Interaction décisionnelle homme-robot: la planification de tâches au service de la sociabilité du robot

Vincent Montreuil

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Paul Sabatier – Toulouse III
Discipline ou spécialité : Systèmes Informatiques

Présentée et soutenue par Vincent MONTREUIL
Le 07 novembre 2008

Titre :
Interaction Décisionnelle homme-robot :
la planification de tâches au service de la sociabilité du robot

---

JURY

M. Philippe PALANQUE	Président
M. René MANDIAU	Rapporteur
M. Abdel-Illah MOUADDIB	Rapporteur
M. David ANDREU	Examinateur
Mme. Sylvie THIEBAUX	Examinateur
M. Félix INGRAND	Examinateur
M. Rachid ALAMI	Directeur de thèse

Ecole doctorale : EDSYS
Unité de recherche : LAAS-CNRS
Directeur de Thèse : Rachid ALAMI

à Pierre, Alexandre et Jean-Claude
qui nous ont quitté trop tôt

Remerciements

Ce manuscrit est l'aboutissement de trois années de travail au LAAS-CNRS à Toulouse. Je souhaite, ici, remercier l'ensemble des personnes qui ont contribué à mon développement cognitif tant sur le plan scientifique que sur le plan humain tout au long de ces trois années.

Tout d'abord, je souhaiterais remercier Malik Ghallab – directeur du laboratoire – et Raja Chatila – chef du groupe de robotique puis directeur du laboratoire – de m'avoir accueilli au sein de l'équipe de robotique du LAAS.

Merci également à mon directeur de thèse Rachid Alami – actuel directeur du groupe de robotique – pour m'avoir accordé du temps et m'avoir prodigué ses conseils tout au long de ces trois années.

Merci à mes deux rapporteurs, Abdel Illah Mouaddib et René Mandiau ainsi qu'à l'ensemble des membres de mon jury de thèse d'avoir consacré une partie de leur temps (que je sais être précieux) à la relecture de ce manuscrit.

Merci à tous ceux qui ont contribué à la réalisation de la partie expérimentale de cette thèse. Ils sont nombreux puisqu'un robot ne saurait fonctionner efficacement sans un ensemble conséquent de lignes de code. Merci tout particulièrement à Aurélie et Maxime pour toutes ces heures passées à interfacer et déboguer nos programmes respectifs. Un grand merci également à tous les maîtres du C++ qui ont pris le temps de m'expliquer les subtilités de ce langage. Un remerciement spécial pour Samir pour avoir apporté des idées neuves qui ont permis une évolution positive de mes travaux.

Il me faut également remercier les personnes qui ont enrichi ces années de travail sur le plan humain. Merci au club des thésards/post-docs de la pose de 16h : Gautier, Sylvain J., Matthieu, Sylvain A., Thierry P. et Martial. J'ai pris beaucoup de plaisir à vous côtoyer et à refaire le monde avec vous. Un grand merci à Jérémi et Pierre pour leurs blagounettes matinales qui donnent de la bonne humeur dès le début de la journée. Il me faut également saluer mes collègues de bureau qui ont réussi à me supporter au quotidien : Léo, Akin, Luis, Waël et Ali. Merci également à ceux avec qui la pause déjeuner permet de faire un « vrai » break : Brice, Matthias, Thierry G. et Rolo. J'espère que ceux dont le nom n'apparaît pas ne m'en tiendront pas rigueur : rassurez-vous

vous conserverez toujours une place dans mes souvenirs.

Enfin, je finirai ces remerciements par un clin d'oeil à toutes les personnes qui m'ont permis de sortir le « nez du guidon » lorsque j'en avais besoin. Merci pour toutes ces soirées de détente qui m'ont permis d'évacuer le stress et d'être plus efficace au laboratoire.

Table des matières

Liste des figures

Introduction

La conception de robots autonomes est un problème complexe. Cette complexité est d'autant plus marquée dans le cadre des robots assistants car ceux-ci seront amenés à agir et à collaborer étroitement avec des partenaires humains pour les seconder dans leur vie quotidienne et/ou professionnelle. Cette proximité directe entre l'homme et le robot impose des contraintes sur le comportement du robot. En effet, un tel robot devra agir de manière à assurer la sécurité physique de ses partenaires humains mais il devra également se comporter d'une manière socialement acceptable. Ainsi, à la réalisation des buts courants du robot vient se superposer une contrainte supplémentaire : produire et exécuter des comportements socialement acceptables.

Le caractère *social* du comportement du robot doit être pris en compte à tous les niveaux de l'architecture logicielle du robot : du niveau fonctionnel jusqu'aux systèmes décisionnels. Depuis maintenant de nombreuses années, les capacités décisionnelles des robots autonomes sont souvent concrétisées par un couple superviseur-planificateur. Le superviseur est en charge de contrôler l'exécution des tâches courantes du robot. Le planificateur, quant à lui, est en charge de fournir des plans permettant au robot de satisfaire ses buts courants. Le planificateur permet donc de conférer une autonomie décisionnelle importante au robot. Cette thèse étudie la façon dont le planificateur peut contribuer à la sociabilité du robot.

La planification de tâches est un domaine de recherche qui a été largement approfondi ces dernières années et qui possède donc un état de l'art important. Toutefois, à la date d'aujourd'hui, la majorité des applications robotiques utilisant de tels planificateurs mettent en oeuvre des robots seuls dans l'environnement ou des robots évoluant aux côtés de personnes expertes. La problématique des robots assistants impose deux contraintes importantes : (1) la nécessité pour le robot d'agir de manière socialement acceptable et (2) la proximité entre le robot et des personnes non-expertes.

La sociabilité du robot impose donc que celui-ci agisse de manière *cohérente* et *lisible* i.e. de façon compréhensible pour ses partenaires humains. Pour cela le robot devra, dans la mesure du possible, se comporter comme un être humain. La collaboration entre êtres humains peut avoir deux déclinaisons possibles : une collaboration de type « dirigeant-subordonné » ou une collaboration entre « partenaires ». Un robot assistant devra donc pouvoir tenir le rôle de « robot serviteur » ou « d'équipier ». Quelque soit le rôle retenu il devra être capable de produire

des plans permettant de réaliser ses buts courants de manière efficace tout en assurant un comportement qui puisse être aisément interprété par ses collaborateurs humains. En effet, un robot équipier ou un robot serviteur agissant de manière non prédictible, i.e. effectuant des actions désordonnées, pourrait susciter un sentiment d'incompréhension chez ses collaborateurs humains et ainsi engendrer une remise en cause de la réalisation de la tâche. Afin d'assurer une bonne sociabilité du robot il convient donc d'intégrer un ensemble de *règles sociales* dans le système de planification du robot pour garantir que les plans produits conduisent à un comportement socialement acceptable.

Ce mémoire est articulé en cinq chapitres. Le premier chapitre contient une tentative de classification des applications robotiques interactives. Cette classification permet de mettre en avant la nécessité de doter un robot assistant d'une grande autonomie décisionnelle lorsque l'interaction qu'il aura avec ses partenaires humains sera de longue durée. L'autonomie décisionnelle des robots est souvent concrétisée par un planificateur de tâches. Le premier chapitre contient également un état de l'art de l'utilisation de la planification de tâches dans des applications robotiques interactives et/ou des applications de collaboration entre agents hétérogènes. Cet état de l'art permet de mettre en évidence les différentes formes que peut prendre la planification de tâches et d'illustrer celles-ci par des applications concrètes proches de celles ciblées par ce manuscrit. Cette analyse permettra de préciser le type de planification retenue pour une utilisation sur un robot assistant.

Le second chapitre dresse un état de l'art des différentes approches qui ont été développées pour conférer un comportement *social* à un robot. L'hypothèse la plus communément admise est que l'interaction homme-robot gagne en qualité lorsqu'elle se rapproche d'une interaction homme-homme. Les différentes approches qui seront présentées dans ce chapitre sont donc basées sur cette hypothèse et permettent d'apporter des solutions concrètes à certains aspects de la sociabilité du robot. Toutefois, ces solutions ne ciblent qu'un aspect *local* du comportement du robot et ne permettent pas de garantir que le comportement *global* de celui-ci soit socialement acceptable. Ce chapitre nous permettra de mettre en avant la nécessité d'introduire des notions de *règles sociales* dans le planificateur de tâches utilisé par un robot assistant.

Le troisième chapitre aborde le problème de l'évaluation de la qualité sociale d'un plan. Pour pouvoir permettre au robot de favoriser des plans socialement acceptables, il faut le doter d'un planificateur utilisant une métrique évaluant le ressenti des partenaires humains du robot par rapport aux plans produits. La conception de notre métrique s'est effectuée en deux étapes : (1) l'identification des critères intervenant dans la métrique et (2) l'établissement d'un processus d'agrégation de ces critères. Ce chapitre détaille les différents critères utilisés et la façon de les combiner pour produire une évaluation numérique reflétant la qualité sociale d'un plan.

Dans le quatrième chapitre nous détaillons l'implémentation d'un planificateur nommé HATP (*Human Aware Task Planner* en anglais) prenant explicitement en compte la qualité sociale

des plans. Ce planificateur a été spécifiquement conçu pour pouvoir fonctionner de manière embarquée sur un robot interactif. Ce chapitre décrit la façon dont la méthode d'évaluation élaborée au troisième chapitre est insérée dans un algorithme de planification issu de l'état de l'art. Ce chapitre détaille également les choix techniques qui ont été effectués pour permettre une utilisation en ligne du planificateur.

Le cinquième chapitre présente les résultats obtenus avec le planificateur HATP dans le cadre de scénarios d'interaction homme-robot. Dans un premier temps, les résultats présentés seront issus de scénarios de simulation inspirés de situations réelles. Ces scénarios de simulation permettront de mettre en avant les aptitudes du planificateur HATP et d'illustrer ses performances. Dans un second temps, nous présenterons les résultats obtenus en utilisant le planificateur HATP sur un robot réel dans le cadre d'un scénario concret.

Liste des publications liées à cette thèse

[Alami 05] R. Alami, A. Clodic, V. Montreuil, E.A. Sisibot, R. Chatila. *Task planning for human-robot interaction.* Dans Proceedings of the Smart Objects and Ambient Intelligence Conference (sOc-EUSAI), pages 81-85, Octobre 2005.

[Clodic 05] A. Clodic, V. Montreuil, R. Alami, R.Chatila. *A decisional framework for autonomous robots interacting with humans.* Dans Proceedings of the 14th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), pages 543-548, Août 2005.

[Alami 06] R. Alami, A. Clodic, V. Montreuil, E.A. Sisibot, R. Chatila. *Toward human-aware robot task planning.* Dans Papers from 2006 AAAI Spring Symposium "To bodly go where no human-robot team has gone before", pages 39-46, Mars 2006.

[Alami 06a] R. Alami, R. Chatila, A. Clodic, S. Fleury, M. Herbb, V. Montreuil, E.A. Sisbot. *Towards human-aware cognitive robots.* Dans Proceedings of the 5th International Cognitive Robotics Workshop (CogRob), Juillet 2006.

[Clodic 06] A. Clodic, S. Fleury, R. Alami, R. Chatila, G. Bailly, L. Brethes, M. Cottret, P. Danes, X. Dollat, F. Elisei, I. Ferrane, M. Herrb, G. Infantes, C. Lemaire, F. Lesrasle, J. Manhes, P. Marcoul, P. Menezes, V. Montreuil. *Rackham : an interactive robot-guide.* Dans Proceedings of the 15th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), pages 502-509, Septembre 2006.

[Clodic 07] A. Clodic, R. Alami, V. Montreuil, S. Li, B. Wrede, A. Swadzba. *A study of interaction between dialog and decision for human-robot collaborative task achievement*. Dans Proceedings of the 16th International Workshop on Robot and Human Interactive Communication (RO-MAN), pages 913-918, Août 2007.

[Clodic 07a] A. Clodic, M. Ransan, R. Alami, V. Montreuil. *A management of mutual belief for human-robot interaction*. Dans Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), pages 1551-1556, Octobre 2007.

[Montreuil 07] V. Montreuil, A. Clodic, M. Ransan, R. Alami. *Planning human centered robot activities*. Dans Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), pages 2618-2623, Octobre 2007.

[Clodic 08] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, R. Chatila. *SHARY : A Supervision System Adapted to Human-Robot Interaction*. Dans Proceedings of the International Symposium on Experimental Robotics (ISER), pages 229-238, Juillet 2008.

1

Robotique interactive et planification

La conception de robots autonomes est un problème complexe en soi mais cette complexité est d'autant plus marquée pour les robots interactifs. En effet, un robot partageant son espace avec des humains se doit d'agir en préservant leur sécurité physique mais il doit également se comporter de manière socialement acceptable. Ces contraintes doivent être prises en compte à tous les niveaux de la conception du robot : de la perception jusqu'aux systèmes décisionnels de haut-niveau.

Depuis maintenant de nombreuses années, les architectures de contrôle trois niveaux ont été largement utilisées pour le contrôle des robots autonomes (voir [Alami 98] pour un exemple). Le schéma de principe d'une architecture trois niveaux est illustré par la figure 1.1. Une telle architecture est composée de :

✓ un niveau fonctionnel constitué de toutes les routines de contrôle des capteurs et des actionneurs du robot,

✓ un niveau intermédiaire dédié au contrôle d'exécution, ce niveau est chargé de vérifier à tout moment la validité des requêtes destinées au niveau fonctionnel,

✓ un niveau délibératif qui contient les systèmes décisionnels de haut-niveau.

Le niveau délibératif est souvent concrétisé par un couple superviseur-planificateur. Le superviseur (aussi appelé « exécutif ») permet de contrôler le bien fondé des actions en cours et de déterminer les buts à réaliser selon l'état courant du monde. Le planificateur est utilisé pour conférer un haut degré d'autonomie décisionnelle au robot, il permet à ce dernier d'établir une

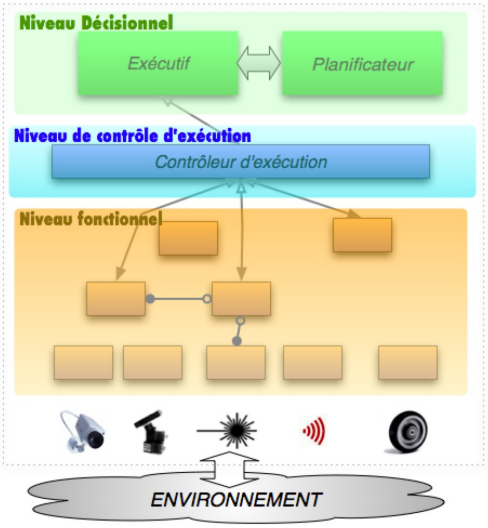

FIG. 1.1 – Structure d'une architecture de contrôle trois niveaux.

succession d'actions à réaliser permettant de satisfaire les buts courants. Le planificateur est un outil puissant mais n'est pas toujours nécessaire, cela dépend de l'application finale visée.

Les applications étudiées dans ce manuscrit sont celles où le robot agit en collaboration avec l'homme de manière à être perçu comme un compagnon par ses partenaires humains. Il convient de s'interroger sur l'utilité de l'outil de planification pour ces applications. Cette utilité sera directement liée au degré d'autonomie décisionnelle qui devra être conféré au robot. Nous nous proposons d'étudier cela en nous intéressant aux différentes formes que peut prendre la collaboration entre l'homme et le robot.

1.1 Les différentes formes de collaboration homme-robot

Les avancées réalisées ces dernières années dans tous les domaines de la robotique (perception, décision, etc.) ont permis d'apporter une nouvelle dimension à certains types d'applications robotiques. Ainsi, les premiers travaux sur la collaboration homme-robot ont porté sur l'aptitude d'un robot à être aisément contrôlé par un opérateur humain. Les progrès qui ont été effectués ces dernières années permettent d'envisager une modification du rôle de l'humain pour en faire un collaborateur et non plus un opérateur. Cela signifie que l'humain agit en partenaire et non plus en « donneur d'ordres » vis-à-vis du robot. Les outils décisionnels mis en oeuvre dans ces deux types d'applications robotiques sont très différents, nous nous proposons maintenant de mettre en évidence ces différences en tentant d'établir une classification des applications collaboratives homme-robot.

La classification proposée est basée sur deux critères : (1) le degré de partage de l'espace d'action entre l'homme et le robot et (2) la durée de la collaboration entre ces deux agents. Grâce à ces deux critères nous avons identifié trois grandes familles comme l'illustre la figure 1.2.

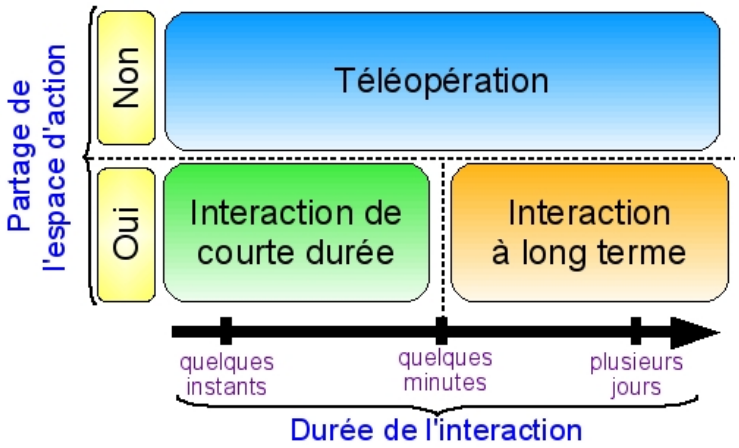

FIG. 1.2 – Proposition de classification des différentes types de collaboration homme-robot.

1.1.1 La téléopération

Nous allons maintenant préciser les caractéristiques propres à la téléopération selon notre classification. Il est important de noter qu'il ne s'agit pas ici de faire un état de l'art complet des applications téléopératives, nous nous focaliserons uniquement sur les aspects interactifs liés à la téléopération.

La téléopération est caractérisée par un faible degré de partage de l'environnement d'action entre l'homme et le robot. En effet, les applications de téléopération proposent bien souvent des situations où le robot est amené à évoluer dans un environnement totalement distinct de celui de son opérateur humain. Dans un tel cadre, la collaboration entre l'homme et le robot est alors réduite à une interface (parfois très sophistiquée) pouvant prendre des formes très variées. Cette interface a deux rôles : (1) transmettre/traduire les requêtes de l'homme en ordres compréhensibles par le robot et (2) centraliser/expliciter pour l'opérateur humain les informations collectées par le robot sur son environnement. L'interface est alors le seul moyen de communication existant entre le robot et son opérateur humain. Dans de telles situations, la qualité de la collaboration peut alors être quantifiée en se basant sur trois caractéristiques :

- l'ergonomie de l'interface entre l'homme et le robot,
- les fonctionnalités interactives permettant la communication entre l'homme et le robot,
- le degré d'autonomie du robot.

L'ergonomie de l'interface homme-robot. Dans le cadre de la téléopération il est clair que l'ergonomie de l'interface est un point clef puisque l'interface est le seul moyen pour l'opérateur humain de percevoir l'environnement dans lequel évolue le robot. De nombreux travaux se sont donc naturellement portés sur la conception d'interfaces sophistiquées facilitant au maximum la prise de décision de l'opérateur humain. On peut, par exemple, citer des travaux tels que [Kawamura 03] qui proposent une interface basée sur un ensemble d'agents logiciels. Chacun de ces agents est responsable d'une partie des informations provenant du robot et doit adapter l'affichage produit en fonction des préférences de l'utilisateur et des données courantes. Enfin, il

existe un agent particulier dont le rôle est d'organiser graphiquement les différents affichages issus des autres agents. Dans de tels cas, l'amélioration de la qualité de la collaboration homme-robot tient à la conception d'une interface adaptant son ergonomie en fonction des données perçues par le robot.

Les fonctionnalités interactives. Cette caractéristique rejoint partiellement celle de l'ergonomie de l'interface. En effet, l'ajout de fonctionnalités interactives dans une application de téléopération a pour but de permettre à l'opérateur humain de contrôler le robot de manière plus intuitive. La combinaison de l'interface graphique et des fonctionnalités interactives telles que la reconnaissance vocale et la reconnaissance gestuelle permettent de créer des interfaces dîtes multimodales (voir [Fong 01a] et [Fong 01b] pour un exemple).

L'autonomie du robot. Dans le cadre de la téléopération, il est clair que le degré d'autonomie du robot joue également un rôle sur la qualité de la collaboration entre le robot et son opérateur humain. La collaboration gagne en qualité lorsque le robot a un degré d'autonomie suffisamment avancé permettant à l'opérateur de se consacrer uniquement aux décisions de haut-niveau. C'est dans cet esprit que des travaux sur l'Autonomie Ajustable ont été menés (voir [Dorais 98] et [Goodrich 01] pour des exemples). Le principe de l'Autonomie Ajustable est de doter le robot d'une autonomie décisionnelle à degré variable afin de permettre à son opérateur humain de choisir à quel niveau de décision il souhaite intervenir. Ainsi l'opérateur est libre de choisir entre un robot à piloter intégralement ou un robot « exécutant intelligent », il existe également des niveaux intermédiaires entre ces deux extrêmes où le robot prend des décisions dans un nombre de cas limités et fait appel à l'opérateur humain dans les autres cas.

La téléopération est un cadre d'application permettant une collaboration homme-robot où le robot joue un rôle d'exécutant plus ou moins intelligent et l'homme un rôle d'opérateur prenant les décisions de plus ou moins haut-niveau. La planification de tâches peut-être utilisée en téléopération pour doter le robot d'un mode de fonctionnement avec un haut degré d'autonomie. Toutefois, cette planification ne concernera que les actions du robot puisqu'il ne partage pas l'espace d'actions avec l'homme. L'impact social des actions du robot sur l'opérateur humain reste donc limité.

1.1.2 L'interaction de courte durée

Nous allons maintenant détailler les caractéristiques propres à l'interaction de courte durée selon notre classification. De même que pour la téléopération, nous n'aborderons ici que l'aspect interactif de ce type d'applications.

Les applications interactives de courte durée sont caractérisées par un partage de l'espace d'actions entre l'homme et le robot et par une interaction entre les deux agents d'une durée limitée dans le temps. Le partage de l'espace est un point très important puisqu'il oblige le système décisionnel du robot à prendre explicitement en compte le partenaire humain à tout

moment étant donné que ce dernier peut influencer leur environnement commun et donc agir par ce biais sur le bien fondé des actions du robot. Parmi les applications robotiques interactives de courte durée existantes, on peut citer :

➢ **les robots guides de musée** : le robot Sage [Nourbakhsh 99] au musée de l'Histoire Naturelle de Carnegie ou le robot Rackham [Clodic 06] à la Cité de l'Espace à Toulouse,

➢ **les robots cherchant leur chemin au milieu d'une foule** : les robots Grace et Georges ([Simmons 03], [Gockley 04]) lors du challenge AAAI,

➢ **les robots réceptionnistes** : le robot Valérie ([Gockley 05], [Gockley 06]) qui a accueilli des visiteurs à l'Université de Carnegie Mellon.

➢ **les robots de transport d'objets** : le robot CERO [Huttenrauch 02] servant d'assistant de transport dans un institut de recherche.

Le critère de brièveté de l'interaction entre le robot et son partenaire humain joue un rôle très important dans la conception de ce type d'applications. En effet, la collaboration entre l'homme et le robot ne durant que quelques instants il est très souvent inutile de doter le robot de capacités décisionnelles très évoluées. Ce type d'applications se prête naturellement à un contrôle du robot par un automate à états fini de haut-niveau. En effet, ce formalisme est tout à fait adapté aux robots dédiés à des interactions de courte durée puisqu'il permet une exécution systématique de « recettes » garantissant une réponse efficace et rapide aux besoins du partenaire humain. Cet automate à états fini est très souvent associé à un système d'apprentissage permettant d'adapter légèrement le comportement du robot en fonction du vécu existant entre le robot et son partenaire humain courant.

Toutefois, cette approche atteint rapidement ses limites lorsque l'interaction entre le robot et son(ses) partenaire(s) humain(s) se prolonge. En effet, il existe des études de cas [Dauthenhahn 05] qui montrent la nécessité pour le robot d'agir de manière prédictible (i.e. éventuellement répétitive) mais des travaux tels que [Lund 03] révèlent une perte d'intérêt rapide des partenaires humains du robot lorsque ce dernier répète trop souvent les mêmes comportements. Le comportement du robot est alors perçu comme « non-intelligent » et provoque un sentiment d'étouffement chez son partenaire humain qui se sent bloqué dans un cycle d'exécution. Il est possible de contourner cette difficulté en modifiant légèrement le comportement du robot à chaque exécution [Jensen 02] ou en débloquant de nouveaux comportements au fur et à mesure que l'interaction se prolonge [Kanda 05]. Cependant, une fois que le partenaire humain aura expérimenté l'ensemble des comportements du robot, il risque de s'en désintéresser. Il apparaît donc que ces méthodes permettent seulement de prolonger la « durée de vie » de l'interaction sans apporter une solution pérenne dans le temps.

Pour les applications où l'interaction homme-robot est de courte durée l'utilisation de « recettes » est parfaitement adaptée puisque cela permet au robot de fournir une réponse rapide et efficace aux requêtes de son partenaire humain. La planification de tâches présente donc un intérêt limité pour ce type d'application puisqu'elle pénalise la réactivité du robot. Toutefois,

on peut constater que le formalisme des automates à états finis ne permet pas de garantir une interaction de *bonne* qualité sur une longue durée.

1.1.3 L'interaction à long terme

Les interactions à long terme nécessitent de doter le robot de systèmes décisionnels avancés afin que celui-ci puisse faire preuve de capacités d'adaptation inhérentes au bon déroulement de son échange avec son partenaire humain. Dans de telles applications le robot a souvent un nombre restreint de partenaires mais a des interactions régulières de longue durée avec ceux-ci ce qui implique qu'il devra être capable d'adapter et de varier son comportement vis-à-vis de son(ses) partenaire(s) courant(s). Nous nous intéresserons particulièrement aux applications dans lesquelles le robot agit directement sur l'état du monde. Un cadre d'applications qui nous paraît pertinent est celui du robot domestique ou du robot assistant dans le milieu professionnel.

De telles applications nécessitent une autonomie décisionnelle importante de la part du robot, en effet, il devra être capable de raisonner sur la répartition des tâches entre lui-même et son(ses) partenaire(s) humain(s) ainsi que sur la façon de réaliser ces tâches. Nous allons montrer que la planification de tâches est un outil précieux pour ce type d'applications car elle permettra au robot d'élaborer des plans pour réaliser ses buts courants.

1.2 Interaction homme-robot à long terme et planification : état de l'art

La section 1.1 nous a permis de montrer l'utilité de l'outil de planification de tâches pour des applications robotiques interactives avec des interactions de longue durée. Toutefois, la planification de tâches peut prendre des formes très diverses selon la nature des décisions que doit prendre le robot. Si l'on souhaite doter le robot d'un « faible » niveau d'intelligence on pourra se limiter à un modèle succinct de ses actions alors que si l'on souhaite qu'il soit pleinement « conscient » des conséquences de ses actes sur le monde il faudra enrichir le modèle utilisé. Trois grandes familles de planification de tâches ont été identifiées dans [Ghallab 04]. Si on classe ces familles par ordre croissant de complexité des modèles de tâches on obtient :

- la planification de projet,
- l'ordonnancement et l'allocation de ressources,
- la synthèse de plans.

Nous nous proposons de suivre cette classification et d'illustrer l'utilisation de chacune de ces familles dans des applications collaboratives et/ou interactives entre agents hétérogènes et plus particulièrement dans des applications mêlant des robots et des partenaires humains. A la date d'aujourd'hui il existe peu d'applications de ce type et peu d'entre elles abordent les problèmes décisionnels qui y sont liés. Il est important de préciser que, là encore, nous nous focaliserons sur les aspects interactifs de ces applications.

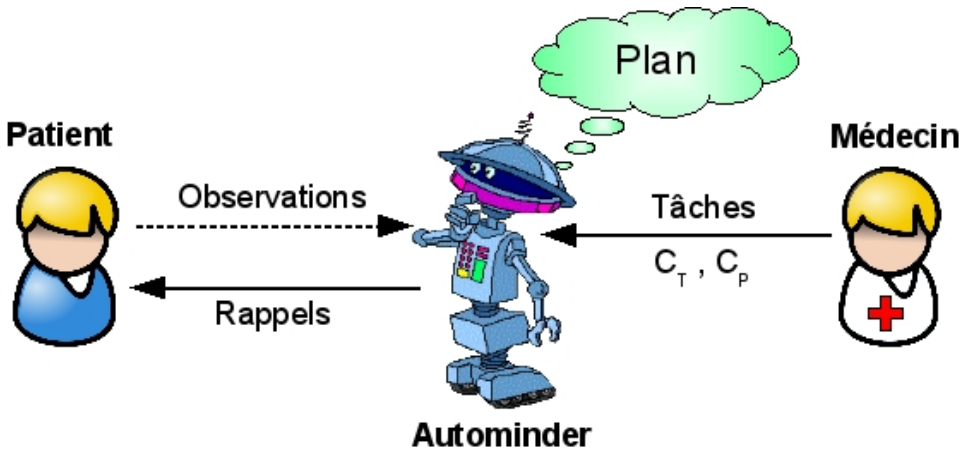

FIG. 1.3 – Principe d'utilisation du robot Autominder.
Le médecin spécifie l'ensemble des tâches à réaliser par le patient dans la journée ainsi que les contraintes C_T et C_P liées à ces tâches. Autominder produit un plan conforme aux spécifications du médecin, observe l'activité quotidienne du patient et lui rappelle les tâches qui n'ont pas été réalisées.

1.2.1 La planification de projet

Dans ce type de planification, le modèle des tâches est réduit à sa plus simple expression. Le modèle d'une tâche peut alors s'écrire comme un triplet $<N,C_T,C_P>$ où N est le nom de la tâche (ce nom est unique), C_T est un ensemble de contraintes temporelles et C_P est un ensemble de contraintes de précédence. Les contraintes temporelles associées à une tâche désignent les intervalles de temps pendant lesquels les instants de début et de fin de l'action peuvent avoir lieu. Les contraintes de précédence associées à une tâche définissent les intervalles de temps la séparant des autres tâches à réaliser. Le rôle du planificateur est alors de trouver un agencement temporel des différentes tâches qui satisfasse l'ensemble des contraintes. Il est important de noter que ce modèle n'inclut pas la notion de conditions d'application d'une tâche ni la notion des effets produits par la réalisation d'une tâche. Ainsi, avec un tel modèle le robot ne raisonne ni sur la faisabilité ni sur la finalité des tâches mais uniquement sur le moment où elles doivent être réalisées. La faisabilité et la finalité des tâches sont traitées dans une phase préliminaire.

Le robot Autominder [Pollack 03] est un robot interactif utilisant cette approche de la planification. Ce robot a été conçu pour prêter assistance aux personnes victimes de problèmes de mémoire en leur rappelant les actions quotidiennes qu'elles pourraient oublier (prendre leurs médicaments, etc.). L'aspect principal de cette application est donc le support cognitif des patients. Le robot n'opère pas réellement d'actions dans l'environnement. La figure 1.3 illustre le principe d'utilisation de ce robot. Dans cette application, le système de planification du robot Autominder reçoit en entrée la liste des activités quotidiennes de son partenaire humain (liste fournie par le médecin du patient) ainsi qu'un ensemble de contraintes portant sur les instants de réalisation de ces activités et sur leur ordre. Le planificateur utilise alors le formalisme DTP (*Disjunctive Temporal Problem* en anglais) détaillé dans [Pollack 02] pour déterminer un ordonnancement valable pour la journée et vérifiant les contraintes spécifiées. Un plan produit par Autominder est un 4-uplet $< S, O, L, B >$, où S est l'ensemble des différentes étapes du

plan (les actions à effectuer par le patient), et O, L et B sont respectivement les contraintes d'ordre, les liens causaux et les contraintes d'instanciation. Le système est capable d'adapter cet ordonnancement en ajoutant ou supprimant des actions lorsqu'il reçoit les requêtes associées de la part du médecin. Le robot est alors en charge de vérifier que le patient exécute les actions de l'ensemble S en respectant les différentes contraintes. En cas d'oubli d'une action par le patient, Autominder intervient pour lui rappeler de réaliser l'action qui n'a pas été opérée.

Il apparaît que la planification de projet permet de créer des ordonnancements de tâches de manière robuste et adaptable dans le temps. Toutefois, ce niveau de raisonnement nécessite la présence d'un superutilisateur en charge de spécifier les contraintes à respecter afin que le plan résultant permette d'aboutir aux objectifs visés. Ainsi, l'autonomie décisionnelle du robot ne porte pas sur la façon dont doivent être satisfaits les buts courants mais sur l'instant de leur réalisation. La collaboration avec un robot doté d'un tel planificateur passe par la spécification de la totalité des actions à opérer ainsi que des contraintes permettant de les organiser. Le robot sera alors perçu comme un simple « exécutant » incapable de prendre des initiatives simples. Ce type de planification n'est donc pas adapté à la conception de robots assistants autonomes « actifs ».

1.2.2 L'ordonnancement et l'allocation de ressources

Dans ce type de planification, le modèle de tâches utilisé dans la planification de projet est enrichi pour y inclure des contraintes sur les ressources utilisées lors de la réalisation de la tâche. Ainsi, le modèle d'une tâche devient un 4-uplet $<N,C_T,C_P,R>$ où N est le nom unique de la tâche, C_T l'ensemble des contraintes temporelles portant sur les instants de début et de fin de la tâche, C_P l'ensemble des contraintes de précédence portant sur l'ordre des tâches à exécuter et R l'ensemble des ressources utilisées par la tâche. Un planificateur basé sur cette approche de la planification doit alors déterminer un agencement temporel des tâches respectant les contraintes temporelles, les contraintes de précédence et les contraintes de ressources.

Ce type de planification a été mis en oeuvre dans un système d'exploitation appelé HRI/OS [Fong 06]. Ce système d'exploitation est dédié à l'interaction homme-robot, il a été conçu pour faciliter la communication entre humains et robots dans le cadre de la réalisation collaborative de tâches. Il fournit un cadre générique permettant aux différents agents en présence de communiquer entre eux afin qu'ils puissent s'informer de leurs aptitudes propres, de leurs tâches courantes et de l'état d'avancement de leurs actions.

Comme l'illustre la figure 1.4, ce système d'exploitation est un système centralisé basé sur un ensemble d'agents logiciels. Les agents (humains et robots) sont tous représentés par des proxys qui ont en charge de gérer les requêtes ne nécessitant pas l'intervention de l'agent qu'ils représentent. Pour chaque tâche à accomplir une requête est émise au gestionnaire de ressources qui dresse alors la liste triée par ordre de priorité des agents aptes à réaliser cette tâche selon la situation courante (positions des agents, temps nécessaire, ressources disponibles, etc...). Une

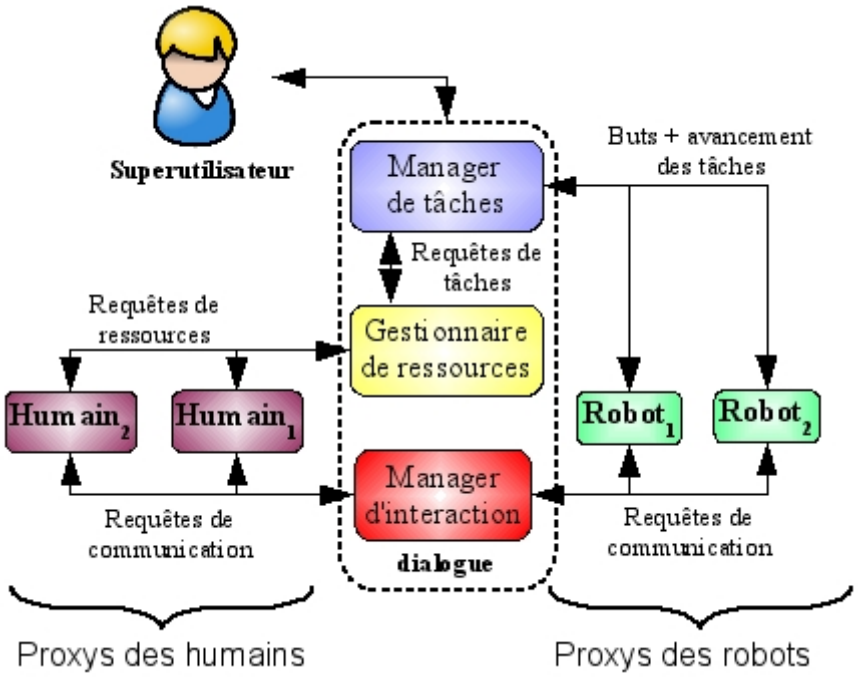

FIG. 1.4 – Structure du système d'exploitation HRI/OS.
Chaque agent exécutant est représenté par un proxy qui dialogue avec un système centralisé dont le rôle est de répartir les tâches à réaliser, contrôler le bon déroulement des opérations et faciliter la communication entre agents.

fois cette liste établie elle est transmise au manager de tâches qui va interroger les agents dans l'ordre de la liste. Le manager d'interaction permet d'établir le dialogue entre les agents lorsqu'un d'entre eux en émet la requête. La planification dans HRI/OS se situe donc dans le gestionnaire de ressources qui va déterminer l'ordre dans lequel les tâches à accomplir doivent être réalisées et qui va gérer les aptitudes et les ressources de chaque agent pour déterminer la répartition des rôles. Dans ce type de planification les tâches à accomplir ne sont pas déterminées par le système mais sont insérées par un superutilisateur chargé de surveiller le déroulement global des évènements.

