n^{'})\right) = \begin{cases} P_0\left(s_0^{'} \middle| s_0\right) & i = 0 \\ P_i\left(s_i^{'} \middle| s_i, a_i, s_0^{'}\right) & 1 \leq i \leq n \end{cases}$$

Ce qui veut dire, les caractéristiques externes changent en se basant seulement sur les caractéristiques externes précédentes, et le nouvel état local de chaque agent dépend seulement de l'état local précédent, l'action prise et les caractéristiques externes courantes. Une indépendance similaire peut être formalisée pour la fonction d'observation.

- *DEC-MDP à observations indépendantes*, un DEC-MDP factorisé à n-agents est dit à observations indépendantes s'il existe $O_1$ jusqu'à $O_n$ tel que :

$$\forall o_i \in \Omega_i : O\left(o_i \middle| (s_0, \dots, s_n), a, (s^{'}{}_0, \dots, s^{'}{}_n), (o_1, \dots, o_{i-1}, o_{i+1}, \dots o_n)\right) = O\ \ (o_i | s_i, a_i, s_i^{'})$$

Ce qui veut dire, l'observation d'un agent reçue dépend seulement de l'état local, l'état futur et l'action courante de l'agent.

La figure ci-dessous décrit un DEC-MDP avec indépendance des transitions et des observations.

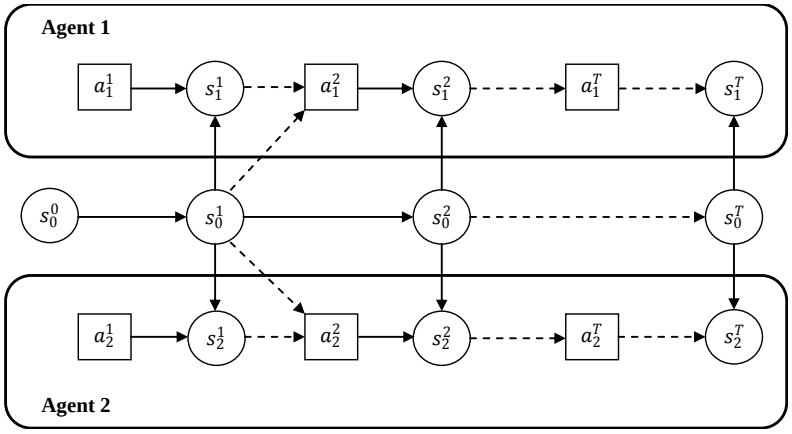


***Figure 3.8*** *DEC-MDP à horizon fini avec indépendance des transitions et des observations*

Dans un DEC-POMDP à transitions et observations indépendantes, seule la fonction de récompense qui dépend de l'action jointe. Toute dépendance entre les agents est capturée dans la fonction de récompense.

Goldman et Zilberstein [Goldman et Zilberstein, 2004] ont montré que la résolution d'un DEC-MDP à transitions et observations indépendantes nécessite que l'agent ne garde en mémoire que sa dernière observation, contrairement au cas général où l'agent doit garder en mémoire toute la séquence de ses observations. Ainsi, la taille des politiques locales des agents diminue. La politique d'un agent $i$ a alors une taille polynomiale en $|S_i| \times T$ et son évaluation peut être réalisée en temps polynomial. Cependant, il existe un nombre exponentiel $|A_i|^{|S_i| \times T}$ de politiques [Beynier, 2006]. De ce fait, Goldman et Zilberstein [Goldman et Zilberstein, 2004] ont prouvé que la résolution d'un DEC-MDP à transitions et observations indépendantes est un problème NP-complet.

Il faut noter que les propriétés d'indépendance (de transition et d'observation) ne permettent pas la réduction de la complexité dans le DEC-POMDP. En effet, chaque agent doit, dans tous les cas, mémoriser toute la séquence de ses observations et la taille des politiques locales reste inchangée.

- *DEC-MDP à récompense indépendante*, un DEC-MDP factorisé à n-agents est dit à récompense indépendante s'il existe $f$ et $R_1$jusqu'à $R_n$ tel que :

$$R\big((s_0, \dots, s_n), a, (s_0^{'}, \dots, s_n^{'})\big) = f\big(R_1(s_1, a_1, s_1^{'}), \dots, R_n(s_n, a_n, s_n^{'})\big)$$

Et

$$R_i(s_i, a_i, s_i^{'}) \leq R_i(s_i, a^{'}{}_i, s_i^{''}) \Leftrightarrow f(R_1 \dots R_i(s_i, a_i, s_i^{'}) \dots R_n) \leq f(R_1 \dots R_i(s_i, a^{'}{}_i, s_i^{''}) \dots R_n)$$

Cela veut dire que la récompense globale est composée d'une fonction des fonctions de récompense locales, chacune d'entre elles dépend seulement de l'état local et l'action locale de l'agent correspondant. Maximiser les fonctions de récompense locales individuellement est suffisant pour maximiser la fonction de récompense globale.

Il faut noter qu'un DEC-MDP à n-agents factorisé avec des transitions, observations et récompenses indépendantes peut être décomposé en $n$ MDPs indépendants locaux. Ainsi, le problème devient P-complet et non intéressant du point de vue contrôle décentralisé. Cependant, si l'une de ces relations d'indépendance est absente, le problème reste une sous classe non triviale des DEC-MDPs.

- *Orientation par des buts,* Un DEC-MDP est dit orienté par des buts (GO-DEC-MDP, *Goal Oriented* DEC-MDP) si les conditions suivantes sont satisfaites :

1. Il existe un sous ensemble non vide $G$ de l'ensemble des états $S$ représentant les états buts du système. Au moins un état $g \in G$ est accessible par une politique jointe.

2. Le problème a un horizon fini $T$.

3. Toute action $a_i$ d'un agent $i$ a un coût $C(a_i) < 0$.

4. La fonction de récompense est telle que $R(s, \langle a_1, \dots, a_n \rangle, s^{'}) = C(a_1) + \cdots + C(a_n)$.

5. Si à l'instant $T$, le système a atteint un état but $s \in G$ alors, une récompense supplémentaire est attribuée au système pour avoir atteint un état but.

- *DEC-MDP orienté par des buts à coût uniforme,* Un GO-DEC-MDP est dit à coût uniforme si toutes les actions ont le même coût.

La résolution de DEC-MDPs à transitions et observations indépendantes, ayant un seul but et un coût uniforme, est P-complet.

- *DEC-MDPs dirigés par les évènements*, Becker et al. [Becker et al., 2004b] ont identifié une autre sous-classe de DEC-MDPs: les DEC-MDPs dirigés par les événements ou *Event Driven Decentralized Markov Decision Processes* (ED-DEC-MDP).

Les ED-DEC-MDPs décrivent une classe des problèmes à observations et transitions dépendantes où les interactions entre les agents correspondent à des dépendances entre leurs actions. Ces dépendances s'interprètent par des contraintes d'exécution entre les actions des agents. En effet, les problèmes du monde réel présentent souvent des contraintes sur l'exécution des actions. Les ED-DEC-MDPs permettent de modéliser certaines contraintes contrairement au formalisme classique des processus décisionnels de Markov décentralisés. Des contraintes de précédences sont en effet générées et représentées à l'aide du formalisme TAEMS (*Task Analysis, Environment Modeling, and Simulation*) [Decker et al., 1993] de description des tâches. Il permet de décrire la structure des tâches et leurs relations. Des contraintes temporelles et des durées d'exécution des actions sont aussi considérées. Les ED-DEC-MDPs supposent que les états locaux des agents sont localement totalement observables. Enfin, l'ED-DEC-MDP est à récompenses indépendantes c'est-à-dire que sa fonction de récompense $R$ peut être décomposée en une somme de fonctions de récompense locales telles que $R\left(\langle s_1, \dots s_n\rangle, \langle a_1, \dots a_n\rangle, \langle s_1^{'}, \dots, s_n^{'}\rangle\right) = \sum_{i=1}^{n} R_i(s_i, a_i, s_i^{'})$.

La résolution d'un ED-DEC-MDP a été prouvée un problème NP-complet. En effet, un ED-DEC-MDP a une complexité exponentielle en espace d'états et doublement exponentielle en nombre de dépendances.

La figure 3.9 résume la complexité des sous-classes des DEC-POMDPs.

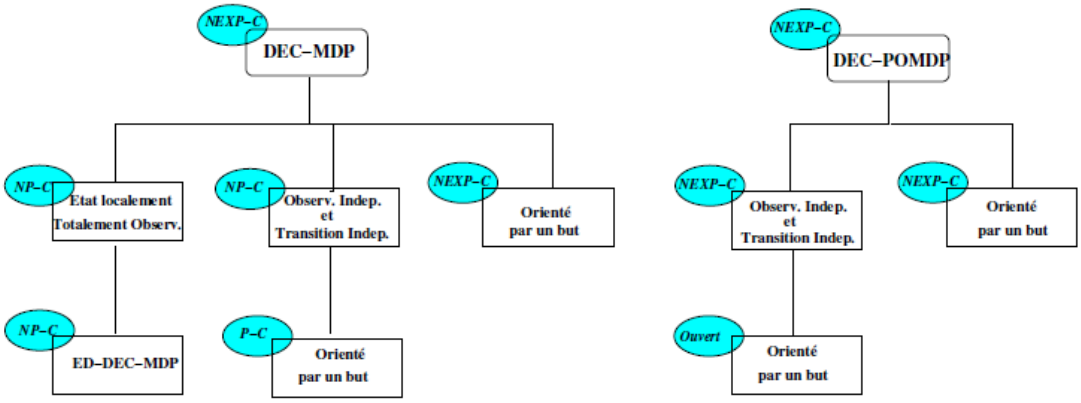


***Figure 3.9*** *Complexité des problèmes des sous-classes de DEC-MDPs et DEC-POMDPs [Beynier, 2006]*

### 3.3.7 Méthodes de résolution des DEC-POMDPs

Résoudre un DEC-POMDP revient à trouver un ensemble de politiques locales, appelé aussi politique jointe, tel que leur exécution décentralisée maximise un critère de performance associé, maximiser la récompense accumulée attendue du système par exemple. Il faut noter que cette récompense est la même pour tout le système. Il n'y a pas de récompenses individuelles, ce qui signifie implicitement que les agents sont coopératifs.

Chaque agent $i$ exécute une action locale $a_i$ basée sur sa politique locale. L'action jointe $a = \langle a_1, \dots, a_n \rangle$ change la configuration du système de son état courant $s$ à un nouvel état $s'$, et une observation jointe $o = \langle o_1, \dots, o_n \rangle$ est produite. Cependant, chaque agent a seulement accès à ses composantes locales $o_i$ de cette observation afin de choisir sa prochaine action.

Comme ça était mentionné à la section 3.3.4, quelque soit l'algorithme optimal utilisé pour résoudre un problème formulé comme un DEC-POMDP, un temps doublement exponentiel au pire cas sera nécessaire. Cependant en dehors des travaux effectués sur les algorithmes optimaux, il existe également des algorithmes utilisant des heuristiques qui, bien qu'ils n'offrent aucune garantie théorique, trouvent tout de même de « bonnes » politiques. Nous allons dans un premier temps nous intéresser aux

algorithmes optimaux existants, en vue de bien comprendre la difficulté de résoudre un DEC-POMDP. Nous verrons par la suite les hypothèses et les méthodes qui ont été mises de l'avant pour les approches d'approximation.

#### 3.3.7.1 Algorithmes de résolution exacte

Une politique optimale à horizon fini peut-être représentée par un arbre de décision $q$, où les noeuds sont marqués par des actions et les arcs par des observations. Une solution d'un DEC-POMDP à horizon $t$ peut être alors vue comme un vecteur $q^t$ d'arbres de décision de profondeur $t$ (appelés aussi arbres de politiques dans la littérature), $q^t = \langle q_1^t, \dots, q_n^t \rangle$, un pour chaque agent, où $q_i^t \in Q_i^t$. La figure 3.10 donne un exemple d'arbres de décisions joints pour un problème avec deux actions et deux observations à horizon 1 et 2 et montre comment le nombre de politiques possibles pour chacun des agents croît de manière exponentielle.

L'ensemble $Q_i^{t+1}$ des politiques de l'agent $i$ à l'horizon $t+1$ peut être construit à partir de l'ensemble des politiques à l'horizon $t$ en ajoutant une étape observation-décision à chaque feuille de l'arbre. Ainsi, pour chaque feuille de $q_i^t \in Q_i^t$, $|\Omega|$ nouvelles feuilles filles sont ajoutées, et pour chacune d'entre elles, une décision est choisie. Le nombre total d'arbres de politique pour un agent est le nombre de combinaison d'assignation d'actions possible étant donné tous les historiques possibles. Pour l'horizon 1, une seule décision doit être prise donc $|A|$ politiques peuvent être suivies. Pour l'horizon 2, pour chaque observation possiblement obtenue après la première action, il va falloir prendre une décision. $|\Omega|$ branches sont donc ajoutées à chacune des politiques initiales, et toutes les assignations d'actions au second niveau sont alors possibles. Ceci résulte en $|A|^{(1+|\Omega|)}$politiques possibles à l'horizon 2. En continuant de manière similaire on trouve qu'il faudra encore ajouter un troisième niveau étant donné chaque séquence possible de deux observations pour se retrouver avec $|A|^{(1+|\Omega|+|\Omega|^2)}$ politiques possibles. L'opération est suivie jusqu'à l'horizon $t$. On voit donc que l'algorithme est doublement exponentiel en l'horizon $t$.

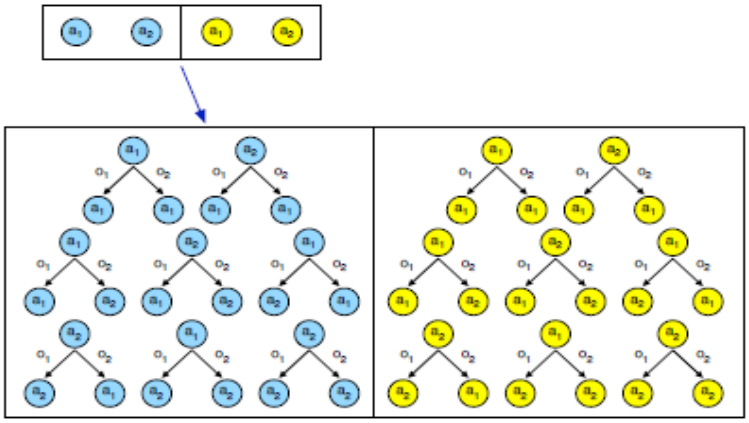

***Figure 3.10*** *L'étape de génération exhaustive quand on passe des politiques de l'horizon 1 aux politiques de l'horizon 2 [Allen, 2009]*

Malheureusement, le problème est également exponentiel en nombre d'agents, puisque le nombre de politiques jointes est le produit des ensembles de politiques de chacun des agents.

Deux différents algorithmes ont été développés pour rechercher la meilleure politique jointe dans cet espace incommensurable : le premier basé sur la programmation dynamique, élimine à chaque étape les politiques qui ne pouvent jamais apporter mieux qu'une autre politique (elles sont alors dites dominées). Le deuxième est, quant à lui, basé sur la recherche arborescente heuristique (A*). Ces deux algorithmes sont détaillés dans ce qui suit.

- *Programmation dynamique*

Dans cette section, on va présenter l'algorithme de programmation dynamique proposé par [Hansen et al., 2004] pour les DEC-POMDPs, qui est le premier algorithme de programmation dynamique générale pour résoudre les DEC-POMDPs d'une manière optimale.

L'idée principale de cette technique consiste à parcourir l'ensemble des politiques de manière incrémentale, en éliminant le plus tôt possible les politiques dominées, évitant ainsi de multiplier exagérément le nombre de politiques possibles. Cependant, du fait que l'élimination de la politique d'un agent peut influencer la meilleure politique

d'un autre agent, l'élimination de politique doit être faite itérativement jusqu'à ce qu'aucun agent ne puisse plus éliminer aucune politique. Ainsi, la programmation dynamique pour les DEC-POMDPs entrelace la génération exhaustive des politiques à l'horizon $t+1$ sachant l'horizon $t$, et l'élimination des politiques dominées dans le nouvel ensemble des politiques possibles pour chacun des agents. Dès que l'algorithme atteint un horizon voulu $T$, une politique est alors choisie, celle qui garantie la meilleure valeur espérée étant donné l'état de croyance initial.

Dans le pire cas, l'algorithme reste doublement exponentiel selon l'horizon voulu. Cependant, l'amélioration par rapport à un algorithme de force brute (*brute force search*) réside dans l'élimination, à chaque étape, des politiques dominées. Bien entendu, une telle élimination dépend du problème étudié. La figure 3.11 décrit l'étape d'élagage des politiques dominées.

Les expérimentations montrent que l'algorithme est beaucoup plus efficace que l'évaluation de toutes les politiques en utilisant la force brute. Cependant, le nombre de politiques à évaluer augmente fortement avec l'horizon ce qui entraîne rapidement une limitation de la taille des problèmes traités. Ainsi, Hansen et al. arrivent à résoudre au maximum jusqu'à l'horizon 4 (4 décisions par agent), des problèmes à 2 agents, 4 états, 2 actions par agent et 2 observations par agent.

Dans le cas des DEC-POMDPs à horizon fini, cet algorithme permet d'obtenir une solution optimale. Si l'horizon est infini, nous aboutissons à un ensemble d'équilibres.

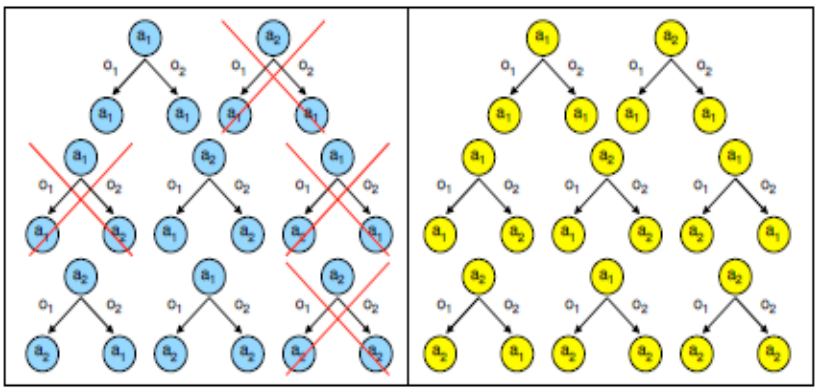

***Figure 3.11*** *Etape d'élimination des politiques dominées [Allen, 2009]*

- *MAA**

Une autre approche appelée *multi-agent* A*, proposée par Szer et al. [Szer et al., 2005], est radicalement différente de la programmation dynamique vue précédemment. Au lieu de calculer les arbres de politiques du bas en haut comme dans les méthodes de programmation dynamique, ils sont construits du haut en bas. Elle consiste à rechercher par profondeur d'abord, les sous politiques les plus prometteuses à la manière d'un algorithme de type A* [Nilsson, 1980]. Ce ne sont donc pas tous les nœuds de l'arbre qui sont complètement explorés; à la place, une heuristique est utilisée pour évaluer les feuilles de l'arbre de recherche. Et c'est le nœud ayant la plus haute estimation qui est exploré à l'étape suivante.

Cette recherche est menée en utilisant une limite supérieure pour la valeur de différentes politiques et ensuite choisir les politiques dans un meilleur ordre. Une étape de la politique est alors fixée et une limite supérieure sur cette politique partielle est trouvée. Encore, la politique qui a la plus grande valeur est choisie et un autre choix d'action est fixé. Cela continue jusqu' à une politique définie totalement est trouvée qui a une valeur qui est supérieure aux valeurs des politiques partielles.

L'efficacité de l'algorithme est étroitement liée à l'heuristique utilisée. Plusieurs heuristiques ont été décrites afin de calculer rapidement une approximation des politiques, et ainsi déterminer quelle solution doit être développée en priorité. Il est proposé de réduire le DEC-POMDP à un MDP ou à un POMDP centralisé pouvant être résolu plus facilement.

Plus l'heuristique sera précise, plus la solution sera trouvée rapidement. Dans le pire cas, la complexité de cet algorithme est équivalente à celle de la recherche exhaustive.

Les résultats expérimentaux montrent que cet algorithme est plus efficace que l'élimination itérative des politiques dominées proposée par Hansen et al. puisque MAA* stocke moins de politiques en mémoire. Ainsi, dans des problèmes où Hansen et al. [Hansen et al., 2004] peuvent aller jusqu'à l'horizon 4, MAA* calcule une solution

jusqu'à l'horizon 5. Au delà, MAA* se trouve également confronté à une explosion de la mémoire nécessaire à la résolution du problème.

D'autres versions de MAA* ont été aussi proposées en utilisant les jeux bayésiens pour fournir une heuristique améliorée. Les travaux récents de l'utilisation de la version généralisée du MAA* avec le clustering [Oliehoek et al., 2009] améliorent le passage à l'échelle en représentant les politiques des autres agents d'une manière concise sans perdre les valeurs.

**Résolution des DEC-MDPs à transitions dépendantes**

Les approches précédentes s'intéressent au problème général des DEC-POMDPs dont la complexité est NEXP. Cependant, nous avons vu précédemment que certaines propriétés du problème permettaient de diminuer sa complexité. Il peut alors être plus facile de calculer la solution optimale.

Becker et al. ont proposé l'algorithme CSA (*Coverage Set Algorithm*) qui permet de résoudre des DEC-MDPs à transitions et observations indépendantes [Becker et al., 2004a], mais également des DEC-MDPs dirigés par les événements [Becker et al., 2004b].

Cet algorithme est principalement divisé en trois phases: la première consiste à la création de MDPs augmentés. Un MDP augmenté représente un problème de décision dans lequel un agent doit calculer sa politique optimale sachant les politiques fixes des autres agents. La deuxième phase consiste à trouver l'ensemble de couverture optimal (*optimal coverage set*) pour les MDPs augmentés qui est l'ensemble de toutes les politiques optimales d'un agent quelle que soient les politiques des autres agents. Trouver cet ensemble permet de réduire l'espace de recherche des politiques et d'améliorer les performances par rapport à une recherche exhaustive.

Lors de la dernière phase, l'algorithme trouve pour chaque politique dans l'ensemble de couverture optimal, les meilleures politiques correspondantes pour les autres agents. La politique jointe trouvée est évaluée et la meilleure combinaison correspond à la politique optimale jointe

La complexité de cet algorithme est exponentielle en espace d'états. Dans le cadre de la résolution des ED-DEC-MDPs, la complexité est également doublement exponentielle en nombre de dépendances. Dans le pire cas, CSA a la même complexité que la recherche exhaustive.

#### 3.3.7.2 Les méthodes de résolution approchée

Les algorithmes exacts présentés précédemment démontrent les limites de la résolution optimale, dues au temps et à l'explosion de la mémoire, même quand il s'agit de problèmes à horizon fini. De plus, il est connu qu'en général résoudre un DEC-POMDP à horizon infini d'une manière optimale n'est pas réalisable, puisque résoudre un POMDP mono-agent à horizon infini est déjà indécidable. C'est pour cette raison que les solutions approximatives ont été introduites qui permettent de résoudre des problèmes à horizon fini ou infini plus large.

- *Accélérer la programmation dynamique*

Certains travaux avaient comme objectif d'accélérer la programmation dynamique pour la génération de politique pour les DEC-POMDPs. Szer et Charpillet définissent un état de croyance multi-agents pour chaque agent qui consiste en une distribution de probabilité sur les états du monde et une croyance sur le comportement futur des autres agents [Szer and Charpillet, 2006]. Inspirer par les techniques basées échantillonnage qui ont été utilisées pour résoudre le POMDP mono-agent (e.g. [Pineau et al., 2006]), la programmation dynamique basée point pour les DEC-POMDPs, échantillonne les croyances probables de l'ensemble d'états de croyance multi-agents possibles et génère des politiques en se basant sur ce sous-ensemble.

La programmation dynamique à mémoire limitée (MBDP: *Memory Bounded Dynamic Programming*) utilise des heuristiques haut-bas pour rechercher les états de croyances probables [Seuken and Zilberstein, 2007]. Ces états de croyance sont ensuite utilisés pour élaguer les sous-arbres de politiques générés à chaque étape de temps de la programmation dynamique, de telle sorte que seulement un nombre fixe de politiques possibles est préservé. De cette manière, MBDP limite la mémoire utilisée par la programmation dynamique. S'il y a peu d'états de croyances atteignables, cet élagage

peut être très efficace, permettant de calculer des politiques pour des horizons plus longs que celles calculées avec un algorithme optimal de programmation dynamique.

Une autre approche est celle proposée par Bernstein et al. [Bernstein et al., 2005], *Bounded Policy Iteration* pour les POMDPs décentralisés (DEC-BPI) qui est une extension de l'algorithme BPI proposé par Poupart et Boutilier [Poupart and Boutilier, 2003] pour le POMDP mono-agent. Comme le BPI, DEC-BPI est un algorithme approximatif qui cherche à améliorer les contrôleurs stochastiques à états finis. Il améliore un ensemble de contrôleurs de taille fixe en utilisant un programme linéaire. Cela est réalisé en itérant à travers les nœuds de chaque contrôleur d'agent et essayer de trouver une amélioration. L'algorithme et le programme linéaire sont très similaires à ceux du BPI à l'exception de l'optimisation qui est faite sur non pas seulement les états du système mais les nœuds des contrôleurs des autres agents (les états de croyances généralisés).

Le programme linéaire cherche une distribution de probabilité sur les actions et les transitions dans le contrôleur courant de l'agent. Si une amélioration est découverte, le nœud est mis à jour en se basant sur les distributions de probabilités trouvées. Chaque nœud pour chaque agent est examiné et l'algorithme se termine quand il n'y a plus d'agent qui peut réaliser des améliorations.

- *Trouver des politiques localement optimales*

D'autres algorithmes se sont intéressés à trouver des politiques localement optimales. Un exemple de ces algorithmes est (JESP: *Joint Equilibium-based Search for Policies*) [Nair et al., 2003]. Dans cette méthode les agents améliorent leurs politiques à tour de rôle en trouvant la meilleure réponse aux politiques fixes des autres agents (leurs co-équipiers). Ce processus est répété d'une manière itérative jusqu'à convergence quand aucun agent ne peut améliorer la récompense de l'équipe en changeant seulement sa propre politique. JESP est garanti de converger, mais pas pour un optimum global parce que les agents sont coopératifs et changent leurs politiques seulement quand ce changement apporte un gain dans la récompense attendue de l'équipe. En pratique, JESP converge rapidement vers un optimum local correspondant à un équilibre de Nash

[Nash, 1950], cependant, dans le pire cas, l'algorithme peut itérer toutes les politiques possibles.

Par la suite une amélioration du JESP a été proposée [Nair et al., 2003] en utilisant la programmation dynamique: DP-JESP (*Dynamic Programming Joint Equilibium-based Search for Policies*). DP-JESP peut traiter des problèmes plus grands que ceux résolus par JESP en raison de l'utilisation du principe d'optimalité pour l'évaluation des politiques. Sur le scénario du tigre à deux agents, JESP résout le problème jusqu'à l'horizon 3 et DP-JESP atteint l'horizon 7. Cependant, ces problèmes restent assez petits par rapport à des problèmes réels.

Une autre extension de l'algorithme JESP est LID-JESP (*Locally Interacting Distributed* JESP), utilise les techniques d'optimisation à contrainte distribuée afin d'accélérer le calcul de la politique dans les domaines où l'interaction entre agents est localisée, ce qui veut dire que les fonctions de transition et d'observation de chaque agent dépendent des actions d'un sous-ensemble des agents co-équipiers [Nair et al., 2005]. Bien que, la majorité des algorithmes de résolution des DEC-POMDPs commencent par une distribution des probabilités sur les états du monde, une autre extension du JESP, CS-JESP (*Continuous [Belief] Space JESP*) résout le DEC-POMDP avec des distributions de croyances initiales arbitraires [Varakantham et al., 2005]. CS-JESP combine JESP avec les techniques de résolution d'un POMDP mono-agent pour trouver des politiques basées sur les croyances et accélérer le calcul par élagage des états de croyances inatteignables.

#### 3.3.7.3 D'autres méthodes de résolution

Il existe d'autres méthodes de résolution pour les DEC-POMDPs qu'on va présenter brièvement dans cette section. [Aras et al., 2007] ont proposé une formulation de *mixed integer linear programming* (MILP) pour la résolution optimale des DEC-POMDPs à horizon fini. Leur approche est basée sur la représentation de l'ensemble de politiques possibles pour chaque agent sous forme de séquence. Dans cette séquence, une politique d'un agent $i$ est représentée comme un sous-ensemble de l'ensemble de séquences (souvent correspond aux historiques action-observation) pour l'agent. Ce

problème peut être interprété par un problème d'optimisation combinatoire, qui Aras et al. proposent de le résoudre en utilisant MILP.

[Oliehoek et al., 2007a, 2008a] ont prouvé aussi que trouver une solution pour les DEC-POMDPs est un problème d'optimisation combinatoire et proposent d'appliquer la méthode de *Cross-Entropy*, une méthode d'optimisation combinatoire qui est devenue populaire grâce à sa capacité de trouver des solutions proche de l'optimale dans des problèmes d'optimisation assez larges. L'algorithme résultant DICE effectue un échantillonnage basé recherche de politique pour résoudre les DEC-POMDPs d'une manière approximative.

Emery-Montemerlo et al. [2004, 2005] proposent d'approcher les DEC-POMDPs en des séries de jeux bayésiens.

[Peshkin et al., 2000] ont étudié comment approximer la solution décentralisée en utilisant l'apprentissage en ligne quand les agents n'ont pas d'information sur le modèle de leur environnement. Guestrin et al. ont étudié les approximations hors ligne qu'ils soient centralisées [Guestrin et al., 2001] ou distribuées [Guestrin and Gordon, 2002], où les dépendances connues entre les actions des agents induisent une structure de passage de message. Dans ce contexte, les agents choisissent leurs actions à tour de rôle et la communication est supposée d'être gratuite, ce qui limite l'efficacité du travail. [Amato et al., 2006, 2007] étudient les techniques de programmation linéaire et non-linéaire pour générer des contrôleurs à états finis de taille fixe pour les DEC-POMDPs. Ces contrôleurs mènent généralement à des solutions sous-optimales, cependant, il a été prouvé qu'ils représentent de meilleures approximations que les autres approches présentées ci-dessus.

[Oliehoek et al., 2013] proposent un ensemble de méthodes de résolution optimale des DEC-POMDPs basée sur l'algorithme GMAA* (*Generalized Multi-agent* A*) [Oliehoek et al., 2009] qui modélise le problème du DEC-POMDP par les jeux bayésiens collaboratifs (CBGs: Collabora*tive Bayesian Games*). Les auteurs proposent d'utiliser une technique incrémentale de clustering afin de réduire la taille des CBGs. Cela peut réduire le nombre de nœuds fils dans l'arbre de recherche GMAA* ce qui augmente l'efficacité de l'algorithme GMAA*. Une expansion incrémentale a été

appliquée, par la suite, sur les nœuds de l'arbre de recherche afin de ne créer un nœud fils que quand il est candidat d'une autre expansion.

Un algorithme approximatif basé sur les algorithmes génétiques est proposé par [Mazurowski and Zurada, 2007]. L'approche évolutionnaire introduite est utilisée comme une technique de recherche de la politique jointe du DEC-POMDP.

Plus récemment, des modèles qui permettent de limiter les dépendances de transition et de récompense ont été introduits. On peut citer, les jeux de Markov dirigés interaction [Spaan and Melo, 2008], DEC-MDPs avec peu d'interactions [Melo and Veloso, 2011], POMDP distribué avec coordination locale [Varakanthan et al. 2009; Velagapudi et al. 2011], EDI-CR (*Event-driven Interaction with Complex Reward*) [Mostafa and Lesser, 2011] et *transition decoupled DEC-POMDPs* [Witwicki and Durfee, 2010; Witwicki 2011]. Pendant que les méthodes développées pour résoudre ces modèles assurent une meilleure scalabilité que les méthodes standards des DEC-POMDPs, ils ne sont pas appropriées pour les problèmes où les interactions entre agents sont étendues. D'autres modèles ont été proposés afin de considérer le facteur du temps et des actions que leur ordre est déjà déterminé [Marecki and Tambe 2007; Beynier and Mouaddib, 2011].

## 3.4 Discussion

Dans les sections précédentes nous avons recensé quelques approches existantes pour la résolution des DEC-POMDPs. Chaque approche se caractérise par un ensemble d'hypothèses et de règles posées sur les problèmes étudiés. Le tableau 3.4 établit une classification des différentes extensions des DEC-POMDPs ainsi que leurs premiers algorithmes de résolution.

Nous avons différencié entre les DEC-POMDPs et les DEC-MDPs (où l'observabilité totale jointe est supposée). Dans un cas pareil les agents n'ont pas une observabilité totale de l'état global du système mais ils ont une observabilité totale jointe. Donc, chacun des agents a une croyance sur l'état global du système, qui est représentée par une distribution de probabilités sur les états globaux.

| Modèle | Résolution optimale | Référence | Résolution approchée | Référence |
|---|---|---|---|---|
| **DEC-POMDP** | Programmation dynamique pour DEC-POMDPs | [Hansen et al., 2004] | Bounded Policy iteration | [Bernstein et al., 2005] |
| | MAA* | [Szer et al., 2005] | Apprentissage par descente de gradient | [Peshkin et al., 2000] |
| | | | Jeux bayésiens | [Emery-Montemerlo et al., 2004] |
| **MTDP** | - | - | JESP, DP-JESP | [Nair et al., 2003] |
| **TI, OI, DEC-POMDP** | CSA | [Becker et al., 2004a] | - | - |
| **IT, IO GO-DEC-MDP ($\|G\| = 1$) et avec coût uniforme** | Opt1Goal | [Goldman et al., 2004] | - | - |
| **IT,IO GO-DEC-MDP ($\|G\| > 1$) et avec coût uniforme** | OptNGoal | [Goldman et al., 2004] | - | - |
| **ED-DEC-MDP** | CSA | [Becker et al., 2004b] | - | - |

***Tableau 3.4*** *Classification des extensions des DEC-POMDPs et leurs premiers algorithmes de résolution*

Bien qu'il ait été prouvé que le formalisme présenté dans ce chapitre est très puissant et expressif pour la prise de décision séquentielle décentralisée et sous incertitude, il présente quelques limites par rapport à la modélisation des problèmes réels.

En effet, les problèmes décentralisés présentent en général un certain nombre de contraintes sur l'exécution des actions. Ces contraintes varient selon le type de problème étudié. Elles peuvent concerner le temps, comme elles peuvent être des contraintes d'ordre entre l'exécution des actions de différents décideurs. De plus, ces décideurs doivent, parfois, gérer des contraintes physiques, comme la limitation de communication ou de ressources. Le modèle DEC-POMDP de base ne permet pas la prise en compte de toutes ces contraintes. Par conséquent, les approches de résolution proposées ne sont pas adaptées aux problèmes réels.

La première extension du DEC-POMDP qui permet la gestion d'une certaine classe de contrainte est l'ED-DEC-MDP proposée par [Becker et al., 2004b]. Néanmoins, Becker propose une résolution exacte de ces problèmes ce qui limite l'applicabilité de son approche à des problèmes de petite taille.

[Beynier and Mouaddib, 2006; Marecki and Tambe, 2007] ont à leurs tours essayé de gérer quelques contraintes d'exécution et améliorer le passage à échelle (*scalability*) des DEC-MDPs [Beynier and Mouaddib, 2011].

Afin de rapprocher nos travaux aux problèmes réels et de pouvoir couvrir un large spectre d'application, il nous semble primordial de continuer sur le même axe de recherche. Les objectifs que nous fixons pour cette thèse concernent la proposition d'une extension des DEC-POMDPs pour la gestion de contraintes d'exécution.

## 3.5 Conclusion

Dans ce chapitre, nous nous sommes intéressées à la prise de décision séquentielle décentralisée où plusieurs acteurs sont en coopération. Nous avons décrit le modèle DEC-POMDP ainsi que ses différentes extensions.

Nous avons détaillé par la suite, les différentes approches existantes pour la résolution de DEC-POMDPs. Les approches de résolution exactes et approximatives ont

été différenciées ce qui nous a permis d'illustrer les limitations du calcul de la solution optimale. Nous nous sommes alors intéressées à l'applicabilité de ces méthodes dans des systèmes réels.

De plus, nous avons pu mettre en avant les déficits des modèles existants quant à la gestion des contraintes du problème ce qui limitent leur applicabilité. Afin de rapprocher ces modèles aux problèmes réels, il nous parait nécessaire de pouvoir modéliser un certain nombre de contraintes.

Nous avons montré aussi que les DEC-POMDPs et les DEC-MDPs peuvent supporter la communication comme un aspect particulier de partage d'information entre les agents ainsi que les différentes issues qui lui sont associées.

Les deux volets: modélisation de contraintes et communication constituent le sujet de cette thèse. Ainsi, dans le chapitre suivant, nous classifierons les travaux existants dans la littérature qui couplent la gestion de contraintes et la communication dans les modèles de la théorie de la décision.

# Chapitre 4

# *Travaux connexes*

*Après avoir introduit dans les deux chapitres précédents le contexte général et les concepts de base de notre travail, nous allons à présent établir le lien entre l'état de l'art que nous venons de réaliser et nos travaux. Ceci va nous permettre de bien cerner notre problématique avant de présenter nos contributions.*

*Dans ce chapitre nous essaierons de recenser plusieurs travaux qui sont en relation avec le travail que nous présentons dans cette thèse. En effet, notre travail se situe à l'intersection de plusieurs axes de recherches à savoir, la gestion des contraintes sur l'exécution des actions dans les problèmes de contrôle décentralisés ce qui fait appel aux techniques du scheduling et la planification stochastique, la communication dans les DEC-POMDPs et les DEC-MDPs, son rôle dans la résolution de ces modèles ainsi que son impact sur leur complexité, et la fiabilité de la communication.*

*Comme nous nous attaquons à des problèmes complexes nécessitant une coordination plus étroite, nous ne pouvons pas ignorer la possibilité de besoin pour la communication entre les décideurs. Même lorsque chaque agent connait la politique initiale de chacun des autres agents, l'incertitude sur les sorties des actions peut créer une ambiguïté sur les états des autres agents et par conséquent ce qu'ils vont faire dans l'avenir. Cette ambiguïté introduit la nécessité d'une coordination au cours de l'exécution d'une politique distribuée. La communication soulève d'importants problèmes où nous pouvons bénéficier de leurs structures spécifiques et leurs interactions structurées.*

*Nous présentons tout d'abord les travaux portants sur la gestion des contraintes dans les problèmes de contrôle décentralisé. Par la suite, nous donnerons un*

*survey des travaux sur la communication dans les modèles de la théorie de la décision.*

## 4.1 Les contraintes d'exécution dans les problèmes de contrôle décentralisé

Plusieurs problèmes décentralisés du monde réel impliquent l'interaction de plusieurs agents afin de réaliser un ensemble de tâches. Lors de la réalisation de ces tâches plusieurs contraintes peuvent être envisagées: des contraintes temporelles, des contraintes de ressources, plusieurs durées d'exécution des actions, des contraintes sur l'ordre d'exécution, l'incertitude, …

Dans ce contexte, "la planification stochastique et l'ordonnancement de tâches sous contraintes", plusieurs travaux ont été réalisés et qui se sont focalisés sur les méthodes markoviennes et les méthodes de planification classique telle que STRIPS [Fikes and Nilson, 1971] qui représente le premier système à introduire la représentation des actions avec pré-conditions/effets.

Depuis, plusieurs formalismes et algorithmes ont été développés pour gérer ce problème et en particulier les contraintes et l'incertitude envisagée.

Plusieurs approches ont été développées afin de faire face aux incertitudes dans le *scheduling* (l'ordonnancement). [Beck and Wilson, 2007] ont adressé le problème du *job shop scheduling* avec des durées des actions probabilistes. Chaque durée est représentée par une variable aléatoire avec un écart type et une variance connus. Pour résoudre ce problème, les algorithmes du *scheduling* déterministe ont été combinés avec la simulation Monte Carlo. L'incertitude dans [Lambrechts et al., 2008] est modélisée par la non-disponibilité de ressources. [Lazarova-Molnar and Mizouni, 2010] adressent des incertitudes sur la durée des tâches, l'allocation des tâches et les décisions arbitraires prises à la volée (*arbitrary on-the-fly decisions*).

Le problème d'ordonnancement et de planification de tâches ayant des durées d'exécution probabilistes, des contraintes temporelles et de précédence a été soulevé dans [Baki and Bouzid, 2005]. Les auteurs ont proposé des algorithmes qui permettent la génération de tous les plans possibles et calculent les coûts de chaque plan avec les utilités respectives. Cependant, aucune modélisation du problème n'a été faite. [Bresina

and Washington, 2000] présentent notamment une approche reposant sur le calcul de l'utilité des plans suffixes et permettant la gestion de contraintes temporelles.

[Boerkoel Jr. and Durfee, 2013] ont adressé le problème d'ordonnancement des agents qui assistent les humains lors de la gestion de leurs activités. Ce problème est confronté à plusieurs contraintes: information distribuée, contraintes implicites, les coûts liés au partage d'information entre agents et la possibilité d'apparition de nouvelles contraintes d'une manière dynamique.

Néanmoins, toutes ces approches ne garantissent pas un comportement optimal et rationnel des agents et nécessitent des phases de re-planification, donc des calculs lors de l'exécution.

Les processus décisionnels de Markov offrent une seconde alternative à la planification sous incertitude. Comme nous l'avons décrit au deuxième chapitre, ces modèles expriment l'objectif de l'agent en termes de maximisation d'une fonction d'utilité espérée. De ce fait, ils permettent l'obtention d'un comportement optimal et rationnel de l'agent. De plus, la politique obtenue fait correspondre une action à tout état possible du système. Ainsi, aucune re-planification n'est nécessaire au cours de la phase d'exécution des actions.

Cependant, comme nous avons noté au chapitre précédent, la formalisation des problèmes réels par les modèles markoviens présentent quelques difficultés. En effet, les MDPs (POMDPs) et les DEC-MDPs (DEC-POMDPs) considèrent les actions comme instantanées et ne permettent pas la représentation de contraintes temporelles explicites sur l'exécution, ni les contraintes de ressources, …

L'intégration des contraintes dans les processus décisionnels de Markov décentralisés a été considérée, en premier, dans le travail de [Becker et al., 2004b] en proposant le modèle ED-DEC-MDP. En effet, les ED-DEC-MDPs traitent des problèmes à observations et transitions dépendantes. De plus, ils permettent la prise en compte de dépendances entre les actions. Des contraintes de précédences sont en effet générées et représentées à l'aide du formalisme TAEMS [Decker and Lesser, 1993]. Des contraintes temporelles et des durées d'exécution des actions ont aussi été considérées.

[Beynier and Mouaddib, 2006, 2011] adressent le problème de gestion des contraintes dans les DEC-MDPs. Tout comme [Becker et al., 2004b], ils considèrent des problèmes à observations et transitions dépendantes ainsi qu'ils gèrent les contraintes temporelles,

les contraintes de précédence et les durées probabilistes des actions. En outre, des contraintes de ressources ont été envisagées. Ce problème a été modélisé indépendamment du formalisme TAEMS. En effet, une nouvelle extension du DEC-MDP a été proposée, OC-DEC-MDP (*Opportunity Cost Decentralized Markov Decision Process*). La notion du coût occasionné a été utilisée afin de gérer la contrainte de précédence entre les actions. Cependant, et puisque les auteurs se sont attaqués à un problème complexe gérant plusieurs contraintes en même temps, une hypothèse relaxante a été posée sur la structure du problème. Cette hypothèse stipule que les actions de chaque agents sont totalement ordonnées et ainsi diminuer les critères de décision au niveau de chaque agent (puisque chaque agent connait *a priori* quelle action il va exécuter). Ce modèle sera détaillé dans le chapitre 6, puisqu'il représente le background de notre travail.

[Marecki and Tambe, 2007] ont amélioré l'OC-DEC-MDP en termes de rapidité et de qualité de la solution. En effet, dans l'OC-DEC-MDP, le temps est supposé discret ce qui résulte en un espace d'états très large. Marecki et Tambe dans leur travail traitent les contraintes temporelles continues afin de manipuler une fonction de valeur sur le temps à la place d'une valeur séparée pour chaque pair (action, intervalle d'exécution) comme dans l'OC-DEC-MDP. Ces deux travaux [Beynier and Mouaddib, 2006; Marecki and Tambe, 2007] supposent qu'aucune communication n'est disponible entre les agents. Mostafa et Lesser dans leur travail [Mostafa and Lesser, 2009] gèrent les contraintes d'exécution tout en laissant l'opportunité aux agents de partager de l'information. Une extension du DEC-MDP a été proposée afin de modéliser ce problème EDI-CR (*Event Driven Interaction with Complex Rewards*).

[Wu and Durfee, 2010] considèrent les problèmes de décision séquentielle et stochastique non seulement dans le choix des actions mais aussi dans l'allocation des ressources dans les deux cas, mono-agent et multi-agents. Les MDPs ont été utilisés afin de modéliser ce problème, avec des extensions pour la gestion de contraintes (ressources limitées).

[Zhu et al., 2014] adressent le problème d'ordonnancement stochastique multi-agents en ligne (*online multi-agent stochastic scheduling*). Deux contraintes ont été considérées: la probabilité du succès dépendante du temps et la durée du processus. Ces deux contraintes dépendent du temps du début du *scheduling*. Les auteurs ont proposé une

nouvelle extension du DEC-MDP, OL-DEC-MDP (*Opportunity Loss* DEC-MDP) afin d'inclure la perte en opportunité dans la décision du *scheduling* pour améliorer la performance globale. Plus le *scheduling* commence tôt, plus la probabilité de terminer un job assigné à un agent augmente. Le coût de perte en opportunité augmente, cependant, si la durée engagée est plus longue. Le modèle OL-DEC-MDP introduit alors une fonction de récompense qui prend en considération la perte d'opportunité et assigne le job arrivé à l'agent avec la récompense la plus élevée.

## 4.2 La communication dans les modèles de la théorie de la décision

Dans ce chapitre nous revenons aux modèles de décision pour des agents coopératifs et nous discutons la question importante de la communication. Cependant, contrairement à ce qu'on pourrait penser, et comme nous avons montré au chapitre 3, la majorité des algorithmes proposés pour trouver la politique optimale pour les DEC-POMDPs et les DEC-MDPs supposent que la communication est coûteuse et impossible [Hansen et al. 2004; Bernstein et al., 2005]. Cette hypothèse est trop sévère, car dans le monde réel, nous ne pouvons pas ignorer la possibilité de nécessité de communication entre les décideurs.

Comme il a été mentionné dans le chapitre 3, la complexité de résolution des DEC-POMDPs et DEC-MDPs est doublement exponentielle [Bernstein et al., 2002]. Cette complexité dépend de la quantité de l'information que chaque agent a sur les autres [Shen et al., 2006]. Une solution pour atténuer cette complexité et qui s'avère plus ou moins prometteuse c'est de permettre aux agents de partager leur information locale afin d'avoir une croyance coopérée sur l'état du monde. Cependant, la communication n'est pas ubiquitaire et induit un coût, raison pour laquelle le problème décentralisé incluant la communication peut être aussi complexe que celui qui n'implique pas de communication [Goldman and Zilberstein, 2004].

Dans ce but, quand la communication est introduite dans le raisonnement sur les problèmes coopératifs décentralisés, de nombreuses hypothèses relaxantes peuvent être faites afin d'alléger le problème variant de la moins sévère à la plus sévère.

Dans cette section, nous présentons quelques hypothèses simplificatrices que nous avons tirées des travaux dans la littérature sur la communication dans les DEC-POMDPs et les DEC-MDPs ainsi que nous donnons une classification des travaux existants dans le cadre de la communication.

### 4.2.1 Hypothèses relaxantes

Plusieurs hypothèses peuvent être posées sur les modèles DEC-MDPs et DEC-POMDPs que ce soit dans le cadre de la communication ou non. Ces hypothèses portent principalement sur l'observabilité des états des agents, le degré d'indépendance entre eux et le protocole de la communication.

#### 4.2.1.1 Observabilité

Une hypothèse simplificatrice peut être faite sur l'observabilité de l'état du domaine. Dans le DEC-POMDP l'état est dit partiellement observable, cependant, il peut être collectivement totalement observable où les observations de l'équipe peuvent identifier l'état du système ce qui revient au modèle DEC-MDP.

Toutefois, même si l'ensemble de toutes les observations identifient l'état du DEC-MDP, chaque agent possède une vue partielle sur les autres agents ce qui peut mener à une complexité doublement exponentielle [Goldman and Zilberstein, 2004].

Une autre extension du DEC-MDP résulte de la factorisation de l'espace d'états. Dans cette extension, l'espace d'état est composé d'ensembles d'états: chaque ensemble est affecté à un agent. L'hypothèse qui peut être faite dans ce contexte (DEC-MDP factorisé) est l'observabilité locale totale où chaque agent peut localement totalement observer son état ce qui mène à une complexité NP-complet [Goldman and Zilberstein, 2004]. Ces hypothèses ont été formalisées dans le chapitre précédent.

#### 4.2.1.2 Hypothèse d'indépendance

Les hypothèses d'indépendance peuvent être appliquées sur les modèles de transition et d'observation. Comme ça était montré dans la section 3.3.6.2 du chapitre précédent, le DEC-POMDP peut être à transitions indépendantes, à observations

indépendantes ou les deux en même temps [Goldman and Zilberstein, 2004; Becker et al., 2004a].

#### 4.2.1.3 Calcul de la politique

Lorsque la communication est introduite dans les DEC-POMDPs (DEC-MDPs), le calcul de la politique peut être fait de différentes manières qui peuvent influer sur la complexité du calcul.

La première forme suppose que la communication est ubiquitaire lors de la planification (offline). Cela réduit le problème de contrôle décentralisé à un problème centralisé à agent unique où chaque agent peut partager, à chaque pas de temps, son observation. Ainsi, ce problème peut être résolu en utilisant des techniques standards de solution de POMDP (MDP). Cependant, cette hypothèse est trop sévère (irréelle, même) puisque dans les applications réelles la communication n'est pas gratuite. Pour rapprocher cette hypothèse à la réalité, l'exécution de la politique jointe est faite d'une manière décentralisée tout en imposant des contraintes de communication [Roth et al., 2005]. Contrairement à cette méthode, dans la deuxième forme, la politique jointe est calculée de manière décentralisée en supposant qu'aucune communication n'est disponible. Puis, dans la phase d'exécution (online), la communication est mise en place afin d'améliorer la récompense cumulée [Becker et al., 2009]. La troisième forme consiste à introduire la communication au moment de la planification et ajouter une action de communication à l'ensemble des actions disponibles aux agents [Nair et al., 2004; Mostafa and Lesser, 2009]. Ainsi, le calcul de la politique doit déterminer pour chaque état la meilleure action à faire ce qui peut être une action de communication ou une action niveau domaine.

#### 4.2.1.4 Les types de la communication

Xuan, Lesser et Zilberstein [Xuan et al., 2001] ont identifié trois types de communication en se basant sur comment la communication est initiée:

- *Tell*, dans ce type l'agent décide d'envoyer un message aux autres agents volontairement. En conséquence, l'émetteur ne va pas avoir aucune information sur les récepteurs.

- *Query*, ici l'agent demande une information particulière d'un ou plusieurs agents qui peut lui être utile pour la poursuite de son exécution.
- *Sync*, ce type consiste à la combinaison du *tell* et *query*, où tous les agents communiquent simultanément leurs informations locales.

#### 4.2.1.5 ET/OU communication

Quand la communication est introduite dans les DEC-POMDPs (DEC-MDPs), une manière simple pour générer les politiques de communication optimales est d'augmenter l'ensemble des actions par l'action de communication en plus des actions du domaine. Ces actions de communication peuvent prendre la place des actions du domaine (OU-Communication) ou peuvent être choisies en combinaison avec les actions du domaine (ET-Communication) où les agents choisissent d'abord de communiquer ou non et par la suite utilisent la sortie de l'action de communication pour prendre une décision sur l'action du domaine [Emery-Montemerlo et al., 2004].

#### 4.2.1.6 La valeur de la communication

Décider quand communiquer dans les problèmes décentralisés coopératifs peut être mesuré par la valeur de la communication (VoC, *Value of Communication*) qui réfère à comment un agent évalue le bénéfice de l'action de la communication.

La valeur de la communication peut être définie comme le gain net de la communication, qui est la différence entre l'augmentation attendue dans le gain des agents et le coût associé à la communication [Becker et al., 2009]. Elle peut être mesurée d'une manière myopique (ce qui veut dire que la communication est possible seulement dans le temps présent) ou en considérant la valeur attendue de l'état résultant de la communication. Une autre approche peut ne pas calculer la valeur de la communication mais la communication est déclenchée quand une certaine condition est satisfaite [Mostafa, 2011].

#### 4.2.1.7 Quand communiquer?

Cette issue de communication nécessite, en premier, le raisonnement sur les effets de la communication sur le comportement de l'équipe et la récompense globale attendue. En second, si les agents ne vont pas communiquer à chaque étape de temps,

dans quelles situations la communication sera un bon choix afin de maintenir la coordination.

#### 4.2.1.8 Quoi communiquer?

La communication peut être implicite ou explicite [Goldman and Zilberstein, 2003]. Implicite, quand les actions de communication affectent les observations vues par les autres agents. Explicite, quand les actions de la communication sont différentes des actions du domaine et que le langage de la communication est attaché explicitement par le designer de l'agent. Dans la communication explicite, généralement, les agents peuvent communiquer n'importe quelle information qui aide les récepteurs à affiner leurs croyances sur ce que l'émetteur va faire.

#### 4.2.1.9 Avec qui communiquer?

En plus de décider quand la communication est nécessaire et quoi communiquer, les agents peuvent aussi déterminer avec qui ils vont partager de l'information. L'utilité majeure de cette heuristique est de limiter le nombre d'agents communicants et ainsi réduire la complexité liée au calcul de la valeur de la communication qui est exponentielle dans le nombre d'agents [Becker et al., 2009]. Cette heuristique peut tirer profit de la structure du problème traité, autrement dit du degré de dépendance entre les agents.

### 4.2.2 Classification des travaux existants

Le raisonnement sur la communication peut être fait hors ligne pendant la phase de la planification ou en ligne durant la phase d'exécution de la politique. Ce dernier est moins complexe en termes de coût de calcul, mais le premier assure une meilleure coordination à long terme. Dans cette sous section nous classifions les travaux existants en se basant sur le temps de raisonnement sur la communication.

#### 4.2.2.1 Communication au temps d'exécution

Prendre les décisions de la communication au temps d'exécution est plus facile que les prendre au temps de planification. Dans cette dernière, les agents doivent raisonner sur tous les messages possibles, cependant, un agent qui prend la décision de

la communication au temps d'exécution n'a besoin de considérer que l'historique d'observation qu'il observe actuellement. Ce type de raisonnement a été adressé dans différentes manières.

Le premier travail était réalisé par [Xuan et al., 2001] dans le contexte des DEC-MDPs à transition et observation indépendantes où les algorithmes à base d'heuristique ont été utilisés pour inclure la communication. L'action de communication était utilisée en combinaison avec les actions niveau domaine (ET-communication). Dans ce travail, les politiques des agents ne dépendent pas seulement de l'état du monde mais de l'information reçue des autres agents. Deux heuristiques de communication ont été utilisées, *"no news is good news"* et *"silent commitment"*. L'agent initie la communication quand il découvre que le plan courant n'est pas atteignable et ainsi tous les agents doivent se synchroniser (*sync*).

Goldman et Zilberstein [Goldman and Zilberstein, 2003] introduisent le modèle DEC-POMDP-Com qui représente explicitement les actions de communication et les messages. Un agent ne doit pas seulement considérer le problème de quand communiquer mais il est libre de choisir n'importe quel message de l'ensemble des messages disponibles. Les agents conditionnent le choix de l'action sur les historiques des observations et les messages reçus précédemment. Deux approches ont été présentées; la première implique que les agents communiquent afin d'atteindre des sous objectifs, et ainsi définir de nouveaux buts, la deuxième implique que les agents agissent d'une manière myopique afin d'optimiser le choix de quand envoyer le message, supposant que la communication ne sera plus disponible après.

[Becker et al. 2005] supposent qu'au temps de planification, la communication n'est pas possible et obtiennent une politique 0-communication. Durant l'exécution, ces agents calculent la valeur de synchroniser leurs états d'une manière myopique en se basant sur une étape dans la dynamique du problème. Puisque l'approche est en ligne - restreinte aux états atteignables - et le modèle est à transition et observation indépendante, le calcul sera facile et simple. Cependant, les décisions de la communication dans ce cas sont seulement optimales supposant qu'il n'y aura plus de communication dans le futur, ce qui parfois résulte dans une sur-communication. Les

améliorations sur le calcul de la valeur de la communication qui varie le degré de myopie de l'agent ainsi que la prise en considération de l'autre agent sont données dans [Becker et al., 2009]. Cependant, comme l'agent devient moins myopique le coût de calcul de la valeur de la communication augmente parce que l'agent a besoin de voir loin quand il décide de communiquer ou non. Les deux travaux considèrent le modèle *sync* de la communication.