Il apparaît que la planification pour l'ordonnancement et l'allocation de ressources permet de gérer une répartition des rôles entre agents afin d'optimiser la réalisation des buts courants. Toutefois, on peut là encore constater la présence d'un superutilisateur en charge de centraliser les requêtes des agents et de faciliter la communication entre eux. Un robot doté d'un tel système de planification nécessitera que l'utilisateur spécifie l'intégralité des tâches à effectuer ainsi que les contraintes sur leur ordre d'exécution. Le robot restera donc dans son rôle de simple « exécutant » ce qui n'est pas compatible avec le type d'applications visées par ce manuscrit.

1.2.3 La synthèse de plans

Dans ce type de planification, le modèle des tâches est encore enrichi pour y intégrer les notions de préconditions d'applicabilité et d'effets des tâches. Ainsi le modèle d'une tâche devient

un triplet $<N,P,E>$ où N est le nom unique de la tâche, P est la liste de préconditions nécessaires à l'application de la tâche et E est l'ensemble des effets produits sur l'état du monde par la réalisation de la tâche. Les contraintes de précédence utilisées dans les modèles de tâches précédents ne sont plus utiles puisque les conditions d'application permettront de déterminer les ordonnancements temporels possibles. Ainsi, un planificateur prévu pour synthétiser des plans reçoit en entrée les descriptions associées à l'ensemble des tâches possibles (cet ensemble est appelé « domaine »), une description donnant l'état courant du monde et une liste de buts à atteindre (cette liste est appelée « problème »). Le planificateur détermine alors les tâches qui doivent être réalisées ainsi que leur agencement temporel.

Ce type de planification est celui qui confère la plus grande autonomie décisionnelle à un robot, il a été très largement utilisé dans le cadre des systèmes collaboratifs et/ou interactifs. La planification de type « synthèse de plans » a été déclinée en deux variantes : la planification réactive et la planification délibérative. La planification réactive repose sur l'établissement d'un plan général qui va être instancié et adapté au fur et à mesure de l'exécution alors que la planification délibérative vise à établir une succession d'actions à réaliser puis de contrôler le bon déroulement de ces actions lors de l'exécution. Dans ces deux déclinaisons, la problématique abordée est l'élaboration de plans dans un contexte multi-agents. A partir de ce postulat, un cadre formel a été défini : la théorie de l'intention jointe.

1.2.3.1 La théorie de l'intention jointe

La théorie de l'intention jointe ([Cohen 90], [Levesque 90], [Cohen 91]) a été développée afin de définir un cadre formel permettant d'établir des buts communs entre plusieurs agents. Dans cette théorie chaque agent possède des croyances qui peuvent être classées en deux catégories : les croyances individuelles et les croyances mutuelles. Une croyance individuelle est une croyance propre à l'agent lui-même alors qu'une croyance mutuelle est une croyance partagée par un ensemble d'agents. La théorie de l'intention jointe définit la notion de *but commun persistant* JPG(Θ,$\mathbf{p}$,$\mathbf{q}$) où Θ est l'équipe associée à la réalisation du but commun $\mathbf{p}$ sous les conditions de pertinence $\mathbf{q}$. Un *but commun persistant* JPG(Θ,$\mathbf{p}$,$\mathbf{q}$) sera valide tant que les conditions suivantes sont satisfaites :

1. Chaque membre de Θ a la croyance mutuelle que $\mathbf{p}$ reste à réaliser,
2. Chaque membre de Θ a $\mathbf{p}$ comme but commun (chaque agent de Θ veut réaliser le but $\mathbf{p}$),
3. Chaque membre μ de Θ a la croyance mutuelle que chacun des membres de Θ a $\mathbf{p}$ comme but individuel (i.e. que WAG(μ,$\mathbf{p}$,Θ,$\mathbf{q}$) est valide) jusqu'à ce que $\mathbf{p}$ soit mutuellement reconnu comme terminé.

Ainsi l'équipe Θ a un but commun persistant JPG(Θ,$\mathbf{p}$,$\mathbf{q}$) valide si tous les membres de Θ ont la croyance mutuelle que $\mathbf{p}$ est à réaliser (condition 1) et que cette réalisation est un but commun (condition 2). La condition 3 permet de s'assurer que chaque agent de Θ va participer à la réalisation de $\mathbf{p}$ tant que tous les membres de Θ n'auront pas la croyance commune que $\mathbf{p}$

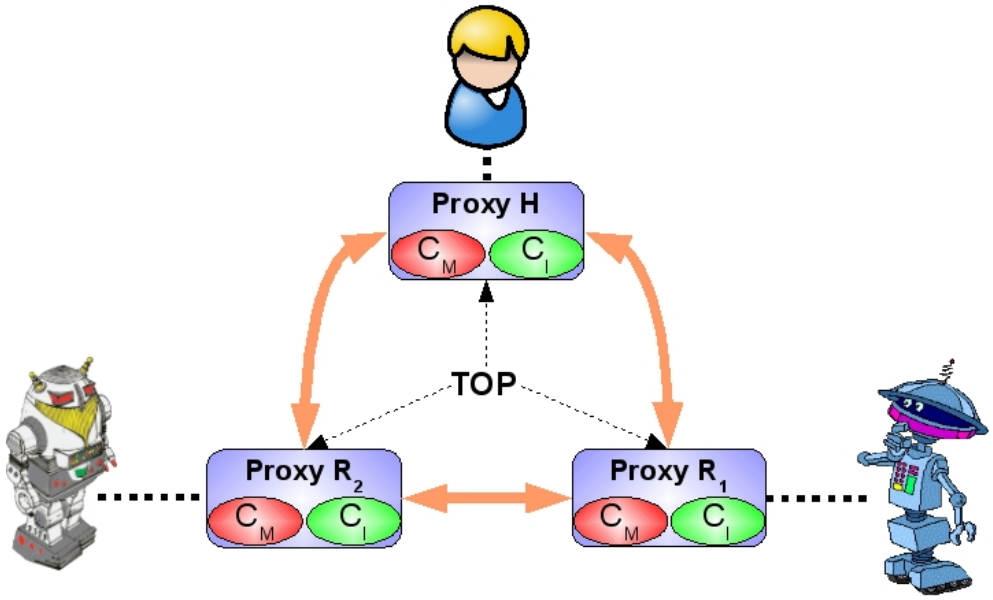

FIG. 1.5 – Structure d'une application STEAM.
Chaque agent (homme ou robot) est représenté par un proxy qui a en charge le maintien des croyances C_M et C_I de l'agent. Les différents proxys partagent un plan commun appelé TOP.

est terminé. Le but **p** peut se terminer par un succès, un échec ou un abandon, ce dernier cas étant déterminé par l'éventualité où les conditions de pertinence **q** ne sont plus valides. Un but individuel WAG(μ,**p**,Θ,**q**) est valide si une des conditions suivantes est satisfaite :

– μ a la croyance individuelle que **p** n'est pas encore réalisé et la croyance individuelle que **p** doit être réalisé,

– μ a la croyance individuelle que **p** est terminé (par un succès, un échec ou un abandon) et s'efforce de rendre cette croyance commune à tous les membres de Θ i.e. μ veut « transformer » cette croyance individuelle en une croyance mutuelle.

La théorie de l'intention jointe permet ainsi de formaliser les croyances que doivent posséder les agents pour démarrer et clore une collaboration. Cela permet donc de définir les actes de communication nécessaires au bon déroulement de la collaboration.

1.2.3.2 La planification réactive

Ce type de planification a été largement utilisé dans des applications multi-agents hétérogènes. Cette technique de planification se base sur un plan global décrit de manière générique qui va être instancié et adapté au fur et à mesure de l'exécution des différentes tâches le composant.

L'infrastructure STEAM décrite dans [Tambe 97] et [Scerri 03] est une approche multi-agents basée sur la planification réactive. Un exemple de mise en place de cette infrastructure est illustrée par la figure 1.5. Dans cette infrastructure, chaque agent réel (homme ou robot) est associé à un proxy. Chaque proxy contient un ensemble de croyances associées à l'agent qu'il représente. Conformément à la théorie de l'intention jointe ces croyances sont divisées en deux

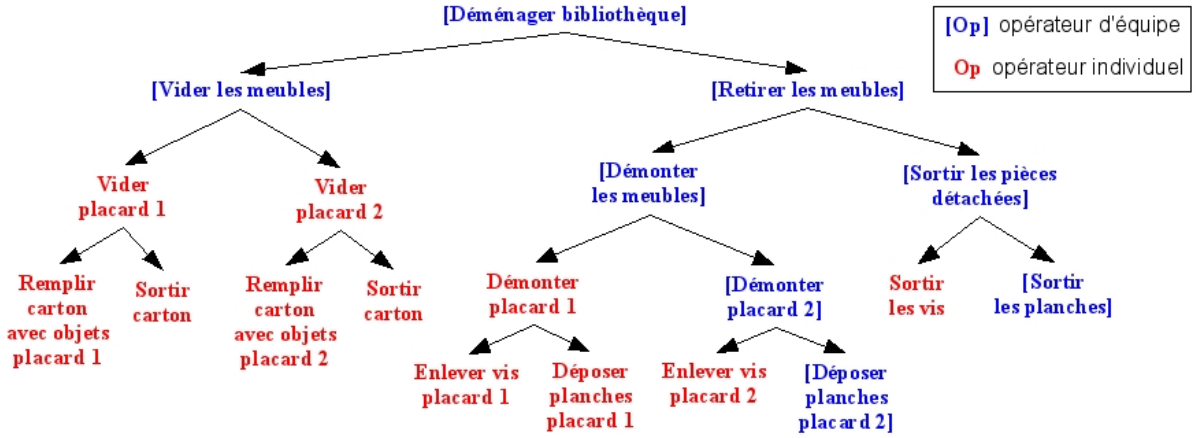

FIG. 1.6 – Exemple de *Team Oriented Plan* (*TOP*).
Un TOP est une structure hiérarchique décomposant une tâche en un ensemble de sous-tâches. Certaines tâches sont à réaliser individuellement alors que d'autres doivent être prises en charge par une équipe d'agents.

ensembles : C_I l'ensemble des croyances individuelles et C_M l'ensemble des croyances mutuelles. Pour chaque agent le proxy associé va alors raisonner sur un plan réactif hiérarchique appelé TOP (*Team Oriented Plan* en anglais) [Scerri 04]. Ce plan va être instancié et analysé au fur et à mesure de l'exécution des différentes étapes le composant. Un proxy peut raisonner de manière autonome sur le TOP lorsque les croyances qu'il contient sont suffisamment fiables ou, dans le cas contraire, peut demander à son agent confirmation des actions à opérer.

Un exemple de TOP est donné par la figure 1.6. Dans un TOP on distingue deux types d'opérateurs : les opérateurs individuels et les opérateurs d'équipe. Un opérateur individuel ne cible qu'un agent unique alors qu'un opérateur par équipe vise toute ou partie de l'équipe affectée à l'opérateur. Chaque opérateur consiste en : (1) des règles de préconditions, (2) des règles d'application et (3) des règles de terminaison. Les règles de préconditions définissent les conditions d'applicabilité de l'opérateur, les règles d'applications décrivent les sous-buts à réaliser pour satisfaire l'opérateur courant (description de la structure hiérarchique) et, enfin, les règles de terminaison définissent les conditions menant à la fin de l'opérateur (échec, succès ou abandon).

Le fonctionnement de STEAM est directement inspiré de la théorie de l'intention jointe. Pour illustrer cela, considérons l'infrastructure illustrée par la figure 1.5 et le TOP illustré par la figure 1.6. L'équipe d'agents composée de l'homme H et des robots R_1 et R_2 doit réaliser l'opérateur d'équipe « déménager la bibliothèque ». Les agents commencent donc par établir un *but commun persistant* (JPG) sur cette tâche. Pour cela, les proxys s'échangent des messages afin d'établir les croyances mutuelles et individuelles nécessaires au maintien du JPG. Une fois le JPG établi, chaque agent consulte le TOP qu'il a en mémoire et applique donc la recette fournie par celui-ci pour la tâche « vider les meubles » (la tâche « retirer les meubles » sera traitée lorsque la tâche « vider les meubles » aura été accomplie). On opère ainsi pour toutes les équipes ou sous-équipes d'agents affectées à des opérateurs d'équipe. Lorsqu'un agent est affecté à un opérateur individuel, il utilise ses croyances individuelles et les recettes fournies par le TOP

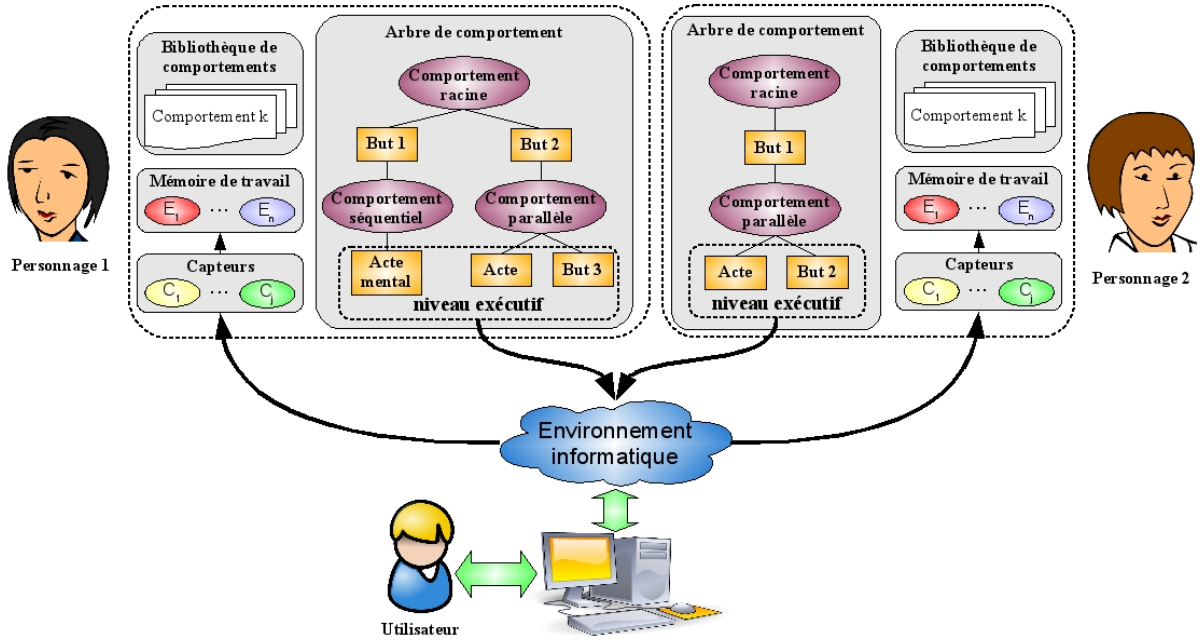

FIG. 1.7 – Structure de fonctionnement du langage ABL.
Chaque agent logiciel est associé à un ensemble de structures définissant son comportement : l'état de ses capteurs, une mémoire de travail, une bibliothèque de comportements et un arbre décrivant les comportements en cours de réalisation.

afin d'établir un plan individuel pour la tâche associée. Une fois qu'un agent a un plan établi il démarre l'exécution des actions associées jusqu'à ce qu'il obtienne la croyance individuelle que la tâche associée soit terminée. Lorsqu'un opérateur d'équipe est achevé, les proxys des membres de l'équipe ayant participé à sa réalisation s'échangent des messages afin d'établir les croyances mutuelles nécessaires à la fin du JPG associé à l'opérateur. Afin de limiter les échanges massifs de messages entre proxys, STEAM intègre une méthode probabiliste déterminant si l'importance de la croyance à établir justifie le coût lié à l'exécution du protocole de communication.

Le TOP utilisé pour la réalisation des tâches qui doivent l'être est toujours le même d'une exécution à l'autre. Par contre, les préconditions attachées à chaque opérateur permettent de n'exécuter que ce qui doit l'être en fonction de l'état courant du monde. Ainsi, si on considère le TOP de la figure 1.6 et que dans l'état du monde courant les meubles de la bibliothèque ont déjà été vidés alors l'opérateur « Vider les meubles » ne sera pas exécuté et les agents passeront directement à la réalisation de l'opérateur « Retirer les meubles ». De même, l'exécution d'un TOP pourra être différente d'une fois sur l'autre si les sous-équipes associées aux opérateurs d'équipe varient.

La planification réactive a également été utilisée pour la conception d'applications logicielles mettant en scène des agents intelligents, cela a donné naissance au langage ABL ([Mateas 02b], [Mateas 02a], [Mateas 04]). Ce langage est utilisé pour décrire le comportement à adopter par des agents logiciels interactifs évoluant dans une simulation informatique où l'humain joue son

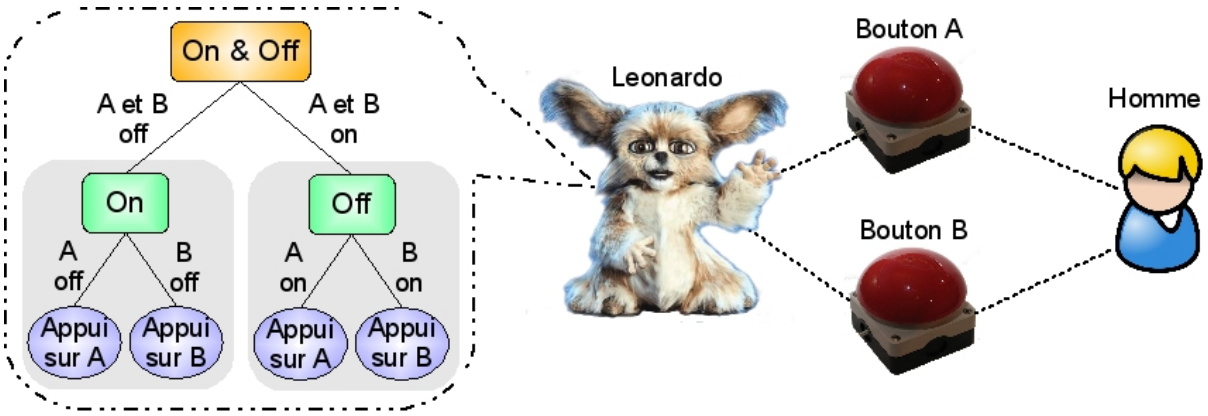

FIG. 1.8 – Tâche *Allumer/Eteindre* avec le robot Leonardo.
Le robot Leonardo représente les tâches qu'il apprend sous la forme d'un arbre hiérarchique. Il est alors capable d'exécuter ces tâches tout en proposant à l'homme de réaliser certaines branches de l'arbre.

propre rôle (voir figure 1.7). Chaque personnage informatique est représenté par un ensemble d'entités logicielles :

- ➢ une bibliothèque de comportements prédéfinis,
- ➢ un ensemble de capteurs lui permettant de percevoir l'environnement simulé,
- ➢ une mémoire de travail contenant les données nécessaires au bon déroulement des comportements en cours,
- ➢ un arbre dynamique décrivant les comportements en cours.

Un comportement élémentaire pour un personnage peut prendre plusieurs formes : une action du personnage dans l'environnement de simulation, l'attente d'un évènement particulier ou un acte mental (i.e. un processus de calcul pur). Un comportement de plus haut-niveau est décrit sous la forme de sous-tâches à réaliser, ces sous-tâches pouvant être réalisées de manière séquentielle ou parallèle. L'arbre de comportement est alors enrichi à chaque décomposition en sous-tâches avec la ou les premières tâches à effectuer et ainsi de suite jusqu'à ce que le comportement complet ait été réalisé. La mémoire de travail contient un ensemble de processus créés dynamiquement qui permettent de contrôler l'exécution des comportements en cours.

Le langage ABL permet de décrire des comportements *joints* ciblant un ensemble d'agents (i.e. de personnages animés et/ou l'utilisateur). Un comportement *joint* est caractérisé par la nécessité de s'assurer que tous les agents concernés souhaitent y participer. Ainsi, lorsqu'un comportement joint est lancé par un agent, ce dernier entre en contact avec les autres membres de l'équipe concernée afin d'établir les croyances mutuelles nécessaires à l'établissement d'un *but commun persistant* (JPG). Afin de terminer le comportement *joint*, les membres de l'équipe communiquent entre eux afin d'établir les croyances mutuelles nécessaires à la fin du JPG.

La planification réactive a également été utilisée sur le robot interactif Leonardo [Breazeal 04]. Ce robot est doté de grandes capacités interactives : expressions faciales, communication par geste, dialogue, etc... Ce robot est capable d'apprendre comment réaliser certaines tâches puis

de les réaliser collaborativement avec un partenaire humain. Lors de la phase d'apprentissage, Leonardo construit un plan réactif hiérarchique (voir figure 1.8). Lors de la phase d'exécution, les différentes actions/tâches sont réparties entre l'homme et Leonardo. Pour ce faire, Leonardo utilise ces capacités interactives afin d'établir les croyances mutuelles nécessaires à l'établissement d'un *but commun persistant* (JPG). De même, les croyances mutuelles nécessaires à la terminaison du JPG sont établies par les capacités multimodales de Leonardo. On constate donc que, là encore, l'application de la théorie de l'intention jointe se concrétise par des points de synchronisation entre agents au début et à la fin des tâches collaboratives.

Comme l'illustre les exemples précédents, la planification réactive présente l'avantage majeur de conférer aux agents artificiels une autonomie décisionnelle importante et une excellente réactivité vis-à-vis des évènements extérieurs inattendus puisque le plan (et toutes ses éventualités) est connu à l'avance par l'ensemble des agents. Toutefois, cette approche présente deux inconvénients : elle nécessite de pouvoir décrire un plan suffisamment générique pour pouvoir couvrir un maximum de situations possibles et elle nécessite un échange permanent d'informations entre les agents afin de leur permettre de se synchroniser (notamment à l'établissement et à la clôture des tâches collaboratives). Ce type de planification est difficile à mettre en oeuvre dans le cadre des applications robotiques interactives qui nous intéressent. En effet, la quantité d'informations à échanger entre le robot et ses partenaires humains peut vite devenir importante si le nombre de tâches collaboratives est conséquent dans le plan courant. De plus, de par le principe de la planification réactive, le robot devra exécuter la tâche courante qui lui aura été attribuée avant de porter son attention sur la suivante. Si la tâche suivante est une tâche collaborative le robot devra interrompre l'exécution du plan pour communiquer avec ses partenaires afin d'établir les croyances mutuelles nécessaires à l'exécution de la nouvelle tâche. Le robot peut alors apparaître comme un exécutant intelligent limitant son raisonnement à la prochaine tâche sans être capable de mémoriser des consignes à long terme.

1.2.3.3 La planification délibérative

A la différence de la planification réactive, la planification délibérative consiste à générer un plan complet avant de se lancer dans son exécution. L'horizon de raisonnement d'un planificateur délibératif est donc plus important que celui d'un planificateur réactif. Ceci permet donc d'aboutir à une vue plus globale du rôle de chaque agent et permet de réaliser la tâche au mieux. Une fois le plan global établi chaque agent connaîtra les actions qu'il doit réaliser, cela permet une meilleure fluidité lors de l'exécution du plan. Toutefois, en cas d'échec lors de cette exécution il devient nécessaire de produire un plan alternatif complet. Ceci implique qu'une application utilisant un planificateur délibératif aura une moins bonne réactivité face aux évènements imprévus que si elle utilisait un planificateur réactif. De plus, ce type de planification présente également l'avantage de limiter les messages échangés entre agents lors de l'exécution du plan puisque les actions à effectuer ont été déterminées conjointement au préalable. Les applications existantes utilisant la planification délibérative pour la collaboration entre agents hétérogènes peuvent se classer en deux catégories : (1) les systèmes permettant à l'homme d'intervenir dans le processus de

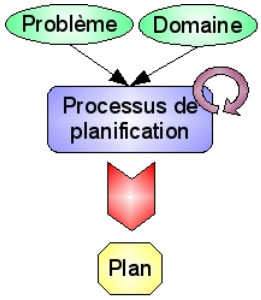

FIG. 1.9 – Principe de fonctionnement d'un planificateur « standard ».
Le planificateur prend en entrée un problème et un domaine puis fonctionne de manière autonome.

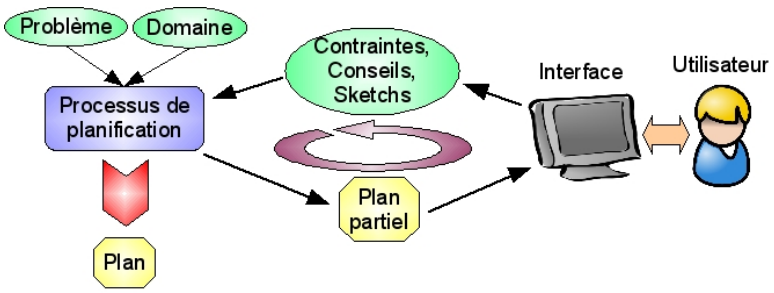

FIG. 1.10 – Planificateur à initiative mixte.
Le planificateur prend en entrée un problème et un domaine. Pendant le processus de planification l'utilisateur peut influencer les choix du planificateur en imposant des contraintes ou en choisissant les branches de l'espace de recherche à explorer.

planification et (2) les systèmes utilisant la planification pour proposer des actions adéquates à l'homme.

Planification à initiative mixte

Le principe de fonctionnement d'un planificateur classique est illustré par la figure 1.9. Le planificateur prend en entrée un domaine décrivant l'ensemble des actions possibles et un problème décrivant les buts à satisfaire. Le processus de planification utilise ces deux descriptions pour produire un plan, le processus de planification agit alors de manière autonome sans intervention extérieure. Un planificateur à initiative mixte permet à l'utilisateur d'intervenir dans le processus de planification comme l'illustre la figure 1.10. Dans un planificateur à initiative mixte, chaque étape du processus de planification permet d'aboutir à un plan partiel qui est présenté à l'utilisateur via une interface. L'utilisateur peut alors influencer la prochaine étape du processus de planification en imposant des contraintes à respecter ou en donnant des conseils sur les options à privilégier.

Le planificateur PASSAT [Myers 02] est un exemple de planificateur à initiative mixte. Ce planificateur est basé sur le formalisme HTN (*Hierarchical Task Network* en anglais). Ce formalisme utilise des opérateurs de bas niveau (les actions de base) et un ensemble de tâches de haut-niveau (appelées « méthodes »). Les méthodes peuvent être décomposées en un ensemble partiellement ordonné d'opérateurs et/ou d'autres méthodes formant ainsi une structure hiérarchique de tâches. Ainsi, lors du processus de planification, un planificateur basé sur le formalisme HTN va développer une structure arborescente jusqu'à ce que les feuilles de cette structure soient toutes des opérateurs. Le planificateur PASSAT permet à l'utilisateur de guider l'évolution de l'arbre de raffinement pendant le processus de planification i.e. de l'aider à choisir parmi les décompositions possibles pour les méthodes qui restent encore à développer. Pour cela, PASSAT a été doté d'un système robuste de traitement de *sketchs* [Myers 03]. Comme défini dans [Myers 97], un sketch est un ensemble de tâches définissant une structure hiérarchique

partiellement ou totalement instanciée. Le processus de planification cherche alors à produire une structure arborescente incluant totalement ou partiellement le sketch. L'objectif final est alors de trouver la structure arborescente qui soit la plus compatible possible avec le(s) sketch(s) défini(s) par l'utilisateur.

L'utilisation des sketchs permet à l'homme d'influencer le processus de planification en privilégiant les choix de décompositions à retenir pour les tâches de haut-niveau. Un sketch permet donc de définir une forme d'heuristique permettant d'accélérer le processus de planification et d'imposer des contraintes sur la forme des plans solutions. La planification à initiative mixte permet donc à l'homme d'influencer le processus de décision d'un autre agent, toutefois, cette approche ne permet pas d'assurer une exécution collaborative du plan.

La planification délibérative a également été utilisée pour permettre à un agent logiciel d'aider un partenaire humain dans ses prises de décision. Cette approche est basée sur la théorie des plans partagés.

Théorie des plans partagés

La théorie des plans partagés ([Grosz 96], [Grosz 99]) est basée sur la notion d'intention de réaliser une tâche. Cette théorie a été développée pour permettre d'établir formellement la composante intentionnelle du discours lorsque celui-ci porte sur la réalisation de tâches par une équipe d'agents. La théorie des plans partagés utilise la notion de « recette ». Une recette est un ensemble de sous-tâches à effectuer pour réaliser une tâche de niveau supérieur. De ce fait, les recettes permettent de construire une hiérarchie de tâches. La réalisation d'une tâche T par une équipe Θ en utilisant la recette R_T consiste alors à répartir les sous-tâches composant R_T parmi les membres de Θ. Afin de représenter les états mentaux des agents au fur et à mesure du déroulement du dialogue, on utilise quatre méta-prédicats : FIP, PIP, FSP et PSP. Ces méta-prédicats sont respectivement associés à des plans individuels complets, des plans individuels partiels, des plans partagés complets et des plans partagés partiels.

Le méta-prédicat FIP (*Full Individual Plan* en anglais) représente l'état mental d'un agent G lorsqu'il a déterminé un plan P_α grâce à une recette R_α complète permettant de réaliser la tâche α (i.e. il a un plan individuel complet). FIP(P_α,G,α,R_α) est valide si :

1. G connaît une recette R_α pour réaliser la tâche α et chaque sous-tâche β_i de R_α,
2. une des conditions suivantes est vraie :
   - ➢ β_i est une action élémentaire et G a l'intention de réaliser β_i lui-même,
   - ➢ β_i est une tâche de haut-niveau et FIP(P_{β_i},G,β_i,R_{β_i}) est valide,
   - ➢ G a l'intention de sous-traiter la réalisation de β_i à un autre agent.

Le méta-prédicat PIP (*Partial Individual Plan* en anglais) représente l'état mental d'un agent G utilisant un plan P_α pour réaliser la tâche α en ayant actuellement trouvé qu'une solution

partielle (i.e. P_α est incomplet). PIP(P_α,G,α) est valide si :

1. G a la croyance qu'il existe une façon de réaliser α mais ne possède actuellement qu'une recette partielle (il a alors l'intention de compléter cette recette).
2. Pour chaque sous-tâche β_i de la recette partielle une des conditions suivantes est vraie :
   - ➢ G a l'intention de réaliser β_i lui-même mais n'a qu'un plan partiel pour le faire,
   - ➢ G a l'intention de sous-traiter β_i à un autre agent mais n'a pas de plan complet pour réaliser l'acte de sous-traitance,
   - ➢ G n'a pas encore l'intention franche de réaliser β_i (il n'a qu'une intention potentielle).

Le méta-prédicat FSP (*Full Shared Plan* en anglais) représente l'état mental d'un groupe d'agent GR lorsqu'il a déterminé un plan P_α grâce à une recette R_α complète permettant de réaliser la tâche α. FSP(P_α,GR,α,R_α) est valide si :

1. Le groupe GR a la croyance mutuelle que tous les membres de GR sont impliqués dans la réussite de la tâche α,
2. Le groupe GR a la croyance mutuelle des sous-tâches β_i à réaliser pour réaliser la tâche α,
3. Pour chaque sous-tâche β_i une des conditions suivantes est vraie :
   - ➢ La tâche β_i est une tâche nécessitant un seul agent et un membre de GR va la réaliser,
   - ➢ La tâche β_i est une tâche nécessitant plusieurs agents et un sous-groupe de GR va la réaliser,
   - ➢ La tâche β_i est une tâche nécessitant un seul agent et le groupe GR a l'intention de sous-traiter la réalisation de β_i à un autre agent extérieur à GR,
   - ➢ La tâche β_i est une tâche nécessitant plusieurs agents et le groupe GR a l'intention de sous-traiter la réalisation de β_i à un groupe d'agents extérieurs à GR.

Le méta-prédicat PSP (*Partial Shared Plan* en anglais) représente l'état mental d'un groupe d'agent GR utilisant un plan P_α pour réaliser la tâche α en ayant actuellement trouvé qu'une solution partielle (i.e. P_α est incomplet). PSP(P_α,GR,α) est valide si :

1. Le groupe GR a la croyance mutuelle que tous les membres de GR sont impliqués dans la réussite de la tâche α,
2. Le groupe GR a la croyance mutuelle qu'il existe une recette permettant de réaliser la tâche α mais la recette actuelle n'est que partielle. Les membres de GR ont un FSP pour réaliser la recette partielle,
3. Pour chaque sous-tâche β_i dans la recette partielle, une des conditions suivantes est vraie :
   - ➢ β_i ne nécessite qu'un agent, un agent membre de GR a l'intention de réaliser β_i mais n'a qu'un plan partiel pour la réaliser,
   - ➢ β_i nécessite plusieurs agents, un sous-groupe de GR a l'intention de réaliser β_i mais n'a qu'un plan partiel pour la réaliser,
   - ➢ β_i ne nécessite qu'un agent, le groupe GR a décidé de sous-traiter la réalisation de β_i à un autre agent mais le groupe GR n'a qu'un plan partiel pour réaliser l'acte de sous-traitance,

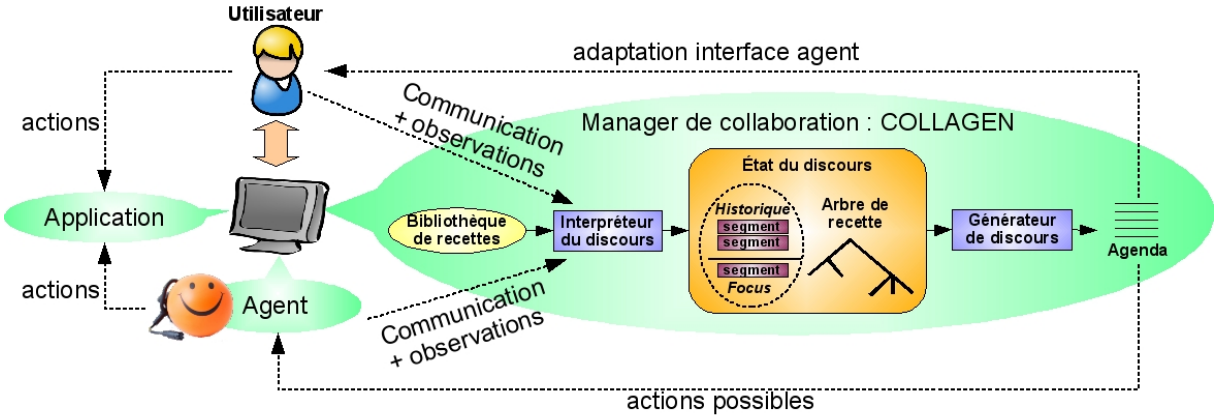

FIG. 1.11 – Structure de l'agent COLLAGEN.
L'agent construit une structure hiérarchique reflétant les intentions de l'utilisateur et observe ses actions sur l'application courante. Il peut alors proposer de l'aide à l'utilisateur en fonction de ce que ce dernier souhaite faire.

- ➢ β_i nécessite plusieurs agents, le groupe GR a décidé de sous-traiter la réalisation de β_i à un autre groupe mais le groupe GR n'a qu'un plan partiel pour réaliser l'acte de sous-traitance,
- ➢ le groupe GR n'a pas encore délibéré à propos de β_i et aucune décision n'a encore été prise sur qui réalisera β_i.

La théorie des plans partagés formalise donc les intentions nécessaires à la constitution d'un plan permettant la réalisation des tâches désirées. Cela permet donc de formaliser la composante intentionnelle du dialogue lorsque celui-ci vise à déterminer la répartition des tâches entre agents collaboratifs. Cette théorie présente l'avantage majeur de faire apparaître explicitement les notions de plans en cours d'élaboration (plans partiels) et de plans intégralement déterminés (plans complets).

Application basée sur la théorie des plans partagés

La théorie des plans partagés a été appliquée pour la conception d'un manager de collaboration appelé COLLAGEN ([Rich 97], [Rich 98]). Le fonctionnement de ce système peut être illustré par la figure 1.11. L'objectif est de permettre à un agent logiciel d'apporter son aide à un partenaire humain qui est en train d'utiliser une application sur un ordinateur. Afin de garantir un fonctionnement efficace et une interaction de bonne qualité une entité appelée *manager d'interaction* est en charge d'analyser le comportement du partenaire humain afin de lui proposer une aide adaptée. Le *manager d'interaction* scrute les actions que le partenaire humain effectue dans l'application et analyse les différentes communications entre ce dernier et l'agent logiciel grâce à un module d'interprétation du discours. Un module est alors en charge de maintenir l'état de la composante intentionnelle du partenaire humain en conservant l'historique des communications (en se focalisant sur les plus récentes) et en construisant un arbre de recettes représentant le *plan partagé partiel* (PSP) courant. Cette structure permet d'établir les actions

qui restent à effectuer et permet donc d'établir une liste d'actions associées à des priorités. Ces informations sont stockées dans l'agenda. Ce dernier est alors utilisé par l'agent soit pour agir directement sur le logiciel soit pour adapter la communication entre l'agent et son partenaire humain. Ce principe est répété jusqu'à obtention par l'humain d'un *plan individuel complet* (FIP).

Dans le cadre de robots assistants, il est souhaitable que le robot soit capable de proposer et de discuter de plans permettant de satisfaire les buts fixés par ses partenaires humains. La collaboration entre l'homme et le robot se construit donc autour d'un plan commun. Ceci implique que le robot soit capable de prévoir un certain nombre d'actions à effectuer dans le futur. Il doit donc être capable de planifier sur un horizon à moyen terme. Cette contrainte impose donc l'utilisation de la planification délibérative. De plus, dans le cadre des applications qui nous intéressent, la phase préalable de négociation entre les agents pour déterminer quel sera le rôle de chacun lors de l'exécution du plan reste indispensable mais ne doit pas être trop fréquente. En effet, il est préférable pour un robot assistant d'établir les croyances mutuelles pour l'ensemble du plan avant de débuter l'exécution de celui-ci plutôt que d'établir ces croyances mutuelles au fur et à mesure de la réalisation du plan. Cela permettra notamment de limiter le nombre des phases d'échanges d'informations entre agents et donc d'assurer une meilleure autonomie de l'ensemble des agents.

1.3 Autres sources d'inspiration

L'état de l'art que nous venons de présenter est orienté vers les applications interactives où l'homme collabore avec des agents autonomes physiques ou logiciels. Toutefois, il existe d'autres approches de la planification pour la coopération notamment dans les systèmes multi-robots/multi-agents. Il ne s'agit pas ici de faire un état de l'art de ce type d'applications mais de mettre en évidence l'intérêt qu'elles peuvent présenter dans le développement de robots assistants. On peut par exemple citer de manière non-exhaustive :

✓ La coopération de robots par processus décisionnels de Markov. Dans ce type d'approche la coopération entre les robots se fait par la construction de politiques. On peut par exemple citer des travaux tels que [Beynier 06] qui utilisent des processus de Markov décentralisés partiellement observables.

✓ La coopération par négociation et allocation de tâches. Dans ce type d'approche la coopération se traduit par la répartition des tâches à exécuter entre les différents agents. Cette thématique est abordée dans [Botelho 99] par exemple.

✓ La coordination/fusion de plans. Dans ce type d'approche, les plans individuels produits pour chacun des agents sont combinés/comparés pour éviter les conflits entre agents (voir [Gravot 01] pour un exemple).

✓ L'anticipation du comportement des autres agents. Dans ce type d'approche chaque agent est doté de processus décisionnels lui permettant d'anticiper le comportement des autres agents et ainsi d'agir de façon à garantir un comportement global cohérent. Ce type d'approche a par exemple été utilisé par la simulation de trafic routier [Doniec 06].

Bien que cette thèse n'aborde pas en détail ces différentes approches, il est important de noter qu'elles utilisent des concepts présentant des liens évidents avec la problématique des robots assistants. Toutefois, ces travaux sont basés sur des approches très différentes de la notre, nous ne les détaillerons donc pas davantage dans ce manuscrit.

1.4 Conclusion

Dans la première partie de ce chapitre nous avons tenté d'établir une classification des différents types d'applications robotiques nécessitant une interaction homme-robot. Cette classification est basée sur deux critères : (1) le partage de l'espace d'action entre l'homme et le robot et (2) la durée de l'interaction. A l'aide de cette classification, nous avons établi que la planification de tâches est un outil présentant des avantages majeurs pour les applications robotiques interactives où l'homme et le robot se côtoient longuement et agissent dans le même environnement.

La planification de tâches peut se décliner selon plusieurs approches. Nous avons donc étudié ces différentes approches dans la seconde partie de ce chapitre en étudiant des exemples concrets d'applications. Enfin, nous avons opté pour l'utilisation de la planification délibérative en raison de l'autonomie décisionnelle nécessaire aux applications qui nous intéressent.

2

Comportement social d'un robot

Le chapitre précédent nous a permis de préciser le « quoi » et le « pourquoi » des travaux décrits dans ce manuscrit : nous nous intéressons à des applications robotiques de service dans lesquelles le robot agit comme un assistant. Dans un tel cadre, le robot doit être doté d'une grande autonomie décisionnelle, c'est pourquoi nous nous sommes intéressés à la planification de tâches. Il convient maintenant de s'interroger sur le « comment » i.e. sur la façon dont un planificateur de tâches peut contribuer à la qualité *sociale* du comportement du robot. Nous allons donc commencer par définir les caractéristiques inhérentes à un comportement qualifié de *social*.

2.1 Définition d'un comportement social

2.1.1 Les caractéristiques du comportement social

La notion de *comportement social* peut prendre de nombreuses formes concrètes. La définition que nous retiendrons ici est celle d'une activité jointe entre l'homme et le robot pour réaliser une tâche commune tout en assurant une interaction de bonne qualité. Ceci signifie que nous nous intéressons au cas d'un robot physique qui agit sur les mêmes objets que son partenaire humain et que ces deux agents partagent le même espace d'actions. Dans cette situation, le robot se situe à la même échelle que l'homme. Dans ce contexte, l'hypothèse la plus généralement admise est que la qualité d'une interaction homme-robot est d'autant plus grande que cette interaction est similaire à celle d'un échange homme-homme. L'observation d'interactions entre des hommes

et des agents artificiels ont permis à *Klein et al.* d'identifier dix *challenges* ([Klein 04]). Ces challenges nous semblent pertinents et sont compatibles avec notre vision du comportement social d'un robot :

1. **Le contrat de base** : la faculté pour l'homme et le robot de se mettre d'accord sur le fait qu'ils vont travailler ensemble,
2. **Modélisation des autres agents** : pour agir comme un membre de l'équipe, le robot doit être capable de modéliser fidèlement ses partenaires humains,
3. **Prédictibilité** : le robot doit agir de manière prédictible vis-à-vis de ses partenaires humains,
4. **Directabilité** : si le robot a reçu une information remettant en cause le bien fondé de certaines actions en cours, le robot doit pouvoir modifier ses actions mais aussi être capable d'informer ses partenaires d'en faire de même,
5. **Expression d'intentions** : le robot doit être capable de rendre explicite à ses partenaires humains son état courant et ses intentions,
6. **Interprétation des comportements** : le robot doit être capable d'interpréter les comportements de ses partenaires humains afin d'en déduire leurs intentions,
7. **Négociation des buts** : le robot doit être capable de « dialoguer » avec ses partenaires humains sur les buts à satisfaire et sur la façon de les réaliser,
8. **Collaboration** : le robot doit être autonome mais doit intégrer explicitement la notion de collaboration dans son système de raisonnement,
9. **Gestion de l'attention** : le robot doit être capable de se focaliser sur les données importantes à observer,
10. **Contrôle des coûts** : le robot doit tenir compte des coûts engendrés par ses actions au sein de l'équipe.

Ces challenges ont été identifiés en analysant des interactions entre hommes et représentent les différentes caractéristiques que doit posséder un agent artificiel afin de pouvoir assurer une interaction homme-robot d'aussi bonne qualité que si son partenaire humain interagissait avec un autre être humain. Ainsi, le robot se doit de pouvoir communiquer efficacement avec ses partenaires humains (challenges 1, 4 et 7). Le robot doit également agir de manière collaborative, compréhensible et efficace (challenges 3, 8 et 10). Cela impose donc que le robot soit capable de prendre en compte les spécificités de ses partenaires (challenge 2) et de se focaliser sur les évènements importants (challenge 9). Enfin, le robot se doit de rendre compréhensible son comportement (challenge 5) mais aussi de comprendre et d'interpréter les comportements de ses partenaires humains (challenge 6).

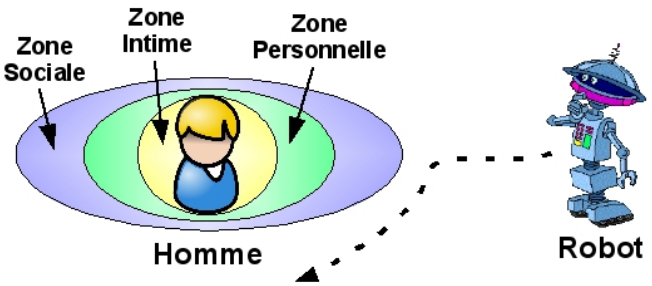

FIG. 2.1 – Zones de proximité d'un être humain selon les études de Hall.

2.1.2 Production de comportements sociaux

Nous nous intéresserons ici à la *production* de comportements sociaux par le robot i.e. nous n'aborderons pas les aspects relatifs à la perception. Les challenges du 2.1 peuvent se concrétiser à différents niveaux de l'architecture logicielle du robot en partant du niveau fonctionnel vers le niveau décisionnel. Les différents travaux qui se sont intéressés à la production de comportements sociaux par un robot se sont principalement focalisés sur la reproduction partielle du comportement humain. On peut classer ces différents travaux en trois catégories :

- la prise en compte de l'homme dans la réalisation des actions,
- l'utilisation de modèles cognitifs,
- la reproduction du raisonnement humain.

La prise en compte de l'homme dans la réalisation des actions permet de s'assurer que le robot a un comportement compréhensible (challenges 3 et 5) lorsqu'il exécute une étape d'une tâche. L'utilisation de modèles cognitifs permet de conférer une « dimension plus humaine » au robot et donc de permettre à ses partenaires humains de lui attribuer des intentions (challenge 5). Enfin, la reproduction du raisonnement humain permet de doter le robot de représentations mentales compréhensibles par ses partenaires humains facilitant ainsi la communication (challenges 1 et 7) mais également l'attribution d'intentions au robot (challenge 5). Nous nous proposons de détailler ces différentes approches afin de préciser la contribution qu'elles peuvent apporter à la production de comportements sociaux du robot. Nous pourrons alors étudier dans quelle mesure l'utilisation de la planification de tâches peut contribuer au comportement social du robot.

2.2 Prise en compte de l'homme dans la réalisation des actions

Cette approche consiste à considérer l'homme au niveau de l'exécution des actions du robot. La présence d'êtres humains à proximité directe du robot impose que ce dernier exécute ses actions de façon à assurer leur sécurité physique. Toutefois, cette contrainte n'est pas suffisante pour garantir un comportement social du robot. Ce dernier doit également prendre en compte la façon dont il exécute ses actions car cela peut influencer sur sa relation avec son partenaire humain.

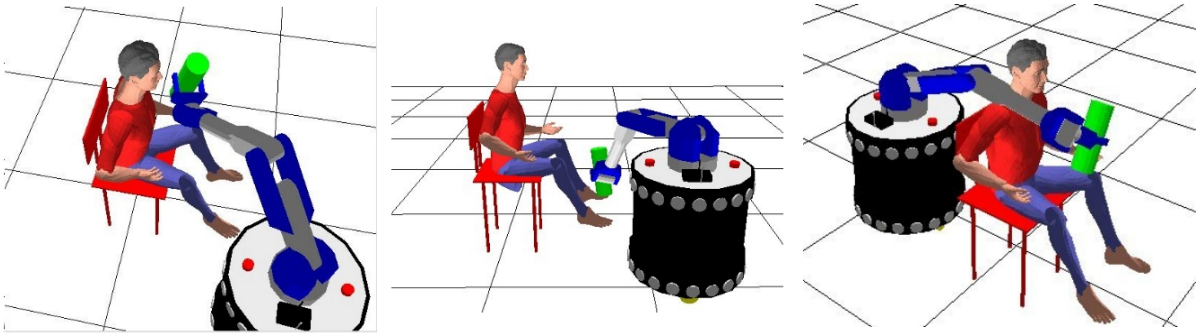

FIG. 2.2 – Problèmes de placement du robot pour un échange d'objets avec l'homme.
Les figures de gauche et du centre illustrent des situations où la position de l'objet est problématique pour sa saisie par l'homme. La figure de droite illustre une situation où le placement du robot peut provoquer une gêne pour son partenaire humain.

Un robot se déplaçant dans un environnement dans lequel évoluent des êtres humains ne peut se contenter de considérer ceux-ci comme des obstacles qu'il peut contourner et frôler à sa guise. En effet, au-delà des risques physiques, la proximité entre le robot et un être humain peut véhiculer chez ce dernier l'idée que le robot souhaite interagir avec lui. Une telle situation irait à l'encontre du challenge 5. A partir de l'étude d'échanges entre hommes, Hall a défini différentes zones de proximité pour les êtres humains et la signification associée à chacune d'elles [Hall 66]. La figure 2.1 définit les différentes zones de proximité d'un être humain. La zone intime est à une distance inférieure à 45cm et ne doit être franchie par le robot qu'en cas d'interaction nécessitant un contact physique. La zone personnelle correspond à une distance comprise en 45cm et 1.2m, elle est utile pour une interaction proximale i.e. un échange personnel entre l'homme et le robot. La zone sociale correspond à une distance comprise entre 1.2m et 3.5m et est utilisée pour une interaction plus formelle entre l'homme et le robot. Enfin, tout ce qui est au-delà de la zone sociale est appelée zone publique, la présence du robot dans cette zone n'implique aucune interaction avec l'homme. Des travaux tels que [Pacchierotti 05] se sont intéressés à l'application de ces zones de proximité pour le déplacement d'un robot en présence d'hommes. Dans cette étude le robot essaye de rester, autant que faire se peut, dans la zone publique des humains présents afin d'éviter toute mauvaise interprétation de ses intentions. Cet aspect social du déplacement est inhérent à la proximité entre l'homme et le robot et doit être pris en compte par ce dernier lors de ses déplacements.

Des travaux se sont focalisés sur la planification de trajectoires du robot lorsqu'il doit se déplacer dans un environnement où des humains sont présents [Sisbot 05]. Dans un tel contexte, le choix d'une trajectoire par le robot doit prendre en compte les contraintes de distances exposées au paragraphe précédent tout en évitant de surprendre au maximum les humains en présence. Pour cela il faut privilégier des trajectoires qui restent à bonne distance des humains et qui leur permettent de conserver au maximum le robot dans leur champ de vision. De ce fait le comportement adopté par le robot lors d'une action de déplacement pourra être interprété comme socialement acceptable.

L'action de donner un objet à un homme peut sembler triviale pour un être humain mais elle obéit pourtant à un certain nombre de contraintes sociales qu'un robot se doit de respecter. La figure 2.2 illustre des situations qui pourraient apparaître si l'homme n'était pas pris en compte lors de la réalisation de l'action par le robot. Ainsi, lorsque le robot souhaite transmettre un objet à son partenaire humain, il doit se placer face à lui et lui tendre l'objet de manière à ce que l'homme puisse le saisir le plus aisément possible. A partir de ces observations, des travaux tels que [Sisbot 07] se sont intéressés au développement de techniques de placement et de planification de mouvement pour un bras robotisé lorsque le robot est amené à transmettre un objet à son partenaire humain.

De même, l'attitude physique d'un robot lorsqu'il est engagé dans une conversation joue un rôle important dans l'aspect social de son comportement. Lorsqu'un robot dialogue avec un partenaire humain il se doit de respecter des règles d'engagement et de désengagement durant le déroulement de la conversation. Les travaux disponibles dans [Sidner 03a] et [Sidner 03b] mettent en évidence l'importance du suivi des mouvements de la tête et de l'orientation du regard lors d'une discussion. On voit là encore que la réalisation d'une action élémentaire du robot se doit d'être exécutée en tenant compte de la présence des humains à proximité du robot.

Les travaux ciblant la prise en compte de l'homme dans la réalisation des actions du robot permettent de garantir, à un niveau local, un comportement social du robot. Cet aspect est nécessaire pour assurer une bonne qualité de l'interaction entre l'homme et le robot, toutefois, cela ne suffit pas à garantir un comportement global socialement acceptable. En effet, le fait qu'un robot soit capable de se déplacer, de dialoguer et de manipuler un objet de manière socialement acceptable ne signifie pas qu'il soit capable de combiner ces de manière à exprimer un comportement social plus général. La prise en compte de l'homme dans la réalisation des actions doit donc être complétée par un processus décisionnel de haut-niveau intégrant lui aussi des contraintes sociales et garantissant au robot un comportement global socialement acceptable.

2.3 Utilisation de modèles cognitifs

Le comportement social d'un robot peut également être concrétisé en donnant une dimension intelligible par l'homme au robot. Pour cela, on introduit des modèles cognitifs dans l'architecture logicielle du robot afin de simuler les émotions et/ou la personnalité de celui-ci. L'émotion prédominante et/ou la personnalité courante sont alors utilisées pour adapter le comportement du robot à un niveau local (i.e. dans le déroulement de l'exécution des actions) mais aussi à un niveau plus global en influençant les successions d'actions qui seront effectuées. La personnalité et/ou l'émotion dominante du robot dépend(ent) du contexte et/ou de l'identité de son partenaire humain. Le robot est alors capable d'adapter son comportement au contexte et à son partenaire ce qui peut être interprété comme une intention du robot. On rejoint là le challenge 5.

La figure 2.3 illustre un exemple d'introduction d'un modèle cognitif dans l'architecture

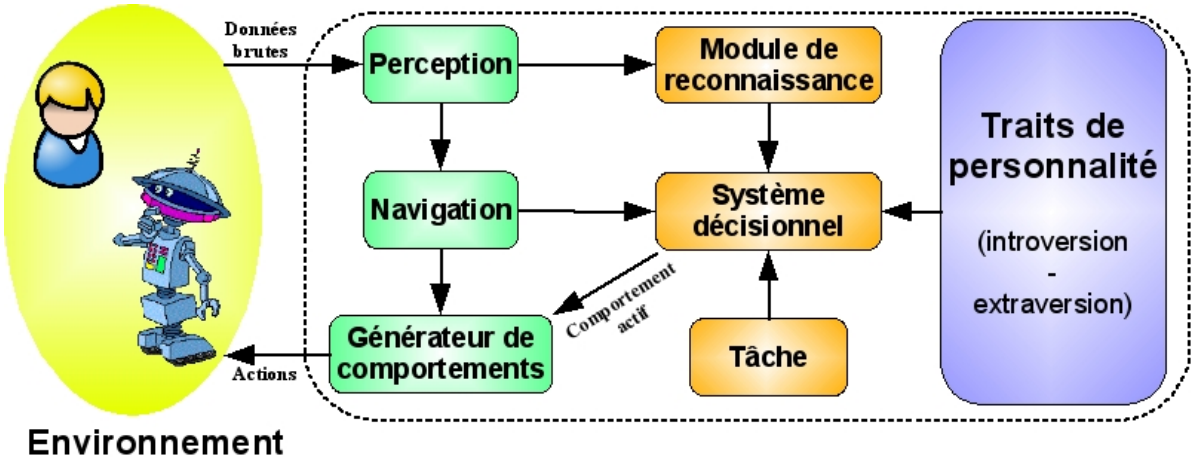

FIG. 2.3 – Architecture logicielle de contrôle d'un robot utilisant un modèle cognitif.
Le trait de personnalité retenu pour le robot influence le système décisionnel central et la façon dont les actions seront réalisées.

de contrôle du robot [Tapus 06]. Dans cette étude, le robot est utilisé pour apporter soutien et encouragements aux personnes ayant subi un AVC. Durant leur convalescence, ces patients effectuent des exercices physiques simples pour pouvoir retrouver la mobilité nécessaire aux gestes quotidiens. Le rôle du robot est alors de surveiller les exercices effectués par le patient et de l'encourager lorsqu'il semble faiblir. Le modèle cognitif utilisé est relativement simple, le robot peut revêtir deux personnalités : introvertie ou extravertie. La personnalité adoptée par le robot se traduit par un comportement local dicté par les règles suivantes :

Comportement du robot	**Robot Introverti**	**Robot Extraverti**
Volume de la voix	faible	fort
Débit de paroles	faible	rapide
Phrases utilisées	encouragements	défi

Ainsi un robot introverti utilise une voie douce avec un faible débit de paroles en utilisant des phrases encourageantes telles que : « Je sais que c'est dur mais rappelle toi que c'est pour ton bien. ». A l'opposé, un robot extraverti s'exprime d'une voie forte et rapide en utilisant des phrases lançant des défis à son partenaire humain : « Tu peux mieux faire, je le sais ! ». La personnalité influence également le comportement global du robot. En effet, un robot extraverti enchaînera rapidement les différentes actions de dialogue alors qu'un robot introverti les espacera davantage. Les résultats contenus dans [Tapus 06] montrent que la qualité de l'interaction entre l'homme et le robot s'améliore si le robot choisit une personnalité similaire à celle de son interlocuteur. Ainsi il devient envisageable de concevoir un robot capable de s'adapter à son partenaire humain courant en optant pour la personnalité qui lui ressemble le plus. Le robot pourrait alors exhiber son état courant (challenge 5) à travers les différences de ses comportements locaux.

L'insertion de modèles émotionnels dans l'architecture logicielle de contrôle du robot présente des avantages similaires. Une architecture basée sur cette approche a été développée [Malfaz 04] et utilisée sur un robot réel [Salichs 06]. Cette architecture est illustrée par la figure 2.4. Il s'agit

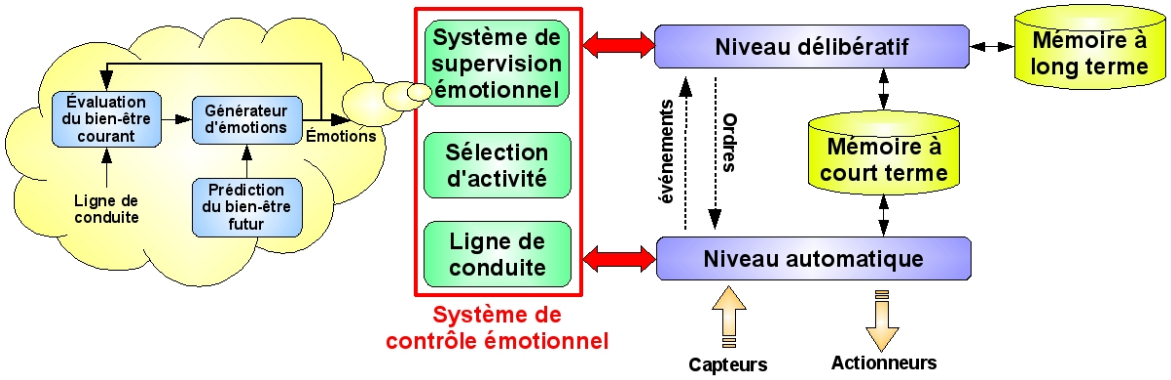

FIG. 2.4 – Exemple de superviseur émotionnel pour robot interactif.
Le système de contrôle émotionnel détermine la ligne de conduite que doit tenir le robot. A partir de cette ligne de conduite et de l'évolution du « bien-être » du robot, le superviseur émotionnel détermine « l'émotion » courante du robot et sélectionne une activité lui permettant d'augmenter au maximum son « bien-être » futur.

d'une architecture deux niveaux, le niveau automatique a en charge la gestion des processus pour les capteurs et les actionneurs et le niveau délibératif est en charge de déterminer les buts de haut-niveau et la façon de les réaliser. Ces deux niveaux sont associés à deux mémoires : la mémoire à court terme permet de stocker les informations changeantes propres à l'exécution courante alors que la mémoire à long terme contient des informations considérées comme *invariables* dans le temps. L'architecture est complétée par un *superviseur de contrôle émotionnel*. Le principe de ce superviseur est de simuler les émotions au sein du robot afin d'influencer les prises de décision effectuées au niveau délibératif. Le superviseur de contrôle émotionnel est composé de trois modules : (1) le module de ligne de conduite, (2) le module de sélection d'activité et (3) le système de supervision émotionnelle. Le module de ligne de conduite contient un nombre fini de lignes de conduite prédéfinies : *survie* (batterie faible ou dommage physique), *interaction sociale* (communication avec un partenaire humain) et *agenda* (exécution des actions planifiées). Les différentes lignes de conduites sont ensuite agrégées grâce à une combinaison linéaire pour déterminer la ligne de conduite dominante. En fonction de la ligne de conduite dominante le module de sélection d'activité définit les actions à effectuer (recharger la batterie, favoriser l'interaction avec l'homme ou poursuivre l'exécution du plan). Étant donné la ligne de conduite dominante le système de supervision émotionnelle va déterminer « l'émotion » courante du robot. Pour cela, le bien-être du robot est quantifié numériquement, la variation de cette valeur numérique va déterminer l'apparition d'une émotion. Pour cela, le robot calcule une estimation de son bien-être futur selon l'activité qui a été retenue. Le robot peut alors déterminer une estimation de la variation de son « bien-être » dans un avenir proche et en déduire l'émotion prédominante :

Variation du bien-être	Augmentation	Baisse faible	Annulation	Baisse due à l'activité choisie
Émotion prédominante	Joie	Tristesse	Peur	Colère

Les changements d'émotion peuvent ensuite être traduits par des expressions faciales afin de les rendre compréhensibles aux partenaires humains du robot. Le robot devient alors capable

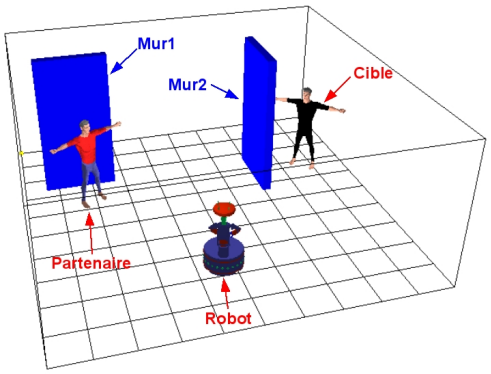

FIG. 2.5 – Exemple de situation de la tâche RECON.
Le robot et son partenaire humain doivent isoler leur cible en quadrillant le « secteur » de manière collaborative.

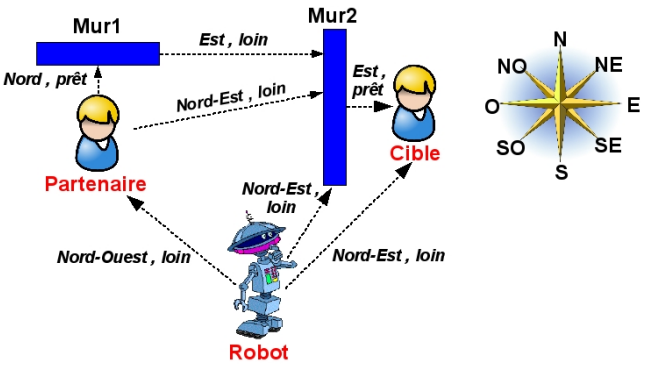

FIG. 2.6 – Carte cognitive associée à l'exemple de situation de la tâche RECON.
La carte cognitive contient une estimation des directions et des distances entre les entités de l'environnement. Les directions sont basées sur la rose des vents et les distances sont évaluée de manière grossière (proche, loin, très loin, etc...).

d'exprimer son état courant (challenge 5) d'une manière intuitive pour ses partenaires humains.

L'insertion de modèles cognitifs dans l'architecture logicielle de contrôle du robot permet de conférer à ce dernier une dimension plus acceptable par l'homme. Ainsi, si le robot est capable d'exprimer son émotion dominante (à l'aide d'expressions faciales par exemple) ou d'adapter son comportement au contexte, ses partenaires humains peuvent l'humaniser et lui attribuer des intentions. Le comportement du robot peut donc être jugé socialement acceptable.

2.4 Reproduction du raisonnement humain

Afin de permettre à l'interaction homme-robot de gagner en qualité, il faut que le robot se comporte de manière à ce que cette interaction soit la plus proche possible d'une interaction homme-homme. Une façon d'y parvenir est de doter le robot de représentations et/ou de systèmes de raisonnement directement inspirés de l'homme. Cette approche présente l'avantage majeur de faciliter la communication entre l'homme et le robot puisque les données utilisées et la façon de les manipuler par le robot sont intelligibles par son partenaire humain. Toutefois, ce type d'approche cible généralement un ensemble restreint de tâches.

Considérons le cas des données spatiales. Les données spatiales manipulées par un robot sont principalement des données numériques basées sur un point pris comme origine. Cette représentation n'est pas adaptée pour permettre une communication efficace entre l'homme et le robot. En effet, un homme définit sa position à l'aide de lieux topologiques et sa direction à l'aide de points de référence tels que les points cardinaux. Il convient donc de trouver un moyen permettant de faire le lien entre la représentation spatiale numérique utile du robot et la représentation spatiale symbolique utilisée par l'homme. La représentation numérique peut être utilisée pour construire une *carte cognitive* [Kennedy 07]. Cette carte cognitive contient les

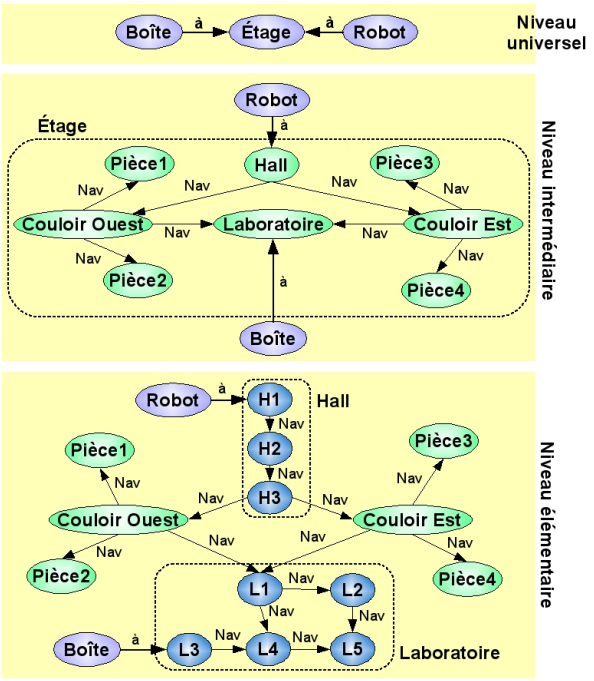

FIG. 2.7 – Exemple de graphe multi-hiérarchique annoté.
Cette représentation permet de structurer hiérarchiquement l'environnement. Les noeuds de ce graphe sont décomposés selon les agents et/ou les objets concernés par la tâche courante.

positions relatives et les orientations entre les éléments importants de l'environnement. L'utilité de cette carte est de permettre au robot de communiquer avec son partenaire humain sur la position relative d'une cible mobile qu'ils cherchent à isoler. Cette tâche est appelée *tâche RECON*. La figure 2.5 illustre un exemple de situation : le robot perçoit la cible mais le partenaire humain du robot ne la voit pas, le robot doit donc guider son partenaire de façon à ce qu'il puisse se rapprocher de la cible. Dans cette situation, le robot peut concevoir la carte cognitive illustrée par la figure 2.6. Cette carte contient uniquement des indications d'orientation basées sur les points cardinaux et des informations de proximité relative entre agents et objets. Ainsi, le robot peut aisément communiquer avec son partenaire en lui indiquant des orientations basées sur la rose des vents et des notions de distances qualitatives (proche, loin, etc.). Cette représentation cognitive de l'espace permet au robot de transmettre aisément des indications compréhensibles à son partenaire humain mais lui permet également d'interpréter les indications que son partenaire lui fournit.

D'autres travaux se sont intéressés à une reproduction partielle du raisonnement humain en se basant sur une représentation hiérarchisée de l'environnement [Galindo 04]. Dans cette approche l'environnement dans lequel évolue le robot est modélisé à l'aide d'un graphe multi-hiérarchique annoté. La figure 2.7 illustre un exemple d'un tel graphe. Le plus haut niveau du graphe est appelé *niveau universel* et le plus bas niveau est appelé *niveau élémentaire*. Chaque noeud du graphe représente un lieu, un objet ou un agent. Chaque arc du graphe représente la relation qui existe entre les deux noeuds. Des noeuds d'un même type peuvent être

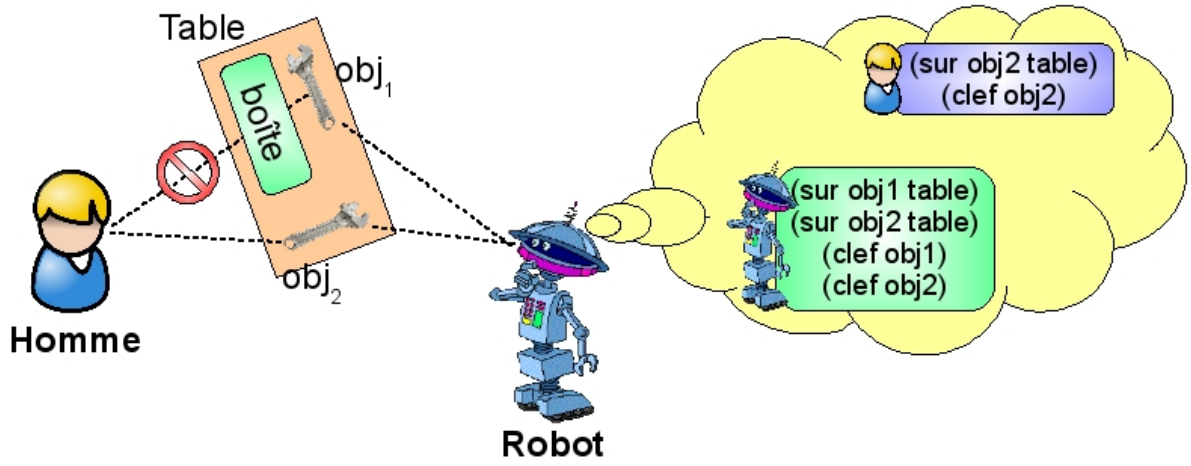

FIG. 2.8 – Exemple de situation où la prise de perspective permet de faciliter la communication entre l'homme et le robot.
Le robot dispose de deux représentations du monde : une basée sur ses propres croyances et une autre basée sur les croyances estimées de l'homme. Lorsque ce dernier demande au robot de lui apporter la clef, le robot détecte plusieurs options possibles puisqu'il perçoit deux clefs. Il utilise alors la seconde représentation du monde pour lever l'ambiguïté liée à la requête de l'homme.

regroupés pour former un *super-noeud* au niveau hiérarchique supérieur. Cette représentation de l'environnement permet au robot de raisonner de manière grossière puis d'affiner ce raisonnement en décomposant les super-noeuds concernés. Ainsi, dans l'exemple de la figure 2.7, si le robot doit aller chercher la boîte il va décomposer le super-noeud *Étage* ce qui donnera le graphe du niveau intermédiaire. Les super-noeuds *Hall* et *Laboratoire* sont ensuite développés afin de définir les positions exactes du robot et de la boîte. Les super-noeuds *Couloir Ouest* et *Couloir Est* seront ensuite développés afin de définir le meilleur trajet possible pour le robot. Cette approche présente l'avantage de représenter explicitement des concepts de haut-niveau manipulés par le robot. Cela permet une communication plus efficace entre l'homme et le robot puisque ce dernier peut adapter son discours en fonction du niveau hiérarchique qui intéresse son interlocuteur. De plus, cette approche présente l'avantage de reproduire un raisonnement proche de celui d'un être humain : raisonnement de haut-niveau qui est ensuite affiné jusqu'à parvenir à une série d'actions élémentaires. Ainsi, le partenaire humain du robot peut lui attribuer des intentions de la même façon qu'il le fait avec un autre homme.