Carlin and Zilberstein [Carlin and zilberstein, 2009] ont étendu le travail de [Becker et al., 2005] afin de gérer le modèle DEC-POMDP dans lesquels les agents sont capables de communiquer leurs historiques d'observation. Quand décider sur la communication dans le DEC-POMDP, chaque agent doit raisonner sur un long nombre de séquences de décision-observation pour les autres agents, un tel raisonnement consomme du temps et d'espace. Afin de pallier à ce problème, une méthode de compression de ces séquences a été proposée. Tout comme les travaux de [Becker et al. 2005, 2009] le type *sync* de la communication a été utilisé.

Roth, Simmons and Veloso [Roth et al., 2005] adressent les DEC-POMDPs et supposent une communication ubiquitaire au temps de planification, ainsi ils obtiennent un POMDP mono-agent qui sa politique est après utilisée comme une solution approximative pour le DEC-POMDP. Durant l'exécution, si l'historique d'observation de communication d'un agent affecte positivement les performances de l'équipe alors la communication est déclenchée. Ainsi, la communication est considérée sans coût. Une fois que l'agent reçoive un historique d'observation, il élague les croyances qui ne s'accordent pas avec les historiques reçus, à la suite desquelles il peut trouver que la communication de son propre historique est utile, déclenchant ainsi un nouveau cycle de communication. Les auteurs adressent aussi dans d'autres travaux la question de quoi communiquer en raisonnant sur les croyances jointes, les deux types *tell* et *query* ont été utilisés ainsi que ET-communication [Roth et al., 2006a, 2006b, 2007].

[Wu et al., 2011] propose un algorithme de planification en ligne pour les DEC-POMDPs qui aide à faire face à la grande complexité de résoudre de tels problèmes d'une manière offline. De plus, la communication a été introduite afin d'augmenter le degré de coordination entre agents. La phase de communication prend place avant la

phase de planification. Chaque agent décide d'abord si la communication est nécessaire et si les ressources sont disponibles. Si la communication est nécessaire mais les ressources ne sont pas disponibles, les agents vont reporter la communication à l'étape suivante, ce qui permet la gestion de la communication avec délai. Sinon, les agents vont synchroniser leurs informations locales. La communication est initiée si une inconsistance dans l'historique de croyances est détectée. Dans ce travail, l'action de communication est considérée en combinaison à l'action du domaine (ET-communication).

#### 4.2.2.2 Communication au temps de planification

A la place de produire une politique séparée de la communication, [Nair et al., 2004] étendent les actions du domaine en ajoutant une action de communication: *sync* (OU-communication). Si un agent choisit l'action *sync*, il initie une synchronisation où lui et les autres agents diffusent leurs historiques d'observation depuis la dernière synchronisation. *Sync* permet aux agents de passer à un seul état de croyance synchronisé sur les états du monde.

Néanmoins, comme nous avons déjà mentionné, introduire la communication à la phase de planification, nécessite, en général l'énumération de tous les messages possibles ainsi que leurs effets sur les agents. Malheureusement, dans le cas de ce travail, le nombre de messages est aussi large que l'ensemble de tous les historiques d'observations possibles. Afin de remédier à ce problème, l'algorithme *COMMUNICATIVE-DP-JESP* proposé par [Nair et al., 2004] intègre une stratégie de communication chaque *K* étapes de l'algorithme JESP. En effet, puisqu'un agent ne peut pas savoir exactement quelles sont les observations que les autres agents ont reçues au temps d'exécution, l'algorithme permet aux agents de synchroniser périodiquement leurs historiques d'observations afin de réduire la probabilité d'un comportement non désiré.

[Spaan et al., 2006] ont établit un nouveau modèle où les messages sont envoyés comme une partie des vecteurs d'action des agents et ils sont reçus à l'étape suivante comme une partie des vecteurs d'observations des récepteurs. A l'instar des modèles COM-MTDP et DEC-POMDP-Com, ce modèle ne nécessite pas un langage de communication explicite mais, à la place, traite les sémantiques de la communication

comme une partie du problème d'optimisation. Ils présentent aussi une méthode itérative pour calculer la politique jointe de l'équipe d'une manière décentralisée, tout en intégrant la communication dans le processus de raisonnement. Etant donné un ensemble de politiques fixes pour tous les agents sauf l'agent $i$, le processus de calcul de la politique convertit le DEC-POMDP en un POMDP de la perspective de l'agent $i$. L'information contenue dans le message a été mesurée par l'entropie. Cependant ce modèle est restreint aux agents à observations et transitions indépendantes.

L'inconvénient majeur du raisonnement sur la communication offline est l'augmentation de la taille du problème comme résultat de suivre chaque action niveau domaine par une action de communication. Même si ce problème peut être modélisé, il est en général difficile de le résoudre. L'approche proposée par Mostafa et Lesser [Mostafa and Lesser, 2009] afin de gérer ce problème consiste à construire un problème plus petit qui ses politiques optimales d'action et de communication sont proches de celles du problème original. Ils ont traité les domaines où les interactions entre agents sont limitées et présentées explicitement dans le modèle EDI-CR. Cette structure spécifique du problème a été exploitée afin de définir à l'avance les possibilités de communication et ainsi réduire la taille du problème. L'approche proposée a permis aussi de gérer les contraintes d'exécution des actions.

Un autre travail qui traite la communication à la phase de planification est [Beynier and Mouaddib, 2010] où la communication est ajoutée au modèle OC-DEC-MDP qui a été proposé dans [Beynier and Mouaddib, 2005, 2006]. *Communicative* OC-DEC-MDP permet à un agent de communiquer la date de fin de sa dernière action exécutée avec succès. La raison derrière le choix d'une telle sémantique est la relation de précédence entre les actions des agents. Le bénéfice de la communication est mesuré par l'amélioration dans l'utilité espérée du récepteur prenant compte de son coût (consommation de temps et de ressources). Cependant, il y a une certaine indépendance entre les MDPs des agents, parce que l'effet de communication est calculé seulement par l'agent émetteur étant donné la politique du récepteur durant la dernière itération. Cela est possible grâce à la topologie simple du graphe de la mission. Ce travail sera détaillé dans le chapitre 6.

Le travail précédent utilise une certaine séparation où chaque agent résout son MDP d'une manière individuelle étant donné la dernière politique connue des autres agents. Cette séparation facilite le processus de coordination. Un différent type de séparation est utilisé dans le travail de [Witwicki et al., 2007] où l'exécution des agents est séparée en décidant, en premier, sur un ensemble d'engagements (*commitments*) et ensuite chaque agent calcule sa politique tout en respectant ces engagements. La première phase, chercher l'espace d'engagements, est basée sur un algorithme heuristique. La deuxième phase est basée sur la programmation linéaire afin de forcer les agents à respecter ces engagements. L'avantage d'utiliser cette approche à la place de la communication consiste à réduire la taille de l'espace de politiques. Cependant, si un engagement n'est pas satisfait, il n'y a pas une manière de prévenir l'agent dépendant. Une extension de ce travail [Witwcki et al., 2009] permet à chaque agent de mieux modéliser les autres agents au détriment d'une taille importante du problème. Cependant, cela est restreint au modèle ED-DEC-MDP.

Messias, Spaan et Lima dans leur travail [Messias et al., 2011] permettent la communication entre agents ce qui réduit le problème décentralisé à un MPOMDP. L'objectif consiste, alors, à exploiter les dépendances entre agents afin de réduire la communication. En effet, quand les dépendances entre les décisions des agents sont dispersées, la croyance sur l'état local est, en général, une information suffisante pour l'agent afin d'identifier l'action qu'il va entreprendre. Cette particularité a été utilisée dans ce travail afin de réduire la communication entre agents.

Le travail dans [Melo et al., 2012] exploite aussi les interactions entre les agents afin d'optimiser les décisions de la communication dans les DEC-POMDPs. En effet, dans les domaines avec des interactions locales, le besoin de la communication peut être réduit, du à l'influence limitée entre les agents. Cette idée est exploitée en dérivant des POMDPs locaux qui optimisent le comportement de communication de chaque agent. Le type *query* de la communication a été utilisé afin de permettre à chaque agent de demander l'état local de l'autre agent quand c'est nécessaire.

Dans le tableau 4.1, nous classifions quelques travaux sur la communication en se basant sur les hypothèses simplificatrices présentées précédemment.

### 4.2.3 La communication stochastique

La plupart des travaux réalisés dans le cadre de la communication dans les DEC-POMDPs et les DEC-MDPs ne posent aucunes contraintes sur la communication, mais à la place ils supposent que la communication est instantanée et elle s'effectue avec succès. Récemment, quelques travaux ont été réalisés en considérant la communication stochastique où les messages peuvent être perdus, corrompus ou retardés. [Spaan et al., 2008; Oliehoek and Spaan, 2012] ont considéré le problème de la communication retardée par une ou plusieurs étape de temps et l'incertitude sur le succès de la communication (avec une probabilité de succès) dans le modèle DEC-POMDP. Les jeux bayésiens ont été utilisés afin de résoudre le DEC-POMDP.

Wu et al. [Wu et al., 2011] dans le modèle de communication proposé, permettent de traiter la communication avec délai en retardant la communication à l'étape suivante de décision. [Matignon et al., 2012] introduisent une nouvelle méthode de résolution orientée interaction pour les modèles de décision décentralisés qui gèrent le partage limité d'information et les *breaks* dans la communication. Cette méthode de résolution est basée sur une fonction de valeur distribuée considérant la dernière étape de temps où les agents ont communiqué avec succès. Les messages corrompus ont reçu peu d'attention, cependant, dans la planification décentralisée [Valtazanos and Steedman, 2014].

## 4.3 Discussion

Les processus décisionnels de Markov décentralisés sont un formalisme mathématique très puissant qui permet la modélisation des problèmes de décision à contrôle décentralisé. Cependant, la grande complexité liée à la résolution de ces modèles limite leurs applicabilités aux différents problèmes issus du monde réel. Afin de réduire cette complexité, la littérature dans ce domaine était divisée entre deux grands volets: des travaux qui se focalisent sur la communication entre agents afin d'augmenter l'information que chaque agent a sur les autres et des travaux qui se focalisent sur l'exploitation de la structure du problème (les interactions structurées entre les agents).

Raisonner sur la décision de la communication peut être fait au temps de planification ou au temps d'exécution. De la classification effectuée, nous pourrons remarquer que la majorité des travaux optent à introduire la communication au temps d'exécution, cela revient à la difficulté liée à sa considération à la phase hors ligne qui mène à une augmentation dans la taille de l'espace du problème. Néanmoins, la complexité du problème reste toujours importante à cause de la décentralisation des agents. Afin de remédier à ce problème, les chercheurs du domaine ont posé plusieurs hypothèses simplificatrices qui concernent soit le modèle (observabilité, indépendance, …), soit le protocole de la communication.

L'exploitation des interactions structurées entre agents afin de réduire le nombre d'agents communicants représente un axe de recherche très actif. Cela influence positivement la complexité du problème.

## 4.4 Conclusion

Dans ce chapitre, nous venons de passer en revue quelques travaux existants dans la littérature et qui sont en liaison avec nos travaux de thèse. Nous avons donné une classification des travaux incluant la communication en se basant sur plusieurs hypothèses relaxantes afin d'alléger la complexité des DEC-MDPs. Comme ça été montré, la communication stochastique a reçu peu d'attention dans la littérature. Dans notre travail et comme nous allons voir dans les chapitres suivants, nous allons proposer un nouveau modèle de communication qui permet la prise en considération des messages perdus lors de la prise de décision par les agents.

Afin de résoudre les problèmes de planification qui nous sont posés, nous allons tout d'abord devoir nous pencher sur la mise en place d'une modélisation appropriée permettant en particulier la gestion du temps, de l'incertitude et des contraintes du problème. Nous allons commencer par la proposition d'une modélisation des contraintes d'exécution dans les MDPs à un seul agent, chapitre 5. Par la suite, nous étendons cette modélisation aux modèles markoviens décentralisés, chapitre 6.

# Chapitre 5

# *Un MDP pour la gestion de contraintes*

*Dans les chapitres 2 et 3 nous avons pu dégager quelques limitations liées à l'utilisation des MDPs et des DEC-MDPs pour la résolution de problèmes concrets, par exemple la planification des actions dans des colonies de robots autonomes. Les MDPs et les DEC-MDPs ne permettent, en effet, pas la prise en compte de certaines données du problème comme les contraintes d'exécution des actions.*

*Ce chapitre présente une partie de nos contributions dans le cadre de la gestion des contraintes dans les DEC-MDPs. L'objectif final de notre travail consiste à étendre les modèles DEC-MDPs afin de prendre en considération les contraintes complexes d'exécution des tâches, tout en introduisant la communication entre agents.*

*Nous allons à présent proposer une extension du modèle MDP à un seul agent afin de gérer les différentes contraintes sur l'exécution des actions, à savoir les contraintes temporelles, les contraintes de précédences et l'incertitude sur les durées d'exécution des actions [Abdelmoumène and Belleili, 2011, 2012]. Nous commençons par une formalisation du problème posé (section 5.1). Par la suite, nous présentons le modèle proposé avec la construction adaptée de chaque composant du MDP.*

## 5.1 Description du problème

Le problème que nous adressons consiste à planifier et ordonnancer un ensemble des actions ayant des durées probabilistes, des contraintes temporelles et des contraintes de précédences. Nous allons modéliser ce problème par un MDP et ainsi définir une adaptation de chaque composant du MDP afin de gérer ces contraintes.

Le problème de décision que nous considérons est caractérisé par: $\langle A, C_T, C_P, \Delta, C \rangle$ où:

- $A$ est l'ensemble des actions que l'agent doit exécuter tout en respectant les contraintes posées.
- $C_T = \{[EST_i, LET_i], i = 1..|A|\}$ où, $EST_i$ est la date de début au plus tôt de $a_i$ et $LET_i$ est sa date de fin au plus tard.
- $C_P$ représente l'ensemble des prédécesseurs d'une action. En effet, chaque action $a_i$ a un ensemble de prédécesseurs spécifiant l'ensemble des actions qui doivent être exécutées avant $a_i$. On distingue deux types de contraintes de précédences:

  -*Contraintes conjonctives:* une contrainte de précédence conjonctive existe entre une action $a$ et un ensemble d'actions $a_1, a_2, \dots, a_n$ si $a$ ne peut être exécutée que si $a_1, a_2, \dots, a_n$ sont toutes exécutées.

  -*Contraintes disjonctives:* une contrainte de précédence disjonctive existe entre une action $a$ et un ensemble d'actions $a_1, a_2, \dots, a_n$ si $a$ ne peut être exécutée que si une des actions $a_1, a_2, \dots, a_n$ est exécutée.

- Chaque action $a_i$ dispose de plusieurs durées d'exécution, chacune d'entre elles est associée à une probabilité. On note: $\Delta = \{(d_i, p_i) | \, p_i = P(duration = d_i)\}$.
- A chaque durée $d_i$ d'une action $a_i$, on associe un coût d'exécution. D'une manière générale le coût est une fonction de la durée: $C = f(d)$

***Définition.*** Une action qui a des contraintes de précédence et une date de début au plus tôt ne peut pas être exécutée que si tous ses prédécesseurs sont exécutés (dans le cas de contrainte de précédence conjonctive) ou un de ses prédécesseurs est exécuté (dans le cas de contrainte de précédence disjonctive) et sa

date de début au plus tard est satisfaite. D'où une action peut avoir un délai quand sa date de début au plus tôt n'est pas satisfaite même si ses prédécesseurs ont terminé leur exécution.

***Hypothèse.*** On suppose que l'ensemble des actions à planifier est connu à l'avance. On ne considère pas les actions qui viennent dynamiquement.

La figure 5.1 présente un graphe de tâches (actions) avec contraintes.

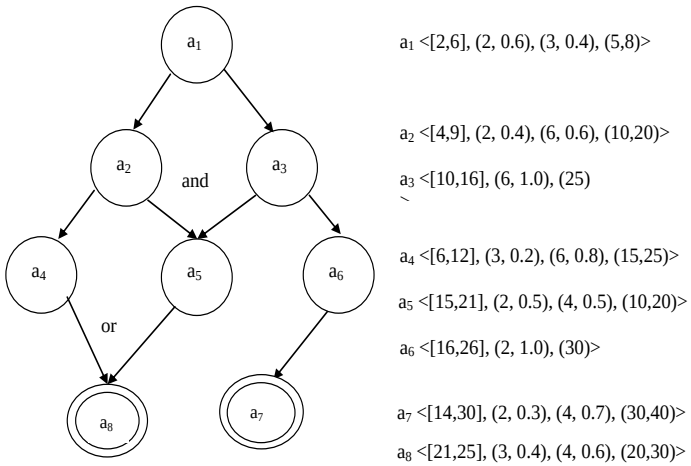


***Figure 5.1*** *Un graphe représentant l'ensemble des actions avec contraintes*

Dans cet exemple, l'agent doit exécuter l'action $a_8$ ou l'action $a_7$ pour accomplir sa mission. Ainsi la mission peut être exécutée avec différents coûts selon la consommation de ressources (temps) et la qualité qui dépend des circonstances de l'exécution de la mission. Puisque les actions $a_7$ et $a_8$ ne peuvent pas être exécutées que si les actions qui les précèdent sont exécutées sans violation de contraintes temporelles, on a besoin d'une stratégie de planification pour accomplir la mission avec des coûts minimes. Plusieurs plans d'exécution sont possibles: $(a_1, a_2, a_4, a_8), (a_1, a_2, a_3, a_5, a_8), (a_1, a_3, a_6, a_7), \ldots$

## 5.2 Modèle proposé

La modélisation du problème présenté ci-dessus par un MDP revient à définir l'espace d'états, l'espace des actions, la fonction de transition et la fonction de récompense. Les actions correspondent aux actions du problème (tâches); les autres

paramètres nécessitent une modélisation spécifique du problème, ce qui va être adressé dans les sous-sections ci-dessous.

### 5.2.1 Construction de l'espace d'états

Puisque notre problème présente des contraintes temporelles et des contraintes de précédence, il est nécessaire d'encapsuler la dernière action exécutée dans l'état. En effet, l'exécution de l'action peut être faite dans de différents intervalles d'exécution qui dépendent de la date de début et la durée probabiliste. Alors, l'état doit encapsuler l'action courante ainsi que son intervalle d'exécution.

La question qui se pose maintenant, est, comment modéliser les contraintes de précédence?

Comme l'état du MDP doit être markovien (résume l'historique), l'idée que nous proposons consiste à utiliser un état factorisé dans lequel on garde les informations suivantes: la dernière action exécutée avec son intervalle d'exécution et pour chaque action nous spécifions une variable aléatoire qui prend ses valeurs dans un domaine de valeurs, ainsi, nous définissons l'état factorisé comme suit: $s_i = \langle a_i, [t_i, t_i^{'}], V \rangle$ où $t_i$ est une date de début possible de l'action $a_i$, $t_i^{'}$ est sa de fin et $V$ est un vecteur: $V = \big(val(a_1), val(a_2), \dots, val(a_n)\big)$, $val(a_i)$ est la valeur prise par la variable aléatoire associée à l'action $a_i$ spécifiant différents statuts de chaque action. Nous avons identifié quatre valeurs possibles: *E: Enable* ce qui veut dire que l'action est prête à être exécutée (tous ses prédécesseurs sont exécutés); *D: Disable* ce qui veut dire que l'action n'est pas prête à être exécutée, ses prédécesseurs n'ont pas encore terminé leurs exécutions; *S: Success* qui veut dire que l'action a été exécutée avec succès; *F: Failure* ce qui veut dire que l'action a échoué.

La construction des différents intervalles d'exécution de chaque action est faite en appliquant à la première action toutes les durées possibles et en les propageant dans tout le graphe des actions. Une description du calcul des intervalles d'exécution est présentée ci-dessous.

- *Quand $a_i$ n'a pas de prédécesseurs:*

$$I_{a_i} = \left\{ I_{a_i} = [t_i, t_i^{'}] \;\middle|\; \begin{matrix} t_i = max(Start\_time, EST_i)\,, t_{i_j}^{'} = t_i + d_{i_j} \\ j=1..n \end{matrix} \right\} \qquad (5.1)$$

Où *start_time* est le temps dans lequel la mission commence, $n$ est le nombre des durées possibles de l'action $a_i$.

- *Quand $a_i$ a des contraintes de précédences:* On distingue deux cas

(1) $a_i$ a un seul prédécesseur $a_{i-1}$:

$$I_{a_i} = \left\{ I_{a_i} = [t_i, t_i^{'}] \;\middle|\; \begin{matrix} t_i = max(EST_i, \{ET_{i-1}\}),\ t_i^{'} = t_i + d_{i_j} \\ j=1..n \end{matrix} \right\} \qquad (5.2)$$

Où $ET_{i-1}$ est l'ensemble des dates de fin de l'action $a_{i-1}$ et $n$ est le nombre de durées possibles de l'action $a_i$.

(2) $a_i$ a plus d'un seul prédécesseur $a_1, a_2, \ldots, a_{i-1}$:

$$I_{a_i} = \left\{ I_{a_i} = [t_i, t_i^{'}] \;\middle|\; \begin{matrix} t_i = max\left(EST_i, max\{ET_1, ET_2, \ldots ET_{i-1}\}\right),\ t_i^{'} = t_i + d_{i_j} \\ j=1..n \end{matrix} \right\} \qquad (5.3)$$

Où $n$ est le nombre de durées possibles de l'action $a_i$.

Dans le tableau (5.1), nous montrons un exemple des intervalles d'exécution possibles générés après avoir appliqué la propagation temporelle donné ci-dessus, quand $start\_time$ est égale à 2 [Abdelmoumène and Belleili, 2011, 2012].

Les intervalles en gris correspondent à ceux qui violent les contraintes temporelles de l'action. Quand une action est exécutée et que ses contraintes temporelles sont violées, tous ses successeurs ne peuvent pas être exécutés. Par exemple, quand l'action $a_2$ est exécutée dans l'intervalle [5,11], les actions $a_4$ *et* $a_5$ ne peuvent pas être exécutées.

| **Actions** | **Contraintes temporelles** | **Contraintes de précédences** | **Intervalles d'exécution possibles** |
|---|---|---|---|
| $a_1$ | [2,6] | Non | [2,4], [2,5] |
| $a_2$ | [4,9] | $a_1$ | [4,6], [4,10], [5,7], [5,11] |
| $a_3$ | [10,16] | $a_1$ | [10,16], [11,17], [13,19], … |
| $a_4$ | [6,12] | $a_2$ | [6,9], [7,10], [6,12], [7,13] |
| $a_5$ | [15,22] | $a_2$ *et* $a_3$ | [16,18], [16,20] |
| … | … | … | … |

***Tableau 5.1*** *Un exemple d'intervalles d'exécution possibles quand* $start\_time = 2$

Ainsi, en appliquant la propagation temporelle, nous obtenons tous les intervalles d'exécution possibles dans lesquels les actions peuvent être exécutées. Cette information (intervalle d'exécution) sera ajoutée à la dernière action exécutée et à la combinaison des valeurs des variables aléatoires qui chacune d'entre elles reflètent le statut de l'action correspondante, pour former l'état factorisé.

**Exemple 1.** Un état possible correspondant à l'action $a_2$ est : $<a_2$,[4,6],(*S, **S**, E, E, D, D, D,D*)> , les deux premières actions ont été exécutées avec succès impliquant que l'action $a_3$ sera assignée à $E$ puisque son prédécesseur $a_1$ a été exécuté, la valeur $E$ sera affectée aussi à l'action $a_4$ puisque son prédécessuer $a_2$ a été exécuté avec succès. Cet état représente un état de succès parce que l'action $a_2$ a été exécutée avec succès tout en respectant ses contraintes temporelles.

Un autre état possible correspondant à l'action $a_2$ est : $<a_2$,*[5,11],S, **F**, E, D, D, D, D,D* $>$ , cet état est un état d'échec parce que les contraintes temporelles ont été violées. Il faut noter que l'action $a_4$ ne peut pas être exécutée parce que son prédécesseur (action $a_2$) a échoué.

Initialement, l'agent se trouve dans un état de succès, aucune action n'a été exécutée. Ainsi, nous notons l'état initial comme suit: *(' ',[start_time,start_time ],(E, D, D, ..., )*) ce qui veut dire que la première action est prête à être exécutée (*E*), toutes les autres actions sont à *disable* (*D*). Les états peuvent être classés en états de succès quand les contraintes temporelles sont respectées, états d'échec quand les contraintes temporelles sont violées, états terminaux avec succès de la mission et états terminaux avec échec de la mission.

### 5.2.2 Construction de la fonction de transition

La probabilité qu'une action transite d'un état $s_i$ associé à l'action $a_i$ à un état $s_{i+1}$ associé à l'action $a_{i+1}$ correspond à la probabilité de l'intervalle d'exécution de l'état $s_{i+1}$. Cette probabilité est obtenue à partir des probabilités que l'action $a_{i+1}$ a commencé son exécution $P\big((start(a_{i+1}) = t_{i+1})\big)$ et la probabilité de la durée d'exécution $P(duration(a_{i+1}) = d_{i+1})$.

On pose $I_{a_{i+1}} = [t_{i+1}, t'_{i+1}]$ un intervalle d'exécution possible calculé durant la construction de l'espace d'état:

$$\begin{aligned} P\big(I_{a_{i+1}} = [t_{i+1}\ ,\ t'_{i+1}]\big) &= P(start(a_{i+1}) = t_{i+1}) \times P(end(a_{i+1}) = t'_{i+1}) \\ &= P(start(a_{i+1}) = t_{i+1}) \times P(t'_{i+1}/t_i) \end{aligned} \tag{5.4}$$

$P(t'_{i+1}/t_i) = P\big(duration = d_{i+1} = t'_{i+1} - t_{i+1}\big)$ est donnée dans la description du problème.

$P(start(a_{i+1}) = t_{i+1})$ est calculée différemment, elle dépend de l'action si elle possède des prédécesseurs ou non.

1) Quand $a_{i+1}$ n'a pas de prédécesseurs

$$P(start(a_{i+1}) = t_{i+1}) = 1 \tag{5.5}$$

**Exemple 2.** On prend les intervalles d'exécution de l'action $a_1$ qui sont [*2,4*] *et* [*2,5*]

$$P(I_{a_1} = [2,4]) = 1 * P(duration(a_1) = 2) = 0.6$$

$$P(I_{a_1} = [2,5]) = 1 * P(duration(a_1) = 3) = 0.4$$

2) Quand l'action $a_{i+1}$ a un seul prédécesseur *a'*

$$P(start(a_{i+1}) = t_{i+1}) = P(t_{i+1} \geq t'_{a'}) \qquad (5.6)$$

**Exemple 3.** Dans l'exemple l'action $a_2$ a comme prédécesseur l'action $a_1$, alors la probabilité que $a_2$ commence son exécution au temps 4 est égale à la probabilité que $a_1$ termine son exécution à un temps inférieur ou égale à 4:

$$P(t_{a_2} = 5) = P(t'_{a_2} \leq 5) = 0.4$$

D'où

$$P(I_{a_2} = [5,11]) = P(t_{a_2} = 5) \times P(d_{a_2} = 11-5 = 6)$$

$$= 0.4 \times 0.6 = 0.24$$

De même pour la probabilité que l'action $a_3$ commence son exécution

$$P(I_{a_3} = [10,16]) = P(s_{a_3} = 10) \times P(d_{a_3} = 6) = 0.4 \times 1.0 = 0.4$$

3) Quand $a_{i+1}$ a plus d'un prédécesseur $a_1, a_2, \ldots, a_m$

$$P(start(a_{i+1}) = t_{i+1}) = P\left(t_{i+1} \geq \max_j \left(t'_{a_j}\right)\right) \qquad (5.6)$$

**Exemple 4.** On prend l'action $a_5$ par exemple qui a deux prédécesseurs $a_2$ *et* $a_3$ et on prend l'intervalle d'exécution $I_{a_5}$ = [*16,18*]

$$P(t_{a_5} = 16) = P\left(\max\{ET_{a_2}, ET_{a_3}\} \leq 16\right)$$

Puisque l'action $a_3$ a le maximum des dates de fin,

$$P(I_{a_5}=[16,18])=P(t_{a_5}=16)\times P(d_{a_5}=2)=1.0\times 0.5=0.5$$

Le tableau 5.2 montre une portion de la matrice de la fonction de transition qui représente les probabilités des transitions entre états.

| | <a1,[2,4],(**S,E,E**,D,D,D,D,D)> | <a1,[2,5],(**S,E,E**,D,D,D,D,D)> | <a2,[4,6],(**S,S**,E,**E**,D,D,D)> | <a2,[4,10],(S,**S**,E,**E**,D,D,D)> | <a2,[5,7],(**S,S,E,E**,D,D,D)> | <a2,[5,11],(S,**F**,E,**D**,D,D,D)> | … … … … | <a3,[11,17],(S,S,**F**,D,D,D,D)> | ….. |
|---|---|---|---|---|---|---|---|---|---|
| Initial state | 0.6 | 0.4 | | | | | | | |
| <a1,[2,4],(S,E,E,D,D,D,D,D)> | | | 0.24 | 0.36 | | | | | |
| <a1,[2,5],(S,E,E,D,D,D,D,D)> | | | | | 0,16 | 0,24 | | | |
| <a2,[4,6],(S,S,E,E,D,D,D,D)> | | | | | | | | | |
| <a2,[4,10],(S,S,E,E,D,D,D,D)> | | | | | | | | | |
| <a2,[5,7],(S,S,E,E,D,D,D,D)> | | | | | | | | | |
| <a2,[5,11],(S,F,E,D,D,D,D,D)> | | | | | | | | 0,24 | |
| <a3,[6,12],(S,S,S,E,E,E,D,D)> | | | | | | | | | |
| <a3,[10,16],(S,S,S,E,E,E,D,D)> | | | | | | | | | |
| <a3,[7,13],(S,S,S,E,E,E,D,D)> | | | | | | | | | |
| <a3,[11,17],(S,S,F,D,D,D,D,D)> | | | | | | | | | |
| <a5,[12,14],(S,S,S,E,S,D,D,E)> | | | | | | | | | |
| <a5,[12,16],(S,S,S,E,S,D,D,E)> | | | | | | | | | |

***Tableau 5.2*** *Une portion de la matrice de fonction de transition*

L'état en gris correspond à l'exécution de l'action $a_2$ avec une violation des contraintes temporelles. Bien que cet état dénote un échec dans l'exécution de l'action $a_2$ et ne va pas activer l'exécution de l'action $a_4$, d'autres alternatives pour accomplir la mission existent. Pour cette raison, cet état n'est pas un état terminal. Cependant, ce n'est pas le cas de l'état où la case est remplie en gris qui est un état terminal à cause de l'exécution échouée des actions $a_2\ et\ a_3$. Il n'y aura pas d'autres actions à exécuter dans ce cas et ainsi toute la mission échoue.