Une des caractéristiques du raisonnement humain est de permettre à un homme de penser comme s'il était un autre individu. Cette capacité s'appelle la prise de perspective (*Perspective Taking* en anglais). Des travaux tels que [Trafton 05] se sont intéressés à la reproduction de cette faculté par un robot. Pour cela, le robot doit être capable de gérer plusieurs représentations concurrentielles de l'environnement afin de lui permettre d'interpréter les requêtes émises par son partenaire humain en se plaçant du point de vue de celui-ci. La figure 2.8 illustre une situation dans laquelle la prise de perspective permet au robot de lever l'ambiguïté liée à la requête émise par son partenaire humain. Dans cette situation, le partenaire humain demande au robot de lui apporter la clef. Cette requête est évidente pour l'humain puisqu'il ne perçoit qu'une seule

clef, cependant, le robot lui en perçoit plusieurs ce qui introduit une ambiguïté. Le robot peut alors faire appel à la représentation mentale de l'état de l'environnement du point de vue de son partenaire humain pour lever cette ambiguïté et lui apporter obj_2 à savoir la clef que le partenaire humain croyait unique.

La reproduction des structures de représentations et/ou du raisonnement humain dans les systèmes délibératifs du robot permet de faciliter la compréhension par l'homme du processus décisionnel du robot. Cette compréhension favorise la communication entre l'homme et le robot et permet au partenaire humain d'attribuer des intentions au robot.

2.5 Contribution de la planification de tâches au comportement social d'un robot

L'ensemble des approches présentées précédemment permettent au robot de se comporter de manière socialement acceptable. Les approches qui visent à considérer l'homme dans la réalisation des actions permettent de garantir un comportement social à un niveau local i.e. lorsque le robot exécute une action. L'introduction de modèles cognitifs et/ou de systèmes émotionnels dans l'architecture logicielle du robot permettent à celui-ci d'exprimer physiquement (expressions faciales par exemple) un ressentiment simulé. Ses partenaires humains peuvent alors plus facilement le comprendre améliorant ainsi la qualité de l'interaction homme-robot. Enfin, la reproduction des représentations et du raisonnement humain permettent une meilleure compréhension du robot par l'homme facilitant ainsi la communication entre eux.

Toutefois, dans aucune de ces approches il n'est pris explicitement en compte le fait qu'une série d'actions particulières du robot peut influencer l'intelligibilité de son comportement par son partenaire humain. En effet, la plupart des approches présentées se focalisent sur la situation présente ou sur l'utilisation de scripts d'actions qui sont exécutés par le robot. Seule la représentation de l'environnement par un graphe multi-hiérarchique annoté permet au robot de planifier lui-même les actions à opérer afin de satisfaire les buts courants. Toutefois, cette approche ne garantit pas que la série d'actions planifiées ne fera pas apparaître un comportement global ou une situation qui pourra être jugé comme socialement inacceptable. Pour cela, il faut que le processus qui permet au robot de déterminer les actions à opérer (i.e. le planificateur) intègre explicitement des notions de règles sociales afin de produire des plans dans lesquels le comportement global du robot garantira une bonne qualité de l'interaction homme-robot.

En effet, garantir une prise en compte de l'homme lors de la réalisation des actions, adapter son comportement de façon à véhiculer une image de personnalité ou reproduire les représentations et le système de raisonnement humain ne suffit pas à garantir que les successions d'actions qui seront réalisées par le robot ne provoqueront pas un non-respect des règles sociales et donc une baisse de la qualité de l'interaction entre l'homme et le robot. La question qui se

pose alors est de savoir quelles caractéristiques du comportement *social* peuvent être considérés au niveau symbolique du planificateur. La planification peut apporter des solutions concrètes à plusieurs des challenges identifiés dans la section 2.1 :

3. **la prédictibilité** : la planification doit fournir au robot des séries d'actions qui forment un comportement global cohérent et sensé pour un observateur humain extérieur,
5. **l'expression d'intentions** : la planification doit fournir au robot une structure permettant de refléter les intentions qui sont associées à la série d'actions qui a été planifiée,
7. **la négociation des buts** : la planification doit fournir les représentations permettant au robot de dialoguer efficacement avec son partenaire humain lorsqu'ils communiquent sur la façon de réaliser les buts courants. Cela implique que si le robot émet une proposition il faut que celle-ci soit cohérente et socialement acceptable,
10. **le contrôle des coûts** : la planification doit intégrer la notion des coûts engendrés par les différentes actions du robot ou de l'homme. Cela permettra au planificateur de comparer la qualité des différents plans produits et donc de proposer à ses partenaires humains des plans efficaces.

2.6 Conclusion

Ce chapitre a permis de montrer que le comportement d'un robot interactif doit être compréhensible par son partenaire humain et que ce critère doit être pris en compte à tous les niveaux de l'architecture logicielle de contrôle du robot. De nombreux travaux se sont portés sur la réalisation des actions, sur l'expression des intentions du robots ou sur le développement de modules de raisonnement facilitant la compréhension du partenaire humain lorsqu'il communique avec le robot. L'ensemble de ces travaux améliorent la qualité de l'interaction homme-robot à un niveau local. Nous avons avancé le fait que la qualité de l'interaction entre l'homme et le robot sera améliorée si les approches locales sont associées à une approche plus globale. Ainsi, la prise en compte du fait que le comportement du robot doit être intelligible par l'homme doit aussi être considérée à un plus haut-niveau et donc être intégrée dans le planificateur symbolique du robot. Nous avons également vu que l'intégration de contraintes sociales dans le système de planification de haut-niveau permettra d'apporter des solutions concrètes à des challenges inhérents aux applications impliquant une coopération efficace entre un homme et un robot, à savoir : la prédictibilité du robot, l'expression des intentions du robot, la gestion des efforts nécessaires à la réalisation des buts et dans une moindre mesure la négociation sur la façon de réaliser les buts courants.

3

Mesure de la qualité sociale d'un plan

Le chapitre 1 nous a permis de justifier l'utilité de doter un robot assistant d'une grande autonomie décisionnelle. Nous avons également établi que la planification délibérative est un outil permettant de concrétiser cette autonomie. Le chapitre 2 nous a permis de mettre en avant que l'impact social du comportement du robot doit être pris en compte à tous les niveaux de contrôle du robot. Afin de permettre au robot d'avoir un comportement global socialement acceptable nous avons choisi d'adapter le système de planification du robot pour qu'il puisse prendre en compte un ensemble de règles sociales.

En robotique, le rôle d'un planificateur délibératif est de produire un plan, i.e. une succession d'actions, permettant au robot d'atteindre ses buts courants. Pour un problème donné il existe de nombreux plans solutions, c'est pourquoi, les planificateurs utilisent couramment une métrique leur permettant de « trier » les plans et de ne retenir que les meilleurs selon les critères composant la métrique. Afin de ne retenir que les plans ayant la meilleure lisibilité pour les partenaires humains du robot, nous avons créé une métrique permettant de mesurer la « qualité sociale » d'un plan. La création de cette métrique s'est déroulée en deux étapes : (1) l'identification des différents critères à prendre en compte et (2) l'élaboration d'un processus d'agrégation de ces critères pour obtenir une estimation numérique.

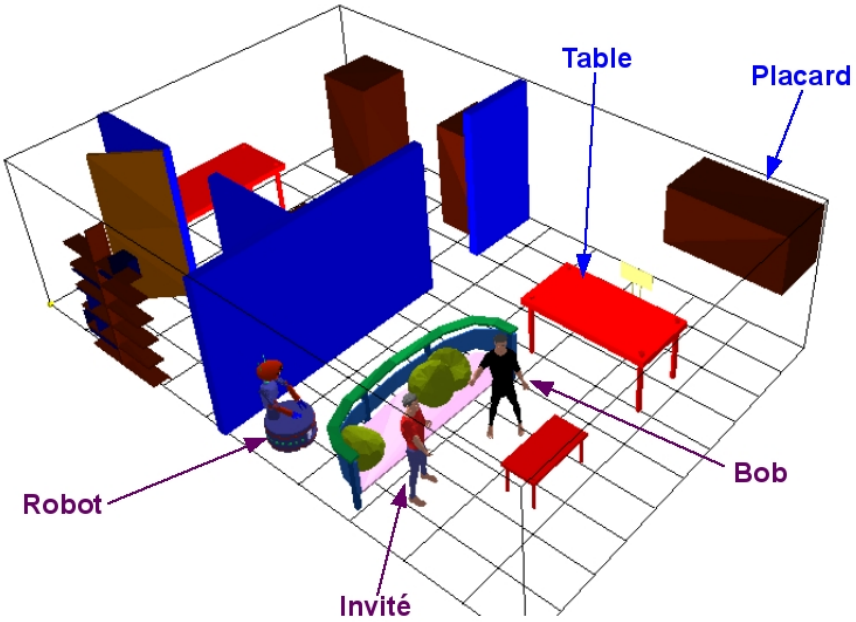

FIG. 3.1 – Situation de départ de l'exemple support
Bob reçoit un invité et souhaite qu'on serve à boire sur la table du salon. Pour cela, il faut y déposer deux verres (qui se trouvent dans le placard) et une bouteille (qui se trouve sur la grande table).

3.1 Exemple support

Afin de mettre en évidence les différents critères à considérer dans la métrique, nous allons utiliser un exemple support. En se basant sur des situations concrètes dans lesquelles pourraient se trouver un robot assistant, nous allons utiliser un environnement similaire à un domicile. Pour cet exemple, nous comparerons plusieurs plans solutions afin d'isoler les critères qui nous permettront de différencier les plans entre eux.

3.1.1 Situation de départ et buts

L'exemple support qui sera utilisé dans la suite de ce chapitre est illustré par la figure 3.1. Ce scénario met en oeuvre un robot interactif appelé « Robot » et deux être humains « Bob » : propriétaire de Robot, et un invité. Dans cette situation Bob souhaite offrir à boire à son invité ce qui se traduit par la nécessité d'apporter la bouteille B à Bob ainsi que de poser deux verres V_1 et V_2 sur la table du salon.

Dans la situation de départ, la bouteille B se trouve sur la grande table et les deux verres V_1 et V_2 sont dans le placard dont la porte est fermée. Robot se trouve à proximité de la porte, Bob et son invité sont tous deux assis sur le sofa. Dans cette situation, il est préférable que l'invité reste assis sur le sofa mais Bob peut participer à la réalisation des objectifs. Pour des raisons de contraintes physiques, Robot ne peut porter, au maximum, que deux objets à la fois.

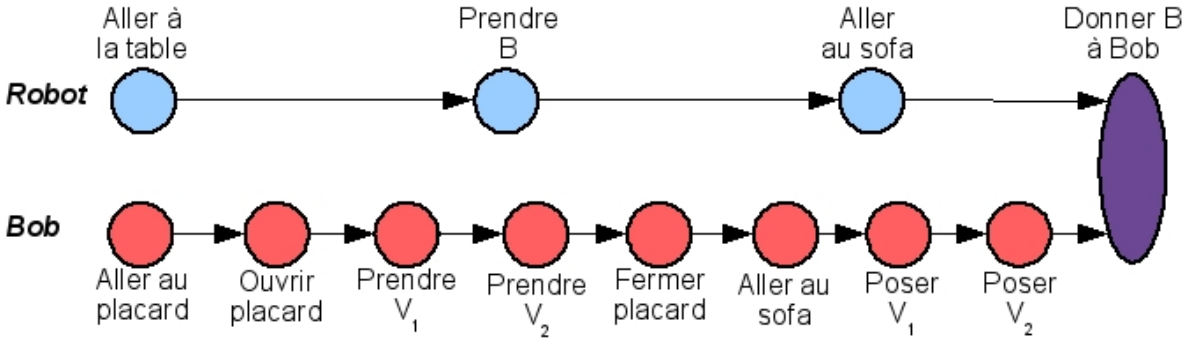

FIG. 3.2 – Exemple de plan solution socialement acceptable.

3.1.2 Plans solutions

Il existe de nombreux plans solutions pour cet exemple selon les agents intervenant dans la réalisation des objectifs et selon l'ordre dans lequel les différents objets sont apportés à leur destination. La figure 3.2 donne un exemple de plan solution qui semble socialement acceptable. En effet, dans ce plan on peut voir que Robot se comporte comme un assistant en apportant son aide dans la réalisation des buts.

Dans toutes les illustrations de plans de ce manuscrit les actions individuelles seront représentées par un cercle sur la ligne d'actions de l'agent correspondant. Une action « jointe » i.e. liée à au moins deux agents sera représentée par un ovale recouvrant les lignes d'actions des agents associés. Par exemple, une action de mouvement n'implique que l'agent qui se déplace, elle est donc représentée par un cercle alors que l'action de donner un objet nécessite deux agents, elle est donc représentée par un ovale.

3.2 Identification des critères de qualité sociale d'un plan

Pour qu'un robot puisse être considéré comme un assistant par ses partenaires humains, il se doit d'agir comme tel et donc d'avoir un comportement adapté à leurs désirs. Il nous a donc fallu déterminer les critères permettant d'identifier les comportements qui peuvent susciter de la gêne ou de l'incompréhension chez les partenaires humains du robot. Les critères que nous avons identifiés sont :

- ✓ la gestion des efforts,
- ✓ les états indésirables,
- ✓ les séquences indésirables,
- ✓ les temps d'inactivité,
- ✓ les liens croisés,
- ✓ les mauvaises décompositions.

Afin d'illustrer ces différents critères, nous allons comparer des plans solutions de l'exemple support à celui de la figure 3.2. Nous analyserons les comportements du robot associés à chacun

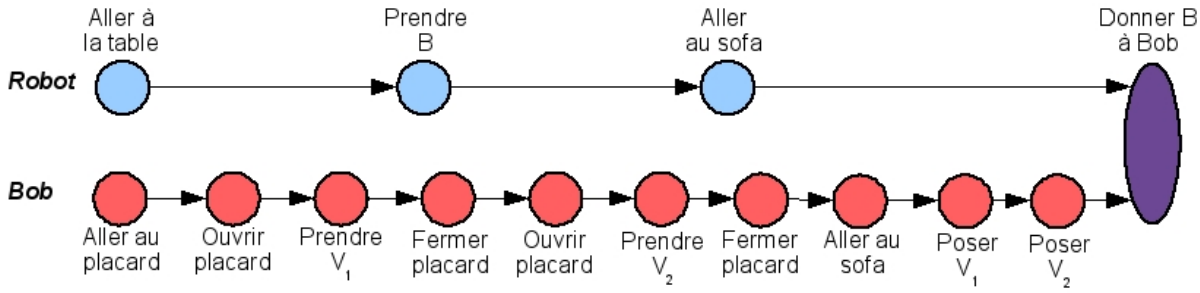

FIG. 3.3 – Exemple de plan solution : plan nécessitant des efforts inutiles de Bob.
Ce plan contient une fermeture suivie immédiatement d'une ouverture du placard. Cet ajout inutile d'actions provoque une perte d'efficacité globale.

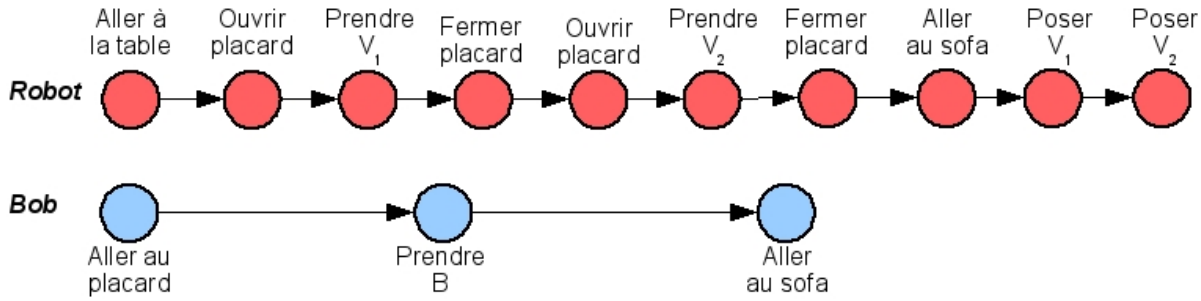

FIG. 3.4 – Exemple de plan solution : plan nécessitant des efforts inutiles de Robot.
Ce plan contient une fermeture suivie immédiatement d'une ouverture du placard. Cet ajout inutile d'actions provoque une perte d'efficacité globale et une baisse de la lisibilité du comportement du robot.

des plans comparés pour en déduire ceux qui ne semblent pas naturels (challenge 3) et/ou qui engendrent une interrogation sur les intentions du robot (challenge 5).

3.2.1 Critère de gestion des efforts

Considérons le plan solution illustré par la figure 3.3. Ce plan permet effectivement d'atteindre les buts fixés mais semble avoir une qualité sociale moindre que celui de la figure 3.2. En effet, les êtres humains lorsqu'ils planifient inconsciemment des actions le font de manière « optimale » i.e. de façon à « économiser » les efforts à fournir pour atteindre les buts fixés. On peut constater que dans ce plan la fermeture et l'ouverture du placard entre la prise de V_1 et V_2 ne fait qu'augmenter la somme d'efforts à fournir par Bob pour atteindre les buts (manquement au challenge 10).

Ce plan soulève un problème majeur pour la phase de négociation précédant l'exécution du plan. En effet si, lors de cette phase de négociation, Robot propose à Bob le plan de la figure 3.3, Bob va instinctivement détecter une non-optimalité grossière et ressentir une incompréhension des intentions de Robot (manquement au challenge 5). La prise en compte des efforts globaux à fournir lors de la réalisation du plan permet également d'éviter de retenir des plans comme celui illustré par la figure 3.4. Ce plan est très similaire à celui de la figure 3.3 sauf que les rôles de Robot et de Bob sont inversés. Du point de vue de la phase de négociation, ce plan ne pose pas de problème immédiat étant donné que le « flux » d'actions effectuées par Bob ne présente pas

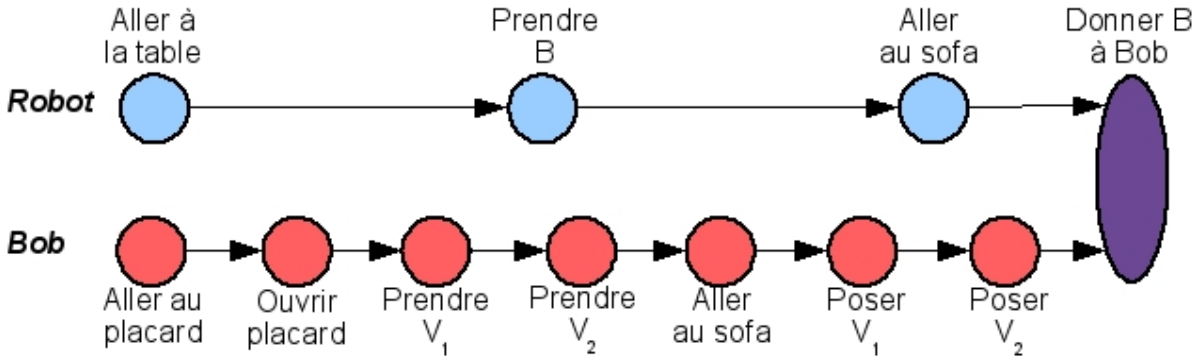

FIG. 3.5 – Exemple de plan solution : plan contenant un état indésirable persistant.
Ce plan contient une ouverture du placard mais aucune fermeture, le placard va donc rester ouvert lorsque le plan aura été exécuté.

de non-optimalité flagrante. Toutefois, ce plan risque de poser problème lors de son exécution. En effet, si un partenaire humain de Robot l'observe lorsqu'il referme le placard pour le rouvrir immédiatement après, le partenaire humain risque, là encore, de ressentir de la confusion et donc d'attribuer de fausses intentions à Robot (manquement au challenge 5).

La prise en compte des efforts nécessaires à l'exécution d'un plan permet d'assurer l'efficacité de la réalisation des différentes tâches. Le critère de gestion des efforts vient en complément de ce contrôle de l'efficacité. En effet, il permettra de diminuer la qualité sociale de plans contenant une(des) non-optimalité flagrante(s) évitant ainsi de susciter un sentiment d'incompréhension chez les partenaires humains du robot.

3.2.2 Critère des états indésirables

La prise en compte des efforts de réalisation permet « d'écarter » les plans présentant des non-optimalités flagrantes, toutefois, il arrive que certaines non-optimalités soient nécessaires. Par exemple, le plan solution illustré par la figure 3.5 nécessite moins d'efforts que le plan solution socialement acceptable de la figure 3.2 mais présente l'inconvénient de laisser la porte du placard ouverte. Il peut s'avérer préférable de favoriser des plans dans lesquels le placard ne reste pas ouvert inutilement. Nous avons donc choisi d'ajouter un nouveau critère basé sur la notion d'états indésirables. Ces états sont à différencier des états interdits. Lors d'un processus de planification, si un état interdit est détecté dans le plan en construction la branche courante de l'espace de recherche est coupée et le planificateur réalise une opération de backtrack. Dans notre approche les états indésirables permettent de diminuer la qualité sociale des plans dans lesquels ils sont présents mais n'engendrent pas de coupures dans l'espace de recherche. Ainsi, dans notre exemple le fait que le placard soit ouvert dans le déroulement du plan n'est pas dérangeant mais c'est le fait qu'il reste trop longtemps ouvert qui pose un problème. Il apparaît donc que la persistance d'un état indésirable puisse faire baisser la qualité sociale d'un plan. Il existe également des cas où la simple apparition, même pour une durée très courte, d'un état indésirable pose des problèmes. On peut, par exemple, imaginer un plan dans lequel Robot tiendrait simultanément,

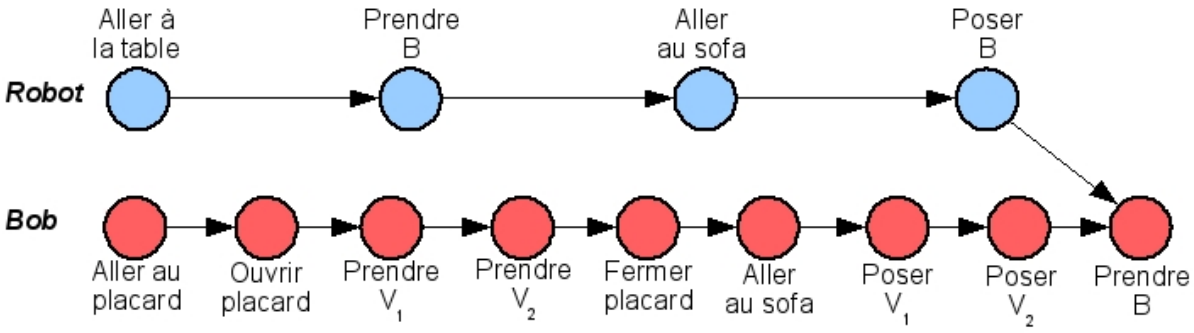

FIG. 3.6 – Exemple de plan solution : plan contenant des séquences indésirables.
Dans ce plan le robot transmet la bouteille à Bob grâce à une combinaison d'actions « Déposer-Prendre » alors qu'il aura été préférable que le robot lui la donne en mains propres.

pendant quelques instants, un chiffon à poussière et des denrées alimentaires ce qui n'est pas acceptable d'un point de vue hygiénique et donc inacceptable d'un point de vue social.

Le principe des états indésirables a déjà été utilisé en planification de tâches mais pour des objectifs différents. Le planificateur TalPlanner [Kvarnstrom 01] planifie dans l'espace d'états et utilise des expressions temporelles pour définir des règles permettant de diminuer l'espace de recherche. Ces règles sont dépendantes du domaine de planification. Ainsi, si lors de la planification le système atteint un état particulier étant donné les actions qu'il reste à effectuer la branche courante de l'arbre d'exploration sera coupée. Pour le planificateur TalPlanner les états indésirables sont utilisés pour couper les branches considérées comme non prometteuses, il s'agit donc davantage d'états interdits. Dans le cadre d'une estimation de la qualité sociale d'un plan, les états indésirables seront utilisés comme une heuristique permettant de favoriser l'exploration de branches aboutissant à des plans ne contenant pas les états non désirés.

En conclusion, nous avons choisi de compléter le critère de gestion des efforts avec un autre critère basé sur la notion d'états indésirables. En effet, ce dernier critère permettra de limiter la qualité sociale de plans dans lesquels apparaissent et, éventuellement, perdurent des états particuliers qui ne sont pas souhaitables évitant ainsi de perturber les habitudes quotidiennes des partenaires humains du robot.

3.2.3 Critère des séquences indésirables

La prise en compte des efforts de réalisation peut provoquer l'apparition ou la persistance d'états indésirables mais peut également provoquer l'apparition de combinaisons d'actions suscitant des sentiments d'incompréhension chez les partenaires humains du robot. Un exemple d'un telle situation est illustré par la figure 3.6. Dans cet exemple, on peut voir que Robot transmet un objet à son partenaire humain en le déposant sur la table forçant ce dernier à s'en saisir. Il se peut que cette situation soit socialement inacceptable notamment si le partenaire humain du robot est en train de patienter pendant que le robot lui apporte l'objet. En effet, il vaut mieux dans cette situation que le robot donne l'objet à son partenaire humain en mains

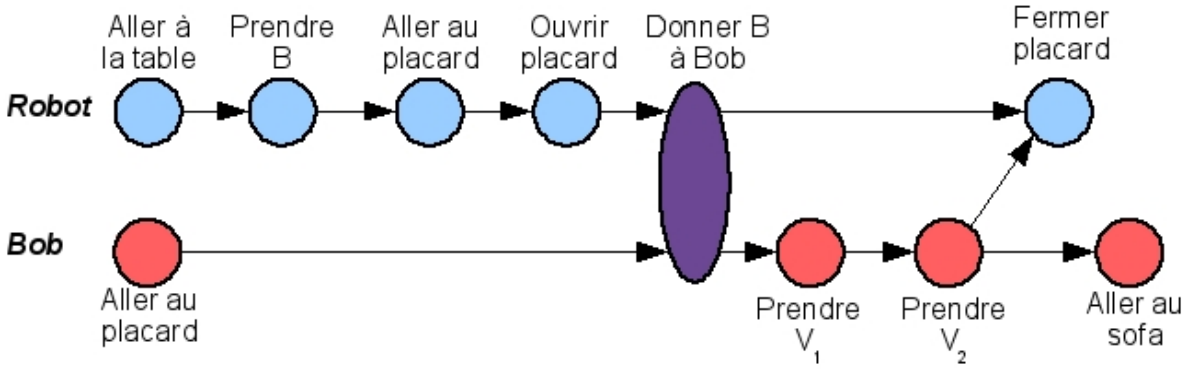

FIG. 3.7 – Exemple de plan solution : plan contenant des temps morts.
Dans ce plan Bob est contraint de rester inactif et de patienter longuement avant que le robot ne lui donne la bouteille.

propres même si cela demande plus d'efforts de réalisation que de le poser sur la table pour que le partenaire humain s'en saisisse.

De plus, le critère des séquences indésirables permet également de contribuer à évaluer encore plus négativement la qualité sociale du plan solution illustré par la figure 3.3. En effet, il est tout à fait envisageable de qualifier de « séquence indésirable » le fait qu'un agent ferme la porte d'un meuble puis l'ouvre aussitôt après. De cette façon, un plan dans lequel cette combinaison d'actions apparaît sera défavorisé au profit de plans excluant ce type de combinaisons.

En conclusion, nous avons choisi d'introduire dans l'estimation de la qualité sociale d'un plan un critère basé sur les séquences indésirables. Ceci permettra de limiter la qualité sociale de plans dans lesquels apparaissent une(des) combinaison(s) d'actions particulières pouvant susciter l'incompréhension des partenaires humains du robot.

3.2.4 Critère des temps d'inactivité

Les soucis d'optimalité lors de la conception du plan évoqués dans la section 3.2.1 doivent également être pris en compte lors de l'exécution du plan. En effet, un plan tel que celui illustré par la figure 3.7 contient un risque important de voir des temps d'inactivité prolongés entre deux actions successives d'un même agent. Ces temps d'inactivité appelés « temps morts » sont préjudiciables à la qualité sociale du plan. En effet, si ces temps morts apparaissent dans le flux d'actions des partenaires humains du robot, ceux-ci peuvent avoir l'impression de perdre leur temps. Le plan retenu doit donc limiter au maximum les temps morts pour les partenaires humains afin d'éviter l'apparition d'un sentiment d'impatience.

Les temps morts dans le flux d'actions du robot peuvent également être préjudiciables. Dans le cas où le robot est contraint d'attendre un évènement particulier pour pouvoir exécuter sa prochaine action, il peut véhiculer une image d'inutilité auprès des éventuels observateurs extérieurs qui seraient présents. Ceci est notamment dû au fait que le robot ne possède pas

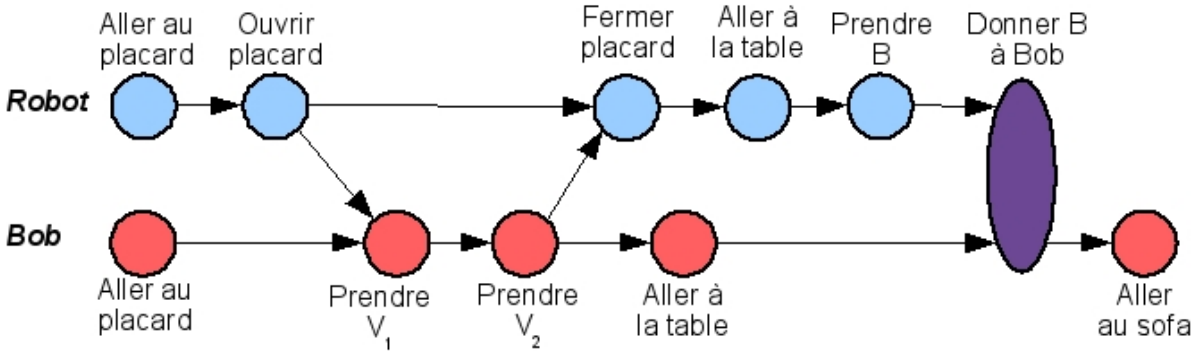

FIG. 3.8 – Exemple de plan solution : plan contenant des liens croisés.
Dans ce plan il existe trop de dépendances entre les actions de Bob et celles du robot ce qui rend son exécution plus difficile.

de moyens expressifs aussi complets qu'un être humain. Ainsi, lorsqu'un robot est en phase d'attente et que celle-ci se prolonge dans le temps, un observateur extérieur pourrait parfaitement interpréter cette attitude comme le fait que le robot a perdu ses repères et ne sait plus quoi faire.

En conclusion, nous avons choisi de compléter l'estimation de la qualité sociale d'un plan grâce au critère des temps d'inactivité. L'introduction de ce critère permettra de favoriser des plans excluant des temps morts trop importants évitant ainsi de véhiculer une image d'inactivité du robot et/ou de susciter l'impatience des partenaires humains du robot.

3.2.5 Critère des liens croisés (dépendance intrinsèque du plan)

Lorsque plusieurs êtres humains ont différentes tâches à réaliser et qu'ils doivent se les répartir, ils essaient instinctivement de le faire de manière à ce que chaque agent puisse exécuter ces tâches le plus indépendamment possible des autres agents. La figure 3.8 illustre un plan solution de notre exemple dans lequel on peut constater que cette notion d'indépendance n'est pas du tout considérée. Ce plan contient de nombreux liens dits « croisés » i.e. il existe beaucoup de liens causaux entre les actions de Bob et de Robot et réciproquement.

La présence de ces nombreux liens rend l'exécution du plan très difficile puisqu'à chaque instant il faut que Bob conserve un oeil sur les activités de Robot afin de déterminer quand lui-même doit agir. Ainsi le partenaire humain du robot peut se sentir « emprisonné » dans un cycle à exécuter. Cela crée une grande dépendance des partenaires humains vis-à-vis du robot. Dans la mesure du possible, il est préférable d'éviter ce type de situations et donc de privilégier des plans dans lesquels chaque agent peut exécuter une série d'actions sans interférer dans le flux d'actions des autres agents comme cela est le cas dans le plan illustré par la figure 3.2.

De plus, des plans contenant de nombreux liens croisés sont fragiles lors de leur exécution par le robot. En effet, si une action du robot ne peut être exécutée qu'à la suite d'une action de son partenaire humain, le robot va devoir focaliser toute son attention (i.e. ses capacités de

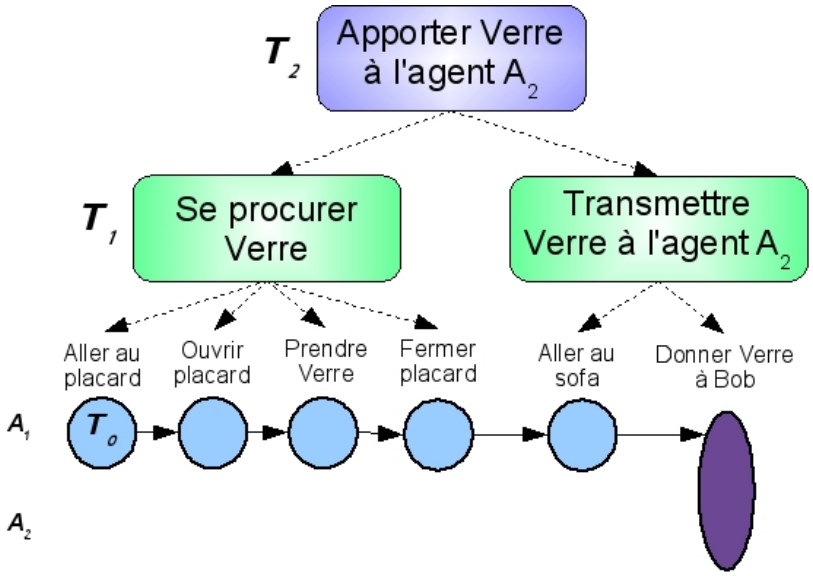

FIG. 3.9 – Exemple de graphe d'intentions.
Dans cet exemple, les quatre premières actions de l'agent A_1 sont réalisées avec l'intention que A_1 obtienne le verre. Cette dernière intention répond à l'intention « supérieure » d'apporter le verre à l'agent A_2.

perception) sur le bon déroulement de l'action de son partenaire et se retrouve donc exposé au risque d'une erreur de perception. De même, si une action du partenaire humain dépend directement de l'exécution d'une action du robot, ce dernier se devra d'agir « rapidement » afin de limiter l'apparition des temps morts évoqués dans la section 3.2.4. Cette obligation d'action rapide peut poser des problèmes pour une action robotique complexe à exécuter. Il vaut donc mieux privilégier des plans dans lesquels ces situations à risque n'apparaissent pas.

Nous avons donc choisi d'introduire un nouveau critère afin de favoriser les plans limitant le nombre de liens croisés i.e. le nombre de liens causaux ayant pour origine une action d'un agent et pour extrémité une action dans laquelle est impliqué un autre agent. Ce critère permettra de limiter les plans engendrant des sentiments de dépendance chez les partenaires humains du robot et de limiter les plans dont l'exécution peut s'avérer fragile.

3.2.6 Critère des mauvaises décompositions

Les critères précédemment établis ont été justifiés par le fait qu'un comportement irrationnel (ou jugé comme tel) du robot peut susciter l'incompréhension de ses partenaires humains. Ces sentiments sont préjudiciables à l'image du robot car si ses partenaires humains sont amenés à s'interroger sur ses intentions il se pourrait que le statut « d'assistant » du robot soit remis en cause. Il convient donc de s'assurer que le comportement du robot transcrit bien ses intentions. Une question se pose alors : comment représenter les intentions du robot ?

La question de la représentation des intentions a déjà été abordée dans plusieurs travaux portant sur des applications diverses. On peut notamment citer les travaux sur la modélisation

de la composante intentionnelle du dialogue tels que la théorie des plans partagés (voir chapitre 1). En se basant sur cette théorie, on peut construire un graphe de recettes [Lochbaum 98] permettant de représenter les intentions de chaque agent en fonction des tâches dans lesquelles il est impliqué. Ainsi, comme l'illustre la figure 3.9, si un agent A_1 doit réaliser l'action T_0 « Aller au placard » il le fera dans le cadre de la tâche T_1 de niveau supérieur « Se procurer Verre » qui est elle-même liée à la tâche T_2 de niveau supérieur « Apporter Verre à l'agent A_2 ».