### 5.2.3 Construction de la fonction de récompense

Les problèmes avec contraintes temporelles sont caractérisés par l'importance du coût dans leur résolution. Le coût représente qu'est ce que nous payons si une action est exécutée dans une certaine durée. Dans d'autres termes, l'exécution d'une action durant

une certaine durée coûte un certain taux de ressources (consommation de ressources, ici notre ressource est le temps).

Dans notre cas, le coût est relatif aux durées d'exécution de chaque action qui peut être implicitement lié à la qualité. En se basant sur ce principe, nous construisons notre fonction de récompense comme suit:

A chaque exécution de l'action $a_i$ qui mène à un état $s_i$ nous associons une récompense positive pour les états de succès (connus à partir de la phase de construction des états). Les valeurs des récompenses positives sont meilleures quand l'action $a_i$ est exécutée dans de meilleures conditions (la durée la plus petite) et elles diminuent quand l'action est exécutée dans des conditions moins meilleures tout en respectant les contraintes temporelles. Pour les états d'échec de l'exécution de l'action $a_i$ dans l'état $s_i$ nous associons des récompenses négatives.

Comme nous avons déjà dit qu'un état qui viole les contraintes temporelles et ne mène pas à la fin de la mission, autrement dit ne mène pas à un état terminal, est associé à une récompense (même négative) qui est supérieure qu'une récompense d'un état qui mène à l'échec total de la mission.

On pose *S, Fp, Ft,* les états de succès, échec partiel et échec total respectivement. Formellement:

$$if\ s=\begin{cases} \boldsymbol{S} & then\ r(s)=scalar\text{-}cost & with\ 0<cost<scalar \\ \boldsymbol{Fp} & then\ r(s)=penalty_{part_{fail}}\text{-}cost & with\ penalty_{part_{fail}}<0 \\ \boldsymbol{Ft} & then\ \ r(s)=penalty_{total_{fail}} & with\ penalty_{total_{fail}}<penalty_{part_{fail}}<0 \end{cases}$$

## 5.3 Conclusion

Dans ce chapitre, nous avons adressé le problème de planification des tâches avec des durées probabilistes, des contraintes temporelles et des contraintes de précédences. Le plan que nous voulons obtenir est un plan robuste qui est capable de mener à bien la mission en dépit d'incertitude relative aux durées des actions. Pour ce faire, nous avons choisi de formaliser ce problème par un processus décisionnel de Markov (MDP) qui est un formalisme mathématique très puissant pour les problèmes séquentiels et stochastiques.

La formalisation que nous avons proposée implique une construction explicite des composants du MDP qui sont à savoir les états, la fonction de transition et la fonction de récompense. Ainsi, nous avons proposé une formalisation structurée de l'état qui satisfait la propriété de Markov qui est impérative dans les modèles MDPs. Nous avons proposé que les états encapsulent toute information relative à la dynamique du problème tel que le statut de l'action, son intervalle d'exécution et l'impact de son exécution sur les autres actions. La fonction de transition est aussi construite et elle dénote les effets qui peuvent résulter quand une action s'exécute à partir d'un état donné. Nous avons calculé dans ce but la probabilité de chaque effet relatif à l'exécution d'une action. Finalement, la fonction de récompense a été spécifiée. Nous avons assigné pour chaque exécution d'une action de différentes récompenses selon le coût et le statut de l'exécution.

Dans le chapitre suivant nous allons étendre la modélisation proposée au cas décentralisé, où plusieurs agents sont impliqués afin d'accomplir une mission commune.

# Chapitre 6

# *Extension du modèle OC-DEC-MDP pour la gestion des plans locaux partiels*

*Plusieurs applications du monde réel impliquent plusieurs agents agissants ensembles sous pression de temps et sous incertitude. Des exemples de ces applications peuvent être trouvés dans les missions d'exploration de la planète de Mars [Becker et al., 2004a], la gestion des désastres [Nair et al., 2002] et la détection décentralisée des évènements climatiques [Kumar and Zilberstein, 2009], …*

*Afin de modéliser ces applications une adaptation du modèle de base DEC-MDP est nécessaire. Dans ce chapitre nous nous intéressons à une sous classe du DEC-MDP qui a été proposée pour justement gérer les contraintes d'exécution qui est à savoir l'OC-DEC-MDP (Opportunity Cost Decentralized Markov Decision Process) [Beynier and Mouaddib, 2005, 2011].*

*Le modèle OC-DEC-MDP présente quelques limitations que nous allons présenter dans ce chapitre et nous allons proposer une extension de ce modèle afin de pouvoir modéliser un large spectre d'applications du monde réel. Nous allons aussi proposer un modèle de communication qui permet aux agents de partager certaines informations locales tout en optimisant le comportement de communication de chaque agent.*

## 6.1 Motivation

Nous motivons la classe de problèmes que nous considérons par deux domaines applicatifs multi-robots, à savoir: l'exploration planétaire et la gestion des désastres.

### 6.1.1 L'exploration planétaire

La première classe de problèmes qui motive notre travail, peut être trouvée dans les scénarios de la NASA, impliquant des robots explorant la surface de Mars [Beaty et al., 2008]. Afin d'augmenter l'information scientifique retournée, les chercheurs envoient des équipes de robots qui doivent agir d'une manière autonome et coopérative. Ces robots sont envoyés pour une période précise et sans l'intervention des humains. Ils peuvent bénéficier de la présence d'un satellite via lequel ils reçoivent leurs tâches. Due à la rotation orbitale du satellite, les agents peuvent communiquer durant un temps fixe. Pendant, le reste de la journée, lorsque le satellite n'est pas correctement aligné, les robots ne peuvent compter sur une quelconque communication. Ils doivent donc agir de manière autonome.

L'exemple de la figure 6.1 implique quatre agents qui leur objectif est de collecter de l'information scientifique tout en exécutant un ensemble d'activités. Cet exemple est tiré de [Witwicki, 2011].

Un satellite en orbite autour de la planète est chargé de la prise de photos et la transmission d'informations entre la terre et Mars. Sur la surface se trouve une station de base qui abrite le centre de données, dont ses activités comprennent l'analyse des données d'imagerie et la compilation des données provenant de plusieurs sources dans des cartes de surfaces détaillées. Deux *rovers*, qui sont situés à la base au début de la mission, doivent se déplacer dans la surface et visiter les différents sites d'intérêt. Le robot 1 est équipé d'outils et de capteurs qu'il peut utiliser pour creuser le sol et l'analyser. Le robot 2 est conçu pour se déplacer plus rapidement et de se positionner dans une série d'emplacements de manière à compiler des données sensorielles en trois dimensions.

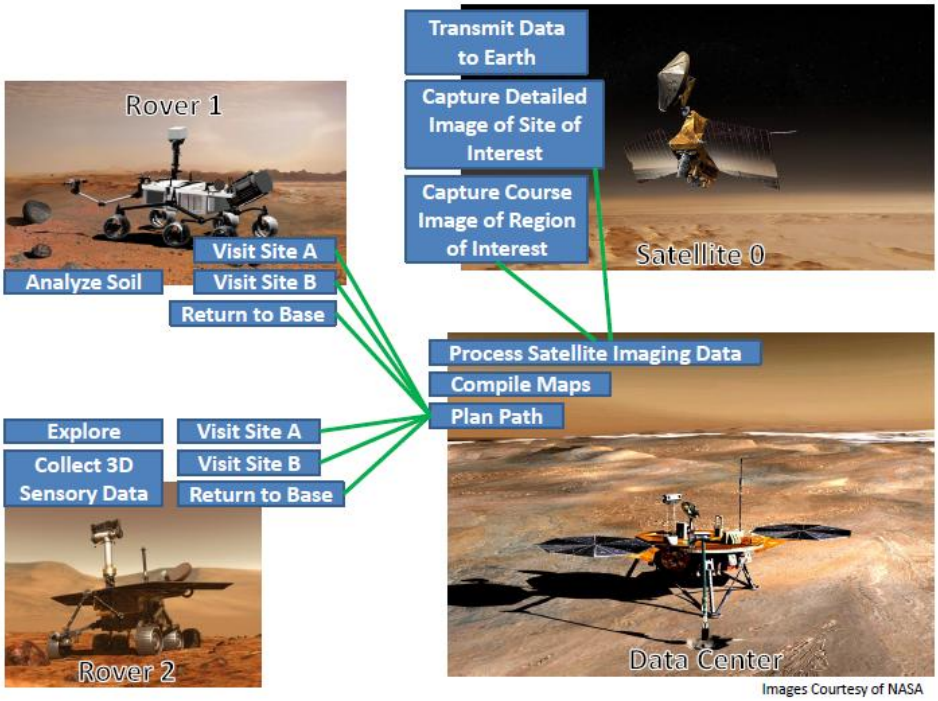


***Figure 6.1*** *Un exemple d'exploration planétaire [Witwicki, 2011]*

En raison de sa vitesse, il a également la capacité d'explorer rapidement les zones inconnues.

Une fois les agents terminent leurs activités, plusieurs objectifs de collecte d'informations scientifiques sont remplis. Ainsi, à chaque tâche accomplie avec succès, une certaine valeur est associée. La productivité globale de la mission peut être quantifiée par la somme des valeurs des tâches accomplies. A chaque ensemble de nouvelles informations recueillies, de nouveaux objectifs peuvent se présenter. Imaginez que chaque jour, les agents ont une nouvelle mission et de nouveaux objectifs qui peuvent variés en fonction des conditions environnementales et sur l'analyse des missions passées. Dans chacune de ces missions, le but collectif des agents est de maximiser leurs valeurs accumulées attendues.

Les agents peuvent chacun recueillir et traiter les données d'une manière individuelle mais pour certaines tâches, ils peuvent profiter des interactions entre eux. Ces activités interdépendantes sont désignées par les lignes à la figure 6.1. Par exemple, le robot 1 ne peut visiter le site A que si l'agent du centre de données lui envoie un chemin. En interagissant de cette manière, la combinaison de leurs activités

coordonnées leur permet d'atteindre les objectifs de mission complexes, comme la localisation des zones présentant des caractéristiques géographiques inhabituelles, la navigation à ces zones, l'analyse des échantillons de sol, …

Une coordination réussie dans ce domaine nécessite de surmonter plusieurs défis difficiles. Les objectifs des agents sont contraints temporellement avec des délais stricts. Dans l'exemple de la figure 6.1, le satellite est limité quand est ce qu'il peut prendre les photos des sites et des domaines d'intérêts parce qu'il est en orbite autour de la planète. La seule source d'énergie des robots est le soleil, ainsi chacun est contraint de terminer ses tâches dans un délai lié à la quantité d'énergie qu'il a stocké et le coucher du soleil. Il y a aussi des contraintes comportementales qui dictent que chaque agent ne peut effectuer qu'une seule activité à la fois. Par exemple, l'agent d'imagerie satellitaire ne peut pas pointer simultanément son appareil photo à deux endroits différents. Par ailleurs, ces robots évoluent dans des environnements partiellement observables, stochastiques, séquentiels, dynamiques et continus. En raison de la connaissance incomplète de l'environnement, celui-ci est considéré comme incertain. De ce fait, l'exécution d'une tâche peut conduire à différents résultats. Une tâche peut alors avoir différentes durées possibles et consommer différentes quantités de ressources. Lorsque ces quantités sont continues, l'exécution d'une tâche peut mener à une infinité de situations différentes. Par exemple, en fonction de la trajectoire prévue pour le robot 1 et des obstacles rencontrés dans cette voie, il peut prendre une quantité variable de temps pour atteindre le site A. Afin de maximiser la productivité dans l'attente, les plans des agents doivent tenir compte de l'incertitude dans leurs actions et interactions.

Afin de coordonner de manière optimale le comportement de l'équipe et d'éviter le gaspillage des ressources, les activités des agents doivent être soigneusement planifiées à l'avance. Ainsi, pour maximiser leur mesure de performance, les robots explorateurs ne doivent donc pas seulement se contenter d'exécuter leurs tâches, ils doivent également s'assurer du respect des contraintes sur l'exécution des tâches.

Les restrictions de communication concernent également les communications entre robots. Dans le cas des robots sur Mars, ces derniers peuvent communiquer directement entre eux ou via un satellite. Comme nous l'avons précédemment expliqué,

la plupart du temps, les robots ne peuvent pas espérer communiquer via le satellite. Toutefois, il leur est possible d'échanger des messages sans passer par le satellite à condition qu'ils soient proches physiquement et que la topologie du site le permette (par exemple qu'aucun obstacle n'empêche la liaison). Lorsque les robots évoluent dans de grands espaces et/ou des terrains accidentés, ce type de communication n'est pas envisageable. En effet, *a priori*, rien ne garantit qu'ils puissent établir un tel échange d'information, il est donc préférable d'éviter le recours à cette forme de communication.

### 6.1.2 La gestion de désastres

Un exemple d'un environnement difficile est considéré dans la compétition *RoboCup Rescue* [Kitano et al., 1999], motivé par le tremblement de terre de *Kobe* en 1995 [Tierney and Goltz, 1997]. *RoboCup Rescue* simule un tremblement de terre dans un environnement urbain. Dans ce scénario l'effondrement des bâtiments a provoqué le blocage des routes et des gens dans les décombres. Dans cet environnement chaotique, des équipes de pompiers, de policiers et des ambulanciers doivent prendre des décisions localement, en se basant sur des informations limitées. Les robots pompiers sont chargés d'éteindre le feu, les robots ambulanciers doivent secourir les victimes et les robots policiers doivent déblayer et évacuer les routes. L'objectif principal de ces services d'urgence est de minimiser les dommages et les pertes. Pour ce faire, il est important de trouver des plans efficaces afin de gérer la situation.

Cependant, cela s'avère difficile en raison des incertitudes (incertitude sur la vitesse avec laquelle le feu évolue, incertitude sur les durées d'extinction des feux et de déblayage des routes, incertitude sur combien d'unités de pompiers doivent être allouées à un site particulier, …). De plus, l'environnement est partiellement observable (chaque robot a une perception limitée à une dizaine de mètres, il ne connaît donc pas l'état de tous les bâtiments, l'emplacement ou l'état de santé de toutes les victimes,…), dynamique (les feux évoluent, la santé des victimes peut se dégrader, …), séquentiel et continu. En outre, des dépendances fortes entre les différents types de robots peuvent être identifiées: les policiers doivent avoir déblayé les routes pour que les pompiers puissent accéder aux feux et que les ambulanciers viennent au secours des victimes.

Les communications entre les différents intervenants sont restreintes. Afin de ne pas saturer les liaisons radios, chaque robot ne peut envoyer qu'un nombre limité de messages. Les ressources des robots sont par ailleurs limitées : un robot pompier ne dispose par exemple que d'une certaine quantité d'eau. Suivant le type de robot la nature des ressources est susceptible de varier, pour les robots pompiers il s'agira d'eau, les ambulanciers quant à eux ne pourront véhiculer qu'un seul blessé à la fois.

En raison de la dynamicité de l'environnement, des contraintes temporelles doivent être respectées. Le désastre doit être géré tout en minimisant le temps et ainsi minimiser les dégats: une victime doit être secourue avant qu'elle ne décède, un feu doit être éteint avant que le bâtiment soit complètement brûlé, etc.

Les chercheurs ont pu modélisé des petits sous-problèmes de gestion de catastrophes par les DEC-POMDPs [Marecki and Tambe, 2007; Nair et al., 2002,2003a,b; Oliehoek and Visser, 2006; Paquet et al., 2005; Abdelmoumène and Belleili, 2014].

## 6.2 Caractéristiques du problème

Les deux exemples donnés dans la section précédente ainsi que tous les problèmes auxquels nous nous intéressons partagent tous plusieurs caractéristiques que nous allons citer ci-dessous.

**Caractéristique 1: Les systèmes multi-agents coopératifs.** Le problème consiste à modéliser un comportement intelligent d'une équipe d'agents qui partagent le même but: maximiser une valeur jointe du groupe. Dans l'exemple de la figure 6.1, la valeur jointe est représentée par la somme attendue des valeurs des actions accomplies avec succès.

**Caractéristique 2: Prise de décisions séquentielles sous incertitude.** Chaque agent possède un ensemble d'actions, avec lesquelles il interagit avec son environnement et peut affecter les autres agents aussi. L'agent ne connait que son état local et son ensemble d'actions. Chaque action est associée à une certaine récompense qui décrit son exécution avec succès. Le modèle qui peut décrire le comportement de chaque agent est le MDP. L'incertitude dans les sorties des actions des agents (qui est dans notre travail représentée par les différentes durées des actions) peut être modélisée par la fonction de

transition du MDP. Le problème de coordination optimale devient alors, décider comment chaque agent doit agir étant donné son état et son ensemble d'actions tel que la séquence de décisions prises maximise l'accumulation attendue des récompense des agents.

**Caractéristique 3: Des contraintes sur l'exécution des actions.** Des exemples présentés précédemment, nous avons pu recenser différentes contraintes que doivent être respectées afin de construire des solutions adaptées aux problèmes posés.

- *Contraintes temporelles*, définissent pour chaque action, une fenêtre temporelle durant laquelle elle peut être exécutée.
- *Contraintes de précédences*, interprètent une certaine dépendance entre les actions: " une action ne peut pas être exécutée que si son prédécesseur a terminé son exécution ". La relation de précédence peut être présente entre les actions du même agent ou bien entre les actions de différents agents.
- *Contraintes de ressources*, l'exécution des actions dépend de la disponibilité de ressources. Dans cette thèse, nous considérons un seul type de ressource: le temps.
- *Communication limitée*, dans certains types de problèmes la communication est une ressource coûteuse et limitée, voire impossible. Par conséquent, les agents ne connaissent pas les états et les actions des autres agents, sauf si une certaine forme de communication a été spécifiée.

Puisque les actions des agents impliquent le respect de plusieurs contraintes, les décisions sur quelles actions exécutées et quand les exécuter doivent être planifiées à l'avance pour ne pas perdre du temps et des ressources. Il faut noter que cette liste de contraintes n'est pas exhaustive.

## 6.3 Exemple présentant le problème considéré

La figure 6.2 présente un problème simple impliquant deux agents qui ont comme objectif d'exécuter leurs plans locaux. Comme les plans locaux des agents sont partiels chaque agent doit choisir entre deux ou plusieurs actions alternatives. Selon la situation courante, les actions alternatives ont des gains différents.

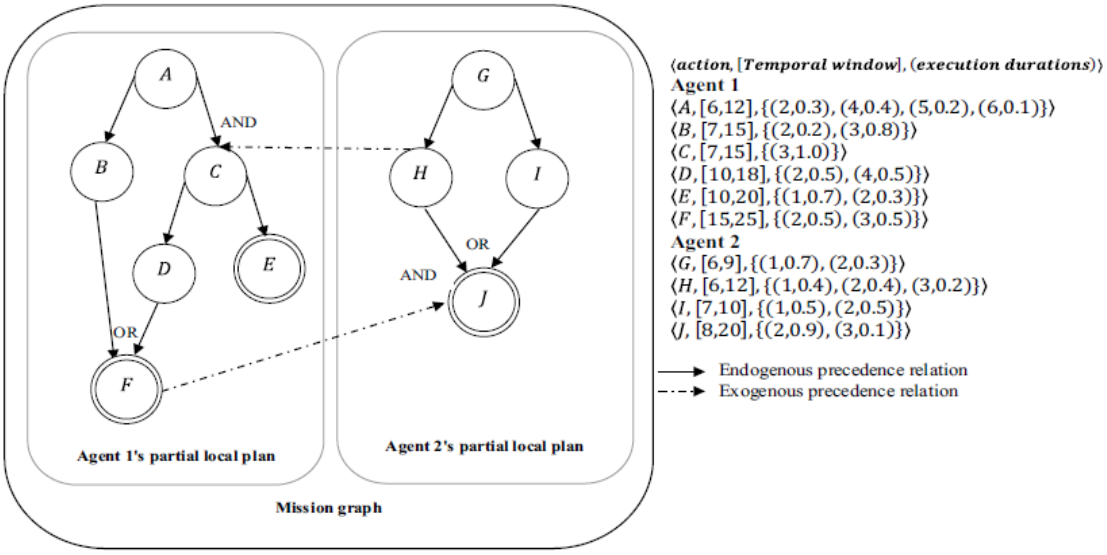


***Figure 6.2*** *Un graphe de mission présenté par des plans locaux partiels de chaque agent*

Plus les dépendances locales entre les actions, il existe un autre type de dépendances entre les actions des agents. Par exemple, dans la figure 6.2, l'action $J$ de l'agent 2 ne peut pas être exécutée que quand les actions $H$ ou $I$ du même agent ont été exécutées et l'action $F$ de l'agent 1 est exécutée. L'incertitude résulte des durées des actions. En effet, une action peut avoir plusieurs durées d'exécution possibles chacune est associée à une probabilité. Ces probabilités sont obtenues d'une manière empirique [Witwicki and Durfee, 2011]. De plus, l'exécution de chaque action est limitée par une fenêtre temporelle qui exprime sa date de début au plus tôt et sa date de fin au plus tard. L'exécution avec succès de chaque action est associée à une valeur. Le gain total de l'équipe est quantifié par les valeurs cumulatives des actions exécutées.

Puisque l'environnement est dynamique et présente des contraintes sur l'exécution des actions, plusieurs situations inattendues peuvent avoir lieu et qui nécessitent un changement dans les plans des agents. Par exemple, l'agent 2 peut choisir d'exécuter l'action $I$ à la place de l'action $H$ et l'agent 1 peut choisir d'exécuter l'action $C$ à la place de l'action $B$. Dans ce cas l'agent 1 va attendre l'action $H$ de l'agent 2 qui ne va jamais être exécutée. Cela nous conduit à l'échec total de la mission.

La communication peut gérer de telles situations. Puisque la communication dans ces domaines est coûteuse, elle doit être limitée. Plusieurs chemins existent menant à

l'accomplissement de la mission, mais le but est de trouver le chemin global qui maximise le gain total du groupe et qui évitent l'échec total de la mission. Ce chemin est interprété par un plan optimal (politique) pour chaque agent, qui est calculée en résolvant le DEC-MDP approprié.

## 6.4 Contexte théorique

Le modèle OC-DEC-MDP représente le background de notre travail. Dans cette section, nous allons présenter ce modèle ainsi que son algorithme de résolution.

### 6.4.1 Processus décisionnel de Markov décentralisé avec coût occasionné

Le modèle OC-DEC-MDP est proposé par Beynier et Mouaddib [Beynier and Mouaddib, 2005, 2011], afin de gérer les contraintes temporelles où chaque action doit être exécutée respectant un intervalle d'exécution spécifique, les contraintes de précédences où chaque action ne peut être exécutée avant la fin d'exécution d'un sous ensemble d'actions des autres agents et les durées probabilistes d'exécution des actions. Pour $n$ agents, ce modèle est composé de $n$ MDPs locaux. Chaque MDP est composé des actions totalement ordonnées, des états factorisés, une fonction de transition et une fonction de récompense. Chaque composant doit subir à un traitement spécifique basé sur la propagation temporelle [Bresina and Washington, 2000] pour prendre en compte les contraintes sur l'exécution des actions. Toutes les actions des agents sont considérées dans un seul graphe de mission qui est acyclique où les nœuds sont les actions et les arcs sont les contraintes de précédences.

Selon le type du problème étudié et puisque la communication entre les agents n'est pas permise, trois types d'états ont été identifiés: états de succès quand l'action respecte ses contraintes temporelles, états d'échec partiel (PF: *Partial Failure*) quand l'action tente de s'exécuter mais ses prédécesseurs n'ont pas encore terminé leurs exécutions et états d'échec total (TF: *Total Failure*) quand l'action viole ses contraintes. L'OC-DEC-MDP a été prouvé d'être polynomial dans la taille de l'espace d'états. La construction de ce modèle (et sa résolution) repose sur l'hypothèse que les actions de chaque agent sont totalement ordonnées. Cette hypothèse veut dire les actions vont être exécutées tôt ou tard.

En conséquent, les intervalles d'exécution des actions doivent être larges, sinon la probabilité d'échec total d'une action sur l'exécution des autres actions est augmentée (à cause de cette représentation sous forme de chaîne). Par conséquent, le problème de décision de chaque agent est allégé. En effet chaque agent va déterminer seulement quand-est-ce qu'il commence l'exécution d'une action (qui est supposée connue *a priori*).

La résolution de ce modèle est basée sur un coût occasionné (OC: *Opportunity Cost*). Ce coût est une mesure empruntée du domaine d'économie [Wieser, 1889]. Elle est particulièrement utilisée pour gérer les contraintes de précédences. L'OC a été utilisé pour la première fois dans le cadre des MDPs dans [Mouaddib and Zilberstein, 1998] et après il a été étendu pour les DEC-MDPs dans le cas où les agents ont des plans locaux linéaires [Beynier and Mouaddib, 2005, 2011]. L'intuition derrière l'utilisation du coût occasionné est de permettre à chaque agent de prendre en compte les effets de ses propres décisions sur les autres agents afin d'assurer une bonne coordination entre les agents.

Récemment, une version communicative de l'OC-DEC-MDP a été proposée [Beynier and Mouaddib, 2010]. La communication est introduite afin de gérer les situations de mis-coordination entre les agents (interpréter par le nombre des états d'échec partiel). Permettre aux agents de partager l'information au temps de planification a pour conséquences l'augmentation des espaces d'états et d'actions. Afin de remédier à ce problème, un ensemble d'heuristique a été proposées. Ces heuristiques consistent à communiquer la date de fin d'une action précédemment exécutée avec succès aux agents qui dépendent de cette action. Une étude comparative va être donnée dans la section 6.8.

Il est important de noter que l'OC-DEC-MDP et le *communicative* OC-DEC-MDP tel qu'ils sont définis, ne sont pas appropriés pour le cas où les plans locaux des agents sont partiellement ordonnés ce qui est plus proche aux applications réelles. Dans ce dernier (plan partiel), le problème de décision local de chaque agent consiste à deux décisions: quelle action exécuter? Et quand l'exécuter? Ce qui affecte l'algorithme de résolution. En outre, les heuristiques introduites dans [Beynier and Mouaddib, 2010]

afin de rendre la communication possible, ne peuvent pas être appliquées dans le cas où les agents possèdent des plans partiels des actions puisqu'un agent peut choisir de ne pas exécuter une action prédécesseur. Le but majeur de notre travail consiste à étendre l'OC-DEC-MDP et son algorithme de calcul de politique pour gérer des contraintes de précédence plus complexes. Il est vrai que la raison pour laquelle la communication est utilisée dans le travail de [Beynier and Mouaddib, 2010] et notre travail [Abdelmoumène and Belleili, 2016] est la même - réduire les situations de mis-coordination -, la manière dont elle a été considérée est différente puisque la topologie du graphe de la mission et les problèmes de décision ne sont pas identiques.