Une structure hiérarchique similaire a également été utilisée en robotique pour le contrôle d'exécution des tâches du robot. On peut par exemple citer le langage OpenPRS [Ingrand 96]. Ce langage permet au robot de contrôler le bien fondé de ses tâches à différents niveaux d'abstraction. Par exemple dans le cas illustré par la figure 3.9 l'agent A_1 peut effectuer en permanence les contrôles associés aux tâches T_0 à T_2. Ainsi, s'il apparaît dans la base de faits de l'agent A_1 que l'agent A_2 est entré en possession du verre, la tâche T_2 devient inutile et on peut ainsi supprimer la partie inférieure de la hiérarchie qui y est associée.

Un robot interactif doit être capable à tout moment de justifier ces activités courantes et de dialoguer sur ses intentions. De même, il doit être capable de faire preuve de réactivité en contrôlant en permanence le bien fondé de ses actions. Ainsi, il convient de doter le robot d'une structure hiérarchique telle que celles évoquées précédemment. Cela implique donc qu'il faut transformer la structure du plan pour y ajouter une dimension supplémentaire : les tâches de haut-niveau. Ainsi la définition du plan comme un ensemble d'actions partiellement ordonnées peut être étendue pour devenir une structure arborescente dont les feuilles sont partiellement ordonnées.

Ainsi le plan de la figure 3.2 peut être complétée pour donner le plan à structure arborescente de la figure 3.10. Dans ce plan on peut constater que l'action de *Donner B à Bob* est réalisée dans le cadre de la tâche *Transmettre B à Bob*. Il se peut que la structure arborescente d'un plan contienne des liens hiérarchiques reflétant des intentions floues comme par exemple le plan de la figure 3.11. Ce dernier peut paraître d'une bonne qualité sociale si on observe les actions constituées par les feuilles de la structure arborescente, toutefois, on peut constater que l'action *Donner B à Bob* est maintenant associée à la tâche *Se débarrasser de B*. Cela signifie que les tâches de haut-niveau peuvent influencer la façon dont le robot va réaliser les actions associées à ces tâches. En effet, dans le cadre du plan de la figure 3.10 le robot réalisera l'action *Donner B à Bob* et considérera que cette action est achevée lorsqu'il pourra constater que Bob est bien en possession de B alors que dans le cadre du plan de la figure 3.11 l'action *Donner B à Bob* sera considérée comme achevée dès que B quittera la pince du robot. Les tâches de haut-niveau définissent donc le contexte dans lequel les actions du robot vont être exécutées et peuvent donc influencer la lisibilité du comportement du robot.

Nous avons donc choisi d'introduire un nouveau critère pour détecter les intentions floues dans la structure arborescente du plan afin de favoriser des plans assurant un dialogue de meilleure

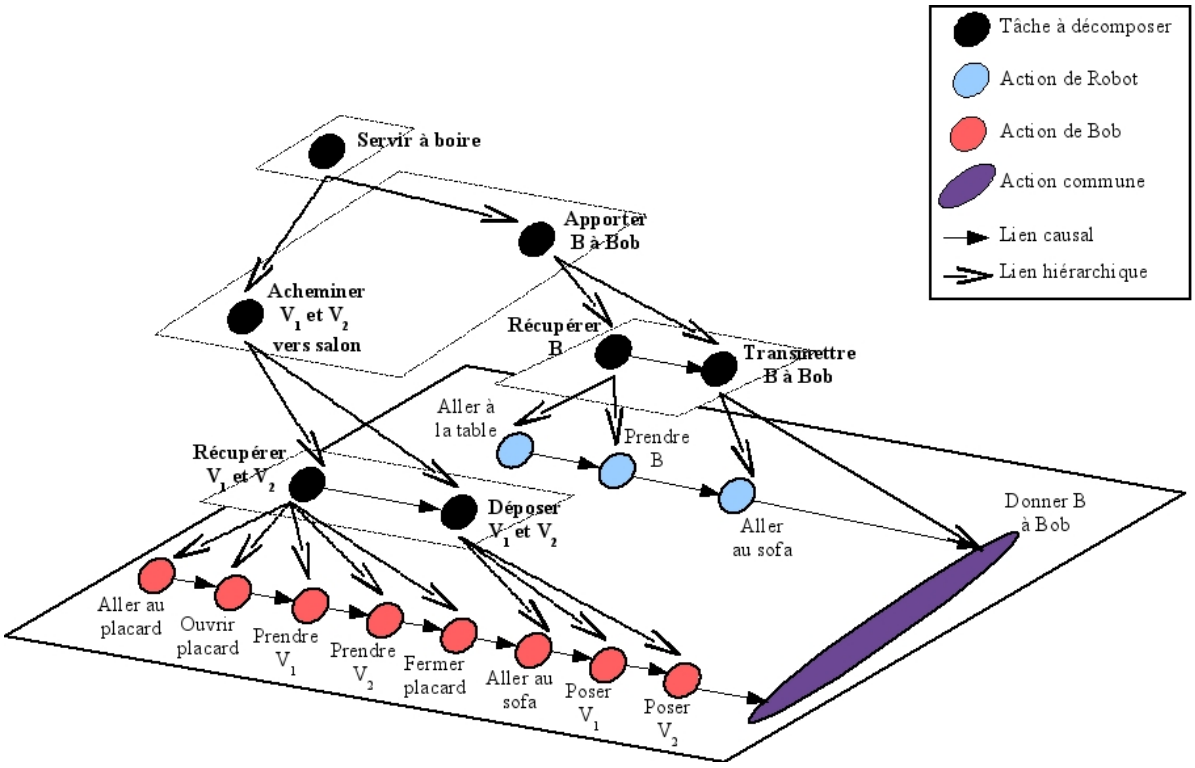

FIG. 3.10 – Exemple de plan solution : plan socialement acceptable avec sa structure arborescente. Dans ce plan on peut constater que l'action *Donner B à Bob* est réalisée dans le cadre de la tâche de plus haut niveau *Transmettre B à Bob* ce qui est cohérent avec la tâche de niveau supérieur *Apporter B à Bob*.

qualité et garantissant un contrôle d'exécution plus adapté. Ce nouveau critère est celui des mauvaises décompositions.

3.3 Définitions pour la planification avec contraintes sociales

Les critères que nous venons d'établir permettent d'associer à un comportement irrationnel du robot une structure partielle aisément détectable dans la structure d'un plan complet. Il nous faut donc étendre la définition d'un domaine de planification standard pour y insérer les descriptions des critères identifiés.

3.3.1 Définition d'un domaine de planification

Un domaine classique de planification délibérative est composé d'un ensemble d'*opérateurs* correspondant aux actions de bases. Un opérateur O est défini comme un n-uplet $<E_O,P,M,C>$ où :

– E_O est l'entête donnant le nom et la liste des paramètres de l'opérateur O,

– P est la liste de préconditions (définies à partir des paramètres de l'opérateur) décrivant les états de la base de faits dans lesquels l'opérateur O est applicable,

– M est la liste de modifications (définies à partir des paramètres de l'opérateur) à apporter à la base de faits lorsque l'opérateur O est appliqué,

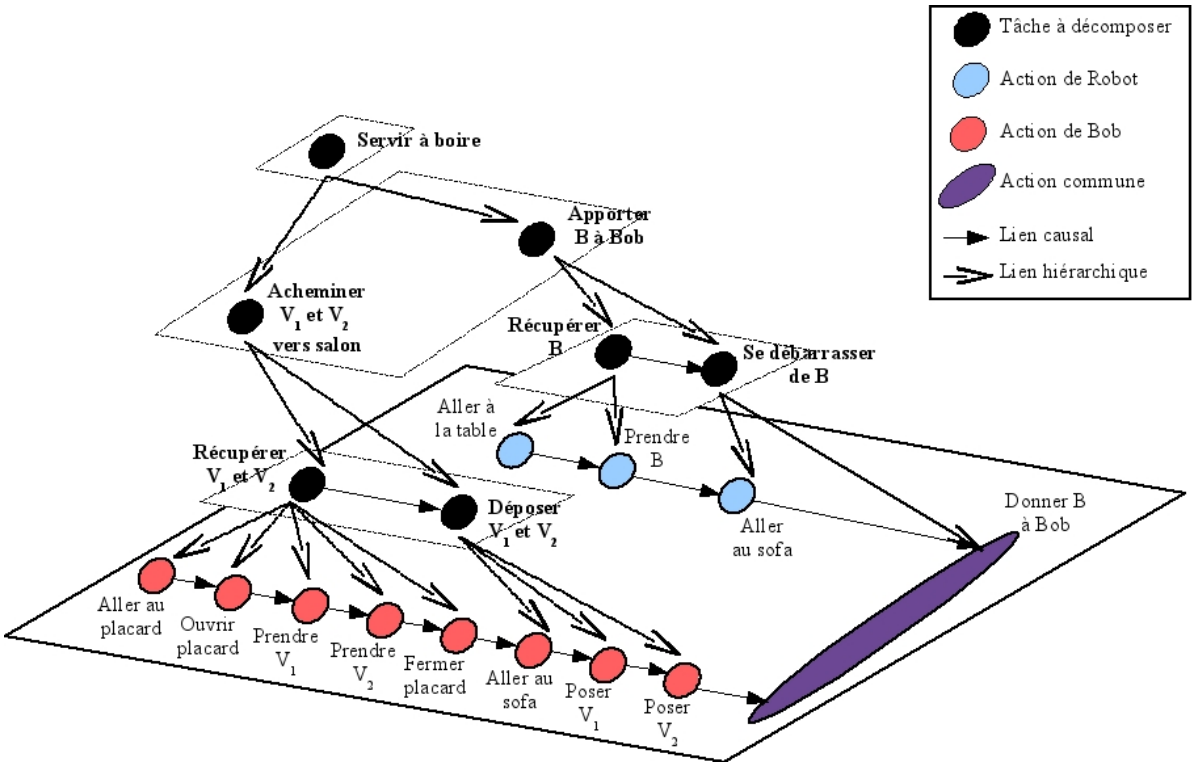

FIG. 3.11 – Exemple de plan solution : plan avec une structure arborescente traduisant de mauvaises intentions du robot.
Dans ce plan on peut constater que l'action *Donner B à Bob* est réalisée dans le cadre de la tâche de plus haut niveau *Se débarrasser de B* ce qui est incohérent avec l'intention liée à la tâche de niveau supérieur *Apporter B à Bob*.

– C est le coût permettant de quantifier la difficulté à réaliser l'action associée (ressources consommées, temps nécessaire, etc.). Le coût est calculé à partir des données liées aux paramètres de l'opérateur.

Selon le formalisme retenu, un domaine de planification délibérative peut également contenir un ensemble de *méthodes* correspondant à des tâches de haut-niveau. Une méthode T est définie comme un n-uplet $<E_T,\{P_u, D_u\}>$ où :

– E_T est l'entête donnant le nom et la liste des paramètres de la méthode T,

– $\{P_u, D_u\}$ est un ensemble de couples préconditions-décomposition pour la méthode T. Si les préconditions P_u (définies à partir des paramètres de T) sont vérifiées étant donné l'état courant du monde alors la méthode T peut être réalisée en achevant la liste partiellement ordonnée de tâches et/ou d'opérateurs D_u.

Ainsi un domaine classique de planification délibérative peut être défini comme un couple $<\{O_i\},\{T_j\}>$ où $\{O_i\}$ est un ensemble d'opérateurs et $\{T_j\}$ est un ensemble de méthodes.

3.3.2 Extension de la définition d'un domaine de planification

Dans le cas d'une planification où la qualité sociale du plan jouera un rôle important, le domaine va être complété afin d'y intégrer la description des structures associées aux critères identifiés dans la section 3.2. Ainsi un domaine de planification délibérative avec prise en compte de la qualité sociale du plan peut être défini comme un n-uplet :

$$< \{O_i\}, \{T_j\}, \{St_k\}, \{Seq_l\}, \{TM_m\}, \{CL_n\}, \{D_p\} >$$

Où :

– $\{O_i\}$ est un ensemble d'opérateurs. Les coûts des opérateurs seront utilisés pour quantifier les efforts nécessaires à la réalisation des actions qu'ils représentent. Chaque opérateur ajouté dans le plan est associé à un ensemble non vide d'agents. On définit ainsi les actions *simples* qui sont associées à un seul agent et les actions *jointes* qui sont associées à au moins deux agents. Par exemple, l'action de Robot *Aller à la table* sera une action simple attribuée à Robot alors que l'action de Robot *Donner B à Bob* sera une action jointe attribuée à Robot et à Bob.

– $\{T_j\}$ est un ensemble de méthodes.

– $\{St_k\}$ est un ensemble d'états indésirables. Un état indésirable étant défini comme une conjonction de faits qui doivent être vrais au même instant.

– $\{Seq_l\}$ est un ensemble de séquences indésirables. Une séquence indésirable Seq_l est définie comme un couple $<O_{Seq_l}, L_{Seq_l}>$ où O_{Seq_l} est un ensemble d'opérateurs et L_{Seq_l} un ensemble de liens définissant un ordre partiel entre les opérateurs de O_{Seq_l}.

– $\{TM_m\}$ est un ensemble décrivant les temps d'inactivité préjudiciables à la qualité sociale du plan. Un temps d'inactivité est défini comme un couple $<A_m, S_{A_m}>$ où S_{A_m} est le seuil minimum définissant un temps d'inactivité trop important pour l'agent A_m.

– $\{CL_n\}$ est un ensemble décrivant les liens croisés pouvant agir sur la qualité sociale du plan. Un lien croisé est défini comme un couple $<A_{S_n}, A_{D_n}>$ où A_{S_n} est un ensemble d'agents participant à l'action à la source du lien et A_{D_n} est un ensemble d'agents participant à l'action à l'extrémité du lien. Pour que le lien soit considéré comme croisé, il faudra vérifier la contrainte suivante : $\exists\ a \in A_{D_n}$ tel que $a \notin A_{S_n}$.

– $\{D_p\}$ est un ensemble de mauvaises décompositions. Une mauvaise décomposition est définie comme un triplet $<N_p, CL_p, HL_p>$ où N_p est un ensemble de méthodes et/ou d'opérateurs, CL_p est un ensemble de liens de précédence entre les membres de N_p et HL_p est un ensemble de liens hiérarchiques entre les membres de N_p.

L'ensemble des critères décrits dans les ensembles $\{St_k\}$, $\{Seq_l\}$, $\{TM_m\}$, $\{CL_n\}$ et $\{D_p\}$ forment ce que nous appellerons dorénavant les *règles sociales* du domaine de planification. Une règle sociale sera considérée comme violée lorsque la structure qui lui est associée sera présente dans la structure globale du plan.

3.3.3 Définition d'un problème de planification

Un problème classique de planification délibérative est un triplet $< S_W, G, M >$ où :
- S_W est un ensemble de faits décrivant l'état initial du monde,
- G est l'ensemble des buts à réaliser ou l'ensemble des tâches à accomplir,
- M est une métrique permettant de comparer les plans entre eux et de ne conserver que les plus prometteurs.

La métrique M est optionnelle et n'apparaît que lorsque le problème consiste à trouver des plans qui doivent être optimisés selon des critères définis.

3.3.4 Extension de la définition d'un problème de planification

Dans le cas d'une planification où la qualité sociale du plan jouera un rôle important, la description du problème va être complétée afin d'y intégrer les fonctions d'évaluations propres aux différents critères qui ont été identifiés dans la section 3.2. Ainsi un problème de planification délibérative avec prise en compte de la qualité sociale du plan peut être défini comme un n-uplet

$$< S_W, G, E, \{C_{St_k}\}, \{C_{Seq_l}\}, \{C_{TM_m}\}, \{CL_n\}, \{D_p\}, M >$$

Où :
- S_W est un ensemble de faits décrivant l'état initial du monde.
- G est l'ensemble des buts à réaliser ou l'ensemble des tâches à accomplir.
- E est la fonction d'évaluation associée à la gestion des efforts. Cette fonction permettra de fusionner les coûts des opérateurs présents dans le plan courant.
- $\{C_{St_k}\}$ est l'ensemble des fonctions d'évaluation associées aux états indésirables. Chaque état indésirable St_k a une fonction d'évaluation associée C_{St_k}.
- $\{C_{Seq_l}\}$ est l'ensemble des fonctions d'évaluation associées aux séquences indésirables. Chaque séquence indésirable Seq_l a une fonction d'évaluation associée C_{Seq_l}.
- $\{C_{TM_m}\}$ est l'ensemble des fonctions d'évaluation associées aux temps d'inactivité. Chaque temps d'inactivité TM_m a une fonction d'évaluation associée C_{TM_m}.
- $\{C_{CL_n}\}$ est l'ensemble des fonctions d'évaluation associées aux liens croisés. Chaque lien croisé CL_n a une fonction d'évaluation associée C_{CL_n}.
- $\{C_{D_p}\}$ est l'ensemble des fonctions d'évaluation associées aux mauvaises décompositions. Chaque mauvaise décomposition D_p a une fonction d'évaluation associée C_{D_p}.
- M est une métrique permettant d'agréger les fonctions d'évaluation associées aux différents critères afin de comparer les plans entre eux et de ne conserver que les plus prometteurs.

La métrique M est, cette fois-ci, obligatoire et se basera sur la fonction d'évaluation E ainsi que sur les évaluations provenant des ensembles $\{C_{St_k}\}$, $\{C_{Seq_l}\}$, $\{C_{TM_m}\}$, $\{CL_n\}$, $\{D_p\}$ pour pouvoir estimer la qualité sociale d'un plan courant.

		Critères				
		C_1	C_2	C_3	...	C_N
Alternatives	A_1	a_{11}	a_{12}	a_{13}	...	a_{1N}
	A_2	a_{21}	a_{22}	a_{23}	...	a_{2N}
	...	...	...	...	...	...
	A_M	a_{M1}	a_{M2}	a_{M3}	...	a_{MN}
Poids relatifs		W_1	W_2	W_3	...	W_N

TAB. 3.1 – Matrice de décision pour une analyse à N critères portant sur M alternatives. Dans cette matrice a_{ij} correspond au coût ou au profit lié à la valeur du critère j pour l'alternative i.

3.4 Processus d'agrégation des critères

En planification, les métriques peuvent servir à « trier » les plans solutions afin de privilégier les plans les plus prometteurs. Les métriques sont souvent composées de plusieurs critères. Ces critères sont agrégés afin de produire une évaluation numérique pour chaque plan, cette évaluation permet alors de ramener l'ensemble des plans solutions sur une échelle numérique et ainsi de les trier du moins intéressant au plus prometteur. La section 3.2 nous a permis d'identifier les critères à considérer lors d'une planification avec prise en compte de la qualité sociale du plan. Il reste cependant à trouver la façon de les agréger pour pouvoir créer une métrique reflétant la qualité sociale des plans.

3.4.1 Analyse multi-critère

La comparaison de deux plans sur un critère donné est relativement simple. Cependant, dans le cadre d'une planification avec prise en compte de la qualité sociale du plan, cette comparaison va porter sur l'ensemble des critères définis dans la description du domaine. Il va donc falloir choisir entre plusieurs plans solutions en se basant sur divers critères de décision simultanément. Ce type de décision relève donc de l'analyse multi-critère.

Le principe de l'analyse multi-critère est d'agréger plusieurs critères afin de prendre une décision basée sur des objectifs parfois contradictoires. L'application d'une analyse multi-critère portant sur la sélection d'une solution parmi un ensemble se fait selon les étapes suivantes :

1. Dresser la liste des alternatives (solutions) possibles,
2. Identifier les critères à prendre en compte dans la décision,
3. Comparer les solutions selon chaque critère indépendamment les uns des autres,
4. Agréger les jugements de l'étape 3 pour déterminer la meilleure solution.

Les divergences entre les différentes méthodes d'analyse multi-critère se situent majoritairement dans la réalisation de l'étape 4. L'étape 3 varie parfois quelque peu d'une technique à l'autre selon que le jugement est qualitatif ou quantitatif. Dans la plupart des méthodes d'analyse

multi-critère, l'importance relative des critères est concrétisée par l'utilisation de poids dans une combinaison linéaire. On appelle « matrice de décision » le tableau illustré par le tableau 3.1 dans lequel les a_{ij} représentent les coûts ou les profits associés à l'alternative i selon le critère j. Nous nous intéresserons dans la suite à des matrices contenant uniquement des coûts. Une matrice de décision est une matrice de dimension $M \times N$ où M est le nombre d'alternatives et N le nombre de critères.

La méthode d'analyse multi-critère la plus employée est la méthode de la somme pondérée (*Weight Sum Method* en anglais). Elle est basée sur le principe d'une utilité additive pour chaque critère. Pour déterminer la solution à retenir on cherche l'alternative i telle que :

$$A_{WSM} = \min_{i} \sum_{j=1}^{N} a_{ij} \times W_j \text{ pour i = 1 à M}$$

L'hypothèse d'additivité des utilités est souvent restrictive. Cette difficulté peut-être contournée en ramenant les coûts de la matrice de décision sur des échelles relatives. Dans le cadre de l'estimation de la qualité sociale d'un plan, les comparaisons sont souvent qualitatives : un plan Pl_1 est préféré à un plan Pl_2 sans pour autant pouvoir déterminer précisement une estimation numérique reflétant cette préférence. Cette contrainte rend difficilement utilisable toute méthode d'analyse multi-critère basée sur un dimensionnement numérique attaché à chaque critère. Il nous faut donc nous tourner vers une méthode permettant de combiner des estimations qualitatives. La méthode que nous avons retenue est la méthode du « Processus d'Analyse Hiérarchique » (*Analytic Hierarchical Process* en anglais).

3.4.2 Processus d'Analyse Hiérarchique (AHP)

Ce processus a été développé dans les années 70 par Thomas Saaty. Son principe d'application est détaillé dans [Saaty 99] et [Saaty 00]. Cette méthode générique a été conçue pour faciliter la prise de décision basée sur plusieurs critères (parfois intangibles) dans des environnements complexes. Cette méthode a été déclinée sous de nombreuses formes et est basée sur une solide base mathématique qui est explicitée dans [Forman 01].

Le processus de décision AHP permet de modéliser un problème d'optimisation multi-critère qualitatifs sous la forme d'une hiérarchie comme l'illustre la figure 3.12. Cette structure hiérarchique peut éventuellement contenir des niveaux supplémentaires si un critère est amené à se décomposer en une combinaison de sous-critères. Cet aspect n'intervient pas dans notre métrique, il ne sera donc pas détaillé ici. Le principe de la méthode AHP est d'obtenir pour chaque alternative une valeur numérique représentant le niveau de préférence de celle-ci par rapport aux autres alternatives. Cette valeur numérique est appelée « désirabilité ». Après utilisation du processus AHP l'alternative finalement retenue sera celle dont la désirabilité sera la plus élevée. L'application de la méthode AHP s'effectue en opérant les étapes suivantes :

1. Déterminer toutes les combinaisons binaires entre critères,

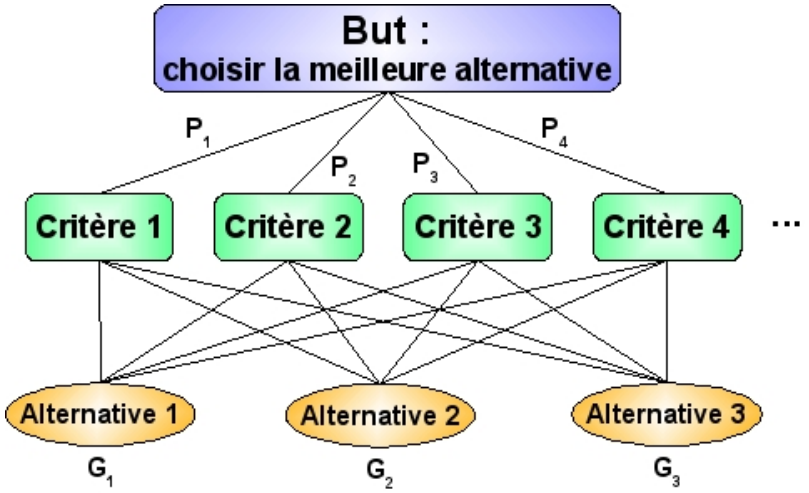

FIG. 3.12 – Modélisation hiérarchique d'un problème avec le processus AHP.

2. Déterminer les priorités locales entre critères,
3. Synthétiser les priorités globales P_i entre critères,
4. Pour chaque critère considéré indépendamment des autres, déterminer la désirabilité locale de chaque alternative,
5. Synthétiser la désirabilité globale G_t de chaque alternative.

Les étapes 1 et 2 se concrétisent par la conception de la matrice de comparaison des paires de critères. Il s'agit d'une matrice carrée N × N (N étant le nombre de critères) de la forme :

	Critère 1	**Critère 2**	**Critère 3**	...	**Critère N**
Critère 1	1	C_{12}	C_{13}	...	C_{1N}
Critère 2	C_{21}	1	C_{23}	...	C_{2N}
Critère 3	C_{31}	C_{32}	1	...	C_{3N}
...	...	...	...	...	...
Critère N	C_{N1}	C_{N2}	C_{N3}	...	1

Les coefficients C_{ij} représentent le degré d'importance numérique du critère i par rapport au critère j i.e. la priorité locale du critère i par rapport au critère j. Pour déterminer ces coefficients, on utilise une échelle comparative qualitative des critères en les comparant deux à deux. Le tableau 3.2 permet ensuite de convertir ces comparaisons en une évaluation numérique. Pour que la matrice globale soit parfaitement cohérente certaines contraintes doivent être respectées :

$$\begin{cases} \forall i,\ C_{ii} = 1 \\ \forall i,\ \forall j \neq i,\ C_{ij} \times C_{ji} = 1 \\ \forall i,\ \forall j \neq i,\ \forall k \neq i \text{ et } k \neq j,\ C_{ik} = C_{ij} \times C_{jk} \end{cases} \tag{3.1}$$

La méthode AHP peut être utilisée avec une matrice de comparaison qui ne soit pas parfaitement cohérente mais qui s'en approche. De telles situations peuvent apparaître si, lors

Importance qualitative entre les critères i et j	**Valeur numérique de C_{ij}**
Le critère i est extrêmement moins important que le critère j	1/9 1/8
Le critère i est énormément moins important que le critère j	1/7 1/6
Le critère i est beaucoup moins important que le critère j	1/5 1/4
Le critère i est légèrement moins important que le critère j	1/3 1/2
Le critère i est d'importance égale avec le critère j	1
Le critère i est légèrement plus important que le critère j	2 3
Le critère i est beaucoup plus important que le critère j	4 5
Le critère i est énormément plus important que le critère j	6 7
Le critère i est extrêmement plus important que le critère j	8 9

TAB. 3.2 – Tableau de priorité qualitative entre critères dans un processus AHP.

de la transcription des priorités locales dans la matrice de comparaison des paires de critères certaines « approximations » sont présentes. Pour illustrer cela, prenons l'exemple suivant : le critère i est deux fois plus important que le critère j et ce dernier est trois plus important que le critère k. On peut alors s'attendre à ce que le critère i soit six fois plus important que le critère k, mais il se peut que lors de la transcription du problème par « l'utilisateur » la priorité locale C_{ik} prenne la valeur 5 ou 7. La matrice de comparaison des paires n'est alors plus parfaitement cohérente mais n'est pas pour autant totalement incohérente. D'une manière plus formelle, la matrice de comparaison des paires de critères n'est pas parfaitement cohérente si :

$$\exists i,\ \exists j \neq i,\ \exists k \neq i \text{ et } k \neq j,\ C_{ik} \neq C_{ij} \times C_{jk}$$

Dans de tels cas, la méthode AHP inclut un processus de vérification du degré de cohérence de la matrice. Nous ne détaillerons pas davantage cet aspect de la méthode AHP car notre métrique est basée sur une matrice de comparaison parfaitement cohérente. Ainsi notre matrice de comparaison des paires de critères contient uniquement des 1 sur sa diagonale puisque le degré d'importance d'un critère ne saurait être différent de lui-même. La deuxième contrainte permet de s'assurer que le degré d'importance C_{ji} est l'inverse du degré d'importance C_{ij}. Ainsi si le critère i est beaucoup plus important que le critère j alors, pour rester cohérent, il faut que le critère j soit beaucoup moins important que le critère i. La dernière contrainte permet de maintenir la cohérence globale de la matrice, à savoir si le critère i a un degré d'importance C_{ij} par rapport au critère j et que ce dernier a un degré d'importance C_{jk} par rapport au critère k alors le critère i doit avoir un degré d'importance $C_{ik} = C_{ij} \times C_{jk}$ par rapport au critère k.

L'étape 3 de la méthode AHP permet de déterminer les priorités globales entre les critères.

Préférence qualitative entre les alternatives t et u selon le critère i	**Valeur numérique de α_{tu}^{i}**
L'alternative t est extrêmement moins préférée à l'alternative u	1/9 1/8
L'alternative t est très fortement moins préférée à l'alternative u	1/7 1/6
L'alternative t est fortement moins préférée à l'alternative u	1/5 1/4
L'alternative t est légèrement moins préférée à l'alternative u	1/3 1/2
Indifférent	1
L'alternative t est légèrement préférée à l'alternative u	2 3
L'alternative t est fortement préférée à l'alternative u	4 5
L'alternative t est très fortement préférée à l'alternative u	6 7
L'alternative t est extrêmement préférée à l'alternative u	8 9

TAB. 3.3 – Tableau de préférence qualitative entre alternatives dans un processus AHP.

Pour cela il faut suivre la procédure suivante :

3.1 Faire la somme des valeurs de chaque colonne.

3.2 Diviser chaque élément de la matrice par le total de sa colonne.

3.3 Calculer la moyenne des éléments de ligne de la matrice. On obtient un vecteur $[P_0; P_1; ...; P_N]$ où P_i est le coefficient de priorité du critère i.

On a donc :

$$P_i = \frac{1}{N} \times \sum_{j=1}^{N} \left(\frac{C_{ij}}{\sum_{k=1}^{N} C_{kj}} \right) \tag{3.2}$$

Dans le cas d'une matrice parfaitement cohérente (i.e. qui vérifie les contraintes de l'équation 3.1) le vecteur des priorités correspond au vecteur propre normalisé de la matrice de comparaison des paires de critères. La méthode de calcul ci-dessus est une méthode approchée permettant de déterminer une approximation de ce vecteur propre dans le cas où la matrice de comparaison n'a pas une cohérence parfaite. Dans toute la suite de ce chapitre, nous nous placerons dans la situation d'une matrice de comparaison parfaitement cohérente.

Les coefficients de priorité vérifient la propriété : $\sum_{i=1}^{N} P_i = 1$. Plus le degré d'importance d'un critère i par rapport à l'ensemble des autres critères sera grand, plus le coefficient de priorité P_i sera important.

Les étapes 4 et 5 du processus AHP consistent à déterminer les désirabilités locales et globales de chaque alternative. Pour cela, on utilise une échelle comparative qualitative (voir tableau 3.3) qui permet d'exprimer un degré de préférence numérique entre les alternatives selon un critère

donné. Pour chaque critère i, on écrit la matrice de second degré correspondante M_i :

	Alternative 1	**Alternative 2**	...	**Alternative M**
Alternative 1	1	α^i_{12}	...	α^i_{1M}
Alternative 2	α^i_{21}	1	...	α^i_{2M}
...	...	...	...	...
Alternative M	α^i_{M1}	α^i_{M2}	...	1

Les coefficients α^i_{tu} représentent la préférence locale de l'alternative t par rapport à l'alternative u en se basant uniquement sur le critère i. Ils permettent de se limiter à des décisions « simples » mono-critères binaires (i.e. entre seulement deux alternatives). Les coefficients α^i_{tu} des matrices de second degré doivent vérifier les mêmes propriétés de cohérence que les coefficients de la matrice de comparaison des paires de critères, à savoir :

$$\begin{cases} \forall t,\ \alpha^i_{tt} = 1 \\ \forall t,\ \forall u \neq t,\ \alpha^i_{tu} \times \alpha^i_{ut} = 1 \\ \forall t,\ \forall u \neq t,\ \forall v \neq t \text{ et } v \neq u,\ \alpha^i_{tv} = \alpha^i_{tu} \times \alpha^i_{uv} \end{cases} \tag{3.3}$$

Dans toute la suite de ce chapitre nous nous placerons dans le cas de matrices de second degré parfaitement cohérentes. L'étape suivant consiste à agréger les jugements mono-critères binaires précédents afin de produire le degré de préférence de chacune des alternatives possibles en se basant sur un seul critère. On obtient donc des estimations numériques permettant de prendre des décisions mono-critères multi-alternatives, ces valeurs numériques sont appelées « désirabilités locales ». Pour chaque critère i on détermine la désirabilité locale F^i_t d'une alternative t en utilisant le même principe que pour le calcul des priorités locales entre critères. On obtient donc une équation similaire à l'équation 3.2 :

$$F^i_t = \frac{1}{M} \times \sum_{u=1}^{M} \left(\frac{\alpha^i_{tu}}{\sum_{v=1}^{M} \alpha^i_{vu}} \right) \tag{3.4}$$

De même que pour les coefficients de priorité, les désirabilités locales vérifient la propriété $\sum_{t=1}^{M} F^i_t = 1$. La dernière étape du processus AHP consiste à agréger les désirabilités locales afin de produire des estimations numériques permettant de choisir entre les différentes alternatives en se basant sur l'ensemble des critères. Ces estimations numériques sont appelées « désirabilités globales ». Pour déterminer la désirabilité globale G_t associée à chaque alternative t on combine les désirabilités locales en utilisant les priorités globales déterminées précédemment. La désirabilité globale G_t s'obtient en combinant les équations 3.2 et 3.4 :

$$G_t = \sum_{i=1}^{N} P_i \times F^i_t = \frac{1}{NM} \times \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \left(\frac{C_{ij}}{\sum_{k=1}^{N} C_{kj}} \right) \times \sum_{u=1}^{M} \left(\frac{\alpha^i_{tu}}{\sum_{v=1}^{M} \alpha^i_{vu}} \right) \right) \tag{3.5}$$

Où :

- N est le nombre de critères,
- M est le nombre d'alternatives,
- C_{ij} est la priorité locale du critère i sur le critère j,
- α^i_{tu} est la désirabilité locale de l'alternative t par rapport à l'alternative u selon le critère i.

La désirabilité globale d'une alternative t est une valeur numérique reflétant le degré de préférence de l'alternative t par rapport à l'ensemble des autres alternatives. Cette estimation numérique est obtenue à partir de l'agrégation des comparaisons binaires mono-critères α^i_{tu}. L'alternative finalement retenue sera celle qui possède la plus grande désirabilité i.e. dont le degré de préférence sera le plus élevé. Ainsi la méthode AHP permet de prendre des décisions multi-alternatives multi-critères à partir de décisions « simples » basées sur un seul critère à la fois et ne portant que sur des comparaisons d'alternatives deux à deux.

3.5 Métrique de qualité sociale d'un plan

Comme nous l'avons vu dans la section 3.3.4, un problème de planification avec prise en compte de la qualité sociale du plan est un n-uplet :

$$< S_W, G, M, E, \{C_{St_k}\}, \{C_{Seq_l}\}, \{C_{TM_m}\}, \{CL_n\}, \{D_p\} >$$

où la métrique M utilise la fonction d'évaluation E et les évaluations issues des ensembles $\{C_{St_k}\}$, $\{C_{Seq_l}\}$, $\{C_{TM_m}\}$, $\{CL_n\}$, $\{D_p\}$ pour pouvoir estimer la qualité sociale d'un plan. La métrique M sera basée sur la méthode AHP détaillée dans la section 3.4.2.

Afin de pouvoir être intégrée efficacement dans un planificateur de tâches, il est souhaitable que la métrique puisse être évaluée partiellement au cours du processus de planification afin de limiter l'espace de recherche. Afin de permettre une évaluation au fur et à mesure des plans trouvés, le planificateur utilisera un plan de référence (le meilleur plan actuellement trouvé) et le comparera au plan partiel en cours d'élaboration. Si ce dernier est moins "prometteur" que le plan de référence alors la branche courante de l'espace de recherche sera coupée et on effectuera une opération de *backtrack*. Nous nous sommes donc intéressés au cas particulier de la méthode AHP appliquée à une décision portant sur deux alternatives.

3.5.1 La méthode AHP avec deux alternatives

Dans notre approche la comparaison des plans se fera deux à deux car elle sera effectuée pendant le processus de planification. Nous allons donc nous intéresser au cas particulier de la méthode AHP lorsque celle-ci porte sur deux alternatives uniquement. Dans le cadre d'une décision basée sur N critères mais ne portant que sur deux alternatives A_1 et A_2, l'équation 3.5

nous permet de déterminer la désirabilité de A_1 :

$$\begin{aligned} G_1 &= \sum_{i=1}^{N} P_i \times F_1^i = \sum_{i=1}^{N} \left(P_i \times \frac{1}{2} \times \sum_{u=1}^{2} \left(\frac{\alpha_{1u}^i}{\sum_{v=1}^{2} \alpha_{vu}^i} \right) \right) \\ &= \frac{1}{2} \times \sum_{i=1}^{N} \left(P_i \times \left(\frac{\alpha_{11}^i}{\alpha_{11}^i + \alpha_{21}^i} + \frac{\alpha_{12}^i}{\alpha_{12}^i + \alpha_{22}^i} \right) \right) \end{aligned}$$

En utilisant les propriétés des coefficients α_{tu}^i vues dans l'équation 3.3, on peut écrire :

$$\begin{aligned} G_1 &= \frac{1}{2} \times \sum_{i=1}^{N} \left(P_i \times \left(\frac{1}{1 + 1/\alpha_{12}^i} + \frac{\alpha_{12}^i}{1 + \alpha_{12}^i} \right) \right) \\ G_1 &= \sum_{i=1}^{N} \left(P_i \times \frac{\alpha_{12}^i}{1 + \alpha_{12}^i} \right) \end{aligned} \tag{3.6}$$

De la même manière on détermine la désirabilité de A_2 :

$$\begin{aligned} G_2 &= \sum_{i=1}^{N} \left(P_i \times \frac{\alpha_{21}^i}{1 + \alpha_{21}^i} \right) \\ &= \sum_{i=1}^{N} \left(P_i \times \frac{1/\alpha_{12}^i}{1 + 1/\alpha_{12}^i} \right) \\ &= \sum_{i=1}^{N} \left(P_i \times \frac{1}{1 + \alpha_{12}^i} \right) \end{aligned} \tag{3.7}$$

Pour déterminer la valeur de G_1 (respectivement G_2) il suffit donc de comparer les deux alternatives critère par critère et de combiner les désirabilités locales grâce à l'équation 3.6 (respectivement 3.7). Comme on compare deux alternatives, on retiendra l'alternative A_1 si $G_1 > G_2 \Leftrightarrow G_1 - G_2 > 0$ sinon on retiendra l'alternative A_2.

3.5.2 Utilisation de la méthode AHP pour la comparaison de deux plans

Nous allons maintenant nous intéresser à l'application des formules de la section 3.5.1 dans le cas particulier où les deux alternatives sont concrétisées par deux plans. Nous nous intéressons, ici, au cas où les deux plans sont totalement instanciés, la comparaison d'un plan complet avec un plan partiel sera abordée dans la section suivante. Étant donné la définition vue dans la section 3.3.4 pour un problème de planification avec prise en compte de la qualité sociale du plan, les estimations des différents critères sociaux du plan Pl peuvent être exprimées dans un vecteur V_{Pl} :

$$\begin{aligned} V_{Pl} &= [\ V_{Pl}^i\ ] \\ &= [\ S_E\ [S_{St_1} ... S_{St_{N_S}}]\ [S_{Seq_1} ... S_{Seq_{N_Q}}]\ [S_{TM_1} ... S_{TM_{N_T}}]\ [S_{CL_1} ... S_{CL_{N_L}}]\ [S_{D_1} ... S_{D_{N_D}}]\ ] \end{aligned}$$

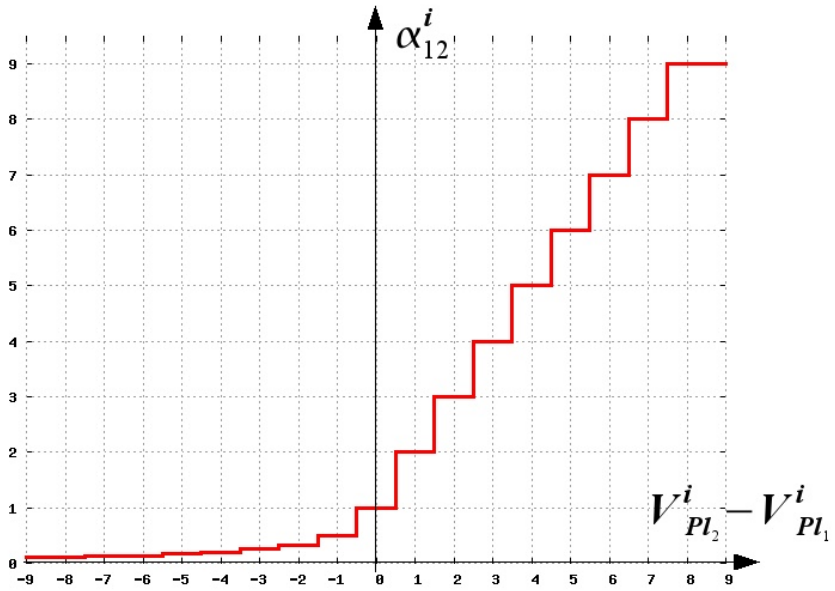

FIG. 3.13 – Comparaison de deux plans : courbe de conversion des scores du plan en désirabilité locale.
Lors de la comparaison de deux plans, cette courbe permet de convertir la différence des scores associés à la règle sociale i en une désirabilité locale exploitable par le processus AHP.

Où :
- S_E est le score associé au critère de gestion des efforts,
- $[S_{St_1}...S_{St_{N_S}}]$ sont les scores associés aux différents états indésirables,
- $[S_{Seq_1}...S_{Seq_{N_Q}}]$ sont les scores associés aux différentes séquences indésirables,
- $[S_{TM_1}...S_{TM_{N_T}}]$ sont les scores associés aux différentes règles sociales liées aux temps d'inactivité,
- $[S_{CL_1}...S_{CL_{N_C}}]$ sont les scores associés aux différentes règles sociales liées aux liens croisés,
- $[S_{D_1}...S_{D_{N_D}}]$ sont les scores associés aux différentes mauvaises décompositions.

Si on note N_C la dimension du vecteur représentant un plan, la comparaison de deux plans Pl_1 et Pl_2 consiste à comparer les scores composant les deux vecteurs associés V_{Pl_1} et V_{Pl_2} afin de déterminer les préférences α_{12}^i pour i = 1 à N_C et d'appliquer les formules 3.6 et 3.7. Pour chaque critère i la valeur $V_{Pl_1}^i$ (respectivement $V_{Pl_2}^i$) représente la contribution du critère i à la dégradation de la qualité sociale du plan Pl_1 (respectivement Pl_2). Ainsi, plus un coefficient $V_{Pl_k}^i$ est grand plus la qualité sociale du plan Pl_k selon le critère i diminue. C'est en se basant sur la différence entre les deux contributions $V_{Pl_1}^i$ et $V_{Pl_2}^i$ que la préférence locale α_{12}^i est établie. Pour cela, nous utilisons la courbe illustrée par la figure 3.13. Ainsi, si $V_{Pl_1}^i > V_{Pl_2}^i$ le plan Pl_2 sera préféré donc on aura $\alpha_{12}^i \leq 1$ et réciproquement.

3.5.3 Évaluation partielle

Afin de pouvoir intégrer efficacement la métrique établie dans notre planificateur, il nous faut pouvoir faire une estimation partielle des désirabilités globales G_t pendant le processus de

planification (i.e. avec un plan partiel) afin de pouvoir arrêter l'exploration des branches peu prometteuses de l'espace de recherche. Considérons la comparaison d'un plan Pl_1 complètement instancié avec un plan $\widehat{Pl_2}$ partiellement instancié. On associe au plan Pl_1 le vecteur V_{Pl_1} et on associe au plan partiel $\widehat{Pl_2}$ le vecteur $\widehat{V_{Pl_2}}$ (dans l'état courant du plan). On note Pl_2 le plan solution qui résulterait de la poursuite du processus de planification à partir de $\widehat{Pl_2}$. Le vecteur V_{Pl_2} est le vecteur associé au plan Pl_2. On notera $\widehat{\alpha_{12}^i}$ la désirabilité locale partielle entre le plan Pl_1 et le plan partiel $\widehat{Pl_2}$ selon le critère i. Si on suppose que la violation d'une règle sociale ne peut pas compenser la violation d'une autre règle sociale, on peut émettre l'hypothèse que les coefficients $\widehat{V_{Pl_2}^i}$ ne peuvent qu'augmenter au fur et à mesure que $\widehat{Pl_2}$ se rapproche de Pl_2. Ainsi, on a donc :

$$\begin{aligned} & V_{Pl_2}^i \geq \widehat{V_{Pl_2}^i} \\ \Rightarrow \quad & V_{Pl_2}^i - V_{Pl_1}^i \geq \widehat{V_{Pl_2}^i} - V_{Pl_1}^i \\ \Rightarrow \quad & \alpha_{12}^i \geq \widehat{\alpha_{12}^i} \end{aligned} \tag{3.8}$$

Comme la fonction $f : x \longrightarrow \frac{x}{1+x}$ est strictement croissante sur $\mathbb{R}^{+*}$, on peut utiliser l'inéquation 3.8 avec l'équation 3.6 pour écrire :

$$\begin{aligned} & \alpha_{12}^i \geq \widehat{\alpha_{12}^i} \Longleftrightarrow \left(\frac{\alpha_{12}^i}{1+\alpha_{12}^i}\right) \geq \left(\frac{\widehat{\alpha_{12}^i}}{1+\widehat{\alpha_{12}^i}}\right) \\ \Longrightarrow \quad & G_1 \geq \sum_{i=1}^{N} \left(P_i \times \frac{\widehat{\alpha_{12}^i}}{1+\widehat{\alpha_{12}^i}}\right) \end{aligned} \tag{3.9}$$

De même, comme la fonction $g : x \longrightarrow \frac{1}{1+x}$ est strictement décroissante sur $\mathbb{R}^{+*}$, on peut utiliser l'inéquation 3.8 avec l'équation 3.7 pour écrire :

$$\begin{aligned} & \alpha_{12}^i \geq \widehat{\alpha_{12}^i} \Longleftrightarrow \left(\frac{1}{1+\alpha_{12}^i}\right) \leq \left(\frac{1}{1+\widehat{\alpha_{12}^i}}\right) \\ \Longrightarrow \quad & G_2 \leq \sum_{i=1}^{N} \left(P_i \times \frac{1}{1+\widehat{\alpha_{12}^i}}\right) \end{aligned} \tag{3.10}$$

En combinant les inégalités 3.9 et 3.10 on peut écrire :

$$\begin{aligned} G_1 - G_2 \quad & \geq \quad \sum_{i=1}^{N} \left(P_i \times \frac{\widehat{\alpha_{12}^i}}{1+\widehat{\alpha_{12}^i}}\right) - \sum_{i=1}^{N} \left(P_i \times \frac{1}{1+\widehat{\alpha_{12}^i}}\right) \\ \Leftrightarrow G_1 - G_2 \quad & \geq \quad \sum_{i=1}^{N} \left(P_i \times \frac{\widehat{\alpha_{12}^i} - 1}{\widehat{\alpha_{12}^i} + 1}\right) \end{aligned}$$

L'alternative finale retenue sera le plan de référence Pl_1 si les préférences globales G_1 et G_2

vérifient $G_1 - G_2 > 0$, soit si :

$$\boxed{\sum_{i=1}^{N} \left(P_i \times \frac{\widehat{\alpha_{12}^i} - 1}{\widehat{\alpha_{12}^i} + 1} \right) > 0} \tag{3.11}$$

La condition donnée par l'équation 3.11 permettra donc d'appliquer une optimisation de type *branch-and-bound* pendant le processus de planification afin de réduire la taille de l'espace de recherche. Cette équation peut s'interpréter comme la condition permettant d'arrêter la construction d'un plan partiel $\widehat{Pl_2}$ suffisamment « mauvais » dans son état courant pour qu'il ne puisse jamais être préféré au plan de référence.

3.6 Conclusion

Dans ce chapitre nous avons identifié les critères quantitatifs nous permettant de détecter les plans dans lesquels le comportement du robot peut être jugé irrationnel. Nous avons également proposé une métrique permettant de combiner des estimations qualitatives portant sur chacun des critères identifiés. Cette métrique nous permet de comparer deux plans entre eux afin de conserver le plus socialement acceptable. Nous avons également établi une méthode permettant de faire cette comparaison entre un plan solution complet de référence et un plan partiel en cours de formation. Ce chapitre nous a donc permis d'établir les bases théoriques et pratiques d'une métrique utilisable par un planificateur pour produire des plans ayant une bonne qualité sociale.

4

Le planificateur HATP

Grâce aux chapitres précédents nous avons établi qu'un robot assistant doit être doté d'une grande autonomie décisionnelle. Cette autonomie peut être concrétisée par un planificateur délibératif. Le caractère social du comportement global du robot dépend alors directement de la qualité sociale des plans fournis par le planificateur. Nous avons également établi une méthode générale permettant de comparer la qualité sociale de deux plans. Il nous faut maintenant nous intéresser à l'insertion de cette méthode d'évaluation au sein d'un planificateur afin de permettre son utilisation en ligne sur un robot.

4.1 Choix du formalisme de planification

Le principe de fonctionnement d'un planificateur de tâches est illustré par la figure 4.1. Le planificateur prend en entrée trois éléments :

1. *le domaine* : il contient la description de l'ensemble des actions qui peuvent être effectuées et éventuellement la description des tâches de haut-niveau qui peuvent être décomposées en sous-tâches,
2. *la base de faits* : elle contient un ensemble de faits qui décrivent l'état du monde au commencement du processus de planification,
3. *le problème* : il contient l'ensemble des buts/tâches qui doivent être satisfaits/réalisées.

Le domaine du planificateur est une description constante tout au long de l'exécution de l'application alors que les deux autres éléments changent selon le contexte courant du robot.

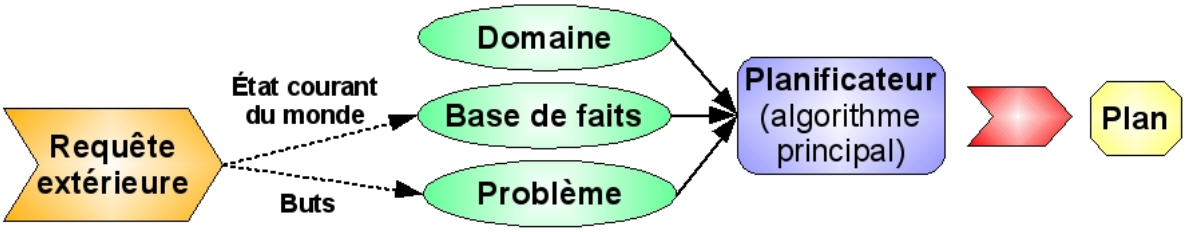

FIG. 4.1 – Principe de fonctionnement d'un planificateur de tâches.

A chaque fois que le robot a besoin d'établir une *procédure* pour réaliser ces buts courants, un élément de son architecture logicielle (le plus souvent le superviseur) émet une requête vers le planificateur. Cette requête permet alors de constituer la base de faits à partir de l'état courant du monde et le problème à partir des buts à satisfaire. L'objectif du planificateur est alors de trouver une combinaison d'actions (i.e. un plan) permettant d'amener la base de faits de son état courant vers un état qui permette de résoudre le problème (i.e. qui satisfasse les buts).

Il existe de nombreux formalismes de planification, une liste exhaustive est disponible dans [Ghallab 04]. La plupart de ces formalismes prennent en entrée un domaine contenant uniquement la description des actions élémentaires qui peuvent être effectuées. Le formalisme utilisé permet alors d'explorer tout ou partie de l'ensemble des combinaisons possibles d'actions jusqu'à ce qu'un(des) plan(s) solution(s) soit(soient) trouvé(s). Il existe des formalismes qui combinent la description des actions élémentaires avec la description de tâches de haut-niveau. Ces tâches de haut-niveau forment des *recettes/scripts* permettant de combiner d'autres tâches et/ou des actions élémentaires afin de satisfaire des buts de haut-niveau. Comme nous l'avons vu dans la section 3.2.6, les tâches de haut-niveau permettent d'exprimer les intentions du robot, elles sont utiles pour le contrôle d'exécution du plan et elles contribuent à l'évaluation de ce dernier. Il convient donc de retenir un formalisme où ces tâches de haut-niveau apparaissent explicitement. Le formalisme HTN (*Hierarchical Task Network* en anglais) est donc particulièrement indiqué. Le formalisme HTN permet de construire une structure hiérarchique qui va évoluer au fur et à mesure que les tâches de haut-niveau vont être décomposées en sous-tâches. Ce formalisme permet d'effectuer un raisonnement étape par étape i.e. on détermine la décomposition d'une tâche puis on détermine la décomposition de ses sous-tâches et ainsi de suite. Ce type de raisonnement est proche d'un raisonnement humain et facilite donc la compréhension entre l'homme et le robot.

Afin de valider notre approche des tests préliminaires [Montreuil 05] ont été réalisés sur un planificateur HTN générique reconnu pour son efficacité : SHOP2 [Nau 03]. SHOP2 n'étant pas conçu pour recevoir les descriptions de nos critères d'évaluation de qualité sociale, il a fallu adapter la description de son domaine pour y incorporer une version simplifiée de notre système d'évaluation. Ces tests ont fourni des résultats encourageants : ils ont mis en évidence la possibilité d'influencer les plans produits par le planificateur en jouant sur l'importance des différentes règles sociales. Toutefois, le pouvoir expressif de SHOP2 étant trop limité pour ce que nous souhaitions faire, la description des domaines de planification est rapidement devenue très

complexe. De plus, nous avons très rapidement ressenti la nécessité d'introduire les règles sociales au sein du processus de planification plutôt que dans une phase secondaire de traitement. Nous avons donc opté pour le développement d'un planificateur appelé HATP (*Human Aware Task Planner* en anglais) basé sur le formalisme HTN et spécifiquement conçu pour une utilisation concrète de notre approche sur un robot assistant.

Nous nous proposons maintenant de détailler les structures et les descriptions qui permettent de définir les éléments en entrée d'HATP : la base de faits, le domaine et le problème. Afin d'illustrer le pouvoir expressif de notre langage de description nous comparerons ces possibilités de description avec celles du langage de référence de la planification à savoir le langage PDDL (*Planning Domain Definition Language* en anglais) [Ghallab 98]. Afin de ne pas alourdir le manuscrit nous limiterons cette comparaison à un point de vue qualitatif sans aller jusqu'au détail de la syntaxe. Dans la suite nous détaillerons également la structure d'un plan produit par HATP, nous justifierons les choix techniques qui ont été retenus afin de rendre le planificateur utilisable en ligne et, enfin, nous décrirons l'algorithme principal de HATP.

4.2 Structure et description de la base de faits

La base de faits est l'élément qui contient la description de l'état courant du monde. Cette description est composée d'un ensemble de faits. Dans de nombreux planificateurs, la base de faits est composée d'un ensemble de prédicats du premier ordre. Dans le planificateur HATP, la base de faits est composé d'un ensemble d'entités. Chaque type d'entité possède un ensemble d'attributs définis par la description du domaine. Ainsi, dans HATP, un fait est défini comme la valeur particulière prise par un attribut d'une entité à un instant donné. Cette représentation de la base de faits permet de la structurer facilitant ainsi l'accès aux données (i.e. aux faits) pendant le processus de planification. La figure 4.2 donne un exemple de la description d'une base de faits HATP, cette description se déroule en 4 étapes :

1. *Définition des types d'entités* : cette étape permet de définir les types d'entités qui seront utilisées pour décrire l'état du monde. Un seul type d'entité est prédéfini, il s'agit du type *Agent* qui représente les individus capables d'agir dans l'environnement.
2. *Définition des attributs* : cette étape permet de décrire l'ensemble des attributs associés à un type d'entité. Un attribut peut être statique ou dynamique selon si sa valeur restera invariable ou non pendant le processus de planification. Un attribut peut être une donnée atomique (i.e. une valeur unique) ou un ensemble de valeurs. Chaque valeur est typée, il peut s'agir d'un nombre, d'une chaîne de caractères, d'un booléen ou d'un type d'entité défini à l'étape 1.
3. *Déclaration des entités* : cette étape permet de créer les entités qui composeront la base de faits en associant un nom d'entité à un type et donc à un ensemble d'attributs tels que définis à l'étape 2.
4. *Initialisation de la base de faits* : cette étape permet d'initialiser les valeurs des attributs pour toutes les entités déclarées à l'étape 3. Cette initialisation permet notamment de

donner des valeurs aux attributs statiques.

Lors de la phase d'initialisation il existe trois opérations permettant de modifier un attribut :
- Le symbole "=" permet d'affecter une valeur à un attribut atomique,
- Le symbole "<<=" permet d'ajouter une valeur à un attribut ensembliste,
- Le symbole "=>>" permet de retirer une valeur d'un attribut ensembliste.

L'exemple de déclaration de la figure 4.2 permet de définir un environnement composé de trois lieux (*Cuisine*, *Salon* et *Chambre*) dans lequel évolue un agent nommé *Bob* qui peut porter aux maximum deux objets en même temps (attribut *maxPossessions*). Dans la situation initiale *Bob* se trouve au *Salon* (attribut *positionTopologique*) et tient en main le *Verre* et la *Bouteille* (attribut *possessions*). Comme *Bob* porte le nombre maximum d'objets qu'il peut, on considère qu'il a les mains pleines (attribut *mainsPleines*). Les attributs *possesseur* de *Verre* et de *Bouteille* sont également mis à jour afin d'assurer la cohérence de l'ensemble de la base de faits.

Afin d'illustrer la différence entre un fait exprimé sous la forme d'un prédicat du premier ordre et un fait dans HATP considérons l'exemple de l'attribut *positionTopologique* associé au type d'entité *Agent*. Dans notre exemple le fait à représenter est que *Bob* a une position topologique qui correspond à une zone symbolique appelée *Salon*. Pour exprimer cela avec un prédicat du premier ordre on ajouterait dans la base de faits l'élément (*positionTopologique Bob Salon*) alors que dans HATP on affecte la valeur *Salon* à l'attribut *Bob.positionTopologique*.

Le langage PDDL permet de déclarer une base de faits composées de prédicats du premier ordre. Dans sa première version [Ghallab 98], ce langage permet de déclarer une base de faits contenant des informations typées. Pour cela, l'entête de la déclaration du domaine inclut une liste des types présents dans la base de faits ainsi qu'une description de la structure des prédicats. Cela correspond donc aux deux premières étapes de la déclaration d'une base de faits HATP. Par exemple, dans le langage PDDL1 la déclaration de prédicat *(at ?x - objet ?l - lieu)* permet de définir dans le domaine courant le prédicat *at* qui prend en premier paramètre un objet (représenté par la variable *?x*) et en second un lieu (représenté par la variable *?l*). Toutefois, bien qu'il soit possible d'émuler le comportement d'un fait basé sur un ensemble de valeurs la déclaration explicite de vecteurs n'est pas possible dans le langage PDDL. On constate donc que, concernant la déclaration de la base de faits, le pouvoir expressif de HATP est similaire à celui de PDDL tout en permettant une déclaration plus aisée d'ensemble de valeurs.

4.3 Structure et description du domaine de planification

Un domaine de planification basé sur le formalisme HTN contient les descriptions de l'ensemble des actions élémentaires possibles ainsi que les descriptions des différentes tâches de haut-niveau. Nous avons vu dans la section 3.3.2 que la prise en compte de la qualité sociale du

```
factdatabase
{
  // Etape 1 : Definition des types d'entites
  define entityType Lieu;
  define entityType Objet;

  // Etape 2 : Definition des attributs
  define entityAttributes Agent {
    static atom string nom;
    dynamic atom Lieu positionTopologique;
    dynamic set Objet possessions;
    static atom number maxPossessions;
    dynamic atom bool mainsPleines;
  }

  define entityAttributes Objet {
    static atom number poids;
    dynamic atom Agent possesseur;
  }

  // Etape 3 : Creation des entites
  Cuisine = new Lieu;
  Chambre = new Lieu;
  Salon = new Lieu;
  Bob = new Agent;
  Verre = new Objet;
  Bouteille = new Objet;

  // Etape 4 : Initialisation des attributs
  Bob.nom = "Bob";
  Bob.positionTopologique = Salon;
  Bob.possessions <<= Verre;
  Bob.possessions <<= Bouteille;
  Bob.maxPossessions = 2;
  Bob.mainsPleines = true;

  Verre.poids = 100;
  Verre.possesseur = Bob;

  Bouteille.poids = 1000;
  Bouteille.possesseur = Bob;
}
```

FIG. 4.2 – Exemple de déclaration d'une base de faits dans HATP.

plan impose d'inclure dans le domaine les descriptions associées aux différents critères permettant son évaluation. Un domaine HATP contient donc la description des actions élémentaires, des tâches de haut-niveau et des règles sociales permettant la mesure de la qualité sociale du plan.

4.3.1 Description d'une action HATP

Dans un domaine classique de planification les actions élémentaires sont décrites par des *opérateurs*. Dans le cadre d'une planification délibérative, cette description inclut :

– les *préconditions* : elles correspondent aux conditions d'application de l'opérateur. Elles sont souvent constituées d'une conjonction de faits.

– les *effets* : ils correspondent aux conséquences de l'application de l'opérateur sur l'état du monde. Les effets sont souvent décrits sous la forme d'un ensemble de faits à retirer et d'un ensemble de faits à ajouter dans la base de faits.

– le *coût* (optionnel) : c'est une valeur numérique qui représente la difficulté à réaliser l'action associée. Cette difficulté peut être liée aux ressources consommées par l'action et/ou aux efforts à fournir pour sa réalisation.

Dans HATP, chaque opérateur est appelé *action* et sera associé à un ensemble d'agents qui réaliseront l'action correspondante. La description d'une action HATP contient 5 éléments :

– une *entête* qui contient le nom de l'action et la liste de paramètres typés nécessaires à sa définition. Les agents associés à la réalisation de l'action sont les agents qui seront présents dans la liste de ses paramètres une fois l'action instanciée.

– une *liste de préconditions*. La conjonction de faits associée est décrite par un ensemble de tests portant sur les attributs des paramètres de l'action. Les tests peuvent être des tests d'égalité (symbole `==`), des tests de différence (symbole `!=`) ou des tests d'appartenance à un ensemble (symbole `>>`).

– une *liste d'effets*. Dans HATP, un fait étant défini comme la valeur prise par un attribut d'une entité à un instant donné, les effets d'une action décriront les changements de valeurs qui doivent être opérés sur les attributs des paramètres de l'action. Les opérations qui peuvent être effectuées dans les effets sont les mêmes que celles de la phase d'initialisation de la base de faits (affectation d'une valeur atomique, ajout/retrait d'une valeur dans un ensemble) mais ne peuvent porter que sur les attributs dynamiques des paramètres. Il est également possible de déclarer des effets conditionnels i.e. des effets qui ne s'appliqueront que si la conjonction de faits qui y est associée est vraie.

– une *fonction de calcul de coût* qui permettra de déterminer le coût associé à l'application de l'action. Cette fonction est écrite en C++ et pourra prendre en arguments des paramètres de l'action. Ceci permet d'effectuer des calculs de coûts complexes basés sur l'ensemble des attributs des entités passées en arguments de la fonction. La seule contrainte est que le coût renvoyé par cette fonction doit être $\geqslant 0$.

– une *fonction de calcul de temps* qui permettra de donner une estimation numérique de la durée minimale et de la durée maximale de l'action. A l'image de la fonction de calcul du coût, la fonction de calcul de temps sera écrite en C++ et pourra se baser sur l'ensemble des

```
action Donner(Agent A1, Agent A2, Objet Obj)
{
  // Declaration des preconditions
  preconditions {
    A1 != A2;
    Obj >> A1.possessions;
    Obj.possesseur == A1;
    A1.positionTopologique == A2.positionTopologique;
    A2.mainsPleines == false;
  };

  // Definition des effets
  effects {
    A1.possessions =>> Obj;
    A2.possessions <<= Obj;
    Obj.possesseur = A2;
    A1.mainsPleines = false;

    if{A2.possessions.size() == A2.maxPossessions;}
    {A2.mainsPleines = true;}
  };

  // Fonctions de calcul du cout et du temps
  cost { DonnerFonctionCout(A1,A2,Obj); };
  duration { DonnerFonctionDuree(A1,A2); };
}
```

FIG. 4.3 – Exemple de description de l'action *Donner* dans HATP.

attributs des entités qui lui sont passées en arguments.

La description de l'action *Donner* permettant à un agent $A1$ de donner l'objet *Obj* à un agent $A2$ est illustrée par la figure 4.3. Selon cette description, l'action $Donner(A1, A2, Obj)$ ne peut s'opérer que si :

- ✓ l'Agent $A1$ n'est pas le même Agent que $A2$,
- ✓ *Obj* est dans les *possessions* de $A1$,
- ✓ Le *possesseur* de *Obj* est $A1$ (il y a là une redondance avec le test précédent),
- ✓ $A1$ et $A2$ ont la même position topologique,
- ✓ $A2$ n'a pas les mains pleines.

Lorsque l'action est réalisée, *Obj* est retiré des possessions de $A1$ pour être ajouté dans les possessions de $A2$ et l'attribut *possesseur* de *Obj* est mis à jour. $A1$ ne peut plus avoir les mains pleines, on actualise donc l'attribut $A1.mainsPleines$. On utilise alors un effet conditionnel pour déterminer si $A2$ a les mains pleines, pour cela on compare la dimension de l'attribut *possessions* de $A2$ avec le nombre maximum d'objets que cet Agent peut porter à la fois. En cas d'égalité, l'attribut *mainsPleines* de $A2$ reçoit la valeur *true*. Les agents qui seront associées à la réalisation de cette action seront les agents représentés par les paramètres $A1$ et $A2$.

Le langage PDDL permet, lui aussi, de décrire des actions utilisant des paramètres typés. Celles-ci contiennent également une liste de préconditions et une liste d'effets. De même, le

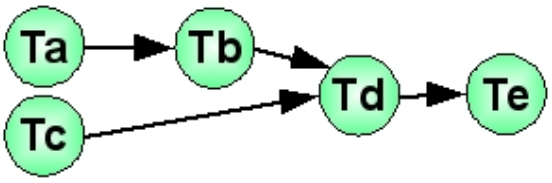

FIG. 4.4 – Exemple de liste de tâches partiellement ordonnées.

```
1    1 : Ta;
2    2 : Tb > 1;
3    3 : Tc;
4    4 : Td > 2, > 3;
5    5 : Te > 4;
```

FIG. 4.5 – Description de la liste de tâches ci-contre.

langage PDDL permet une déclaration d'effets conditionnels tout comme le langage HATP. Toutefois, le pouvoir expressif des préconditions/effets du langage PDDL est plus important que celui de HATP. En effet, PDDL permet de décrire des expressions logiques permettant de chercher un ensemble de faits correspondants à des prédicats particuliers. Par exemple, il est possible de décrire des préconditions/effets portant sur l'ensemble des prédicats $(at\ ?x\ ?l)$ présents dans la base de faits grâce à la syntaxe $(forall\ (at\ ?x\ ?l)\ ...)$. Ce type d'expression n'est pas possible dans HATP puisque les préconditions et les effets ne peuvent porter que sur les paramètres de l'action. De plus, la version 2.1 de PDDL [Fox 03] permet d'introduire des notions temporelles dans le domaine de planification notamment en associant des durées aux actions mais surtout en permettant de déclarer des préconditions/effets attachés à des instants précis ou à des durées. Si nous reprenons la déclaration de l'action *Donner* de la figure 4.3, le langage PDDL2.1 permet, par exemple, de déclarer *explicitement* que pour cette action la précondition $A1.positionTopologique == A2.positionTopologique$ doit être maintenue tout au long de l'action et que l'effet $Obj.possesseur = A2$ n'est effectif qu'à l'instant de fin de l'action. On constate donc que même si une action HATP et une action PDDL ont la même structure le pouvoir expressif temporel de PDDL est nettement supérieur à celui de HATP.

4.3.2 Description d'une méthode HATP

Dans un domaine de planification HTN les tâches de haut-niveau sont très souvent appelées *méthodes*. Une méthode peut être décomposée en une(des) liste(s) partiellement ordonnée(s) de sous-tâches (i.e. d'autres méthodes et/ou d'actions). La description d'une méthode HATP contient 3 éléments :

- une *entête* qui contient le nom de la méthode et la liste de paramètres typés nécessaires à sa définition,
- une *liste de conditions* définissant le but associé à la tâche courante. Si ces conditions sont vérifiées dans l'état courant du monde alors la tâche se réduira à un ensemble vide,
- une *liste de couples* $<P_u,D_u>$ définissant l'ensemble des décompositions possibles pour la tâche. Ainsi pour chaque couple u, si les préconditions P_u sont vérifiées alors la liste partiellement ordonnée de tâches D_u est une façon possible de réaliser la tâche de haut-niveau associée à la méthode décrite.

Une décomposition D_u est une liste partiellement ordonnée de tâches. Pour décrire efficacement une telle liste, il faut pouvoir décrire l'ensemble des tâches la composant ainsi que les liens causaux existant entre ces différentes tâches. Dans le langage HATP, la description d'une décomposition

se fait en associant un identifiant numérique unique à chaque tâche. Les liens causaux sont quant à eux décrits grâce à une relation de précédence entre ces identifiants numériques. Le symbole permettant de décrire cette relation est “>”. La description d'un lien s'effectue lors de la déclaration de la tâche correspondant à la terminaison de ce lien, cela permet d'éviter toute apparition d'un cycle dans la liste de tâches. Afin d'illustrer cette syntaxe, considérons la liste de tâches partiellement ordonnées illustrée par la figure 4.4, la description associée est donnée par la figure 4.5. Les tâches Ta et Tc sont à réaliser en premier, elles sont donc déclarées dans les premières lignes et ne sont associées à aucune terminaison de lien (i.e. il n'y a pas de symbole "$>$" après l'entête de ces deux tâches). La tâche Ta est affectée à l'identifiant "1" et la tâche Tc à l'identifiant "3". Comme il y a un lien partant de Ta et se terminant sur Tb, la déclaration de cette dernière tâche est suivie du symbole "$>$" puis de l'identifiant de Ta à savoir "1". Vient ensuite la tâche Td qui se déroule après Tb et Tc, on complète donc la déclaration de Td avec deux combinaisons "$>$ *identifiant*" correspondant aux deux liens se terminant sur Td. Le principe reste le même pour la tâche Te.

```
1  method AcquerirObjet(Agent Ag, Objet Obj)
2  {
3    // But associe a la methode
4    goal {
5      Obj.possesseur == Ag;
6      Obj >> Ag.possessions;
7    };
8
9    // Preconditions-Decomposition 1
10   {
11     preconditions {
12       Obj.possesseur != NULL;
13       Obj.possesseur != Ag;
14     };
15
16     subtasks {
17       1:SeDeplacer(Ag,Ag.positionTopologique,Obj.possesseur.positionTopologique);
18       2:Donner(Obj.possesseur,Ag,Obj) > 1;
19     };
20   }
21
22   // Preconditions-Decomposition 2
23   {
24     preconditions {
25       Obj.possesseur != NULL;
26       Obj.possesseur != Ag;
27     };
28
29     subtasks {
30       1:SeDeplacer(Obj.possesseur,Obj.possesseur.positionTopologique,
31                    Ag.positionTopologique);
32       2:Donner(Obj.possesseur,Ag,Obj) > 1;
33     };
34   }
35 }
```

FIG. 4.6 – Exemple de description de la méthode *AcquerirObjet* dans HATP.

```
1  method QuitterPiece(Agent Ag, Lieu Pl)
2  {
3    goal { Ag.positionTopologique != Pl; };
4
5    {
6      preconditions {
7        Ag.positionTopologique == Pl;
8      };
9
10     subtasks {
11       // Selection d'un Lieu different de Pl
12       Pl2 = select(Lieu,{ Pl2 != Pl });
13       1:SeDeplacer(Ag,Pl,Pl2);
14     };
15   }
16 }
```

FIG. 4.7 – Exemple de description HATP de la méthode *QuitterPiece* nécessitant une sélection de paramètres.