Les situations de mis-coordination dans [Beynier and Mouaddib, 2010] sont interprétées par l'état d'échec partiel et parviennent du fait que l'action commence son exécution avant que ses prédécesseurs terminent leurs exécutions. Dans notre étude, ces situations résultent aussi du fait que l'agent prédécesseur peut choisir de ne pas exécuter l'action prédécesseur. Comme nous allons voir dans les prochaines sections, les heuristiques que nous allons utilisées dans notre proposition sont appropriées pour la nouvelle topologie du graphe de la mission où les actions des agents sont partiellement ordonnées. Dans ce qui suit nous allons expliquer le principe de l'algorithme d'OC selon [Beynier and Mouaddib, 2011].

### 6.4.2 Le principe de l'algorithme du coût occasionné

L'OC algorithme comme plusieurs autres algorithmes de calcul de politique, commence avec une politique initiale qui consiste à la politique de la date de début au plus tôt. Cette politique est, par la suite, améliorée dans plusieurs itérations jusqu'à ce qu'aucune amélioration ne soit possible. Chaque itération de cet algorithme consiste en deux phases. La première phase est une phase de calcul des utilités espérées, des politiques et des valeurs d'OC pour chaque état en se basant sur la politique initiale. La seconde phase effectue une mise à jour de la fonction de transition en se basant sur la politique courante trouvée dans l'itération précédente afin de préparer pour une autre itération de l'algorithme. La raison derrière la mise à jour des probabilités de transition est que ces probabilités sont calculées en se basant sur la politique initiale fixée au début.

Afin de gérer les contraintes de précédence, l'algorithme du coût occasionné commence du bas par les actions feuilles pour lesquelles il n'y a pas de successeurs (action $J$ et $E$ de l'exemple présenté dans la figure 6.2) en haut jusqu'aux actions racines (actions $A$ et $G$ de la figure 6.2). La raison pour laquelle il faut commencer par les actions feuilles est que ces actions ont un coût occasionné égale à zéro, puisqu'elles n'ont influence sur aucune autre action. Pour chaque état à partir duquel les actions courantes (actions feuilles, en premier) peuvent être exécutées, l'OC algorithme calcule son utilité espérée, sa politique et les valeurs du coût occasionné. Ces valeurs (valeurs d'OC) sont ensuite propagées pour tous les états prédécesseurs possibles des autres agents. A ce moment, chaque agent peut calculer sa politique en considérant ses récompenses moins le coût occasionné propagé. A partir de cette nouvelle politique, la fonction de probabilité est mise à jour afin d'exécuter une novelle itération de l'algorithme.

## 6.5 Description du problème

Le problème de décision que nous considérons est caractérisé par: $\langle \propto, A, C_T, C_P, \Delta, R, C_{com}, \Sigma \rangle$, et est une extension du problème considéré au chapitre 5 pour le cas décentralisé, où:

- $\propto$, est un ensemble d'agents.
- $A$, est un ensemble d'actions.
- $C_T = \{[EST_i, LET_i], i = 1..|A|\}$ où, $EST_i$ est la date de début au plus tôt de $a_i$ et $LET_i$ est sa date de fin au plus tard.
- $C_P = \{EndoPred \cup ExoPred\}$. Chaque action $a_i$ a un ensemble de prédécesseurs endogènes (noté $EndoPred(a_i)$) spécifiant l'ensemble des actions locales qui doivent être exécutées avant $a_i$, et un ensemble de prédécesseurs exogènes (noté $ExoPred(a_i)$), spécifiant l'ensemble des actions qui doivent être exécutées avant $a_i$ par les autres agents.
- Chaque action $a_i$ dispose de plusieurs durées d'exécution, chacune d'entre elles est associée à une probabilité. On note: $\Delta = \{(d_i, p_i) | \, p_i = P(duration = d_i)\}$.
- $R$, est une fonction de récompense. L'exécution avec succès de l'action $a_i$ est associée à une récompense non négative, $R_i$.

- $C_{com}$ , est le coût de la communication.
- Σ, est l'ensemble de messages que les agents peuvent partagés. La sémantique de ces messages sera discutée dans la prochaine section.

Nous allons encoder ce problème de décision par le modèle OC-DEC-MDP. Ce dernier est étendu afin de prendre en considération le coût et le langage de communication. Comme nous avons mentionné dans la section précédente, l'OC-DEC-MDP est composé d'un ensemble de MDPs locaux, un pour chaque agent. Chaque MDP est un tuple $\langle S, A, P, R, \Sigma, \mathrm{C}_{com} \rangle$, où $S$ est un ensemble fini d'états, $A$ est un ensemble fini des actions incluant l'action de la communication. La dynamique de chaque MDP est maintenue par la fonction de transition $P$. $R$ dénote la fonction de récompense et elle est associé à un état spécifique. Σ est un ensemble fini de messages et $\mathrm{C}_{com}$ est le coût de la communication. La modélisation du problème de décision à contraintes par un OC-DEC-MDP revient à construire chaque composant du MDP de chaque agent. Dans ce qui suit, nous décrivons la construction du modèle.

### 6.5.1 Les actions

Nous calculons pour chaque action plusieurs dates de début et de fin possibles en prenant en considération les contraintes de précédences (locales et globales), l'incertitude sur les durées des actions et les fenêtres temporelles. Nous notons $ST_i$ et $ET_i$ les ensembles des dates de début possibles et des dates de fin possibles de l'action $a_i$, respectivement. Par exemple pour l'action $B$ de la figure 6.2, nous obtenons $ST_B = \{8,10,11,12\}, ET_B = \{10,11,12,13,14,15\}$.

### 6.5.2 Les états

Puisque l'action $a_i$ a plusieurs dates de début et de fin, nous pouvons déduire de différents intervalles d'exécution dans lesquels $a_i$ peut être exécutée. Ces intervalles constituent une information utile pour décider sur les transitions des états puisqu'ils nous indiquent quand l'action a commencé, quand est ce qu'elle se termine et combien d'unités de temps elle a consommé. Comme nous traitons les décisions de la communication, le message constitue une information utile aussi pour les décisions des agents. Ainsi, pour satisfaire la propriété de Markov, l'état doit être factorisé et doit

encapsuler la dernière action exécutée $a_i$, son intervalle d'exécution $[t_i, t_i']$ (où $t_i$ est une date de début possible de l'action $a_i$ et $t_i'$ est la date de fin associée), et le message $\Sigma$ s'il existe. Exemple, dans la figure 6.2, nous obtenons pour l'action *B:*

$\langle B, [8,10] \rangle$, $\langle B, [8,11] \rangle$, $\langle B, [10,12] \rangle$, $\langle B, [10,13] \rangle$, $\langle B, [11,13] \rangle$, $\langle B, [11,14] \rangle$, $\langle B, [12,14] \rangle$, $\langle B, [12,15] \rangle$, $\langle B, [8,10], \Sigma \rangle$, $\langle B, [8,11], \Sigma \rangle$, $\langle B, [10,12], \Sigma \rangle$, $\langle B, [10,13], \Sigma \rangle$, $\langle B, [11,13], \Sigma \rangle$, $\langle B, [11,14], \Sigma \rangle$, $\langle B, [12,14], \Sigma \rangle$ et $\langle B, [12,15], \Sigma \rangle$.

Il faut noter que pas tous les états seront augmentés par des messages. Comme nous allons l'expliquer par la suite, les agents ne vont pas toujours communiquer à n'importe quel état mais la communication est permise dans des points bien spécifiques. Comme dans le modèle OC-DEC-MDP, trois types des états sont distingués. Etats de succès, l'agent transite à un tel état quand l'action considérée est exécutée avec succès respectant ses contraintes. Etat d'échec partiel (PF), quand l'agent tente d'exécuter une action qui ses prédécesseurs exogènes n'ont pas encore terminé leurs exécutions. Etat d'échec total (TF), quand l'action exécutée viole ses contraintes.

### 6.5.3 La fonction de transition

La fonction de transition calcule la probabilité de transition d'un état $s_i$ associée à l'action $a_i$, à un état résultat $s_{i+1}$ associé à l'action $a_{i+1}$. Le calcul se fait de la racine du graphe de la mission aux feuilles.

On pose $s_i$ et $s_{i+1}$, tel que: $s_i = \langle a_i, I_{a_i} \rangle, s_{i+1} = \langle a_{i+1}, I_{a_{i+1}} \rangle$[1].

$$P(s_{i+1}|s_i, a_{i+1}) = P(I_{a_{i+1}})$$

Cette probabilité dépend de la probabilité que l'action $a_{i+1}$ sera exécutée dans l'intervalle $I_{a_{i+1}}$. On pose $I_{a_{i+1}} = [t_{i+1}, t_{i+1}']$. D'où:

$$P(s_{i+1}|s_i, a_{i+1}) = P(start(a_{i+1}) = t_{i+1}) \times P\left(end(a_{i+1}) = t_{i+1}'\right) \qquad (6.1)$$

[1] Dans l'explication du calcul de la fonction de transition, nous ne considérons que des états sans messages.

Cette probabilité correspond à la probabilité que l'exécution de l'action $a_{i+1}$ commence au temps $t_{i+1}$ et se termine au temps $t'_{i+1}$.

La première partie de l'équation (6.1) dépend de la politique initiale, d'une part: si la politique initiale de l'état $s_i$ dicte l'exécution de $a_{i+1}$ au temps $t_{i+1}$, $P_\pi(\pi(s_i) = (a_{i+1}, t_{i+1})) = 1$, sinon, cette probabilité est égale à zéro. D'autre part, $P(start(a_{i+1}) = t_{i+1})$ dépend de la probabilité que les prédécesseurs exogènes de $a_{i+1}$ ont fini leur exécution à un temps inférieur ou égale à $t_{i+1}$.

La deuxième partie de l'équation (6.1) correspond à la probabilité que $a_{i+1}$ consomme $d_{i+1} = t'_{i+1} - t_{i+1}$ unités de temps et qui est donné dans la description du problème ($P\big(duration = d_{i+1} = t'_{i+1} - t_{i+1}\big)$).

Par exemple, dans la Figure 6.2, la probabilité de passer d'un état $s_A = \langle A, [6,8]\rangle$ à un état résultat $s_B = \langle B, [8,10]\rangle$, est donné par:

$$P(s_B|s_A, B) = P(I_B = [8,10]) = P(start(B) = 8|s_A) \times P(end(B) = 10|s_A)$$

On suppose que la politique initiale de l'agent 1 consiste à exécuter l'action $B$ au temps 8, $P_\pi(\pi(s_A) = (B, 8)) = 1$. D'où:

$$P(start(B) = 8|s_A) = 1$$

Puisque l'action $B$ n'a pas de prédécesseur exogène, on ne considère pas la probabilité que ces prédécesseurs exogènes n'ont pas fini leur exécution.

De la description du problème, on a:

$$P(end(B) = 10|s_A) = P(d_B = 10 - 8) = 0.2$$

D'où:

$$P(s_B|s_A, B) = 1 \times 0.2 = 0.2$$

Puisqu'on maintient le même modèle que [Beynier and Mouaddib, 2011] (OC-DEC-MDP), où trois types d'états sont introduits, nous distinguons trois types de

transition. Transition réussie quand l'action $a_{i+1}$ est exécutée dans l'intervalle $I_{a_{i+1}} = [t_{i+1}, t'_{i+1}]$ respectant ses $EST_{i+1}$ et $LET_{i+1}$ ($EST_{i+1} \leq t_{i+1} < t'_{i+1} \leq LET_{i+1}$). Transition d'échec partiel quand l'action $a_{i+1}$ commence son exécution trop tôt avant que ses prédécesseurs terminent leurs exécutions. Transition d'échec total quand l'action $a_{i+1}$ est exécutée dans l'intervalle $I_{a_{i+1}} = [t_{i+1}, t'_{i+1}]$ violant son $LET_{i+1}$ ($t'_{i+1} > LET_{i+1}$).

### 6.5.4 La fonction de récompense

L'agent reçoit une récompense qui consiste à un nombre positif pour être dans un état de succès. L'état d'échec partiel est associé à une récompense égale à zéro puisqu'aucune transition ne sera faite. Quant à l'état d'échec total, l'agent est pénalisé sur le fait d'être dans un tel état. Cette pénalisation consiste en un nombre négatif ajouté à la perte dans la récompe(nse de toutes les actions qui sont atteignables à partir de l'action qui a échoué et exécuter par le même agent. Si un agent opte à communiquer, le coût de la communication sera soustrait de la récompense reçue.

### 6.5.5 Le langage de communication

Dans notre travail, les sémantiques des messages sont attachées explicitement. Nous proposons trois types de messages chacun avec une sémantique spécifique. Ces sémantiques seront expliquées dans la section 6.6.2.

## 6.6 Calcul de la politique jointe

Cette section est dédiée à la résolution du modèle construit dans la section précédente. Nous allons présenter, en premier, comment l'OC algorithme original est étendu pour le cas du plan local partiel. Après, nous allons donner et argumenter le modèle de communication proposé et nous allons détailler le calcul de la politique jointe selon notre problème prenant en compte des plans locaux partiels et la décision de la communication.

### 6.6.1 Un algorithme de coût occasionné pour des plans locaux partiels

L'adaptation de l'algorithme du coût occasionné pour gérer des plans locaux partiels nécessite de traiter deux questions importantes. La première question concerne deux problèmes de décision qui consistent en quelle action à exécuter et quand exécuter cette action, à la place du problème de décision traiter dans l'original OC algorithme pour les actions totalement ordonnées où le problème de décision concerne quand commencer l'unique action suivante.

La deuxième question concerne quand une action est considérée pour calculer son utilité espérée, sa politique et les valeurs d'OC (noté par $(V, \pi, OC)$, respectivement) au temps de planification. En effet, et en résultat des contraintes de précédence (locales et globales), les actions les plus dépendantes doivent être définies et ainsi elles peuvent être évaluées $(V, \pi, OC)$, au temps de planification, au même niveau. Pour ce faire, nous transformons les plans locaux partiels avec contraintes de précédences exogènes, en un seul graphe de mission acyclique et nous procédons à une décomposition en niveau du graphe de mission. Pour chaque niveau, nous définissons l'ensemble de nœuds (actions) qui appartiennent à ce niveau.

#### 6.6.1.1 Décomposition en niveau

Afin de pouvoir appliquer l'OC algorithme, le processus de division doit commencer des nœuds feuilles aux nœuds racines.

Nous commençons avec le niveau $L_k (k = 0)$ contenant les nœuds feuilles (actions) qui n'ont pas de successeurs ni exogènes ni endogènes. Le prochain niveau $L_{k+1}$ contient les prédécesseurs de ces nœuds (actions) dans $L_k$.

Pour calculer l'utilité espérée, la politique et les valeurs d'OC $(V, \pi, OC)$ pour chaque action, deux conditions $(C_1, C_2)$ doivent être satisfaites dans chaque niveau.

$C_1$: Les valeurs $(V, \pi, OC)$ d'une action $a$ dans un niveau $L_k$ sont calculées si les actions ayant le même prédécesseur que $a$ sont dans le même niveau $L_k$. Sinon, la considération de $a$ sera reportée au prochain niveau.

$C2$: L'action dépendante doit être considérée (calcul d'OC) avant ses prédécesseurs. Sinon, les actions prédécesseurs (contraignantes) doivent être reportées au prochain niveau.

#### 6.6.1.2 Exemple illustratif

Dans cette sous-section, nous illustrons le processus de décomposition en niveau en se référant à l'exemple de la figure 6.2.

Dans notre exemple, nous commençons par $L_0 = \{J, E\}$, (nœuds feuilles). $L_0$ doit être vérifié pour les deux conditions $(C_1, C_2)$, comme $C_1$ n'est pas satisfaite pour $E$ (parce que l'action $D$ qui a le même prédécesseur $C$ n'est pas encore considérée), $E$ est reportée pour le niveau $L_1$.

$L_0$ est alors mis à jour, $L_0 = \{J\}$. Le niveau suivant $L_1$ contient les actions prédécesseurs des nœuds dans $L_0$ $(J)$ avec le nœud reporté de $L_0$ $(E)$. Le résultat est $L_1 = \{E, H, I, F\}$. Le niveau $L_1$ sera alors vérifié. Les deux conditions ne sont pas satisfaites. $C_2$ n'est pas vérifiée pour $H$ (parce que $C$ a comme prédécesseur $H$ n'est pas encore considérée) et $C_1$ n'est pas vérifiée pour $E$ (pour la même raison citée ci-dessus). D'où, $H$ et $E$ sont reportés au niveau $L_2$. Le niveau $L_1$ est mis à jour $L_1 = \{I, F\}$. A chaque fois un niveau est mis à jour, les nœuds restants doivent être à nouveau vérifiés. $C_1$ n'est pas satisfaite pour $I$, ce nœud sera reporté au niveau $L_2$. Alors, le niveau résultant $L_1$ contient seulement le nœud $F$ $(L_1 = \{F\})$…

Comme il est montré à la figure 6.3, le nœud feuille $E$ est reporté deux fois et est considéré au niveau 2 au lieu des niveaux 0 et 1. Les nœuds $H$ et $I$ sont reportés trois fois, du niveau $L_1$ au niveau $L_4$. La Figure 6.3 décrit aussi le processus du OC algorithme tel qu'il est expliqué dans la section 6.4.2[2].

[2] Sur la Figure 6.3 nous schématisons seulement quelques états de succès (sans messages) pour simplification.

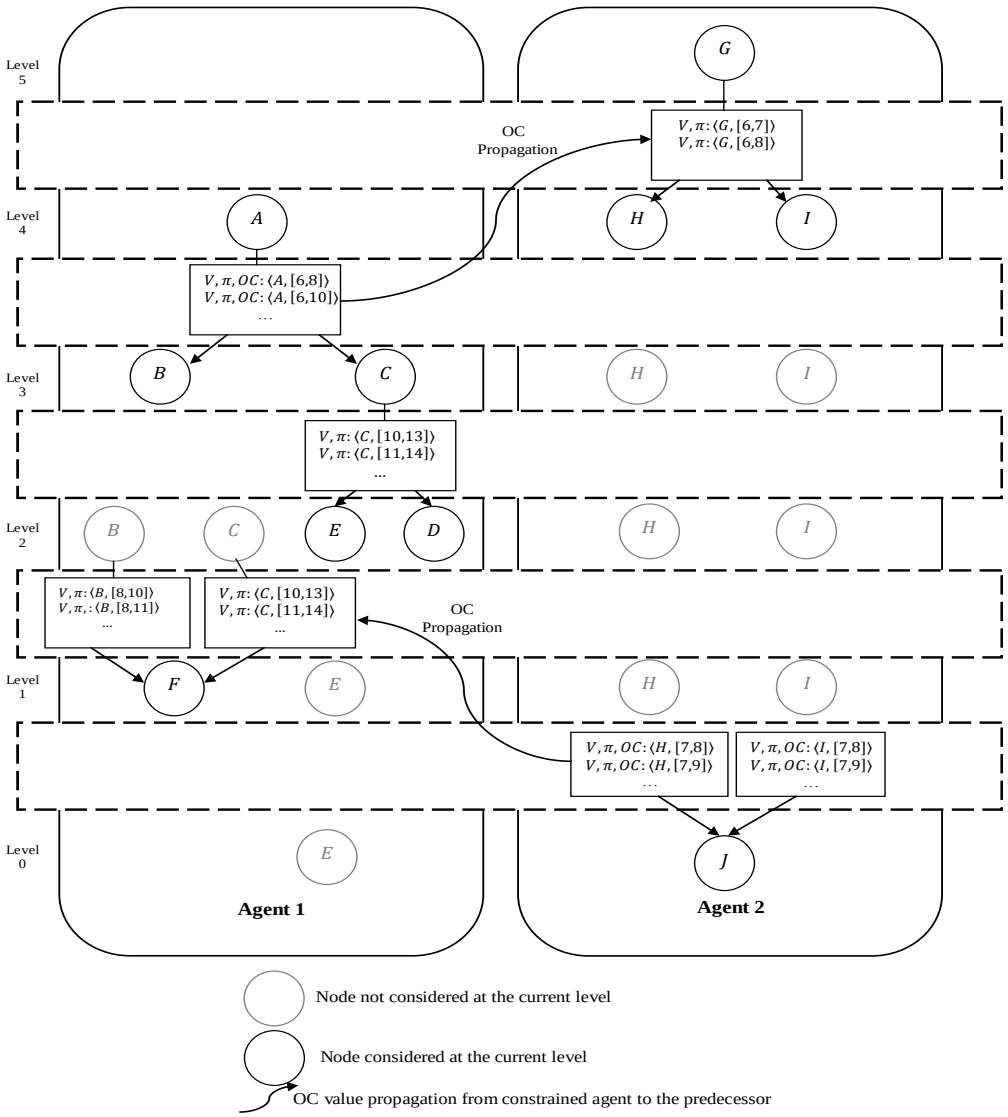


***Figure 6.3*** *Le processus du OC algorithme appliqué à l'exemple de la figure 6.2*

## 6.6.2 Modèle de communication

Nous avons proposé d'introduire la décision de la communication afin de réduire la mis-coordination causée par la relation de précédence. Cependant, introduire la communication au temps de planification mène à une augmentation significative dans la taille du problème. Cette dernière concerne l'espace des actions et l'espace des états: quand la décision de la communication est toujours disponible, l'espace d'actions sera augmenté dramatiquement parce qu'une décision de communication sera ajoutée pour chaque action du domaine. Quant à l'espace d'états, l'augmentation résulte du fait que pour chaque état, nous avons besoin de distinguer entre les états avec messages et les

états sans messages capturés dans l'état factorisé, et pour chaque état avec message nous spécifions le contenu possible du message pour que nous puissions calculer les valeurs et les politiques respectives des états.

Afin d'éviter cette augmentation dans la taille du problème, nous tirons profit de la structure du problème et nous introduisons des heuristiques utiles pour spécifier quand la décision de la communication est disponible, que doit être communiqué et avec qui communiquer. Il faut noter que ces questions ont déjà été traitées dans la littérature [Beynier and Mouaddib, 2010; Mostafa and Lesser, 2009; Roth et al., 2005], mais les réponses relatives (heuristiques) que nous proposons sont spécifiques au type du problème que nous traitons dans cette thèse.

Dans ce qui suit, nous expliquons les heuristiques que nous allons utiliser afin de réduire la taille du problème.

#### 6.6.2.1 H1: Quand communiquer?

Afin de réduire l'espace d'actions, nous exploitons la structure du problème puisque nous traitons des relations de précédence entre les agents. Cela va limiter l'information partagée aux points de précédences que nous allons appelés points de coordination. De plus, la décision de la communication est seulement disponible au niveau de l'agent dépendant quand il passe à un état d'échec partiel. Ainsi, pour chaque état d'échec partiel, une politique est calculée qui peut dicter une action de communication ou non.

#### 6.6.2.2 H2: Avec qui communiquer?

Dans le cas où la politique d'un état d'échec partiel indique la communication, l'ensemble des agents concernés sont les agents contraignants (prédécesseurs) concernés par l'échec partiel. La raison derrière l'utilisation de cette heuristique consiste à restreindre le nombre d'agents pour lesquels l'espace d'état sera augmenté par les messages qui sont les agents prédécesseurs et les agents dépendants.

#### 6.6.2.3 H3: Quoi communiquer ?

Dans cette étude, nous proposons d'attacher explicitement les sémantiques des messages partagés par les agents afin d'améliorer les performances globales de la mission.

Puisque dans la communication il y a un émetteur (l'agent dépendant) et un ou plusieurs récepteurs (les agents contraignants) nous allons analyser les contenus possibles des messages pour l'agent dépendant (successeur) et l'agent(s) prédécesseur(s).

*Au niveau de l'agent dépendant:* l'agent utilise la *query* communication et demande l'action dont il a besoin de ses prédécesseurs, afin qu'il puisse poursuivre son exécution. On pose $a_i$ une action associée à l'agent prédécesseur (contraignant) et que l'agent dépendant en a besoin, le contenu du message sera donc: $\Sigma = a_i$.

*Au niveau du prédécesseur(s):* la décision de la communication n'est pas disponible à ce point. Ainsi, à chaque instant, l'agent prédécesseur reçoit une requête, il doit répondre à l'agent dépendant (des agents coopératifs). De plus, nous supposons, dans un premier temps, que les messages ne seront pas perdus ou corrompus et prennent une unité de temps dans chaque direction (send → receive → respond, prend 2 unités de temps). Par la suite, nous supposons que le message peut être perdu soit dans la phase de transmission soit dans la phase de réception et nous proposons un mécanisme pour gérer une telle situation dans la section 6.6.5.

La réponse de l'agent prédécesseur dépend de sa politique courante. On suppose que l'agent prédécesseur est dans l'état $\langle a, [t, t^{'}], \Sigma = a_i \rangle$.

- Si sa politique courante est différente de l'action demandée par l'agent successeur, $a_i$ ( $\pi\langle a, [t, t^{'}], \Sigma = a_i \rangle \neq (a_i, t_i)$), alors le contenu du message réponse sera $\Sigma = \infty$. Il faut noter que cette situation peut décrire deux cas possibles: quand l'action en question ne sera jamais exécutée (parce que l'agent chargé de son exécution a pris un chemin différent, ne contenant pas cette action) ou l'agent ne peut pas prédire sa politique future.

Si sa politique courante correspond à l'action en question ($\pi\langle a, [t, t^{'}], \Sigma = a_i\rangle = (a_i, t_i)$ ce qui veut dire exécuter $a_i$ au temps $t_i$), alors l'agent prédécesseur (contraignant) envoie à l'agent dépendant le message $\Sigma = val = max\{ET(a_i, t_i)\}$. Ce message consiste en la plus grande date de fin de l'ensemble $ET$ des dates de fin associées à $(a_i, t_i)$, puisque l'agent prédécesseur ne peut pas connaître la date de fin de son action avant la phase d'exécution.

### 6.6.3 Calcul du coût occasionné (OC)

Le coût occasionné est calculé par l'agent dépendant et propagé pour les agents prédécesseurs. On pose $a_i$ une action prédécesseur exogène de l'action $a_j$. Ainsi, les valeurs d'OC sont calculés par l'agent chargé de l'exécution de l'action $a_j$.

Les valeurs d'OC sont calculées pour chaque $\Delta t$ qui consiste au délai causé par chaque date de fin possible de l'action $a_i$ sur la date de début de l'action $a_j$. Ces valeurs consistent en la différence entre l'utilité espérée ($V$) de l'état associé à $a_j$ quand elle commence son exécution au temps $t_j$ sans délai (à sa première date de début possible) et l'utilité espérée de l'état quand son ensemble de date de début est restreint par un certain $\Delta t$, (figure 6.4). Formellement:

$$OC(a_j, \Delta t) = V^{\Delta t=0}\big(\langle a_j, [t_j, t_j + d_j]\rangle\big) - V^{\Delta t}\big(\langle a_j, [t_j + \Delta t, (t_j + \Delta t) + d_j]\rangle\big) \quad (6.2)$$

Où, $\langle a_j, [t_j, t_j + d_j]\rangle, \langle a_j, [t_j + \Delta t, (t_j + \Delta t) + d_j]\rangle$ sont des états possibles associés à l'action $a_j$, $V$ est l'utilité espérée (valeur) de ces états, $t_j$ est la première date de début possible de l'action $a_j$. Le calcul de $V$ sera donné dans la prochaine section.

Les valeurs OC calculées (pour chaque $\Delta t$) sont propagées à l'agent prédécesseur (chargé de l'exécution de l'action $a_i$) qui va calculer à son tour sa politique en prenant compte des valeurs OC reçues.

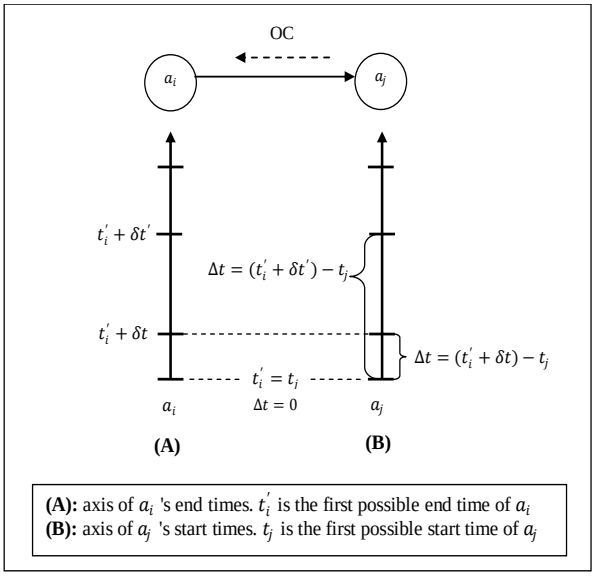


***Figure 6.4*** *Le calcul du coût occasionné*

### 6.6.4 Calcul de la politique

Dans cette sous-section, nous montrons comment la valeur et la politique de chaque état seront calculées. La première est basée sur le principe d'optimalité de Bellman [Bellman, 1957] et la deuxième consiste en une équation de Bellman augmentée par la notion du coût occasionné. Cette phase de calcul permet à chaque agent de choisir, dans chaque état, la meilleure action à exécuter considérant son utilité espérée (fonction de valeur), le coût occasionné induit sur les autres agents et le coût de la communication.