La description de la figure 4.6 définit une méthode appelée $AcquerirObjet(Ag, Obj)$ permettant à l'agent Ag de se procurer l'objet Obj actuellement possédé par un autre agent. Cette tâche se résume à ne rien faire si Ag possède déjà Obj d'où la présence des tests sur les attributs *Obj.possesseur* et *Ag.possessions* dans la section *goal*. Si la tâche n'est pas vide, il existe deux façons de réaliser cette tâche : (1) Ag se déplace vers le possesseur courant de Obj puis ce dernier donne Obj à Ag ou (2) c'est le possesseur de Obj qui se déplace jusqu'à Ag avant de lui donner Obj.

HATP a également la capacité de sélectionner par lui-même les paramètres de certaines sous-tâches comme l'illustre l'exemple de la figure 4.7. Cet exemple définit une méthode permettant au robot de quitter le lieu où il se trouve en sélectionnant un autre lieu vers lequel il va se rendre. Pour cela, on associe à la variable $Pl2$ un Lieu qui doit être différent de Pl, la décomposition qui s'ensuit utilise alors $Pl2$ dans les paramètres des tâches la composant.

La déclaration de méthodes dans PDDL a été envisagé dans la version 1 de ce langage. Comme le confirme [Fox 03]. Cette capacité n'a pas été totalement abandonnée mais a été retirée dans la version 2.1 en raison d'une non-utilisation de cet aspect dans les compétitions internationales de planification. Le langage PDDL ne permet donc pas de définir des tâches de haut-niveau.

4.3.3 Description des critères sociaux

Afin de compléter la description de l'HTN pour former un domaine de planification avec prise en compte de la qualité sociale du plan, il faut ajouter la description des différentes règles sociales associées aux différents critères identifiés dans le chapitre 3. Nous ne détaillerons ici que la description des règles sociales, l'exploitation de ces descriptions sera détaillée dans la section 4.4.

La gestion des efforts Comme nous l'avons vu dans la section 3.2.1, la production de plans nécessitant le minimum d'efforts est un point important car il permet de garantir l'efficacité des solutions retenues. Les efforts nécessaires à la réalisation globale d'un plan sont calculés en effectuant la somme des coûts des actions composant ce plan. Pour limiter les efforts nécessaires à la réalisation d'une tâche, HATP retiendra de préférence les plans qui ne contiennent que des actions utiles agencées de façon à ce que leur évaluation limite les coûts qui y sont associés. La description du domaine HATP n'inclut pas de description spécifique au critère de *gestion des efforts*, celui-ci étant « morcelé » dans les fonctions de calcul de coûts des différentes actions du domaine.

```
1  undesirableState FoodAndCleaning
2  {
3    // Priorite locale du critere
4    priority = 4;
5
6    // Declaration des variables utiles a la description
7    Objet Obj1,Obj2;
8    Agent Ag;
9
10   // Description de la conjonction de faits
11   conditions {
12     Obj1 != Obj2;
13     Ag.type == "robot";
14     Obj1.categorie == "nourriture";
15     Obj2.categorie == "nettoyage";
16     Obj1 >> Ag.possessions;
17     Obj2 >> Ag.possessions;
18   }
19
20   // Fonction de calcul de la contribution
21   penalty{FoodAndCleaningFunction(Ag,Obj1,Obj2)};
22 }
```

FIG. 4.8 – Exemple de description d'un *état indésirable* dans HATP.

Les états indésirables Un *état indésirable* apparaîtra dans le plan si un ensemble de faits particuliers sont tous vrais en un instant t donné. Nous décrivons donc un *état indésirable* sous la forme d'une conjonction de faits. Une exemple de description d'un *état indésirable* est illustré par la figure 4.8. Le premier élément de la description de toute règle sociale sera toujours sa priorité descriptive. Cette priorité permet de déterminer l'importance de la règle sociale par rapport au critère de *gestion des efforts*, elle sera utilisée pour synthétiser la matrice de comparaison des paires de critères (voir section 4.4 pour plus de détails). Un *état indésirable* contient ensuite les déclarations des variables nécessaires à sa description. Ces variables sont utilisées pour décrire un ensemble de tests représentant la conjonction de faits formant le corps de l'état indésirable. Les tests qui peuvent être opérés sont similaires à ceux décrivant les préconditions d'une action. Enfin, à chaque *état indésirable* est associée une fonction de calcul écrite en C++ qui permettra de calculer la contribution de l'*état indésirable* courant à la dégradation de la qualité sociale du plan. Considérons l'exemple de la figure 4.8. Pour décrire cet état indésirable, il nous faut trois variables : 1 Agent et 2 Objets. La première condition permet de préciser que les deux objets

doivent être différents. Les tests suivants permettent de spécifier que l'objet $Obj1$ doit appartenir à la catégorie "nourriture" alors que l'objet $Obj2$ doit appartenir à la catégorie "nettoyage". Enfin, l'état indésirable sera détecté si les autres faits sont vrais i.e. si Ag possède simultanément $Obj1$ et $Obj2$. Pendant le processus de planification, lorsque cette règle devra être évaluée, HATP cherchera toutes les combinaisons de variables qui unifient les conditions décrites. Il testera ensuite chacune d'elles en vérifiant s'il existe au moins un instant pendant lequel toutes les conditions sont vérifiées en même temps.

```
undesirableSequence PutdownAndPickup
{
  // Priorite locale du critere
  priority = -2;

  // Declaration des variables utiles a la description
  Agent Ag1,Ag2;
  Meuble M1;
  Objet Obj;

  // Contraintes entre les variables
  conditions {
    Ag1 != Ag2;
    Ag1.type == "robot";
    Ag2.type == "humain";
    M1.interieur == true;
  }

  // Description de la sequence
  sequence {
    1:DeposerDans(Ag1,Obj,M1);
    2:FermerMeuble(Ag1,M1) ~> 1;
    3:OuvrirMeuble(Ag2,M1) > 2;
    4:PrendreDans(Ag2,Obj,M1) ~> 3;
  }

  // Fonction de calcul de la contribution
  penalty{PutdownAndPickupFunction(Obj,M1)};
}
```

FIG. 4.9 – Exemple de description d'une *séquence indésirable* dans HATP.

Les séquences indésirables Une *séquence indésirable* est une liste d'actions partiellement ordonnées à détecter au niveau le plus bas du plan. Un exemple de description d'une *séquence indésirable* est illustrée par la figure 4.9. Comme toute règle sociale, la description commence par la valeur de la priorité descriptive. La description se poursuit par la déclaration des variables nécessaires à la description puis par un ensemble de conditions qui permettent d'exprimer des contraintes entre les variables. Cette étape est similaire à la déclaration des variables et des conditions d'un état indésirable. L'étape suivante est de décrire la liste d'actions partiellement ordonnées qui forme le corps de la *séquence indésirable*. La syntaxe utilisée est similaire à celle utilisée pour décrire la décomposition d'une méthode. On peut toutefois noter l'apparition du signe "~>" qui permet de décrire un lien de précédence éventuellement éloigné dans le temps alors que le signe ">" exprime un lien de précédence immédiat. Si dans la description de la séquence indésirable,

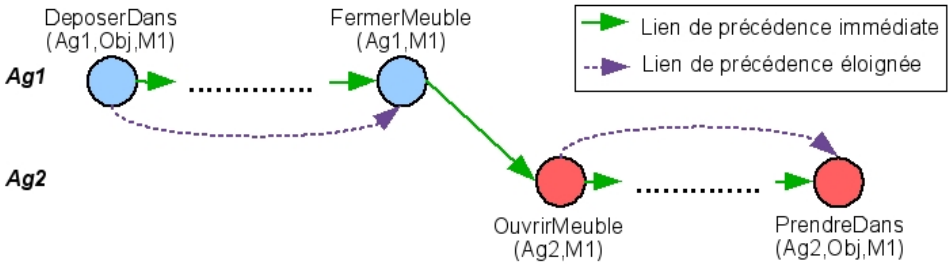

FIG. 4.10 – Représentation graphique d'une *séquence indésirable*.

il existe un lien de précédence immédiat partant de l'action a_1 vers l'action a_2, cela se traduit par le fait que cette règle ne sera détectée dans un plan que si celui-ci comprend un lien causal partant de a_1 et se terminant sur a_2. Si a_1 et a_2 sont séparées par un lien de précédence éloignée la règle sera détectée si dans le plan il y a un chemin partant de a_1 et qui, en suivant les liens causaux, passe par a_2. Afin d'illustrer cela, la liste d'actions partiellement ordonnées décrite dans la figure 4.9 peut être représentée par le graphique de la figure 4.10. Enfin, la description d'une *séquence indésirable* se termine par la déclaration de la fonction C++ qui permettra de calculer sa contribution à la dégradation de la qualité sociale du plan.

```
1  wastedTime
2  {
3    // Priorite locale du critere
4    priority = 0;
5
6    // Liste des agents concernes
7    agents = {Bob, Robot};
8
9    // Fonction de calcul de la contribution
10   penalty{wastedTimeFunction(agents)};
11 }
```

FIG. 4.11 – Exemple de description de la règle sociale des *temps d'inactivité* dans HATP.

Les temps d'inactivité L'utilisation du critère des *temps d'inactivité* dans l'estimation de la qualité sociale d'un plan se traduit par la description d'une unique règle sociale dans le domaine de planification et ce quel que soit le nombre d'agents concernés par cette règle. La figure 4.11 illustre un exemple de description de cette règle. Après la valeur de la priorité descriptive du critère, la description contient la liste des agents concernés par cette règle. Lorsque HATP produit un plan contenant des temps d'inactivité pour les agents concernés, il appelle la fonction C++ de calcul de la contribution en lui passant en paramètre une structure informatique associant à chaque agent concerné la liste de ses durées d'inactivités avec leur ordre d'apparition dans le plan. La fonction peut ainsi calculer la contribution de baisse de qualité sociale du plan en fonction des durées d'inactivité, de leur fréquence et des agents concernés.

Les liens croisés La description utilisée pour l'introduction du critère des *liens croisés* est similaire à celle associée au critère des *temps d'inactivité* comme l'illustre la figure 4.12. La prise en compte des *liens croisés* dans la qualité sociale du plan se traduit par l'apparition d'une

```
controlOfIntricacy
{
  // Priorite locale du critere
  priority = -1;

  // Liste des agents concernes
  agents = {Bob, Robot};

  // Fonction de calcul de la contribution
  penalty{crossedLinksFunction(agents)};
}
```

FIG. 4.12 – Exemple de description de la règle sociale des *liens croisés* dans HATP.

seule et unique description dans le domaine. Cette description commence par la spécification de la valeur de la priorité descriptive et se poursuit par la déclaration des agents concernés par cette règle. Lorsqu'un plan contient des liens causaux en provenance ou à destination d'actions impliquant des agents concernés, la fonction C++ de contribution est appelée avec une structure informatique décrivant l'ensemble des liens croisés. La fonction peut ainsi calculer la contribution de baisse de qualité sociale du plan en fonction du nombre de liens croisés et/ou des agents qu'ils concernent.

Les mauvaises décompositions Pour décrire une *mauvaise décomposition*, il faut pouvoir décrire une structure arborescente partielle composée de listes de tâches partiellement ordonnées reliées par des liens hiérarchiques. Un exemple de description d'une *mauvaise décomposition* est illustré par la figure 4.13. Cette description débute par la spécification de la priorité descriptive, vient ensuite la liste des variables nécessaires à la description et les contraintes existant entre ces variables. Ces deux étapes sont similaires à celles des *états indésirables* et des *séquences indésirables*. L'étape suivante consiste à décrire la structure arborescente partielle. Nous avons repris le même principe que pour les *séquences indésirables* mais il a été étendu afin d'y ajouter une dimension supplémentaire. Cela débute par la description des listes de tâches partiellement ordonnées formant les noeuds de la structure, la syntaxe utilisée est identique à celle employée pour les *séquences indésirables*. La description est complétée par les liens hiérarchiques permettant de lier une tâche d'un des noeuds à un autre noeud. Les liens hiérarchiques peuvent être soit des liens *immédiats* soit des liens *profonds*. De la même façon que dans les *séquences indésirables*, un lien hiérarchique immédiat décrit une décomposition au niveau juste inférieur alors qu'un lien hiérarchique profond permet de définir une décomposition dans l'ensemble du sous-arbre d'une tâche. Afin d'illustrer cela, la figure 4.14 donne une représentation graphique de la structure décrite par la syntaxe de la figure 4.13.

Dans PDDL, la déclaration de critères intervenant dans la métrique d'évaluation des plans n'est possible que depuis la version 3 [Gerevini 06]. En effet, la version 2.1 de ce langage permettait de définir des métriques se basant sur des évaluations numériques telles que les ressources consommées ou le temps total du plan. Toutefois, le seul élément reflétant la structure du plan qui pouvait intervenir dans l'évaluation de cette métrique était le nombre d'actions

```
1  badDecomposition OpenAndClose
2  {
3    // Priorite locale du critere
4    priority = 2;
5
6    // Declaration des variables utiles a la description
7    Agent Ag1,Ag2;
8    Objet Obj;
9    Meuble M1;
10
11   // Contraintes entre les variables
12   conditions {
13     Ag1 != Ag2;
14   }
15
16   // Description de la mauvaise decomposition
17   decomposition {
18     // Description des noeuds
19     node:1 {
20       10:ObtenirObjetDansMeuble(Ag1,Obj,M1);
21     }
22
23     node:2 {
24       20:UnAgentOuvreMeuble(M1);
25       21:UnAgentFermeMeuble(M1) ~> 20;
26     }
27
28     node:3 {
29       30:OuvrirMeuble(Ag1,M1);
30     }
31
32     node:4 {
33       40:FermerMeuble(Ag2,M1);
34     }
35
36     // Description des liens hierarchiques
37     1:10:immediateDecomposition = node 2;
38     2:20:deepDecomposition = node 3;
39     2:30:deepDecomposition = node 4;
40   }
41
42   // Fonction de calcul de la contribution
43   penalty{OpenAndCloseFunction(Ag1,Ag2)};
44 }
```

FIG. 4.13 – Exemple de description d'une *mauvaise décomposition* dans HATP.

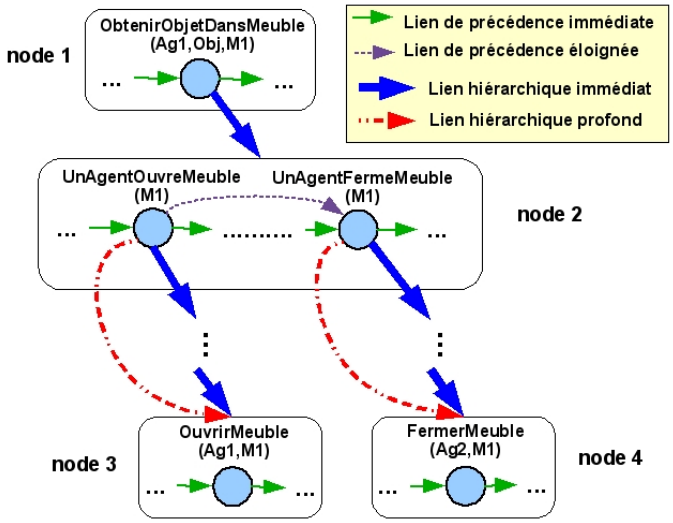

FIG. 4.14 – Représentation graphique d'une *mauvaise décomposition*.

contenues dans le plan. La version 3 de PDDL s'est donc focalisée sur la possibilité d'introduire des critères variés inhérents à la structure du plan dans la métrique utilisée par le planificateur. Pour cela, le domaine inclut la déclaration d'un ensemble de préférences qui sont ensuite quantifiées pour être utilisées dans la métrique. Par exemple, il est possible de déclarer la préférence $p1$ grâce à la syntaxe *(preference p1 (always (cupboard closed)))*. Cette préférence sera considérée comme violée si le plan produit ne maintient pas le fait *(cupboard closed)* à l'état vrai tout au long du plan. La métrique peut alors inclure l'estimation numérique *(is-violated p1)* qui contient le nombre de fois où la préférence *p1* a été violée. Dans PDDL, les préférences peuvent prendre des formes très variées parmi lesquelles on peut citer de manière non exhaustive : des faits qui doivent être vrais/faux tout au long du plan, des faits qui doivent vrais/faux au début ou à la fin du plan, des faits qui doivent être vrais/faux au moins une fois dans le plan, etc.... Ces préférences permettent d'exprimer de manière beaucoup plus riche des règles similaires aux *états indésirables* de HATP. En effet, elles permettent de déclarer des *états indésirables* aussi bien que des *états désirables* en y introduisant une richesse temporelle que HATP ne possède pas. Toutefois, les autres règles sociales que nous avons définies dans notre approche ne peuvent pas être aisément décrites dans le langage PDDL.

4.4 Transcription des règles sociales

Nous avons vu dans la section 3.5.2 que pour chaque plan Pl on associe un vecteur V_{Pl} contenant les scores de chacun des critères sociaux déclarés dans le domaine. Comme la description du domaine HATP ne permet de déclarer qu'une seule règle pour les *temps d'inactivité* et pour les *liens croisés*, on peut simplifier le vecteur de la façon suivante :

$$V_{Pl} = [\ S_E\ [S_{St_1}...S_{St_{N_S}}]\ [S_{Seq_1}...S_{Seq_{N_Q}}]\ [S_{TM}]\ [S_{CL}]\ [S_{D_1}...S_{D_{N_D}}]\ ]$$

Où :

– S_E est le score associé au critère de *gestion des efforts*. Ce score est déterminé en additionnant le coût de l'ensemble des actions présentes dans le plan. Ce score est toujours présent et ce même si le domaine ne contient aucune règle sociale.

– $[S_{St_1}...S_{St_{N_S}}]$ sont les scores associés aux différents *états indésirables* déclarés dans le domaine de planification, ils sont obtenus en faisant appel à leur fonction de contribution respective.

– $[S_{Seq_1}...S_{Seq_{N_Q}}]$ sont les scores associés aux différentes *séquences indésirables* déclarées dans le domaine de planification, ils sont obtenus en faisant appel à leur fonction de contribution respective.

– $[S_{TM}]$ est le score associé au critère des *temps d'inactivité* s'il est déclaré dans le domaine de planification. Il est obtenu en faisant appel à la fonction de contribution qui y est associée.

– $[S_{CL}]$ est le score associé au critère des *liens croisés* s'il est déclaré dans le domaine de planification. Il est obtenu en faisant appel à la fonction de contribution qui y est associée.

– $[S_{D_1}...S_{D_{N_D}}]$ sont les scores associés aux différentes *mauvaises décompositions* déclarées dans le domaine de planification, ils sont obtenus en faisant appel à leur fonction de contribution respective.

Pour pouvoir utiliser le processus d'agrégation des critères élaboré dans le chapitre 3, il faut commencer par déterminer les priorités globales de chacun des critères composant le vecteur V_{Pl}. Pour cela, on utilise les priorités descriptives déclarées dans toutes les règles sociales du domaine. Ces priorités descriptives permettent de définir l'importance du critère décrit par rapport au critère de *gestion des efforts* qui est pris comme référence. Dans la suite de l'explication ce critère sera référencé comme « Critère 1 ». Les priorités descriptives sont des nombres entiers relatifs appartenant à l'intervalle $[-8;8]$. Une valeur de -8 indique que le critère courant est extrêmement moins important que le critère de *gestions des efforts* (i.e. $C_{i1} = 1/9$), une valeur de 0 indique un niveau d'importance est équivalent (i.e. $C_{i1} = 1$) alors qu'une valeur de 8 indique que le critère courant est extrêmement plus important que le critère de *gestions des efforts* (i.e. $C_{i1} = 9$). De manière plus générale, si on note ρ_i la priorité descriptive déclarée dans le domaine pour le critère i, on aura :

$$C_{i1} = \begin{cases} 1 + \rho_i \text{ , si } \rho_i \geq 0 \\ \frac{1}{1-\rho_i} \text{ , sinon} \end{cases}$$

Une fois les priorités locales C_{i1} déterminées, nous utilisons les « propriétés » des coefficients C_{ij} (voir équation 3.1) afin de générer une matrice de comparaison des paires de critères qui soit parfaitement cohérente. Nous déterminons alors les différentes priorités globales en utilisant l'équation 3.2. Une fois ces priorités globales obtenues, il faut synthétiser les désirabilités locales α_{ij} en fonction des scores présents dans les vecteurs V_{Pl}. Lorsqu'on compare deux plans Pl_1 et Pl_2, pour déterminer les désirabilités locales associées à l'ensemble des critères sociaux autres que la *gestion des efforts* on utilise la courbe de conversion illustrée par la figure 3.13. Toutefois, il est difficile d'utiliser cette courbe de conversion pour le critère de *gestion des efforts* car la quantité moyenne d'actions présentes dans le plan (et donc le score moyen de *gestion des efforts*) est très variable selon le degré de complexité du problème courant. Pour pallier cet inconvénient

la désirabilité locale associée à la *gestion des efforts* est calculée en terme d'accroissement relatif du coût total du plan. Si le plan Pl_1 est pris comme plan de référence, on calcule pour Pl_2 le taux d'accroissement du coût global du plan de la façon suivante :

$$a = \frac{S_{E_{Pl2}} - S_{E_{Pl1}}}{S_{E_{Pl1}}}$$

Si ce taux d'accroissement est de 100%, alors le plan de référence Pl_1 sera extrêmement préféré au plan Pl_2 (i.e. $\alpha_{12}^1 = 9$) et, réciproquement, si le taux d'accroissement est de -100%, alors le plan de référence Pl_1 sera extrêmement moins préféré au plan Pl_2 (i.e. $\alpha_{12}^1 = 1/9$).

4.5 Structure d'un plan HATP

Un planificateur délibératif classique délivre généralement des plans sous la forme d'une liste d'opérateurs instanciés partiellement ordonnés. Ceci signifie qu'un plan est composé d'une série d'actions à réaliser dans un ordre précis. Généralement les opérateurs composant le plan sont déterminés séquentiellement, puis le plan est *parallélisé* grâce à une technique détaillée dans [Nau 03]. Toutefois, cette technique est relativement lourde et nécessite de modifier la description du domaine.

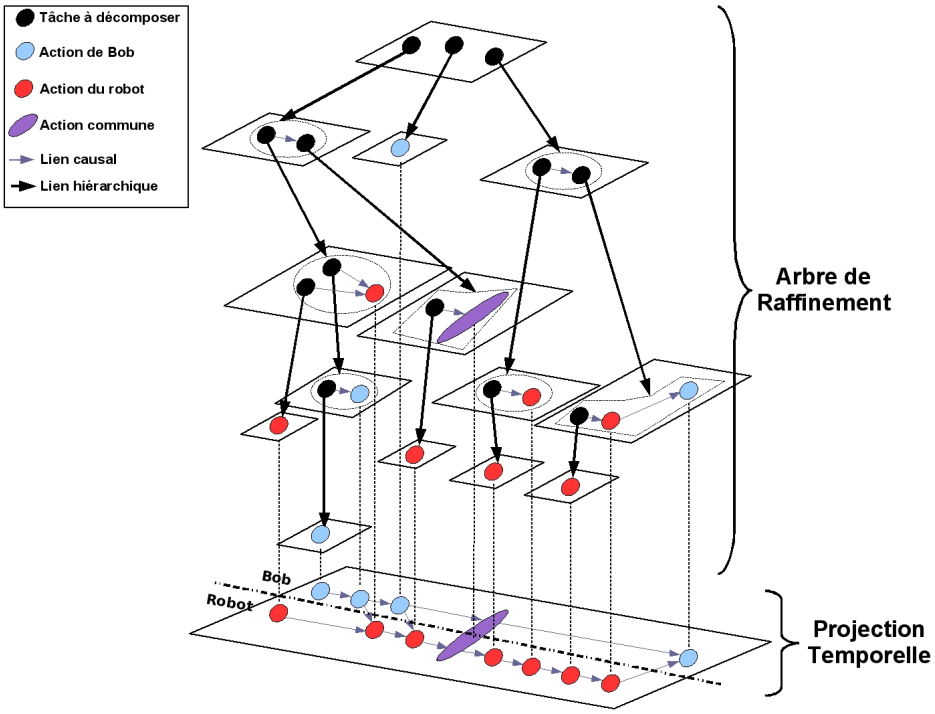

FIG. 4.15 – Structure d'un plan HATP.
Un plan HATP est constitué d'un arbre de raffinement contenant les tâches de haut-niveau et d'une projection temporelle contenant les actions élémentaires ainsi que leur agencement temporel.

Comme nous l'avons établi précédemment, la structure hiérarchique établie pendant le

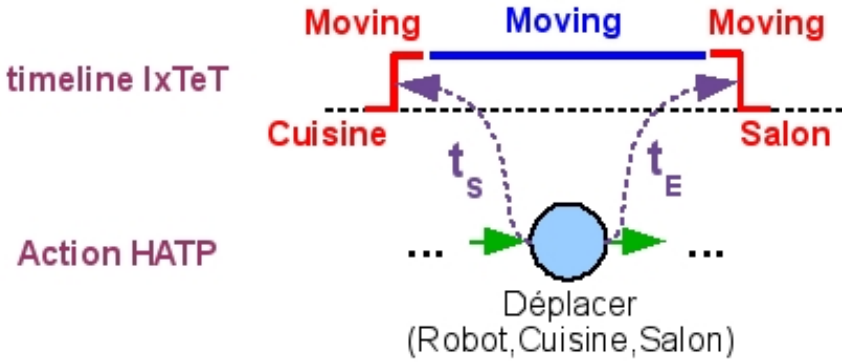

FIG. 4.16 – Portion de timeline IxTeT pour l'attribut Robot.positionTopologique.

processus de planification peut être utilisée pour faciliter l'exécution du plan et pour permettre au robot d'exprimer des intentions. Un plan produit par HATP inclut donc cette structure hiérarchique permettant de préserver la façon dont les tâches de haut-niveau ont été décomposées, cette structure est appelée *Arbre de raffinement*. Toutefois, elle ne suffit pas à décrire complètement un plan. En effet, déterminer la façon dont vont être réalisées les tâches de haut-niveau ne permet pas de décider dans quelle ordre les actions associées vont être exécutées. Pour cela l'arbre de raffinement est complété avec une structure appelée *Projection temporelle* qui a en charge de constituer un treillis temporel définissant l'ordre des actions à exécuter. Un plan produit par HATP peut donc être illustré par la figure 4.15.

La projection temporelle a également pour rôle de constituer une estimation numérique des instants de début et de fin de chacune des actions. Pour cela, elle constitue un treillis temporel numérique à partir des valeurs de durée données par les fonctions de temps des actions. Le treillis temporel est réalisé grâce au moteur principal du planificateur IxTeT [Ghallab 94] qui permet une gestion efficace des contraintes temporelles ([Trinquart 01], [Gallien 06]). Ce treillis permet ensuite d'estimer les *temps d'inactivité* de chaque agent ainsi que la présence d'*états indésirables*.

Pour construire ce treillis, HATP associe à chaque attribut dynamique de la base de faits une *timeline* dans IxTeT. Une timeline est composée d'un ensemble d'évènements et d'assertions qui permettent de conserver les modifications subies par l'attribut au fur et à mesure du déroulement du plan. La figure 4.16 illustre la timeline associée à l'attribut représentant la position topologique du Robot. Lorsqu'une action du plan provoque le déplacement du Robot de la Cuisine vers le Salon, la timeline associée est complétée avec trois éléments :

1. un évènement associé à l'instant t_S (temps de début de l'action) permettant de modifier la valeur de l'attribut de "Cuisine" vers "Moving",
2. un évènement associé à l'instant t_E (temps de fin de l'action) permettant de modifier la valeur de l'attribut de "Moving" vers "Salon",
3. une assertion entre les instants t_S et t_E permettant de maintenir la valeur "Moving" entre ces deux instants.

Les fonctions de calcul de temps des différentes actions présentes dans la projection temporelle permettent de spécifier des contraintes numériques entre les instants de début et de fin de chacune de ces actions. L'ensemble de ces contraintes sont synthétisées et maintenues dans un réseau STN (*Simple Temporal Network* en anglais). Ce réseau constitue le treillis temporel numérique de la projection du plan.

Dans le langage PDDL, un plan est composé d'un ensemble d'actions éventuellement duratives. Ainsi le langage PDDL permet de produire l'équivalent de la *Projection Temporelle* d'un plan HATP mais ne fournit aucun framework permettant d'y associer un arbre de raffinement.

4.6 Problème de planification HATP

Pour un planificateur HTN classique, un problème prend la forme d'une liste partiellement ordonnée de tâches (i.e. d'opérateurs et/ou de méthodes) à réaliser. Cette représentation n'a pas un pouvoir expressif suffisant pour le type d'applications visé par les travaux de ce manuscrit. En effet, un robot assistant doit être capable de prendre en compte les contraintes imposées par son partenaire humain. Pour cela, nous avons doté HATP de la faculté de planifier à partir de plan partiellement défini par le partenaire humain du robot. Un problème HATP prend donc la forme d'une structure arborescente dont les branches sont définies par des *sketchs de plan* similaires à ceux utilisés dans le planificateur PASSAT [Myers 02]. Le partenaire humain peut ainsi imposer les décompositions retenues pour la réalisation des tâches de haut-niveau formant la base du problème.

Pour illustrer cela, reprenons l'exemple support utilisé dans le chapitre 3. Dans cette situation Bob reçoit un invité et souhaite qu'on serve à boire sur la table du salon. La tâche de haut-niveau constituant la base du problème est donc $ServirABoire(Robot, Bob, Invite)$. Si Bob dit à Robot : « Je m'occupe d'apporter la bouteille, va nous chercher des verres et pose les sur la table du salon », alors le problème peut être transcrit sous la forme de l'arbre de tâches illustré par la figure 4.17. Dans cette figure $V1$ et $V2$ représentent les verres, B la bouteille et TS la table du salon. Étant donné les contraintes imposées par Bob, les plans produits par HATP pour ce problème doivent avoir un arbre de raffinement dont la tête est semblable à l'arbre de la figure 4.17.

La description d'un problème HATP doit donc permettre de décrire une structure hiérarchique partielle dont les noeuds sont une liste de tâches partiellement ordonnées. La syntaxe utilisée est donc très similaire à celle utilisée pour décrire les *mauvaises décompositions*. La description d'un problème commence par la description de l'ensemble des noeuds de la structure i.e. de l'ensemble des listes de tâches partiellement ordonnées composant la structure. La deuxième partie de la description permet de décrire les liens hiérarchiques en associant une tâche d'un des noeuds à un autre noeud. La syntaxe illustrée par la figure 4.18 permet de décrire le problème de la figure

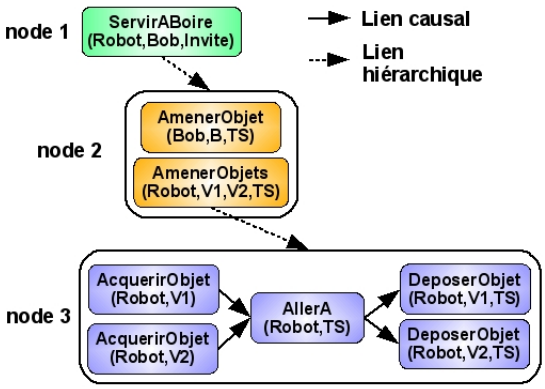

FIG. 4.17 – Problème HATP pour la tâche $ServirABoire(Robot, Bob, Invite)$ avec contraintes du partenaire humain du robot.

```
1  // Description des noeuds
2  node:1 {
3    10:ServirABoire(Robot,Bob,Invite);
4  }
5
6  node:2 {
7    20:AmenerObjets(Robot,V1,V2,TS);
8    21:AmenerObjet(Bob,Bouteille,TS);
9  }
10
11 node:3 {
12   30:AcquerirObjet(Robot,V1);
13   31:AcquerirObjet(Robot,V2);
14   32:AllerA(Robot,TS) > 30, > 31;
15   33:Deposer(Robot,V1,TS) > 32;
16   34:Deposer(Robot,V2,TS) > 32;
17 }
18
19 // Description des liens hierarchiques
20 1:10: decomposition = node 2;
21 2:20: decomposition = node 3;
```

FIG. 4.18 – Description du problème HATP pour la tâche $ServirABoire(Robot, Bob, Invite)$ avec contraintes du partenaire humain du robot.

4.17. Dans l'éventualité où Bob n'aurait imposé aucune contrainte le problème de planification aura été réduit à la seule tâche $ServirABoire(Robot, Bob, Invite)$ et sa description se serait limitée aux quatre premières lignes de la syntaxe de la figure 4.17. Si on considère les contraintes imposées par l'ordre de Bob, il devient alors nécessaire de décrire le noeud formant la base du problème (noeud 1), le sketch de plan décomposant ce noeud (noeud 2) et enfin, le sketch de plan permettant d'imposer la façon dont le robot doit réaliser la tâche qui lui est impartie (noeud 3). La dernière partie de la description permet d'associer les sketchs de plan avec les tâches qui leur correspondent. Pour cela, on associe la tâche $ServirABoire(Robot, Bob, Invite)$ (tâche 10 du noeud 1) avec le noeud 2 et la tâche $AmenerObjets(Robot, V1, V2, TS)$ (tâche 20 du noeud 2) avec le noeud 3.

Dans le langage PDDL, un problème de planification est défini comme un état de la base de faits à atteindre. Ainsi un problème PDDL définit un *but* à atteindre alors qu'un problème HATP est basé sur un ensemble de *tâches* à réaliser décrit sous une forme arborescente partielle.

4.7 Choix techniques pour l'utilisation en ligne

Le planificateur HATP a été développé pour être utilisé en ligne sur un robot. Cette caractéristique impose des contraintes de fonctionnement en temps réel notamment en termes de temps de calcul et d'utilisation de mémoire. Pour trouver une solution concrète efficace à ces contraintes, certains choix techniques ont été nécessaires.

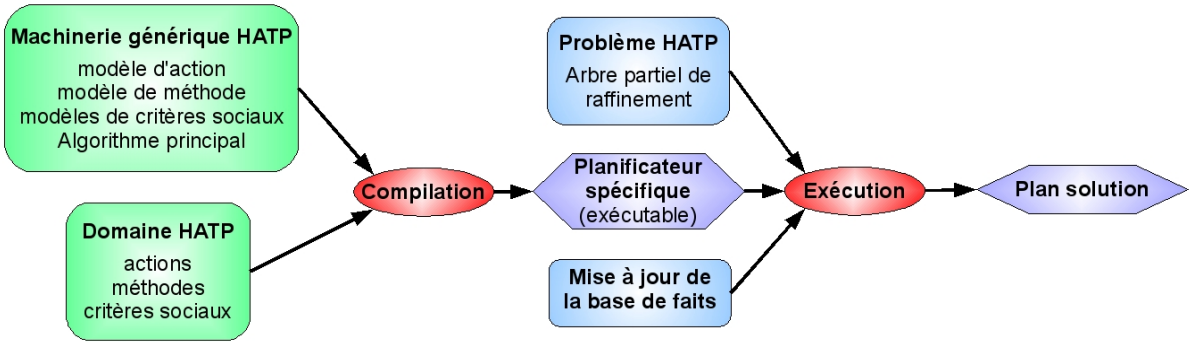

FIG. 4.19 – Principe de compilation de HATP.

4.7.1 Principe de compilation

Il est possible de classer les planificateurs délibératifs en deux familles distinctes : les planificateurs *génériques* et les planificateurs *dédiés*. Les planificateurs génériques utilisent un algorithme central identique quelque soit la forme du domaine de planification alors que les planificateurs dédiés utilisent des domaines spécifiques adaptés à leur algorithme de fonctionnement. De manière grossière on peut résumer les avantages et les inconvénients de chacune des deux approches de la façon suivante :

	Planificateurs dédiés	**Planificateurs génériques**
Gamme d'applications	Limitée	Large
Rapidité d'exécution	Bonne	Moyenne
Utilisation de mémoire	Bonne	Importante

Les planificateurs dédiés ont d'excellentes performances pour une exécution en temps réel mais ne ciblent qu'un nombre limité d'applications. Afin de permettre l'utilisation de planificateurs génériques pour des applications concrètes, les travaux de [Ilghami 03] se sont focalisés sur la génération de planificateurs dédiés à partir d'un processus de planification générique. Ce principe a été repris dans le planificateur HATP. Comme l'illustre la figure 4.19, la description du domaine HATP est utilisée pour générer de manière automatique du code C++ spécifique. Ce code est ensuite compilé pour engendrer un exécutable intégrant directement le domaine de planification et utilisant les bibliothèques génériques du moteur principal d'HATP. Lorsque l'exécutable généré est lancé, il peut recevoir des ordres pour la mise à jour de la base de faits et des requêtes contenant des problèmes de planification. L'apport majeur de ce principe de compilation est qu'il permet de limiter la quantité de mémoire consommée par HATP durant le processus de planification.