Dans ce qui suit, nous détaillons le calcul de la politique au niveau de l'agent dépendant (successeur) et l'agent contraignant (prédécesseur). Nous allons présenter le processus de calcul pour deux agents mais il reste applicable pour n'importe quel nombre d'agents et chaque agent peut être prédécesseur et successeur en même temps (tout en respectant l'aspect acyclique du graphe de mission).

#### 6.6.4.1 Agent dépendant

Cet agent peut déclencher la communication. Pour cette raison, nous différentions entre le calcul de politique de chaque type de ses états.

*a) Etat de succès* $s_i = \langle a_i, [t_i, t_i^{'}] \rangle$

Cet état est associé à l'action $a_i$. L'agent doit décider sur la prochaine action à exécuter et quand l'exécuter. Pour ce faire, la valeur attendue de l'état $s_i$ ($V(s_i)$) est calculée en se basant sur toutes les valeurs des actions ($Q$) que l'agent peut atteindre à partir de l'état $s_i$.

$$V(s_i) = R(s_i) + \max_{p, t_{i_p} \geq t_i^{'}} \left( Q\left( \left( a_{i_p}, t_{i_p} \right), s_i \right) \right) \tag{6.3}$$

Où $p = 1..m$ est le nombre de successeurs de $a_i$, ($a_{i_p}$). $R(s_i)$ est la récompense obtenue du fait d'être dans l'état $s_i$ et elle consiste à la récompense associée à l'exécution avec succès de l'action $a_i$ donné dans la description du problème. $Q\left( (a_{i_p}, t_{i_p}), s_i \right)$ est l'utilité espérée de l'exécution de l'action $a_{i_p}$ au temps $t_{i_p} \geq t_i^{'}$. Elle est déduite à partir des probabilités de transition (Succès, PF et TF) et la valeur de l'état résultant.

La politique de l'état $s_i$ est donné par:

$$\pi(s_i) = \underset{p, t_{i_p} \geq t_i^{'}}{\operatorname{argmax}} \left[ Q\left( \left( a_{i_p}, t_{i_p} \right), s_i \right) - OC\left( \left( a_{i_p}, t_{i_p} \right), s_i \right) \right] \tag{6.4}$$

Où $OC\left( (a_{i_p}, t_{i_p}), s_i \right)$ est le coût occasionné de l'exécution de l'action $a_{i_p}$ au temps $t_{i_p}$ induit sur les successeurs de $a_{i_p}$ qui appartiennent aux autres agents.

*b) Etat d'échec partiel* $s_i = \langle a_i, [t, t+1], t_i^{'} \rangle$ ou $s_i = \langle a_i, [t+1, t'], t_i^{'}, \Sigma \rangle$

Les composants de l'état d'échec partiel sont un peu différents de ceux de l'état de succès. Dans l'état d'échec partiel nous capturons la dernière action exécutée avec succès $a_i$, l'intervalle de PF $[t, t+1]$, où $t$ correspond au temps où l'agent tente d'exécuter la prochaine action dépendante et $t+1$ est le temps où l'agent réalise qu'un ou plusieurs de ses prédécesseurs n'ont pas encore terminé leurs exécutions. Nous

supposons que cela prend une unité de temps (le temps de réalisation que l'agent a passé à un état d'échec partiel), $t_i^{'}$ correspond à la date de fin de $a_i$.

Puisque la communication ne peut être déclenchée que dans l'état d'échec partiel, la décision à ce point diffère en se basant sur la présence ou non des messages. Ainsi, nous distinguons entre le calcul de politique de deux types d'état d'échec partiel (avec et sans messages).

- Etat d'échec partiel sans message $s_i = \langle a_i, [t, t+1], t_i^{'} \rangle$, soit $a_{i+1}$ l'action qui mène à l'état d'échec partiel. Nous distinguons deux scénarios:
  - Attendre pendant certain $\Delta t$ et retenter l'exécution de $a_{i+1}$. Cependant, ce scénario n'est pas adapté à notre cas, parce que si l'échec partiel se reproduit encore une fois, cela va violer les intervalles d'exécution de $a_{i+1}$ et mène à l'échec total de la mission. Cette solution a été adoptée dans [Beynier and Mouaddib, 2005, 2011].
  - Le deuxième scénario consiste à évaluer le gain entre: considérer une autre action successeur (différente de $a_{i+1}$) si elle existe dans le plan partiel ou communiquer avec l'agent(s) contraignant(s) pour demander l'action prédécesseur. Ce scénario est plus approprié pour notre cas puisque nous introduisons la décision de communication. Formellement:

$$V(s_i) = R(s_i) + \max_{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t+1} \left( Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right), V(comm) \right) \quad (6.5)$$

Où $R(s_i) = 0$, parce qu'aucune récompense n'est associée à l'état d'échec partiel $s_i$. $p - \{i+1\}$ sont toutes les actions locales successeurs de $a_i$ à l'exception de $a_{i+1}$ (qui a causé l'échec partiel), $t_{i_{p-\{i+1\}}}$ est le temps d'exécuter une action locale successeur de $a_i$ à l'exception de $a_{i+1}$. $Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right)$ est l'utilité espérée d'exécuter une action successeur de $a_i$ à l'exception de $a_{i+1}$ au temps supérieur ou égale à $t+1$ (temps de la réalisation de l'échec partiel). $V(comm) = Q((a_{i+1}, t_{i+1}), s_i) - C_{com}$, avec $t_{i+1} = (t+1) + 2$ et $C_{com}$ est le coût induit par la communication. Il faut noter que si $t + 1 \geq$

$max\{ST_{a_{i+1}}\}$, $Q((a_{i+1}, t_{i+1}), s_i) = 0$ parce que l'agent ne peut pas retenter l'exécution de l'action en question ($a_{i+1}$), il n'y a pas d'autres dates de début possibles pour cette action.

La politique est alors donnée par:

$$\pi(s_i) = \underset{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t+1}{arg\max} \left( Q\left(\left(a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}}\right), s_i\right) - OC\left(\left(a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}}\right), s_i\right), V(comm)\right) \quad (6.6)$$

- Etat d'échec partiel avec message $s_i = \langle a_i, [t+1, t'], t_i', \Sigma \rangle$

Quand l'état d'échec partiel contient un message, la décision sur la prochaine action à exécuter dépend du contenu de ce message. Quand le message contient $\infty$, l'agent doit choisir une action successeur alternative de $a_i$ à l'exception de $a_{i+1}$. Cela correspond au changement de chemin dans son plan local partiel. L'agent doit changer de chemin dans son plan local partiel aussi quand le message reçu contient une valeur (la plus grande date de fin de l'action prédécesseur) qui est supérieure ou égale au maximum des dates de début possibles de l'action $a_{i+1}$, puisque l'agent peut violer les contraintes temporelles de $a_{i+1}$.

Quand la valeur reçue est inférieure du maximum des dates de début possibles de $a_{i+1}$, l'agent peut choisir entre changer de chemin ou ré-exécuter l'action $a_{i+1}$.

$$V(s_i) = \begin{cases} R(s_i) + \max\limits_{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t'} \left( Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right) \right) & if\ \Sigma = \infty\ or\ \Sigma = val \geq max\{ST_{a_{i+1}}\} \quad (6.7.1) \\ R(s_i) + \max\limits_{\substack{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t' \\ val \leq t_{i+1} < max\{ST_{a_{i+1}}\}}} \left( Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right), Q((a_{i+1}, t_{i+1}), s_i) \right) & if\ \Sigma = val < max\{ST_{a_{i+1}}\} \quad (6.7.2) \end{cases}$$

Après le calcul de l'utilité espérée de l'état $s_i$, nous passons au calcul de sa politique.

$$\pi(s_i) = \begin{cases} \underset{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t'}{argmax} \left( Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right) - OC\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right) \right) & if\ \Sigma = \infty\ or\ \Sigma = val \geq max\{ST_{a_{i+1}}\} \quad (6.8.1) \\ \underset{\substack{p-\{i+1\}, t_{i_{p-\{i+1\}}} \geq t' \\ val \leq t_{i+1} < max\{ST_{a_{i+1}}\}}}{argmax} \left( Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right) - OC\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right), Q((a_{i+1}, t_{i+1}), s_i) - OC((a_{i+1}, t_{i+1}), s_i) \right) & if\ \Sigma = val < max\{ST_{a_{i+1}}\} \quad (6.8.2) \end{cases}$$

Où $R(s_i) = 0$ parce que $s_i$ est un état d'échec partiel. $p - \{i + 1\}$ sont toutes les actions successeurs de l'action $a_i$ sauf $a_{i+1}$ (qui a causé le PF). $t_{i_{p-\{i+1\}}}$ est le temps d'exécution de l'action locale successeur de $a_i$ à l'exception de l'action $a_{i+1}$. $Q\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right)$ est l'utilité espérée de l'exécution de l'action successeur de $a_i$ à l'exception de $a_{i+1}$ au temps supérieur ou égale à $t'$ (le temps où le message est reçu). $t_{i+1}$ est le temps de réexécution de l'action $a_{i+1}$ après la réception du message contenant une valeur, ce temps doit être supérieur ou égale à la valeur reçue et inférieur au maximum des dates de début possibles de $a_{i+1}$ (afin de respecter les contraintes temporelles de $a_{i+1}$). $Q((a_{i+1}, t_{i+1}), s_i)$ est l'utilité espérée d'exécuter l'action $a_{i+1}$ au temps $t_{i+1}$. $OC\left( \left( a_{i_{p-\{i+1\}}}, t_{i_{p-\{i+1\}}} \right), s_i \right)$ est l'OC de l'exécution de $a_{i_{p-\{i+1\}}}$ à $t_{i_{p-\{i+1\}}}$ et $OC((a_{i+1}, t_{i+1}), s_i)$ est l'OC de l'exécution de $a_{i+1}$ à $t_{i+1}$.

*c) Etat d'échec total* $s_i = \langle a_i, [t_i, t_i'], fail \rangle$

A partir de cet état, aucune politique n'est calculée.

$$V(s_i) = -R(s_i) - \sum_{a_{\neq i} \in EndoSuc\ (a_i)} R(s_{\neq i}) \tag{6.9}$$

Où $-\boldsymbol{R(s_i)}$ est la perte en récompense de l'exécution échouée de l'action $\boldsymbol{a_i}$. $\sum_{\boldsymbol{a_{\neq i} \in EndoSuc(a_i)}} \boldsymbol{R(s_{\neq i})}$ consiste à la perte en récompense de toutes les actions restantes qui sont atteignables à partir de l'action échouée $\boldsymbol{a_i}$ et exécutées par le même agent, que nous appelons successeurs endogènes, $\boldsymbol{EndoSuc(a_i)}$).

#### 6.6.4.2 Agent prédécesseur (Contraignant)

Nous rappelons que pour cet agent aucune politique de communication ne sera calculée. L'agent doit synchroniser avec l'agent dépendant si ce dernier déclenche la communication. Comme cet agent n'a pas d'action dépendante, nous ne traitons pas l'état d'échec partiel[3].

*a) Etat de succès* $s_i = \langle a_i, [t_i, t_i'] \rangle$ *or* $s_i = \langle a_i, [t_i, t_i'], \Sigma \rangle$

A partir de l'état de succès, l'agent prédécesseur peut choisir entre les actions successeurs de l'action $a_i$.

$$V(s_i) = R(s_i) - C_{com} + \max_{p, t_{i_p} \geq t_i'} \left( \left( Q\left(a_{i_p}, t_{i_p}\right), s_i \right) \right) \tag{6.10}$$

Quand on calcule la politique de cet état, l'agent doit considérer le coût occasionné induit sur les successeurs de $a_i$ qui appartiennent aux autres agents.

[3] Le cas où cet agent est contraignant et dépendant en même temps peut être pris en compte par notre modèle. Nous distinguons entre les deux cas pour simplifier l'explication.

$$\pi(s_i) = \underset{p, t_{i_p} \geq t_i^{'}}{\operatorname{argmax}} \left[ Q\left(\left(a_{i_p}, t_{i_p}\right), s_i\right) - OC\left(\left(a_{i_p}, t_{i_p}\right), s_i\right)\right] \quad (6.11)$$

Il faut noter que quand $s_i = \langle a_i, [t_i, t_i^{'}]\rangle$, $C_{com} = 0$ et quand $s_i = \langle a_i, [t_i, t_i^{'}], \Sigma\rangle$, $C_{com} = 2$. $R(s_i)$ est la récompense reçue du fait d'être dans l'état $s_i$ et elle consiste à la récompense associée à l'exécution avec succès de l'action $a_i$ donnée dans la description du problème. $Q\left(\left(a_{i_p}, t_{i_p}\right), s_i\right)$ est l'utilité espérée de l'exécution de l'action $a_{i_p}$ au temps $t_{i_p} \geq t_i^{'}$ à partir de l'état $s_i$. $OC\left(\left(a_{i_p}, t_{i_p}\right), s_i\right)$ est le coût occasionné de l'exécution de $a_{i_p}$ au temps $t_{i_p}$ provoqué sur les successeurs de $a_i$ qui appartiennent aux autres agents.

*b) Etat d'échec total $s_i = \langle a_i, [t_i, t_i^{'}], \Sigma, fail\rangle$ ou $s_i = \langle a_i, [t_i, t_i^{'}], fail\rangle$*

$$V(s_i) = -R(s_i) - \sum_{a_{\neq i} \in EndoSuc\ (a_i)} R(s_{\neq i}) - C_{com} \quad (6.12)$$

Où $-R(s_i)$ est la perte en récompense de l'exécution échouée de l'action $a_i$. $\sum_{a_{\neq i} \in EndoSuc\ (a_i)} R(s_{\neq i})$ consiste à la perte en récompense de toutes les actions restantes qui sont atteignables à partir de l'action échouée $a_i$ et exécutées par le même agent, que nous appelons successeurs endogènes, $EndoSuc(a_i)$). $C_{com}$ est le coût de la communication.

### 6.6.5 Gestion des messages perdus

Dans cette sous-section, nous étendons notre proposition afin de gérer les messages perdus. Dans ce cas, l'agent peut attendre indéfiniment pour une réponse ce qui mène à une violation de ses contraintes temporelles. Pour remédier à ce problème, nous augmentons les états d'échec partiel avec un temps d'attente adaptatif $\Delta t$. Ce $\Delta t$ dépend de l'état de l'agent et prend compte de ses contraintes temporelles. Ainsi, pour chaque nouvel état d'échec partiel, l'algorithme de calcul de politique calcule la politique correspondante.

Selon l'équation (6.5), la politique de l'état d'échec partiel $s_i = \langle a_i, [t, t+1], t_i' \rangle$, est soit communiquer, soit exécuter une action alternative de $a_{i+1}$ (exécuter un des successeurs de $a_i$ à l'exception de $a_{i+1}$). La valeur de la communication dans ce cas (le cas de messages perdus) est $V(comm) = Q\big((a_{i+1}, t_{i+1}), s_i\big) - C_{com}$, où $t_{i+1} = (t+1) + \Delta t$. $\Delta t$ correspond au temps d'attente après lequel l'agent peut ré-exécuter $a_{i+1}$ ou exécuter une autre action.

Si après avoir attendu $\Delta t$ unité de temps et aucun message n'a été reçu, l'agent suppose que le message est perdu soit à la phase de transmission ou à la phase de réception, et il doit choisir une autre action à exécuter (avant violation de ses contraintes). Ainsi, nous proposons de calculer $\Delta t$ en se basant sur les contraintes d'exécution des actions alternatives possibles de l'action $a_{i+1}$. Nous posons,

$$\Delta t = \max\{ST_{alter}\} - t_{current}$$

Où $t_{current}$ correspond à la date de fin de l'échec partiel $(t+1)$. $\max\{ST_{alter}\}$ correspond à la plus grande date de début valide tirée des ensembles de dates de début possibles des actions alternatives. Une date de début valide veut dire une date de début possible qui ne viole pas les contraintes temporelles de l'action. Elle est extraite des états de succès de l'action(s) alternative(s) de $a_{i+1}$. La raison derrière le choix de cette valeur est qu'après avoir attendu $\Delta t$ unités de temps et qu'aucun message n'est reçu, l'agent peut toujours choisir une action à entreprendre (une action alternative sur un chemin différent que celui contenant $a_{i+1}$) tout en respectant ses contraintes temporelles. La figure 6.5 montre le développement des états d'échec partiel dans le cas d'une communication réussie (cas 1) avec un temps fixe (égale à 2: 1 unité de temps pour l'envoi et 1 unité de temps pour la réception) et la communication stochastique où les messages peuvent être perdus (cas 2).

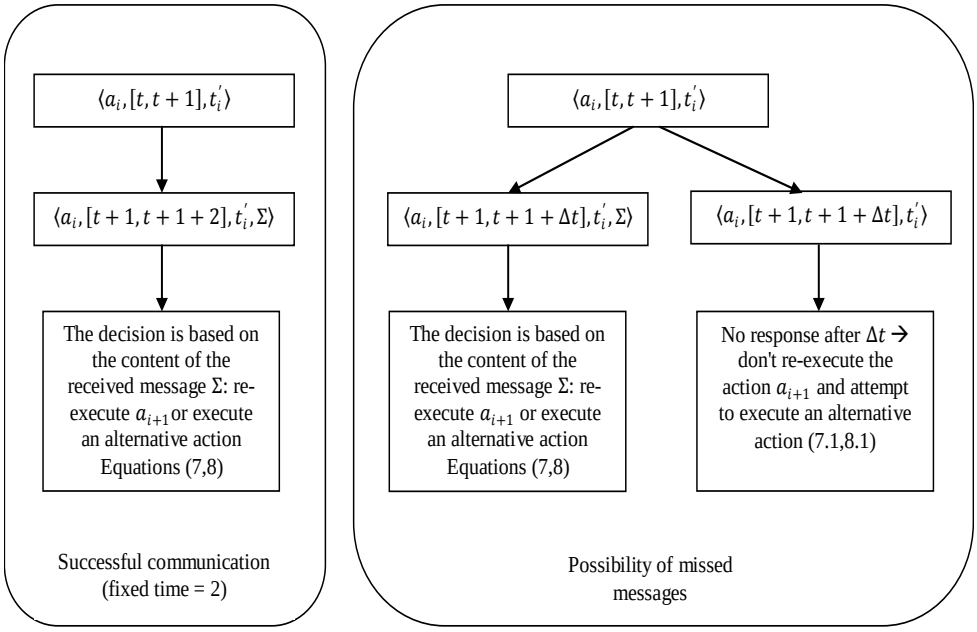


***Figure 6.5*** *Les états d'échec partiel dans la communication avec succès et la communication avec échec*

**Exemple.** Selon l'exemple présenté à la section 6.3, $s = \langle A, [8,9], 8 \rangle, s' = \langle A, [10,11], 8 \rangle$ sont deux états d'échec partiel associés à l'action $C$ quand l'action $H$ n'a pas terminé son exécution aux temps 8 et 10. La politique des états $s$ et $s'$ dicte la communication et $\max\{ST_B\} = 12$ (où $\max\{ST_B\}$ est le maximum des dates de début possibles de l'action $B$ – l'action alternative de $C$ de l'agent 1 extraite des états de succès de $B$).

Nous avons,

$$\Delta t = \max\{ST_B\} - t_{current}$$

D'où, pour l'état $s$, nous avons:

$$\Delta t = 12 - 9 = 3$$

Pour l'état $s'$, nous avons:

$$\Delta t = 12 - 11 = 1$$

Les nouveaux états dans lesquels l'agent va recevoir ou non un message $\Sigma$ sont: $s = \langle A, [9{,}12], 8\rangle$ ou $s = \langle A, [9{,}12], 8, \Sigma\rangle$ et $s' = \langle A, [11{,}12], 8\rangle$ ou $\langle A, [11{,}12], 8, \Sigma\rangle$. Pour chaque état, les valeurs et les politiques seront calculées offline par nos algorithmes de calcul de politiques et de valeurs.

## 6.7 Résultats expérimentaux

Nos expérimentations sont orientées vers la performance et la qualité de la solution contrairement au travail de [Beynier and Mouaddib, 2011] qui est orienté vers la scalabilité de l'approche proposée. En effet, l'OC-DEC-MDP a été prouvé qu'il passe à l'échelle. Notre objectif principal est l'extension de ce modèle aux problèmes avec des plans locaux partiels et la gestion des problèmes qui en découlent en proposant un riche modèle de communication.

Nous avons fait varier le nombre d'agents ainsi que le nombre de dépendances et nous avons évalué les conséquences d'ajout de communication en termes de gain global. Le gain global dans notre cas est mesuré par la récompense cumulée obtenue par les agents après l'exécution de la mission.

Il est important de noter que le type de problèmes que nous traitons souffre de la difficulté d'obtenir des scénarios aléatoires ou des instances qui leurs politiques dictent la communication [Mostafa, 2011]. Les instances générées aléatoirement ne sont pas assez dépendantes pour déclencher la communication. Pour cette raison, pendant la phase de génération de dépendances, le maximum de dépendances ajoutées a été fixé d'une manière empirique, de telle sorte qu'on évite le cycle. Nous rappelons que le graphe de mission doit être acyclique. En outre, nous avons suivi quelques règles afin d'insérer une dépendance. Tout d'abord, l'action dépendante doit être à un niveau supérieur ou égal au niveau de son action prédécesseur dans le graphe de mission. Deuxièmement, dans un niveau donné, il faut avoir au moins une action sans précédence exogène pour que l'agent puisse poursuivre son exécution. Lors de la génération de nos instances, nous avons supposé que le facteur de branchement est égal à 2.

Nous avons généré trois instances du problème:

- La première instance est composée de 15 tâches réparties entre trois agents,
- La deuxième instance est composée de 21 tâches distribuées entre 4 agents et,
- La troisième instance est composée de 40 tâches réparties entre 6 agents.

La topologie de chaque instance est donnée dans le tableau 6.1 avec le nombre d'états générés après la propagation temporelle. Les principaux paramètres qui influent la taille d'une instance qui est mesurée par le nombre d'états sont: le nombre des actions de chaque agent, le nombre de dépendances (les relations de précédence) et la taille de la fenêtre temporelle. Comme nous pouvons le voir dans le tableau 6.1, il y'a une croissance dans l'espace d'états dans le cas de communication avec possibilité de perte de messages par rapport à la communication réussie. Cette augmentation est interprétée par le fait que quand les messages perdus sont considérés, trois types d'états sont considérés après avoir envoyer un message: état d'échec partiel avec $\Sigma = \infty$, état d'échec partiel avec $\Sigma = val$, et état d'échec partiel sans message. Ce dernier état interprète le cas où l'agent dépendant envoie une requête et après avoir attendu $\Delta t$ unités de temps il n'y a pas de réponse (le message est perdu). Ceci est en contraste avec le cas de la communication réussie où l'espace d'états est augmenté par deux types d'état d'échec partiel seulement: état d'échec partiel avec $\Sigma = \infty$ et état d'échec partiel avec $\Sigma = val$.

| Inst – parameters | Nb of agents | Nb Of actions | Nb of levels | Branching factor | Nb of local dependencies | Nb of global dependencies | State space (case 1) | State space (case 2) |
|---|---|---|---|---|---|---|---|---|
| Instance 1 | 3 | 15 | 3 | 2 | 12 | 5 | 211 | 239 |
| Instance 2 | 4 | 21 | 3 | 2 | 17 | 7 | 443 | 466 |
| Instance 3 | 6 | 40 | 4 | 2 | 34 | 15 | 762 | 816 |

***Tableau 6.1*** *la composition de chaque instance du problème (cas 1: communication réussie, cas 2: communication avec échec)*

Il est important de noter que sans l'utilisation des heuristiques proposées à la sous-section 6.6.2, la taille des deux espaces d'états (cas 1 et cas 2) sera plus large. En effet, sans aucune heuristique sur la communication et comme résultat de la communication au temps de planification, nous devons considérer pour chaque état son équivalent avec messages.

Pour ces instances, nous avons mené des expériences qui se composent de deux parties: dans la partie 1, le but est de montrer la nécessité de communication notamment en cas de contraintes temporelles strictes pour éviter l'échec total de la mission. Dans la partie 2, nous avons calculé le gain global des agents lorsque la communication est ubiquitaire (ce qui représente une borne supérieure du gain global) et nous avons mesuré la perte de gain global en introduisant la communication coûteuse. Pour cette partie les deux cas de la communication réussie et la communication avec possibilité de perte de messages sont étudiés.

Les simulations effectuées ont été réalisées en utilisant la plateforme JADE (*Java Agent DEvelopment framework*) [Singh et al., 2011; Prakash et al., 2014]. Après le calcul de la politique de chaque agent (phase hors ligne), nous passons à l'exécution de la politique jointe (phase en ligne en utilisant JADE). Chaque agent $Ag$ a son plan local partiel dans lequel chaque action $a_i$ a une liste de prédécesseurs exogènes $ExoPred(a_i)$, la liste de tous ses états possibles $S$ avec les politiques correspondantes $\Pi(S)$ et les états résultants tel que $S, \Pi(S) \rightarrow S$. Toutes ces informations (description du problème et la politique calculée) représentent l'entrée de la phase en ligne. Quand un agent observe son état, il exécute la politique correspondante et il transite à l'un de ses états suivants. Pour simuler la transition vers un état non déterministe, nous procédons comme d'habitude dans les MDPs [Mostafa and Lesser, 2009; Beynier and Mouaddib, 2011] et nous randomisons entre tous les états qui pourraient en résulter. Depuis le nouvel état résultant, l'agent exécute une nouvelle fois sa politique correspondante et ainsi de suite. Chaque fois que la politique est exécutée, la récompense correspondante est cumulée.

**Partie 1:** La figure 6.6 (a, b et c) montre que la récompense cumulative lorsque les agents peuvent communiquer est toujours supérieure à la récompense cumulée pour

le cas de non-communication même pour les larges contraintes temporelles. Cependant, dans le cas d'intervalles serrés et sans communication, l'échec total de la mission est important. Cela est dû à l'échec partiel des actions dépendantes qui conduit à une violation des contraintes serrées. Lorsque la communication est introduite, l'échec total n'a pas eu lieu parce que les agents opteront toujours pour une politique de communication, même si la communication est coûteuse. La perte dans l'utilité espérée totale induite de la communication est moins importante que l'échec total de la mission, parce que dans le cas d'intervalles serrés le prochain échec sera éventuellement un échec total au lieu d'un échec partiel dans le cas de contraintes plus larges.

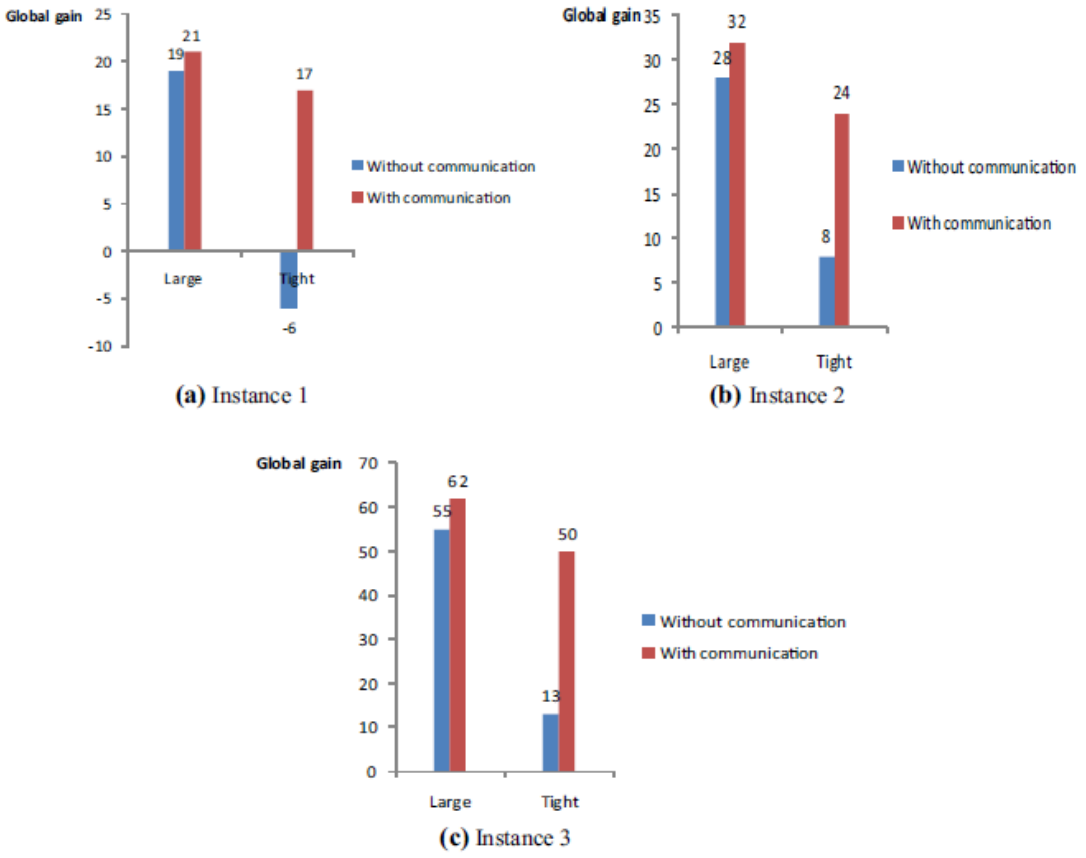


***Figure 6.6*** *Performance des agents*

Comme nous pouvons le voir à la figure 6.6, plus le besoin de la communication est augmentée (matérialisée par une augmentation de dépendances) plus le gain global

des agents est élevé. La raison derrière cette augmentation est que les agents évaluent la valeur de communiquer avant d'adopter une politique de communication, de telle sorte qu'ils trouvent le meilleur compromis entre l'utilité des informations reçues par la communication, le coût de la communication et l'utilité des alternatives sans communication tout en respectant leurs contraintes.

La croissance du nombre de dépendances a deux conséquences principales. Elle augmente la nécessité de la communication, d'une part. D'autre part, elle incrémente le nombre d'intervalles d'exécution. Comme les agents travaillent sous pression de temps, le nombre d'intervalles d'exécution qui ne respectent pas les contraintes temporelles est augmenté. Dans ce cas la communication semble être un bon choix pour les agents afin d'éviter les situations de mis-coordination et l'échec total de la mission et ainsi la performance des agents sera améliorée.

**Partie 2:** Pour les trois instances, nous avons exécuté la politique jointe trouvée hors ligne et nous avons calculé le gain global en mettant le coût de la communication égal à zéro (communication ubiquitaire). Ceci constitue le meilleur gain global que les agents peuvent obtenir. Ensuite, nous avons ajouté un coût de communication ($C_{com}$ =2) nous avons calculé à nouveau le gain global des agents pour le cas où la communication est réussie et elle consomme deux unités de temps, et quand il y a une possibilité de perte de messages. En outre, nous avons fait varier pour chaque instance exécutée le nombre de dépendances à l'égard de la propriété acyclique du graphe de mission.

Comme le montre la figure 6.7 (a, b et c) et la figure 6.8 (a, b et c) il y a une perte en récompense cumulative quand la communication est introduite. En outre pour chaque instance, lorsque le nombre de dépendances augmente, une perte de gain global est capturée.

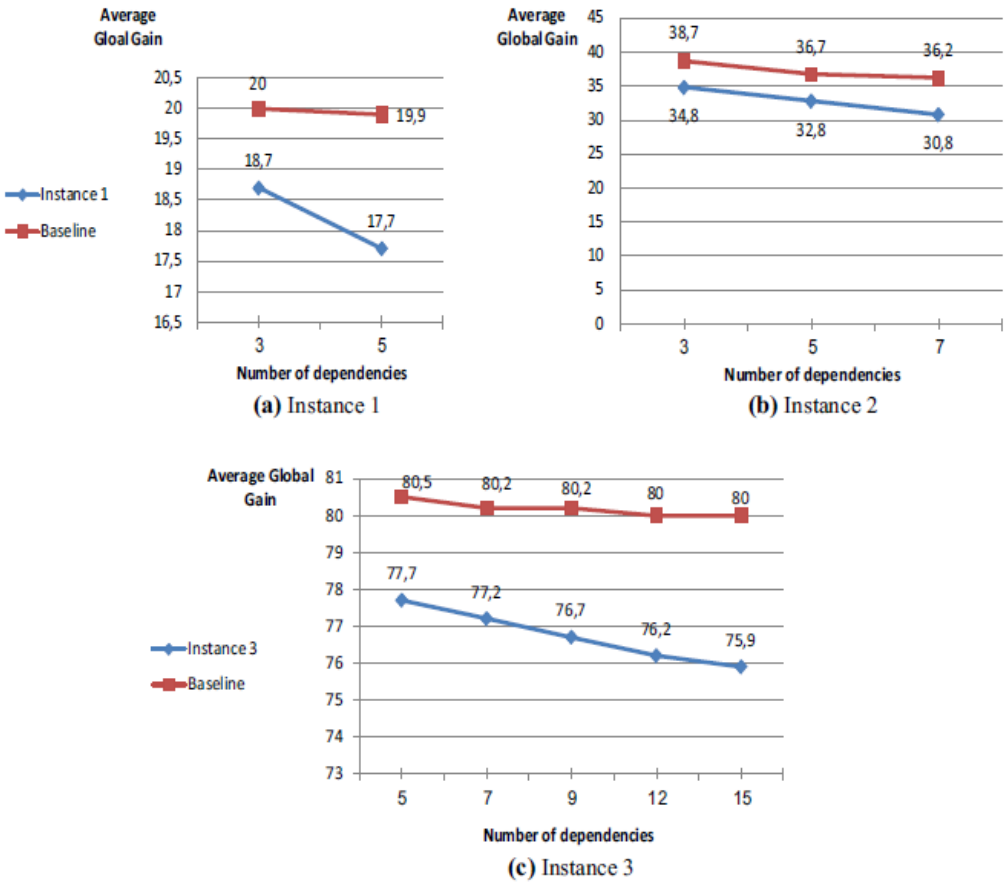


***Figure 6.7*** *L'influence du nombre de dépendances sur le gain total dans la communication avec succès*

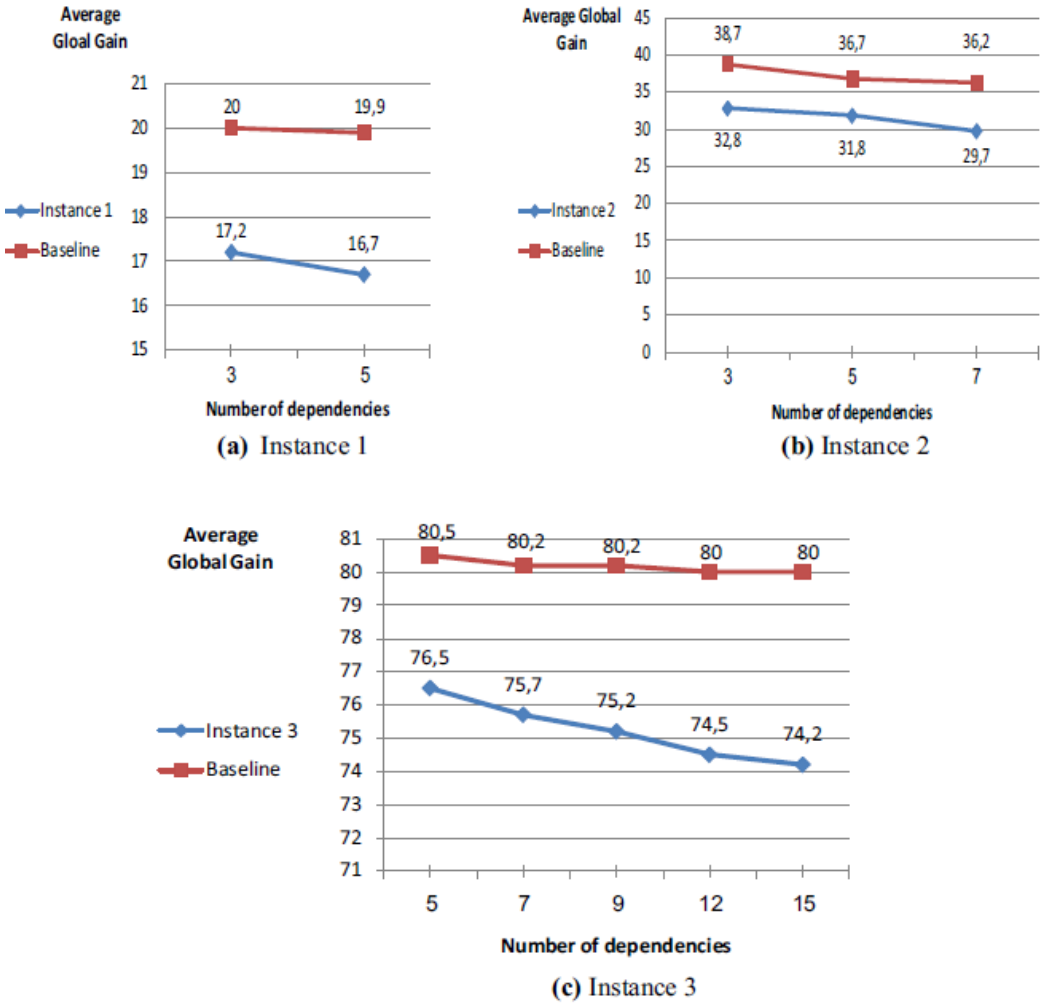


***Figure 6.8*** *L'influence du nombre de dépendances sur le gain total dans le cas de la communication stochastique avec messages perdus*

La diminution de la récompense globale est interprétée par le fait que la communication est plus sollicitée lorsque le nombre de dépendances augmente. En conséquence de l'augmentation de la nécessité de communication, le coût associé est plus considéré.

Lorsque nous avons traité les messages perdus, nous avons observé une perte en récompense cumulative comparant au cas de communication réussie. Ceci peut être expliqué par le fait que l'agent peut ne pas recevoir une réponse à sa requête. Ainsi, il est d'abord pénalisé pour avoir communiqué et il sera obligé de choisir une autre action qui a une récompense moins importante que la première action choisie.

Les résultats expérimentaux ont montré l'efficacité des heuristiques introduites afin de prendre la décision de communication menant à la meilleure solution possible en fonction des contraintes du problème (temporelle et précédence). Ces heuristiques ont pour but de réduire la taille du problème et optimiser l'information à partager.

## 6.8 Etude comparative

Plusieurs travaux ont été réalisés sur la communication dans les modèles de la théorie de la décision, mais le plus proche de notre travail est celui de Beynier et Mouaddib. Dans cette section, nous essayons de surligner les similitudes et les différences entre ces deux travaux [Beynier and Mouaddib, 2010;Abdelmoumène and Belleili, 2016] afin de mieux comprendre comment l'un se situe par rapport à l'autre.

La comparaison entre les deux propositions peut être faite sur deux volets. Nous comparons, dans un premier temps, entre la topologie du graphe de la mission, le modèle de décision et les questions relatives à chaque modèle. Dans un second temps, nous comparons les deux modèles de communication proposés afin de traiter les questions spécifiques à chaque travail. Le Tableau 6.2 compare la topologie de chaque graphe de mission.

Comme le montre le tableau 6.2, le problème de décision considéré dans [Beynier and Mouaddib, 2010] traite des plans linéaires. Cela résulte en une réduction de l'espace de solution dans lequel il y a un seul chemin où chaque nœud est la seule action. Chaque action dans le chemin a différents *schedules* possibles (en raison de ses plusieurs durées). Notre extension résulte en un problème de décision plus difficile et un espace de solution plus large où plusieurs chemins possibles existent (plan partiel). Chaque action dans un chemin a différents *schedules* possibles (en raison de ses plusieurs durées). Cela nous a conduits à proposer un modèle de communication adéquat (présenté à la section 6.6.2) afin de gérer les problèmes résultants des plans partiels.

| Travail / Critère de Comparaison | **[Beynier and Mouaddib, 2010]** | **[Abdelmoumène and Belleili, 2016]** |
|---|---|---|
| **Topologie du graphe des actions (plan local)** | Linéaire | Partiel |
| **Type de dépendances** | -Contraintes de précédence entre les actions des agents (exogènes)<br><br>-Actions locales totalement ordonnées | -Contraintes de précédence entre les actions des agents (exogènes)<br><br>-Actions locales partiellement ordonnées (précédences endogènes) |
| **Problème de décision** | -Quand exécuter l'action connue a priori | -Quelle action exécuter<br><br>-Quand l'exécuter |
| **Pré-conditions relatives au calcul du coût occasionné (OC)** | -L'OC de l'action dépendante doit être calculé avant son prédécesseur | -L'OC de l'action dépendante doit être calculé avant son prédécesseur<br><br>-Deux actions qui ont le même prédécesseur doivent être considérées au même niveau |

***Tableau 6.2*** *Comparaison entre la topologie des graphes de mission*

Nous donnons ci-dessous une comparaison entre les deux modèles de communication. Pour effectuer cette comparaison, nous utilisons les critères présentés au chapitre 4: calcul de politique (temps de raisonnement), ET/OU communication, Quand communiquer?, Avec qui communiquer?, Quoi communiquer?, type de communication (tell, query, sync) et la valeur de la communication (VoC). Nous introduisons aussi quelques autres critères de comparaison, que nous présentons ci-dessous.

- But: pourquoi introduire la communication entre les agents.

- Modèle de l'action de communication: est-ce qu'elle est considérée similaire à l'action niveau domaine ou chaque action représente une sorte différente de l'autre?
- Problème de décision: quelles sont les décisions à prendre par chaque agent à chaque étape de décision.
- Qui déclenche la communication: quel agent initie la communication?
- Flux de l'information: dans quelle direction l'information sera envoyée.
- Méthode de solution: comment la politique jointe est calculée?
- Après communication: l'influence du message reçu sur le comportement des agents. En d'autres termes, comment les agents réagissent après un cycle de communication (réception de message).
- Communication stochastique: traiter les messages perdus, corrompus ou retardés.

Le *communicative* OC-DEC-MDP permet aux agents de communiquer la date de fin d'une action exécutée précédemment avec succès. Le problème avec ce modèle est que l'information envoyée est disponible quand il est trop tard pour l'agent dépendant d'en bénéficier. Ceci résulte de l'exécution asynchrone des actions.

En outre, la communication est utilisée dans une seule direction. Cela veut dire que quand l'agent dépendant arrive à l'action dépendante avant ses prédécesseurs, il est obligé d'attendre ce qui peut affecter ses contraintes temporelles et ainsi les contraintes temporelles des agents qui lui sont dépendants (par propagation).

Dans ce travail et à travers le modèle de communication proposé, nous avons essayé de résoudre ces problèmes en plus d'autres problèmes émergents résultant des actions partiellement ordonnées. Nous avons supposé que l'agent dépendant peut demander l'action dont il a besoin des autres agents (quand il passe à un état d'échec partiel et sa politique dicte la communication). En recevant une requête, l'agent prédécesseur est obligé de synchroniser avec l'agent dépendant. Le langage utilisé dans ce travail, est choisi pour tenir compte de la possibilité de recevoir la requête avant de décider d'exécuter l'action prédécesseur ou non (pour faire face à l'asynchronisme).

| Travail / Critère de Comparaison | **[Beynier and Mouaddib, 2010]** | **[Abdelmoumène and Belleili, 2016]** |
|---|---|---|
| **But** | Réduire la mis-coordination: états de PF | -Réduire la mis-coordination: états de PF<br><br>-Eviter l'échec total de la mission (l'agent dépendant peut attendre indéfiniment pour une action prédécesseur qui ne sera jamais exécutée: plusieurs chemins d'exécution) |
| **Modèle de l'action de communication** | L'action de communication et l'action niveau domaine sont unifiées. Ainsi, l'action de communication a une fenêtre temporelle et des durées probabilistes | La communication est considérée séparément. Ainsi, l'action de communication n'a pas de contrainte et a un coût fixe. |
| **Problème de décision** | Communiquer, attendre, exécuter | Communiquer, exécuter |
| **Temps de raisonnement** | Temps de planification (offline) | Temps de planification (offline) |
| **ET/OU communication** | OU communication | OU communication |
| **Qui déclenche la communication** | Agent prédécesseur (contraignant) | Agent dépendant (successeur) |
| **Quand communiquer** | Après l'exécution avec succès d'une action prédécesseur | Après un état d'échec partiel |
| **Avec communiquer** | Les agents dépendants (successeurs) | Les agents contraignants (prédécesseurs) |
| **Type de communication** | Tell | Query |
| **Flux de l'information** | Unidirectionnel: de l'agent prédécesseur à l'agent dépendant | Bidirectionnel |

| | | |
|---|---|---|
| **Quoi communiquer (le contenu de Σ)** | $\Sigma = t_i^{'}$ (la date de fin de l'action prédécesseur $a_i$) | Pour l'agent dépendant: $\Sigma = a_i$ (action prédécesseur)<br><br>Pour l'agent prédécesseur: $\Sigma = \infty$ ou<br><br>$\Sigma = max\{ET(a_i, t_i)\}$ (la plus grande date de fin de l'ensemble $ET$ des dates de fin associées à $(a_i, t_i)$) |
| **Valeur de la communication** | -Calculer par l'agent prédécesseur indépendamment de l'agent dépendant<br><br>-Equation de Bellman | -Calculer par l'agent dépendant<br><br>-Mesurer par la valeur attendue de l'état résultant après communication et le coût de la communication<br><br>$V(comm) = Q((a_{i+1}, t_{i+1}), s_i) - C_{com}$ |
| **Méthode de solution** | OC algorithme | Version étendue d'OC algorithme pour traiter les plans locaux partiels et le modèle de communication proposé |
| **Après communication** | L'agent dépendant continue à attendre | L'agent dépendant suit le chemin dicté par son état courant et l'information reçue (le contenu du message) |
| **Communication stochastique** | N'est pas gérée | Messages perdus |

***Tableau 6.3*** *Comparaison entre les deux modèles de communication*

De plus, le partage de l'information est réalisé dans deux directions (communication bidirectionnel), afin de maintenir la coordination. En effet, dans [Beynier and Mouaddib, 2010], il y a un découplage entre les agents parce que l'agent prédécesseur décide d'envoyer l'information de façon indépendante sans considérer si cela est nécessaire pour l'agent dépendant et si cela augmente vraiment la récompense totale attendue.

## 6.9 Conclusion

Modéliser des applications distribuées du monde réel en utilisant les DEC-MDPs nécessite une adaptation spécifique afin de prendre en compte les contraintes sur l'exécution des actions.

Dans ce chapitre nous avons étendu un modèle de l'état de l'art OC-DEC-MDP afin de gérer des plans partiels et les décisions de la communication. Dans ce type de problème la communication ne peut être que bénéfique afin d'assurer une bonne coordination entre les agents. Nous avons proposé un riche modèle de communication qui est approprié aux types de problèmes que nous traitons dans cette thèse. La décision sur la politique de chaque agent est basée sur le type de l'état de l'agent. Elle résulte d'un compromis entre l'utilité espérée de l'agent, son coût occasionné sur les autres agents et la valeur de la communication.

Cependant, la communication est coûteuse et doit être limitée. Pour optimiser la décision de la communication, nous avons construits des heuristiques sur le moment de la communication, l'ensemble des agents communicants et ce que doit être communiqué. Nous avons aussi considérés deux cas de la communication: communication avec succès et communication avec échec. Les résultats obtenus sont prometteurs.

# *Conclusion et Perspectives*

Modéliser les applications distribuées du monde réel en utilisant les Processus Décisionnels de Markov Décentralisés (DEC-MDP) nécessite une adaptation spécifique afin de prendre en compte les contraintes d'exécution des actions. Le travail présenté dans cette thèse a eu pour premier objectif de rapprocher le domaine de recherche d'applications multi-agents ainsi que les besoins suscités par le développement de telles applications et les solutions apportées par les recherches en planification de la théorie de la décision. En effet, la théorie de la décision propose de très puissants formalismes qui permettent la prise en considération de différents degrés d'incertitude ainsi que l'aspect séquentiel relatifs à la prise de décision, mais qui restent très limités en termes d'applicabilité.

Nous nous sommes plus particulièrement intéressées par les approches basées sur les processus décisionnels de Markov et les processus décisionnels de Markov décentralisés et sommes parties du constat que très peu de ces travaux pourraient être utilisés dans des applications réelles.

Une première partie bibliographique nous a permis de poser le problème de prise de décision séquentielle sous incertitude dans les deux cas : mono-agent et multi-agents. Nous avons mis en relief les déficits des modèles de la théorie de la décision et plus particulièrement les MDPs et les DEC-MDPs quant à la gestion des contraintes du problème ce qui limite leurs applicabilités ainsi que les difficultés liées à la décision décentralisée.

L'étude de ces modèles nous a permis de mettre en évidence un certain nombre de limitations à la fois dans la modélisation du problème et dans les méthodes de résolution proposées. Ces limitations sont en rapport avec l'applicabilité des MDPs et DEC-MDPs pour la planification dans des applications réelles telles que l'exploration planétaire, la

gestion des désastres, … En effet, les MDPs et les DEC-MDPs ne permettent qu'une modélisation restreinte de données temporelles, chaque action ne durant qu'une unité de temps. A chaque étape de décision, une action est choisie par l'agent (dans le cas du MDP) ou une action pour chaque agent (dans le cas du DEC-MDP), ce qui résulte en une exécution synchrone des actions. Les deux modèles ne prennent pas en compte d'éventuelles contraintes sur l'exécution des actions. De plus, La complexité des algorithmes de résolution des DEC-MDPs existants limitent leur applicabilité et la taille des problèmes modélisés.

Ainsi les MDPs et les DEC-MDPs proposent un formalisme adapté à la résolution de petits problèmes de décision mais lorsqu'il s'agit de résoudre des problèmes de décision décentralisée tels que ceux rencontrés dans les applications réelles, ils restent limités.

A partir de ces limitations nous nous sommes fixées ces principaux objectifs:

- Adapter le modèle MDP afin de pouvoir modéliser le temps et la relation de précédence entre les actions. Le but derrière cette modélisation est d'obtenir un plan de tâche robuste qui est capable de mener à bien la mission en dépit d'incertitude.
- Permettre une modélisation plus riche du temps et des actions partiellement ordonnées dans un modèle de l'état de l'art OC-DEC-MDP.
- Augmenter le degré de coordination en introduisant la communication entre agents et augmenter l'efficacité de la résolution des problèmes envisagés.

**Enrichissement du modèle MDP par la notion du temps et les relations de précédences**

Avant de s'attaquer aux problèmes décentralisés multi-agents nous nous sommes intéressées, dans un premier temps, à l'adaptation du modèle MDP mono-agent pour prendre en considération des contraintes temporelles sur l'exécution des actions, des contraintes de précédence et des durées des actions incertaines. La gestion des contraintes et des incertitudes sur les durées d’exécution des actions a nécessité

l’amélioration de la modélisation du temps et des actions réalisées usuellement dans les modèles Markoviens.

La formalisation que nous avons proposée implique la construction explicite des composants du MDP. Ainsi nous avons proposé une formalisation structurée de l'état qui encapsule l'action, son intervalle d'exécution et l'impact de chaque exécution sur les autres actions. La fonction de transition a aussi été construite et qui désigne les effets qui peuvent en résulter lorsqu'une action est exécutée à partir d'un état donné. Dans ce but, nous avons calculé la probabilité de chaque effet d'exécution des actions. Enfin, nous avons spécifié la fonction de récompense pour chaque état. La fonction de récompense est un élément clé pour déterminer la politique optimale de chaque état. Nous avons attribué à chaque exécution d'une action des récompenses différentes en fonction du coût d'exécution et de l'état d'exécution.

**Extension du modèle de l'état de l'art OC-DEC-MDP pour des plans d'actions partiels et la décision de la communication**

Dans notre deuxième contribution, nous nous sommes intéressées au modèle de l'état de l'art OC-DEC-MDP qui est capable d'adresser les contraintes temporelles, les contraintes de précédences et les durées incertaines des actions. Toutefois, ce modèle est adapté au cas où les agents ont des plans linéaires de leurs actions. Ainsi, nous avons étendu ce modèle afin de modéliser des contraintes d'ordre plus complexes entre les actions. Nous avons présenté l'ensemble de tâches de chaque agent par des plans partiels, sous forme de graphe ET/OU. Une construction explicite de chaque composant du modèle OC-DEC-MDP était réalisée en vue de prendre en compte de toute contrainte posée sur les actions, que ce soit temporelle ou structurelle concernant le graphe de tâches de chaque agent.

En conséquence de cette représentation, deux problèmes de décision ont été considérés: (i) quelle action l'agent doit exécuter et (ii) quand l'exécuter. Cela nous a conduit à adapter l'algorithme basé coût occasionné afin de considérer la structure spécifique du problème et les différentes décisions disponibles présentes à l'agent à chaque étape de temps.

**Proposition d'un riche modèle de communication**

Un autre résultat de la topologie du graphe (plans locaux partiels), est que les agents ont plusieurs chemins pour accomplir leur mission. Par conséquent, l'agent dépendant peut attendre indéfiniment une action prédécesseur qui ne sera jamais exécutée par l'agent contraignant. Pour pallier à ce problème, nous avons proposé d'introduire la communication entre agents. Afin d'assurer une bonne coordination à long terme, nous avons ajouté l'action de communication à l'ensemble des actions des agents et ainsi décider sur le partage d'information au temps de planification.

En revanche, cela conduit à une augmentation de la taille de l'espace d'état du fait de raisonner sur toutes les possibilités de communication à chaque pas de temps.

D'où, notre principal objectif était de limiter l'usage de la communication tout en bénéficiant le maximum de sa présence afin d'assurer une bonne coordination entre les agents. Nous avons tiré profit de la structure spécifique du problème aidé par la relation de précédence pour proposer un riche modèle de communication qui permet aux agents de communiquer dans des points précis appelés points de coordination. Nous avons posés des heuristiques sur le moment de communication (quand communiquer), la sémantique des messages partagés (quoi communiquer) et avec qui communiquer.

Afin de mieux enrichir le modèle de communication proposé, nous avons étudié la fiabilité de la communication. En effet, dans un premier temps, nous avons supposé que les messages sont toujours reçus avec succès. Par la suite, nous avons omis cette hypothèse et nous avons considéré la communication stochastique. Nous avons proposé un mécanisme qui permet de prendre en compte les messages perdus lors de la prise de décision.

Finalement, nous avons procédé à la résolution du DEC-MDP ainsi défini en procédant à la planification des tâches de chaque agent en vue d'une prise de décision décentralisée et autonome en accord avec les contraintes du problème. Notre objectif consistait à la maximisation du gain total cumulé en trouvant un compromis entre coût de la communication, perte en utilité locale et gain en utilité globale.

Les résultats obtenus ont montré l'efficacité de la modélisation proposée ainsi que du modèle de la communication proposé.

**Perspectives de recherche**

En se basant sur les contributions présentées ci-dessus et sur ce que nous avons pu constater en trouvant des solutions aux problèmes posés, ce travail de recherche ouvre la voie vers plusieurs perspectives:

Une première direction de recherche est liée à l'inconvénient majeur du modèle OC-DEC-MDP qui est à savoir le large espace d'états. Nous souhaitons, tout d'abord, de réduire la taille de cet espace en définissant un sous ensemble des états représentatifs atteignables par les agents.

Ce grand espace est le résultat de la discrétisation du temps. [Marecki and Tambe, 2007] ont essayé de gérer ce problème en considérant que le temps est continu et ainsi gérer une fonction de valeur sur le temps pour chaque action à la place d'une valeur séparée pour chaque action et intervalle d'exécution. Il est intéressant d'augmenter le degré de coordination en introduisant la communication et mesurer son apport sur le gain global.

Une deuxième partie de travaux futurs concerne l'enrichissement des contraintes. Nous envisageons d'augmenter la diversité des contraintes pouvant être prises en compte par notre approche et enrichir la représentation des contraintes déjà considérées. Ces contraintes peuvent concerner les types de relations entre les actions des agents et les problèmes de ressources limitées.

Considérer des contraintes sur la communication représente aussi une piste de recherche importante. Il serait intéressant de choisir une application qui a de telles contraintes, et formaliser, par la suite, combien la communication est autorisée (définir un budget de communication, par exemple).

# *Bibliographie*


| | |
|---|---|
| [Abdelmoumène and Belleili, 2011] | Abdelmoumène, H. and Belleili, H. (2011). Une modélisation des contraintes dans les Processus décisionnels de Markov. *In Proceedings of Rencontres sur la Recherche (R²I'2011)*, Tizi Ouzou, Algeria. |
| [Abdelmoumène and Belleili, 2012] | Abdelmoumène, H. and Belleili, H. (2012). Scheduling constrained tasks with Markov Decision Process. *In Proceedings of the second International Symposium on Modeling and Implementation of Complex Systems (MISC'2012)*, Constantine, Algeria. |
| [Abdelmoumène and Belleili, 2014a] | Abdelmoumène, H. and Belleili, H. (2014). Modeling a disaster management with Decentralized Markov Decision Processes. *In Ptoceedings of the first International Conference on Information and Communication Technologies for Disaster Management (ICT-DM'14)*, Algiers, Algeria. |
| [Abdelmoumène and Belleili, 2014b] | Abdelmoumène, H. and Belleili, H. (2014). Policy computation for constrained communicating agents. *In Proceedings of the second World Conference on Complex Systems (WCCS'14)*, Agadir, Morocco. |
| [Abdelmoumène and Belleili, 2016] | Abdelmoumène, H. and Belleili, H. (2016). An extended version of opportunity cost algorithm for communication decision. *Evolving Systems Journal*, 7(1): 41-60. |
| [Abderdeen et al., 2004] | Aberdeen, D., Thiébeaux, S., and Zhang, L. (2004). Decision-Theoretic Military Operations Planning. *In Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pp. 402-412. |
| [Abu Alsheikh et al., 2015] | Abu Alsheikh, M., Hoang, D. T., Niyato, D., Tan, H. P. and Lin, S. (2015). Markov Decision Processes with Applications in Wireless Sensor Networks: A survey. *IEEE Communications Surveys and Tutorials*, 17(3): 1239-1267. |
| [Allen, 2009] | Allen, M. W. (2009). *Agent interactions in decentralized environments*. PhD thesis, University of Massachusetts Amherst. |
| [Amato et al., 2006] | Amato, C., Bernstein, D. S. and Zilberstein, S. (2006). Optimal fixed-size controllers for decentralized POMDPs. *In Proceedings of the Workshop of Multi-agent Sequential Decision Making in Uncertain Domains*, Hakodate, Japan. |
| [Amato et al., 2007] | Amato, C., Bernstein, D. S. and Zilberstein, S. (2007). Solving POMDPs using quadratically constrained linear programs. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* |

*(IJCAI'07)*, pp. 2418-2424, Hyderabad, India.

[Aras et al., 2007] Aras, R., Dutech, A. and Charpillet, F. (2007). Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. *In the Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-07)*, pp. 18-25, Providence, Rhode Island.

[Astrom, 1965] Astrom, K. J. (1965). Optimal control of Markov Decision Processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10: 403-406.

[Baki and Bouzid, 2005] Baki, B. and Bouzid, M. (2005). Scheduling with probability and temporal constraints. *In Proceedings of Advances in Artificial Intelligence AI*IA, 9th congress of the Italian Association for Artificial Intelligence*, pp. 21-32, Milan, Italy.

[Beaty et al., 2008] Beaty, D., Grady, M., May, L. and Gardini, B. (2008). Preliminary planning for an international mars sample return mission, *Report of iMARS Working group*, pp. 1-60.

[Beck and Wilson, 2007] Beck, J. C. and Wilson, N. (2007). Proactive algorithms for Job Shop Scheduling with probabilistic durations. *Journal of Artificial Intelligence Research*, 183-232.

[Becker et al., 2004a] Becker, R., Zilberstein, S., Lesser, V. and Goldman, C. (2004a). Solving transition independent decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22: 423-455.

[Becker et al., 2004b] Becker, R., Lesser, V. and Zilberstein, S. (2004b). Decentralized Markov Decision Processes with event-driven interactions. *In the third International joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'04)*, pp. 302-309, NYC.

[Becker et al., 2005] Becker, R., Lesser, V. and Zilberstein, S. (2005). Analyzing Myopic Approaches for multi-agent communication. *In Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05)*, pp. 550-557, Compiegne, France.

[Becker et al., 2009] Becker, R., Carlin, A., Lesser, V. and Zilberstein, S. (2009). Analyzing myopic approaches for Multi-agent communication. *Computational Intelligence*, 25(1): 31-50.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University, New Jersey.

[Bernstein et al., 2000] Bernstein, D. S., Zilberstein, S. and Immerman, N. (2000). The complexity of decentralized control of Markov Decision Processes. *In Proceedings of the sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp.

| | |
|---|---|
| | 32-37, Stanford, California. |
| [Bernstein et al., 2002] | Bernstein, D. S., Givan, R., Immerman, N. and Zilberstein, S. (2002). The complexity of decentralized control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4): 819-840. |
| [Bernstein et al., 2005] | Bernstein, D. S., Hansen, E. A. and Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. *In Proceedings of the nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1287-1292, Edinburgh, Scotland. |
| [Beynier and Mouaddib, 2005] | Beynier, A. and Mouaddib, A. (2005). A polynomial algorithm for decentralized Markov decision processes with temporal constraints. *In Proceedings of the fourth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 963-969. |
| [Beynier and Mouaddib, 2006] | Beynier, A. and Mouaddib, A. (2006). An iterative algorithm for solving constrained decentralized Markov decision processes. *In the Twenty-First National Conference on Artificial Intelligence (AAAI-06).* |
| [Beynier, 2006] | Beynier, A. (2006). *Une contribution à la résolution des processus décisionnels de Markov décentralisés avec contraintes temporelles*. Thèse de Doctorat, Université de Caen Basse-Normandie. |
| [Beynier and Mouaddib, 2010] | Beynier, A. and Mouaddib, A. (2010). A rich communicative model in opportunistic decentralized decision making. *In Proceedings of ACM International Conference on Web Intelligence and Intelligent Agent Technology*. |
| [Beynier and Mouaddib, 2011] | Beynier, A. and Mouaddib, A. (2011). Solving efficiently decentralized MDPs with temporal and resource constraints, *Journal of Autonomous Agents and Multi-Agent Systems*, 23: 486-539. |
| [Boerkoel Jr. and Durfee, 2013] | Boerkoel Jr., J. C. And Durfee, E. H. (2013). Distributed reasoning for multi-agent simple tempral problems. *Journal of Artificial Intelligence Research*, 47: 95-156. |
| [Bonet and Geffner, 2003a] | Bonet, B. and Geffner, H. (2003a). Faster heuristic search algorithms for planning with uncertainty and full feedback. *In Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 1233-1238, Morgan Kaufmann. |
| [Bonet and Geffner, 2003b] | Bonet, B. and Geffner, H. (2003b). Labeled RTDP: Improving the convergence of real time dynamic programming. *In Proceedings of 13th International Conference on Automated Planning and Scheduling (ICAPS-03)*, pp. 12-21. |

| | |
|---|---|
| [Boutilier, 1996] | Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. *In Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledg (TARK)*, pp. 195-201. |
| [Boutilier et al., 1999] | Boutilier, C., Dean, T. and Hanks, S. (1999). Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1: 1-93. |
| [Bresina and Washington, 2000] | Bresina, J. and Washington, R. (2000). Expected utility distributions for felixble contingent execution. *In the AAAI workshop representation issues for real world planning systems*. |
| [Bresina et al., 2002] | Bresina, J., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. and Washington, R. (2002). Planning under continuous time and resource uncertainty: a challenge for AI. *In Proceedings of International Conference on Uncertainty in Artificial Intelligence*. |
| [Busoniu et al., 2012] | Busoniu, L., Munos, R. and Babuska, R. (2012). A survey of optimistic planning in Markov decision processes. *Jhon Wiley and Sons Inc.* |
| [Carlin and Zilberstein, 2009] | Carlin, A. and Zilberstein, S. (2009). Myopic and non-myopic communication under partial observability. *In IEEE Conference on Intelligent Agent Technology (IAT)*. |
| [Cogill et al., 2004] | Cogill, R., Rotkowitz, M., van Roy, B. and Lall, S. (2004). An approximate dynamic programming approach to decentralized control of stochastic systems. *In Proceedings of the Allerton Conference on communication, control and computing*, pp. 1040-1049, Urbana-Champaign, IL. |
| [Dai et al., 2011] | Dai, P., Weld, M. D. and Goldsmith, J. (2011). Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42: 181-209. |
| [Decker and Lesser, 1993] | Decker, K. and Lesser, V. (1993). Quantitative modeling of complex computational task environment. *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 217-224. |
| [Doshi and Rabinovich, 2011] | Doshi, P. and Rabinovich, Z. (2011). Decision making in Multi-agent settings. *Tutorial in International conference on Autonomous Agents and Multi-Agent Systems (AAMAS'11)*. |
| [Emery-Montemerlo et al., 2004] | Emery-Montemerlo, R., Gordon, G., Schneider, J. and Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *In Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 136-143. |
| [Emery-Montemerlo et al., 2005] | Emery-Montemerlo, R., Gordon, G., Schneider, J. and Thrun, S. (2005). Game theoretic control for robot teams. *In Proceedings of the IEEE* |

*International Conference on Robotics and Automation*, pp. 1175–1181.

[Feng and Zilberstein, 2004] Feng, Z. and Zilberstein, S. (2004). Region-Based Incremental Pruning for POMDPs. *In Proceedings of International Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 146–153.

[Ferns et al., 2012] Ferns, N., Panangaden, P. and Precup, D. (2012). Metrics for Markov decision processes with infinite state spaces. *In Proceedings of International Conference on Uncertainty in Artificial Intelligence (UAI)*.

[Fikes and Nilson, 1971] Fikes, R. and Nilson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, pp. 189-208.

[Garcia, 2008] Garcia, F. (2008). *Processus décisionnels de Markov en intelligence artificielle: principes généraux et applications*, pp. 1-54.

[Gelly et al., 2006] Gelly S.,Wang Y., Munos R. and Teytaud O. (2006). Modification of UCT with patterns in Monte-Carlo Go, *Technical report, INRIA*.

[Golabi et al., 1982] Golabi, K., Kulkarni, R. B. and Way, G. B. (1982). A statewide pavement management system. *Interface*, 12: 5-21.

[Goldman and Zilberstein, 2003] Goldman, C. and Zilberstein, S. (2003). Optimizing information exchange in cooperative multiagent systems. *In International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 137-144.

[Goldman and Zilberstein, 2004] Goldman, C. and Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143-174.

[Guestrin et al., 2001] Guestrin, C., Koller, D., and Parr, R. (2001). Multi-agent planning with factored MDPs. *In Proceedings of Advances in Neural Information Processing Systems*, pp. 1523-1530, Vancouver, British Columbia, Canada.

[Guestrin and Gordon, 2002] Guestrin, C. and Gordon, G. (2002). Distributed planning in hierarchical factored MDPs. *In Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pp. 197-206, Edmonton, Alberta, Canada.

[Guestrin et al., 2003] Guestrin, C., Koller, D., Parr, R. and Venkataraman, S. (2003). Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research*, 19: 399-468.

[Hansen and Zilberstein, 2001] Hansen, E. A. and Zilberstein, S. (2001). LAO*: A heuristic search algorithm that finds solution with loops. *Artificial Intelligence Journal*, 129: 35-62.

| | |
|---|---|
| [Hansen et al., 2004] | Hansen, E. A., Bernstein, D. S. and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games, *In Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI),* pp. 709-715, San Jose, California . |
| [Howard, 1960] | Howard, R. A. (1960). *Dynamic programming and Markov processes.* MIT Press. |
| [Kaelbling et al., 1998] | Kaelbling, L. P., Littman, M. and Cassandra, A. R. (1998). Planning and acting in partially observable sochastic domains. *Artificial Intelligence,* 101: 99-134. |
| [Kitano et al., 1999] | Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjoh, A. and Shimada, S. (1999). Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. *In Proceedings of the International Conference on Systems, Man and Cybernetics*, pp. 739–743. |
| [Kocsis and Szepesvari, 2006] | Kocsis L. and Szepesvari C. (2006). Bandit based Monte-Carlo planning. *In Proceedings of 17th EuropeanConference onMachine Learning (ECML-06),* pp. 282–293, Berlin, Germany. |
| [Kolobov et al., 2009] | Kolobov, A., Mausam and Weld, D. S. (2009). ReTrASE: integrating paradigms for approximate probabilistic planning. *In Proceedings of the Intenational Joint Conference on Artificial Intelligence (IJCAI),* pp. 1746–1753. |
| [Kolobov et al., 2010a] | Kolobov A., Mausam and Weld, D. S. (2010a). Classical planning in MDP heuristics: with a little help from generalization. *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS),* pp. 97-104. |
| [Kolobov et al., 2010b] | Kolobov, A., Mausam and Weld, D. S. (2010b). SixthSense: Fast and reliable recognition of dead ends in MDPs. *In Proceedings of AAAI*. |
| [Kumar and Zilberstein, 2009] | Kumar, A. and Zilberstein, S. (2009). Constraint-based dynamic programming for decentralized POMDPs with structured interactions. *In Proceedings International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-09),* pp. 561-568. |
| [Lambrechts et al., 2008] | Lambrechts O., Demeulemeester E. and Herroelen W. (2008). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling* , 11:121–136. |
| [Lazarova-Molnar and Mizouni, 2010] | Lazarova-Molnar S. and Mizouni R. (2010). Modeling human decision behaviors for accurate prediction of project schedule duration. *Lecture notes in business information processing*. |

| | |
|---|---|
| [Littman, 1996] | Littman, R. (1996). *Algorithms for sequential decision making*. PhD thesis, Brown University. |
| [Marecki and Tambe, 2007] | Marecki J. and Tambe M. (2007). On opportunistic techniques for solving decentralized Markov decision processes with temporal constraints. *In Proceedings of International joint conference on autonomous agents and multi-agent systems (AAMAS).* |
| [Matignon et al., 2012] | Matignon, L, JeanPierre, L. and Mouaddib, A. (2012). Coordinated multi-robot exploration under communication constraints using decentralized Markov decision processes. *In Proceedings of the twenty-sixth AAAI conference on Artificial Intelligence*. |
| [Mausam et al., 2005] | Mausam, Benazera, E., Brafman, R. I., Meuleau, N. and Hansen, E. A. (2005). Planning with continuous resources in stochastic domains. *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI),* pp. 1244-1251. |
| [Mazurowski and Zurada, 2007] | Mazurowski, M. A. and Zurada, J. K. (2007). Solving decentralized multi-agent control problems with genetic algorithms. *IEEE Congress on Evolutionary Computation*. |
| [McMahan et al., 2005] | McMahan, H. B., Likhachev, M. and Gordon, G. (2005). Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. *In Proceedings of the 22nd international conference on Machine learning (ICML-05),* pp. 569–576. |
| [Melo and Veloso, 2011] | Melo, F. S. and Veloso, M. (2011). Decentralized MDPs with sparse interactions. *AI Journal*, 175(11): 1757-1789. |
| [Melo et al., 2012] | Melo, F. S., Spaan, M. T. J. and Witwicki, S. J. (2012). Exploiting sparse interactions for optimizing communication in DEC-MDPs. *In Proceedings Seventh annual workshop on multi-agent sequential decision making (MSDM) held in conjunction with AAMAS*. |
| [Messias et al., 2011] | Messias, J. V., Spaan, M. T. J. and Lima, P. U. (2011). Exploiting sparse dependencies for communication reduction in multi-agent planning under uncertainty. *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI).* |
| [Meuleau et al., 2009] | Meuleau, N., Benazera, E., Brafman, R. I., Hansen, E. A., and Mausam (2009). A Heuristic Search Approach to Planning with Continuous Resources in Stochastic Domains. *Journal of Artificial Intelligence Research (JAIR)*, 34: 27–59. |
| [Mostafa, 2011] | Mostafa, H. (2011). *Exploiting structure in coordinating multiple decision makers.* PhD thesis, University of Massachuestts, Amherst. |
| [Mostafa and Lesser, | Mostafa, H. and Lesser, V. (2009). Offline planning for communication by exploiting structured interactions in decentralized MDP. *In IEEE/WIC/ACM* |

| | |
|---|---|
| 2009] | *International Conference on web intelligence and agent technology (WI-IAT)*, pp. 193-200. |
| [Mostafa and Lesser, 2011] | Mostafa, H. and Lesser, V. (2011). A compact mathematical formulation for problems with structured agent interactions. *In Proceedings of the AAMAS Workshop on Multi-agent Sequential Decision Making in Uncertain Domains.* |
| [Mouaddib and Zilberstein, 1998] | Mouaddib, A. and Zilberstein, S. (1998). Optimal scheduling for dynamic progressive processing. *In Proceedings of European Conference on Artificial Intelligence (ECAI).* |
| [Mundhenk et al., 2000] | Mundhenk, M., Goldsmith, J., Lusena, C. and Allender, E. (2000). Complexity of finite-horizon markov decision process problems. *Journal ACM*, 47(4): 681-720. |
| [Nair et al., 2002] | Nair, R., Tambe, M. and Marsella, S. (2002). Team formation for reformation. *In Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems.* |
| [Nair et al., 2003a] | Nair, R., Tambe, M. and Marsella, S. (2003a). Role allocation and reallocation in multi-agent teams: towards a practical analysis. *In Proceedings of the International joint Conference on Autonomous Agents and Multi-agent Systems*, pp. 552-559. |
| [Nair et al., 2003b] | Nair, R., Tambe, M. and Marsella, S. (2003b). Team formation for reformation in multiagent domains like RoboCupRescue. *In Proceedings of RoboCup-2002 International Symposium.* |
| [Nair et al., 2003c] | Nair, R., Tambe, M., Yokoo, M., Pynadath, D. V. and Marsella, S. (2003c). Taming decentralized POMDPs: Towards efficient policy computation for multi-agent settings. *In Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 705–711. |
| [Nair et al., 2004] | Nair, R., Roth, M. Yokoo, M. (2004). Communication for improving policy computation in distributed POMDPs. *In Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 1098–1105. |
| [Nair et al., 2005] | Nair, R., Varakantham, P., Tambe, M. and Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. *In Proceedings of the National Conference on Artificial Intelligence*, pp. 133–139. |
| [Nash, 1950] | Nash, J. (1950). Equilibrium points in n-person games. *In Proceedings of the National Academy of the USA*, 36(1): 48-49. |
| [Nilsson, 1980] | Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga Publishing, Palo Alto, CA. |
| [Oliehoek and Visser, | Oliehoek, F. and Visser, A. (2006). A hierarchical model for decentralized |

| | |
|---|---|
| 2006] | fighting of large scale urban fires. *In AAMAS'06 Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS)*, pp. 14–21. |
| [Oliehoek et al., 2007] | Oliehoek, F., Kooij, J. F., and Vlassis, N. (2007). A cross-entropy approach to solving Dec- POMDPs. *In International Symposium on Intelligent and Distributed Computing*, pp. 145–154. |
| [Oliehoek et al., 2008] | Oliehoek, F., Kooij, J. F., and Vlassis, N. (2008). The cross-entropy method for policy search in decentralized POMDPs. *Informatica*, 32:341–357. |
| [Oliehoek et al., 2009] | Oliehoek, F., Whiteson, S. and Spaan, M. T. J. (2009). Lossless clustering of histories in decentralized POMDPs. *In Proceedings of The International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 577–584. |
| [Oliehoek and Spaan, 2012] | Oliehoek, F. and Spaan, M. T. J. (2012). Tree-based solution methods for multiagent pomdps with delayed communication. *In Proceedings of AAAI*. |
| [Oliehoek et al., 2013] | Oliehoek, F., Spaan, M. T. J., Amato, C. and Whiteson, S. (2013). Incremental Clustering and Expansion for Faster Optimal Planning in Decentralized POMDPs. *Journal of Artificial Intelligence Research*, 46: 449–509. |
| [Papadimitriou and Tsitsiklis, 1987] | Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3): 441-450. |
| [Paquet el al., 2005] | Paquet, S, Tobin, L., and Chaib-draa, B. (2005). An online POMDP algorithm for complex multi-agent environments, *In Proceedings of the International Joint Conference on Autonomous Agents and Multi-agent Systems.* |
| [Patrascu et al., 2002] | Patrascu, R., Poupart, P., Schuurmans, D., Boutilier, C., and Guestrin, C. (2002). Greedy Linear Value-Approximation for Factored Markov Decision Processes. *In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-02)*, pp. 285–291. |
| [Peshkin et al., 2000] | Peshkin, L., Kee-Eung, K., Meuleau, N. and Kaelbling, L. P. (2000). Learning to cooperate via policy search. *In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 489-496, Stanford, California. |
| [Pineau et al., 2006] | Pineau, J., Gordon, G. and Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27: 335-380. |
| [Poupart et al., 2002] | Poupart, P., Boutilier, C., Patrascu, R., and Schuurmans, D. (2002). Piecewise Linear Value Function Approximation for Factored MDPs. *In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pp. 292–299. |

| | |
|---|---|
| [Poupart and Boutilier, 2003] | Poupart, P. and Boutilier, C. (2003). Bounded finite state controllers. *In Advances in Neural Information Processing Systems 16 (NIPS)*. Vancouver, Canada. |
| [Prakash et al., 2014] | Prakash S., Singh A. and Sammal P. S. (2014). Implementaion of distributed multi-agent system using JADE platform. *Int J Comput. Appl*., 105:12–19. |
| [Puterman, 1994] | Puterman M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley series in probability and mathematical statistics. |
| [Pynadath and Tambe, 2002] | Pynadath, D. V. and Tambe, M. (2002). The communicative multi-agent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research (JAIR), 16*: 389–423. |
| [Rabinovich et al., 2003] | Rabinovich, Z., Goldman, C. V., and Rosenschein, J. S. (2003). The complexity of multi-agent systems: The price of silence. *In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1102–1103, Melbourne, Australia. |
| [Roth et al., 2005] | Roth, M., Simmons, R. and Veloso, M. (2005). Reasoning about joint beliefs for execution-time communication decisions. *In Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*. |
| [Roth et al., 2006a] | Roth, M., Simmons, R. and Veloso, M., (2006a). Decentralized communication strategies for coordinated multi-agent policies. *In Multi Robots systems: From Swarms to Intelligence Automata*. |
| [Roth et al., 2006b] | Roth, M., Simmons, R. and Veloso, M., (2006b). What to communicate? Execution-time decision in multi-agent POMDPs. *In Eighth International Symposium on Distributed Autonomous Robotic Systems*. |
| [Roth et al., 2007] | Roth, M., Simmons, R., and Veloso, M. (2007). Exploiting factored representations for decentralized execution in multi-agent teams. *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*. |
| [Roy et al., 2000] | Roy, N., Pineau, J. and Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000), Hong Kong.* |
| [Russell and Norvig, 2003] | Russell, S. and Norvig, P. (2003). *Artificial Intelligence : A Modern Approach*. Prentice Hall Series. |
| [Sanner et al., 2009] | Sanner, S., Goetschalckx, R., Driessens, K., and Shani, G. (2009). Bayesian Real-Time Dynamic Programming. *In Proceeding of IJCAI*, pp. 1784–1789. |
| [Seuken and Zilberstein, | Seuken, S. and Zilberstein, S. (2007). Memory-bounded dynamic |

| | |
|---|---|
| 2007] | programming for DEC-POMDPs. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2009-2015, Hyderabad, India. |
| [Seuken and Zilberstein, 2008] | Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems journal.* |
| [Shen et al., 2006] | Shen, J., Becker, R. and Lesser, V. (2006). Agent interaction in distributed POMDPs and its implications on complexity. *In Proceedings on the International Joint Conference on Autonomous Agents and Multi-agent Systems*, Hakodate, Hokkaido, Japan. |
| [Singh et al., 2011] | Singh A, Juneja D, Sharma AK (2011) Agent development toolkits. *International Journal of Advanced Technologies*, 2:158–164. |
| [Smallwood and Sondik, 1973] | Smallwood, R. D. and Sondik E. J. (1973). The optimal control of partially observable markov decicion processes over a finite horizon. *Operation Research*, 21:1071-1088. |
| [Spaan et al., 2006] | Spaan, M. T. J., Gordon G. and Vlassis N. (2006). Decentralized planning under uncertainty for teams of communicating agents. *In Proceedings of The fifth international joint conference on autonomous agents and multi-agent systems (AAMAS)*, pp 249–256. |
| [Spaan et al., 2008] | Spaan, M. T. J., Oliehoek, F. and Vlasses, N. (2008). Multi-agent planning under uncertainty with stochastic communication delays. In Proceedings of the International conference on automated planning and scheduling (ICAPS). |
| [Spaan and Melo, 2008] | Spaan, M. T. J. and Melo, F. S. (2008). Interaction-driven Markov games for decentralized multiagent planning under uncertainty. *In Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems.* |
| [Sutton and Barto, 1998] | Sutton R. and Barto A. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA. |
| [Suzuki and Yamashita, 1999] | Suzuki, I. and Yamashita, M. (1999). Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, *28*(4): 1347–1363. |
| [Szer et al., 2005] | Szer, D., Charpillet, F. and Zilbersetin, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. *In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'2005).* |
| [Szer and Charpillet, 2006] | Szer, D. and Charpillet, F. (2006). Programmation dynamique à base de points pour la résolution des dec-pomdps. *14èmes Journées Francophones sur les Systèmes Multi-Agents - JFSMA'2006.* |

| | |
|---|---|
| [Tambe, 1997] | Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR), 7*: 83–124. |
| [Thomas, 2005] | Thomas V. (2005). *Proposition d'un formalisme pour la construction automatique d'interactions dans les systèmes multi-agents réactifs*, Thèse de Doctorat, l'université Henri Poincaré – Nancy 1, France. |
| [Tierney and Goltz, 1997] | Tierney, K. J. and Goltz, J. D. (1997). Emergency response: Lessons learned from the kobe earthquake. *Technical report, Disaster Research Center*, URL http://dspace.udel.edu:8080/dspace/handle/19716/202. |
| [Valtazanos and Steedman, 2014] | Valtazanos A. and Steedman M. (2014). Improving uncoordinated collaboration in partially observable domains with imperfect simultaneous action communication. *In Proceedings The second workshop on distributed and multi-agent planning (DMAP) held in conjunction with ICAPS.* |
| [Varakanthan et al., 2005] | Varakantham, P., Nair, R., Tambe, M., and Yokoo, M. (2005). Winning back the CUP for distributed POMDPs: Planning over continuous belief spaces. *In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS).* |
| [Varakanthan et al., 2009] | Varakantham, P., Kwak, J. young, Taylor, M. E., Marecki, J., Scerri, P., and Tambe, M. (2009). Exploiting coordination locales in distributed POMDPs via social model shaping. *In Proceedings of the International Conference on Automated Planning and Scheduling.* |
| [Velagapudi et al. 2011] | Velagapudi, P., Varakantham, P., Scerri, P., and Sycara, K. (2011). Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. *In Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems.* |
| [Washington et al., 1999] | Washington, R., Golden, K., Bresina, J., Smith, D. E., Anderson, C., and Smith, T. (1999). Autonomous rovers for Mars exploration. In *Proceedings of the IEEE Aerospace Conference*, pp. 237–251, Snowmass, CO. |
| [Wieser, 1889] | Wieser F (1889). *Valeur naturelle (Dernatürliche Wert).* |
| [Witwicki et al., 2007] | Witwicki, S. J. and Durfee, E. (2007). Commitment-driven distributed joint policy search. *In Proceedings of the 6th international joint conference on Autonomous*<br><br>*Agents and Multi-agent Systems*, pp. 480–487. |
| [Witwcki et al., 2009] | Witwicki, S. J. and Durfee, E. (2009). Flexible approximation of structured interactions in decentralized Markov decision processes. *In Proceedings of the 8th international joint conference on Autonomous Agents and Multi-agent Systems*, pp. 1251–1252. |
| [Witwicki and Durfee, 2010] | Witwicki, S. J. and Durfee, E. (2010). Influence-based policy abstraction for weakly-coupled Dec-POMDPs. *In Proceedings of the International* |

*Conference on Automated Planning and Scheduling.*

[Witwicki, 2011] Witwicki, S. J. (2011). *Abstracting Influences for Efficient Multiagent Coordination Under Uncertainty*.PhD thesis, University of Michigan, Ann Arbor, Michigan, USA.

[Wu and Durfee, 2010] Wu, J. and Durfee, E. (2010). Resource-driven mission-phasing techniques for constrained agents in stochastic environments, *Journal of Artificial Intelligence Research (JAIR),* 38: 415-473.

[Wu et al., 2011] Wu, F., Zilberstein, S. and Chen X. (2011). Online planning for multi-agent systems with bounded communication. *Artificial Intelligence Journal,* 175: 487-511.

[Xuan et al., 2001] Xuan, P., Lesser, V. and Zilberstein, S. (2001). Communication decision in multiagent cooperation: Model and Experiments. *In Proceedings of the Fifth International conference on Autonomous Agents.*

[Yoon et al., 2007] Yoon, S., Fern, A., and Givan, R. (2007). FF-Replan: A Baseline for Probabilistic Planning. *In Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS-07),* pp. 352–359.

[Zhu et al., 2014] Zhu, C., Luo, J., Zhang, W. and Liu, Z. (2014). OL-DEC-MDP model for multi-agent online scheduling with a time-dependent probability of success. *Mathematical Problems in Engineering, Hindawi Publishing Corporation.*