4.7.2 Structure multi-thread

Comme l'illustre la figure 4.20, le planificateur HATP a été décomposé en trois threads distincts :

✓ *Le thread de raffinement.* Il permet d'explorer l'espace de recherche, pour chaque plan en cours d'élaboration il construit l'arbre de raffinement et une ébauche de la projection

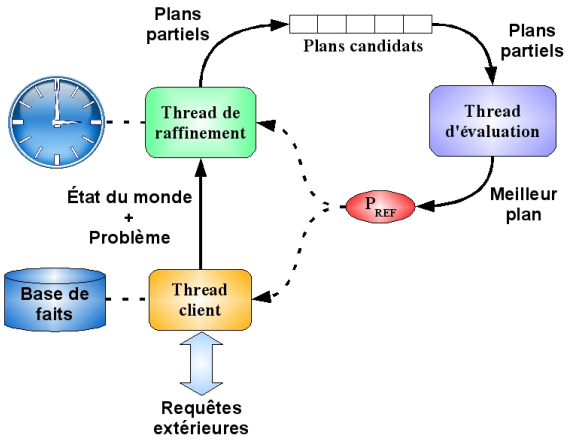

FIG. 4.20 – Structure multi-thread de HATP.

temporelle. Afin de limiter l'espace de recherche, on utilise une optimisation de type *branch-and-bound* en appliquant l'équation 3.11 à l'aide du plan de référence P_{REF}. A chaque fois qu'un plan valide est trouvé par le thread de raffinement il est ajouté dans la pile des plans candidats.

✓ *Le thread d'évaluation.* Son rôle est de construire le treillis temporel numérique du plan candidat courant afin de pouvoir comparer celui-ci au plan de référence P_{REF}. Cette comparaison est effectuée en utilisant l'équation 3.6. Lorsqu'un plan candidat a une désirabilité supérieure à P_{REF} alors il devient le nouveau plan de référence courant.

✓ *Le thread client.* Il permet à HATP de communiquer avec l'extérieur et notamment avec le superviseur du robot. Il a en charge (1) le maintien de la base de faits, (2) la transcription du problème reçu en une structure informatique exploitable par le thread de raffinement et (3) la transcription du meilleur plan trouvé en une structure informatique exploitable par le superviseur.

Lorsque le thread client reçoit un ordre de planification de l'extérieur, il transmet le problème courant au thread de raffinement déclenchant ainsi le processus de planification. Durant ce processus les threads de raffinement et d'évaluation ont des fonctionnements totalement indépendants. Le thread d'évaluation influence « positivement » le thread de raffinement car plus le plan de référence P_{REF} sera proche du meilleur plan possible plus l'optimisation *branch-and-bound* permettra de couper des branches de l'espace de recherche réduisant du même coup la durée de la planification. De plus, cette structure présente l'avantage de paralléliser les calculs du raffinement et les calculs de l'évaluation des plans si le planificateur s'exécute sur une machine multi-processeurs et/ou multi-coeurs.

Nous verrons dans le chapitre 5 qui illustre les résultats obtenus avec HATP qu'on obtient très vite plusieurs plans solutions possibles et que le planificateur entre donc rapidement dans une phase de « sélection » lui permettant d'aboutir à des plans acceptables. Nous avons donc connecté le thread de raffinement à un chronomètre permettant de limiter le temps alloué à l'exploration de l'espace de recherche ce qui nous permet d'obtenir un algorithme pratiquement

« anytime ». Si HATP parvient à explorer la totalité de l'espace de recherche pendant la fenêtre temporelle qui lui est allouée le plan P_{REF} sera le meilleur plan possible, sinon P_{REF} contiendra le meilleur plan associé à la portion de l'espace de recherche exploré.

4.8 Algorithme de planification HATP

Le planificateur HATP étant basé sur le formalisme HTN, son algorithme principal est fortement inspiré de la procédure principale d'un planificateur HTN reconnu pour son efficacité : SHOP2. Nous nous proposons donc de détailler la procédure principale de SHOP2 puis l'algorithme principal de HATP.

4.8.1 Procédure principale de SHOP2

Une version simplifiée de la procédure principale du planificateur SHOP2 est décrite par l'algorithme 4.1. Cette procédure est issue de [Nau 03]. Cet algorithme n'inclut pas les étapes d'élaboration de l'arbre de recherche ni les étapes de backtrack, toutefois, il permet d'illustrer le principe de fonctionnement utilisé par SHOP2 pour produire un plan. Avant de débuter la description de cette procédure, il est important de préciser que dans la description d'un domaine SHOP2 il est possible de définir plusieurs actions et/ou méthodes qui portent le même nom. C'est de cette façon qu'on décrit, dans SHOP2, les différentes décompositions d'une méthode et les variantes des effets des actions. Dans HATP, chaque méthode a un nom unique et l'ensemble de ses décompositions est décrit dans le corps de la méthode. De même, une action HATP a une description unique, les variantes de ses effets étant définies grâce aux effets conditionnels. C'est cette distinction de description dans les domaines qui marque la différence entre l'algorithme SHOP2 et l'algorithme HATP qui sera présenté dans la section suivante.

La procédure principale SHOP2 prend en entrée `s` l'état du monde, `T` la liste de tâches à réaliser (i.e. le problème) et `D` le domaine HTN à utiliser. On initialise le plan `P` par le plan vide (ligne 1) ainsi que l'ensemble T_0 qui contiendra l'ensemble des tâches de `T` sans prédécesseurs (ligne 2). Si la liste de tâches `T` est vide, toutes les tâches ont été réalisées donc on peut sortir de la procédure en renvoyant le plan courant `P` qui est un plan solution (ligne 4). Dans le cas où il reste des tâches de `T` à effectuer on choisit une tâche `t` dans l'ensemble T_0 (ligne 5) puis on applique le traitement approprié selon que `t` est un opérateur (lignes 6-13) ou une méthode (lignes 14-22).

Si `t` est un opérateur, on cherche l'ensemble des opérateurs unifiés (`a`,θ) compatibles avec `t` et dont les préconditions sont vérifiées (ligne 7). Si aucun opérateur unifié n'est trouvé `t` ne peut être réalisée, on sort donc de la procédure avec le statut d'échec (ligne 8). Si au moins un opérateur unifié existe, on en choisit un parmi l'ensemble de ceux qui ont été trouvés (ligne 9) et on met à jour `s`, `P` et `T` en conséquence (lignes 10 à 12). L'état du monde `s` est mis à jour en y supprimant les faits de la liste del(`a`) puis en y ajoutant ceux de la liste add(`a`). Les listes del(`a`)

Algorithme 4.1 : Procédure simplifiée de SHOP2

```
   Données : s, T, D
 1 P = le plan vide
 2 T_0 ← {t ∈ T | predecesseurs(t) = ∅ }
 3 répéter
 4     si T = ∅ alors retourner P
 5     Choisir de manière non-déterministe une tâche t ∈ T_0
 6     si t est un opérateur alors
 7         A ← {(a,θ) | a est un opérateur instancié dans D, θ unifie {head(a),t }, s satisfait
           precond(a)}
 8         si A = ∅ alors retourner "échec"
 9         Choisir de manière non-déterministe une paire (a,θ) ∈ A
10         Modifier s selon del(a) et add(a)
11         Appliquer a à P
12         Modifier T en retirant t et en appliquant θ
13         T_0 ← {t ∈ T | predecesseurs(t) = ∅ }
14     sinon
15         M ← {(m,θ) | m est une instance d'une méthode dans D, θ unifie {head(m), t }, s
           satisfait precond(m), θ est le plus général possible}
16         si M = ∅ alors retourner "échec"
17         Choisir de manière non-déterministe une paire (m,θ) ∈ M
18         Modifier T en remplaçant t par sub(m) et en appliquant θ
19         si sub(m) ≠ ∅ alors
20             T_0 ← {t ∈ sub(m) | predecesseurs(t) = ∅ }
21         sinon
22             T_0 ← {t ∈ T | predecesseurs(t) = ∅ }
23 jusqu'à l'infini
```

et *add*(a) sont déterminées à partir de la description de l'opérateur a. Le plan P est mis à jour en y ajoutant l'opérateur instancié choisi a (ligne 11), la tâche t est alors considérée comme réalisée et doit donc être retirée de la liste T (ligne 12). De plus, afin de respecter les contraintes existant entre les paramètres des différentes tâches à accomplir, on applique l'unification θ à T (ligne 12). Le traitement de la tâche t étant maintenant terminé, on détermine le nouvel ensemble T_0 des tâches sans prédécesseurs (ligne 13) et on réitère la boucle.

Si t est une méthode, on cherche l'ensemble des méthodes unifiées (m,θ) compatibles avec t, dont les préconditions sont vérifiées et telles que l'unification θ soit la plus générale possible (ligne 15). Si aucune méthode unifiée n'est trouvée t ne peut être réalisée, on sort donc de la procédure avec le statut d'échec (ligne 16). Si au moins une méthode unifiée existe, on en choisit une parmi l'ensemble de celles qui ont été trouvées (ligne 17) et on met à jour T en conséquence (ligne 18). La liste de tâches T est mise à jour en remplaçant la tâche t par la décomposition *sub*(m) associée à la méthode unifiée m retenue. De plus, afin de respecter les contraintes existant entre les paramètres des différentes tâches à accomplir, on applique l'unification θ à T (ligne 18). Le traitement de la tâche t étant maintenant terminé, il faut déterminer le nouvel ensemble des tâches sans

prédécesseurs T_0. Afin de limiter l'explosion de l'espace de recherche on détermine l'ensemble T_0 parmi la décomposition $sub(\texttt{m})$ si elle n'est pas réduite à un ensemble vide sinon T_0 est déterminée dans la liste T (lignes 19 à 22). Ces dernières lignes signifient que lorsqu'on sélectionne une tâche `t` qui peut être réalisée grâce à une méthode, on se limitera à la décomposition de cette méthode et de ses sous-tâches jusqu'à ce que cela mène à un ensemble vide ou jusqu'à ce qu'un opérateur soit ajouté dans le plan `P`. Ainsi, lorsqu'une tâche `t` est sélectionnée toutes les autres tâches sont considérées comme *verrouillées* (i.e. elles ne peuvent être sélectionnées) jusqu'à ce que `t` ou une de ses sous-tâches ait été réduite à un ensemble vide ou ait mené à une modification de l'état du monde (i.e. à l'application d'un opérateur dans le plan).

Si on analyse la procédure SHOP2, on constate que les divergences dans l'espace de recherche peuvent avoir trois origines :

- il y a plusieurs tâches sans prédécesseurs qui peuvent être appliquées (ligne 5),
- il y a plusieurs instances possibles d'une action lors de son application (ligne 9),
- il y a plusieurs instances possibles d'une méthode lors de sa décomposition (ligne 17).

Afin de limiter l'espace de recherche exploré, la procédure SHOP2 peut utiliser une optimisation de type *branch-and-bound* dans le cas d'une planification basée sur la qualité des plans produits. Pour cela, SHOP2 associe à chaque plan un score calculé en additionnant les coûts des opérateurs qui y sont présents. Si le score du plan en cours d'élaboration dépasse le score du meilleur plan actuellement en mémoire alors la branche d'exploration est coupée.

4.8.2 Algorithme HATP

L'algorithme 4.2 illustre la procédure principale du thread de raffinement de HATP. La finalité de cet algorithme est de produire les arbres de raffinement et les ébauches de projection temporelle pour tous les plans candidats qui seront transmis au thread d'évaluation.

L'algorithme principal de HATP prend en entrée `s` l'état du monde, `Arb` le problème (i.e. la partie supérieure de l'arbre de raffinement) et `D` le domaine de planification. Cet algorithme commence par l'initialisation d'une liste de tâches `T` (ligne 1), cette liste contient les feuilles de l'arbre `Arb`. On initialise également l'ébauche de projection temporelle `Prj` à un ensemble vide (ligne 2). On entre ensuite dans la boucle principale (lignes 3 à 37) jusqu'à ce que la limite de temps éventuelle soit atteinte ou jusqu'à ce que l'ensemble de l'arbre d'exploration ait été entièrement parcouru. La première étape de la boucle principale consiste à vérifier si la liste de tâches `T` est vide ce qui indiquerait qu'un plan potentiellement valide a été trouvé (lignes 4 à 7). La seconde étape consiste à appliquer une optimisation de type *branch-and-bound* (lignes 8 à 10) afin de limiter l'espace de recherche à explorer. Cette optimisation est réalisée en appliquant l'équation 3.11 s'il existe un plan de référence P_{REF}. Certains critères d'évaluation requièrent la construction du treillis temporel numérique du plan. Or, la construction et le maintien de ce treillis sont coûteux en terme de temps de calcul. L'évaluation partielle ne porte donc que sur le sous-ensemble des règles sociales qui peuvent être évaluées pendant la phase de raffinement : la

Algorithme 4.2 : Algorithme HATP

Données : s, Arb, D
1 T ← *leaves*(Arb)
2 Prj ← ∅
3 **répéter**
4 **si** T = ∅ **alors**
5 P ← produire_plan(Arb, Prj)
6 Ajouter P dans la pile des plans valides
7 Backtracker et Aller à la ligne 4
8 **si** $P_{REF} \neq \emptyset$ **alors**
9 c ← comparaison_partielle(Arb, Prj, P_{REF})
10 **si** $c > 0$ **alors** Backtracker et Aller à la ligne 4
11 **si** ∀ t ∈ T, *lock(t) actif* **alors** ∀ t ∈ T, Déverrouiller t
12 T_0 ← {t ∈ T | *predecesseurs*(t) = ∅, *lock*(t) inactif, t est une action et s satisfait *precond*(t) ou t est une méthode}
13 Créer N(T_0) branches dans l'arbre d'exploration
14 Choisir une tâche t dans T_0 et la branche correspondante
15 **si** t *est une action* **alors**
16 A ← (a,θ) tel que a est une action instanciée dans D, θ unifie {*head*(a),t } et s satisfait *precond*(a)
17 **si** A = ∅ **alors** Backtracker et Aller à la ligne 4
18 Modifier s selon $effects$(a)
19 Appliquer a à Prj
20 Modifier T en retirant t
21 **pour** ∀ t ∈ T **faire** Déverrouiller t
22 **sinon**
23 M ← (m,θ) tel que m est une instance d'une méthode dans D, et θ unifie {*head*(m),t }
24 **si** M = ∅ **alors** Backtracker et Aller à la ligne 4
25 **si** s *satisfait goal(m)* **alors**
26 Retirer t de Arb
27 Retirer t de T
28 Aller à la ligne 4
29 Dec ← {d | d ∈ *decompositions*(m), s satisfait *preconds*(d)}
30 **si** Dec = ∅ **alors**
31 Backtracker et Aller à la ligne 4
32 Créer N(Dec) branches dans l'arbre d'exploration
33 Choisir un membre d dans Dec et la branche correspondante
34 Modifier T en remplaçant t par d
35 Modifier Arb en ajoutant d comme décomposition de t
36 ∀ t ∈ T, t ∉ d Verrouiller t
37 **jusqu'à** *limite de temps atteinte* ou *arbre d'exploration entièrement parcouru*

gestion des efforts, les *séquences indésirables*, les *liens croisés* et les *mauvaises décompositions*. Les règles sociales d'*états indésirables* et de *temps d'inactivité* sont quantifiées lorsque le plan a été transmis au thread d'évaluation et que son treillis temporel numérique a été construit. Il devient alors possible de faire une comparaison complète du plan avec le plan de référence P_{REF} en utilisant l'équation 3.6.

Dans le cas où la liste de tâches `T` contient des tâches qui sont toutes verrouillées, on déverrouille la totalité de `T` (ligne 11). Ce cas de figure peut se produire dans l'éventualité où l'ensemble des tâches sans prédécesseurs et non-verrouillées ont toutes été réduites à des ensembles vides à l'itération précédente. On détermine ensuite T_0 contenant l'ensemble des actions de `T` qui ne sont pas verrouillées, qui n'ont pas de prédécesseurs et dont les préconditions sont satisfaites ainsi que les méthodes de `T` qui ne sont pas verrouillées et qui n'ont pas de prédécesseurs (ligne 12). On crée autant de branches que nécessaire dans l'arbre d'exploration selon la cardinalité de T_0 (ligne 13) puis on choisit une de ces branches en choisissant une tâche `t` de T_0 (ligne 14).

Le traitement de la tâche `t` dépend de sa nature (action ou méthode). S'il s'agit d'une action, on vérifie qu'il existe une instanciation d'une action du domaine `D` qui soit compatible avec `t` (ligne 16), si ce n'est pas le cas on change de branche d'exploration (ligne 17) sinon on opère le traitement pour appliquer l'action (lignes 18 à 21). Pour cela, on modifie l'état courant du monde selon les effets de `a` (ligne 18), on ajoute ensuite `a` à `Prj` (ligne 19), on met à jour `T` (ligne 20) et on déverrouille l'ensemble des tâches de `T` (ligne 21).

Si la tâche `t` est une méthode, on vérifie qu'il existe une instanciation d'une méthode du domaine `D` qui soit compatible avec `t` (ligne 23). Si ce n'est pas le cas on change de branche d'exploration (ligne 24). On vérifie ensuite si les conditions associées au but de `t` sont vraies dans l'état courant du monde (ligne 25), si tel est le cas on retire `t` de `Arb` et de `T` puis on retourne au début de la boucle principale (lignes 25 à 28). Si le but associé à `t` n'est pas réalisé on détermine `Dec` l'ensemble des décompositions possibles de `t` (ligne 29). S'il n'y en a aucune on effectue une opération de backtrack (ligne 30) sinon on crée autant de branches que nécessaire dans l'arbre d'exploration selon la cardinalité de `Dec` (ligne 32). On choisit ensuite une de ces branches (ligne 33), on met à jour la liste de tâches `T` et l'arbre `Arb` selon la décomposition retenue (lignes 34 et 35). Enfin, pour limiter la taille de l'exploration de l'arbre de recherche on verrouille toutes les tâches qui n'appartiennent pas à la décomposition retenue (ligne 36).

Si on analyse la procédure HATP, on constate que les divergences dans l'espace de recherche peuvent avoir deux origines :

– il y a plusieurs tâches qui n'ont pas de prédécesseurs et qui peuvent être appliquées (ligne 14). Dans la suite de ce manuscrit ce cas de figure sera appelé « divergence par parallélisme ».

– il y a plusieurs décompositions possibles pour une méthode (ligne 33). Dans la suite de ce manuscrit ce cas de figure sera appelé « divergence par décompositions ».

L'algorithme principal de HATP est donc très similaire à la procédure SHOP2 notamment dans la façon de traiter les actions et les méthodes. Les différences majeures entre les deux algorithmes résident dans (1) les données constituant un plan, (2) la forme des problèmes de planification et (3) une optimisation *branch-and-bound* de nature différente.

Concernant les informations contenues dans un plan, la différence majeure est que HATP conserve une structure arborescente composées des tâches de haut-niveau associées aux actions présentes dans la projection temporelle. Ainsi HATP est capable de fournir une hiérarchie de tâches au superviseur du robot. Le superviseur peut alors associer à chaque tâche un ensemble de conditions de succès, d'échec et de pertinence conformément à la théorie de l'intention jointe (voir section 1.2.3.1). Lors de l'exécution du plan, cela permet au superviseur de contrôler la validité de l'action courante du robot à différents niveaux. Par exemple, supposons que l'action courante du robot soit $AllerA(Robot, Salon)$ et que cette action soit associée à la tâche de haut-niveau $TrouverAgent(Robot, Bob)$. Dans cet exemple, HATP a introduit dans le plan une action de déplacement du robot vers le salon car selon ses connaissances Bob devait s'y trouver. Si lors de l'exécution du plan le robot est en train de se déplacer et croise Bob dans le couloir menant au salon les conditions de succès associées à la tâche $TrouverAgent(Robot, Bob)$ seront alors vraies mettant un terme à l'action courante sans pour autant invalider le plan.

Concernant la forme des problèmes de planification, SHOP2 prend en entrée une liste partiellement ordonnées de tâches. Nous avons montré que ce type de problème possédait un pouvoir expressif trop limité pour permettre une intégration efficace des éventuelles contraintes sur le plan que pourrait imposer le partenaire humain du robot. Ainsi un problème HATP prend la forme d'une structure arborescente partielle de tâches formant la partie supérieure de l'arbre de raffinement des plans solutions qui seront retenus par le planificateur. Enfin, la dernière différence entre la procédure SHOP2 et HATP porte sur l'optimisation *branch-and-bound*. En effet, le système d'évaluation de qualité des plans utilisé par SHOP2 est une approche additive des coûts associés aux opérateurs présents dans le plan. Dans ce cas de figure, l'optimisation *branch-and-bound* consiste à comparer le coût du plan en cours de construction avec le coût du meilleur plan actuellement en mémoire. L'optimisation *branch-and-bound* revient donc à la comparaison directe de deux valeurs. Pour HATP, l'évaluation de la qualité d'un plan porte sur davantage de critères et utilise un procédé d'agrégation plus complexe. Cela impose donc une évaluation partielle du plan en cours de formation plus « lourde » (on manipule un vecteur et non pas un nombre) et une estimation plus calculatoire pour l'application de cette optimisation.

4.9 Limites des algorithmes SHOP2 et HATP

Lors de nos différents essais avec le planificateur HATP nous avons pu mettre en évidence certaines limites de son algorithme principal. En effet, nous avons pu constater que lorsque le nombre de tâches sans contraintes de précédence augmente dans le problème l'efficacité

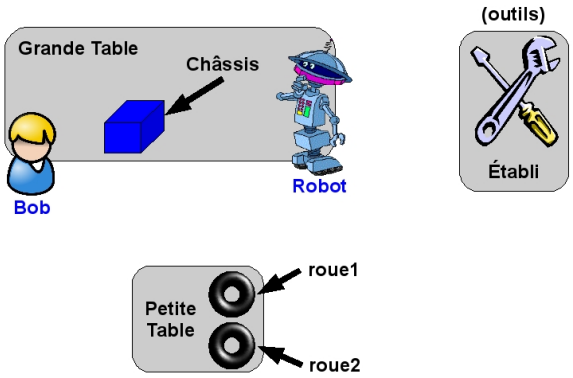

FIG. 4.21 – Situation de départ d'un scénario de construction collaborative d'objets complexes.

du planificateur baisse très rapidement du fait de l'explosion de la dimension de l'espace de recherche à parcourir. Nous avons fait ces constations lorsque nous avons abordé des situations de construction collaborative d'objets complexes. Un exemple de ce type de scénario est disponible dans [Hoffman 06]. Dans cet exemple, l'homme et le robot doivent construire une maquette de véhicule composée de différents objets simples (roues, chassis, etc...) qui doivent être assemblés grâce à des outils (tournevis, clef, pince, etc...). A chaque objet simple est associé un outil particulier. Nous allons ici nous intéresser à un scénario de ce type.

La figure 4.21 illustre la situation de départ du scénario que nous allons utiliser. L'objectif est de construire un objet « complexe » en assemblant le chassis de la maquette avec les deux roues $roue1$ et $roue2$. Pour cela, le $tournevis$ est nécessaire pour fixer la $roue1$ et la $clef$ pour fixer la $roue2$. Au départ les deux roues de la maquette sont présentes sur la petite table, le chassis est sur la grande table et les deux outils sur l'établi. La première partie du problème va consister à apporter toutes les pièces nécessaires au montage sur la grande table qui fera office de « lieu de travail ». Pour cela, $Robot$ se voit affecter la tâche d'aller chercher $tournevis$ et $clef$ puis de les amener sur la grande table pendant que Bob opère la même chose pour $roue1$ et $roue2$. Une fois cette étape réalisée le seconde phase consiste à effectuer le montage de manière collaborative.

Le problème de planification associé à cette situation est illustré par la figure 4.22. Celui-ci se décompose en deux parties : l'acheminement du matériel nécessaire et le montage de la maquette. A propos de la phase d'acheminement, la partie du problème concernant Bob est composé des tâches $AllerChercher(Bob, Roue1, Roue2, petiteTable)$ et $Deposer(Bob, Roue1, Roue2, grandeTable)$ qui sont totalement ordonnées entre elles. L'autre partie du problème est basée sur la même structure mais concerne $Robot$. La figure 4.22 illustre également les seules décompositions disponibles pour les tâches composant la première phase. Chacune des quatre tâches ne peut se décomposer qu'en une seule série de trois actions spécifiques. Cela signifie que quel que soit le plan produit pour le problème global il aura nécessairement une structure comme celle illustrée par la figure 4.23. Il est clair que ce scénario est artificiel et correspond à un cas très spécifique, toutefois, il va nous permettre de mettre en évidence de manière flagrante le phénomène d'explosion combinatoire. On associera à la seconde partie du problème (i.e. le

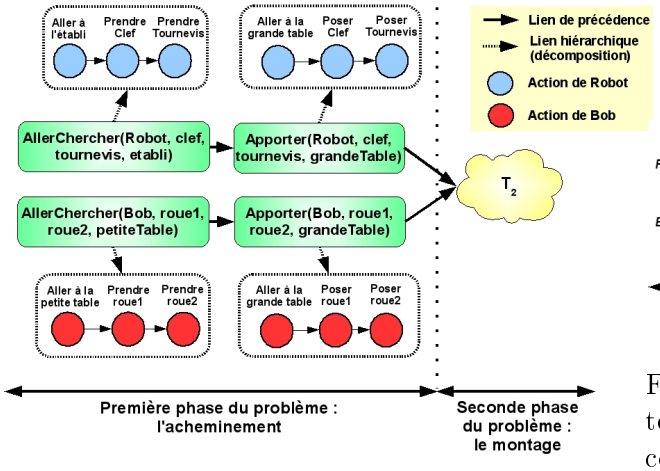

FIG. 4.22 – Problème de planification pour la construction collaborative d'objets complexes.

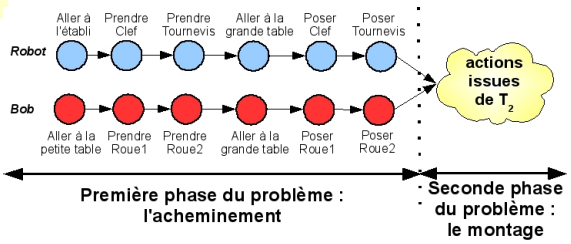

FIG. 4.23 – Début de toutes les projections temporelles pour le problème de construction collaborative d'objets complexes.

montage de la maquette) un arbre de recherche appelé T_2. La caractéristique importante de ce scénario est la présence de « flux » d'actions totalement indépendants. En effet, dans cet exemple il apparaît clairement que pour la première partie du problème le « flux d'actions » de *Robot* n'interfère pas dans celui de *Bob*. On peut ainsi tenter de définir de manière informelle des actions « indépendantes » comme des actions qui n'affectent pas les mêmes attributs de la base de faits, qui n'ont pas de lien de causalité entre elles et qui ne sont pas associées aux mêmes agents. Ainsi dans notre exemple les six actions de *Bob* pour la première phase du problème sont indépendantes des six actions de *Robot* pour cette même phase. Le planificateur pourrait donc traiter ces actions de manière purement séquentielle sans pour autant réduire l'espace des plans solutions. Or l'algorithme de HATP va introduire des divergences de parallélisme dans l'espace de recherche. En effet, l'algorithme de HATP va ajouter les actions des deux flux dans la projection temporelle du plan les unes après les autres et va considérer que deux branches de l'espace de recherche sont distinctes si ces actions ont été « projetées » dans des ordres différents. Ainsi le fait d'ajouter dans la projection d'un plan l'action $AllerA(Bob, grandeTable, petiteTable)$ puis l'action $AllerA(Robot, grandeTable, etabli)$ correspond à une branche différente de l'espace de recherche que la situation où ces deux actions auraient été projetées dans l'ordre inverse. Dans notre exemple, l'algorithme de HATP va donc « dupliquer » certaines branches de l'espace de recherche provoquant ainsi son explosion.

Afin d'étudier ce phénomène nous avons considéré des sous-problèmes de planification composés d'actions indépendantes pour *Bob* et *Robot*. Pour cela, nous avons lancé la planification avec un problème réduit à la première phase du scénario. Nous avons ensuite fait varier la longueur des deux flux d'actions de chacun des deux agents et nous avons recensé la dimension des espaces de recherche produits. Les résultats obtenus sont regroupés dans le tableau 4.1. Il est important de noter que dans tous les cas présentés dans ce tableau la *totalité* des branches produites conduisent à un plan qui est rigoureusement le même. Le tableau 4.1 nous montre que

		Nombre d'actions de Robot						
		0	**1**	**2**	**3**	**4**	**5**	**6**
Nombre d'actions de Bob	**0**	X	1	1	1	1	1	1
	1	1	2	3	4	5	6	7
	2	1	3	6	10	15	21	28
	3	1	4	10	20	35	56	84
	4	1	5	15	35	70	126	210
	5	1	6	21	56	126	252	462
	6	1	7	28	84	210	462	924

TAB. 4.1 – Dimensions des espaces de recherche produits par des problèmes composés de « flux » d'actions indépendantes.

la dimension des espaces produits par la première phase du problème augmente très rapidement et ce même pour un nombre d'actions relativement restreint. Si nous revenons sur notre scénario global, la première phase du problème est composée de deux flux contenant six actions chacun, le tableau 4.1 nous indique donc que cette phase du problème global va créer 924 branches dans l'espace de recherche. Ainsi pour traiter le problème complet HATP devra explorer un espace correspondant à $924 \times T_2$ où T_2 représente l'espace de recherche de la seconde phase du problème global. Si la première phase du problème était traitée de manière séquentielle (i.e. en supprimant les divergences liées au parallélisme des actions) on ne produirait qu'une seule branche pour la première phase du problème se ramenant ainsi à un espace de recherche global de taille T_2.

Ce phénomène que nous venons de mettre en évidence sur un cas particulier peut se généraliser à toute situation dans laquelle un problème contient au moins deux tâches sans contraintes de précédence entre elles et dont les réalisations sont « indépendantes ». Plus ce phénomène apparaîtra de manière « tardive » (i.e. vers la fin du plan) plus l'explosion de l'espace de recherche sera limitée. L'exemple que nous avons pris est donc le cas le plus « défavorable ». Il est clair également que notre exemple utilise un nombre d'actions indépendantes très important, il est très probable que dans le cadre d'une planification réaliste ce nombre soit plus faible et donc que le phénomène soit moins marqué. Bien que cette « faille » de l'algorithme puisse sembler « anecdotique » il faut garder à l'esprit que, dans le cadre d'applications qui nous intéresse, HATP sera amené à planifier en utilisant des domaines où plusieurs agents seront présents et que ceux-ci pourront se voir « affecter » des actions individuelles. Cette situation est un terrain propice à l'apparition du phénomène que nous avons mis en avant.

4.10 Conclusion

Dans ce chapitre nous avons décrit le planificateur HATP qui a été spécifiquement conçu pour pouvoir prendre en compte la qualité sociale des plans qu'il génère. Nous avons détaillé la syntaxe permettant de décrire les éléments en entrée du planificateur. Nous avons illustré le pouvoir expressif de cette syntaxe en la comparant au langage de référence de la planification PDDL. Nous avons vu qu'il y a de nombreuses ressemblances entre les deux langages mais que

leurs pouvoirs expressifs divergent sur plusieurs points :

✓ le langage PDDL permet de décrire des formules logiques ciblant un ensemble de faits alors que HATP ne peut cibler que les faits liés aux paramètres de la description courante,

✓ le langage PDDL permet une gestion des aspects temporels beaucoup plus riches que HATP,

✓ le langage HATP permet une expression plus aisée des critères composant la métrique de notre approche,

✓ le langage PDDL définit un problème de planification comme un ensemble de buts à satisfaire alors qu'un problème HATP est basée sur la réalisation de tâches.

Nous avons également détaillé les choix techniques qui ont été opérés pour permettre une utilisation en ligne de HATP sur un robot réel. Enfin, nous avons décrit l'algorithme principal de notre planificateur et nous avons illustré sa « faiblesse ».

5 Résultats

Dans le chapitre précédent nous avons détaillé l'implémentation et les choix techniques opérés pour la conception du planificateur HATP. Ce planificateur a été conçu pour être utilisé en ligne sur un robot assistant réel, il intègre la définition de règles sociales au sein de son processus de planification permettant ainsi au robot de déterminer les plans les plus adaptés parmi l'ensemble des plans solutions. Nous nous proposons maintenant de valider l'approche développée grâce à un scénario de simulation puis de montrer les résultats obtenus avec HATP sur un robot réel dans le cadre d'un scénario réaliste d'une application robotique assistive.

5.1 Résultats en simulation

Dans un premier temps, nous nous proposons d'illustrer les résultats produits par HATP sur un scénario de simulation. L'analyse des résultats nous permettra de valider l'approche développée mais aussi de mesurer les performances de HATP. Nous avons retenu un scénario qui est une version « réduite » du scénario de l'exemple support du chapitre 3.

5.1.1 Scénario de simulation

La figure 5.1 illustre la situation de départ du scénario que nous allons utiliser. Robot se trouve à proximité de la porte, Bob est assis sur le sofa. Dans ce scénario, Bob souhaite boire un rafraîchissement. Pour cela, il faut qu'il obtienne la bouteille B qui se trouve sur la table et un verre V qui se trouve dans le placard. Le placard est légèrement en hauteur et sa porte est fermée. De plus, il faut que, une fois que Bob aura obtenu le verre et la bouteille, il retourne

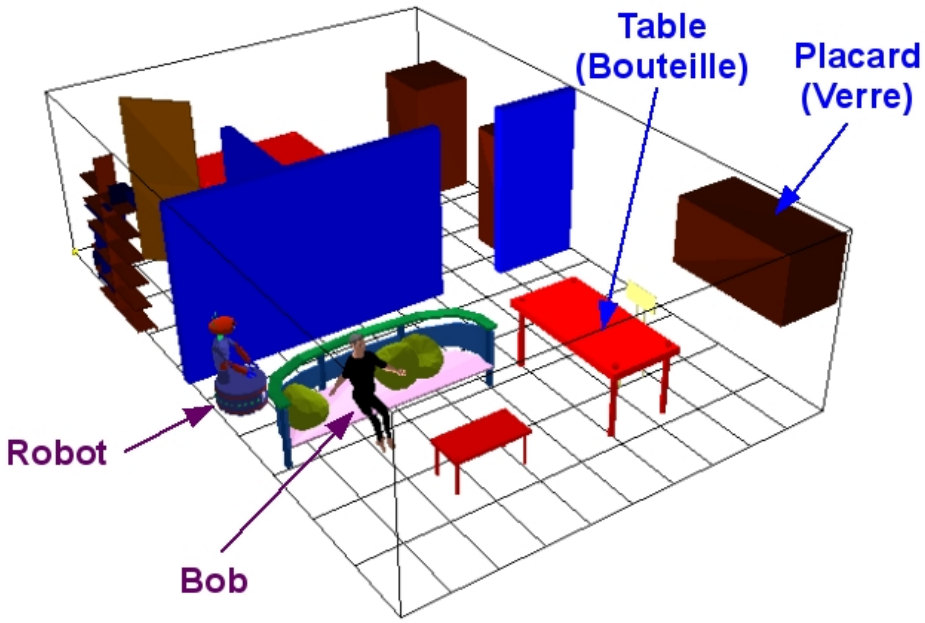

FIG. 5.1 – Situation de départ du scénario simulation.

au sofa s'il n'y est pas déjà. L'ensemble des actions qui peuvent être effectuées par les agents en présence sont : se déplacer, ouvrir/fermer la porte d'un meuble, prendre/poser un objet sur/dans un meuble et donner un objet à un autre agent.

Afin d'illustrer les aptitudes de HATP nous allons considérer plusieurs situations différentes du scénario de simulation. Dans un premier temps nous nous placerons dans une situation où Bob souhaite rester sur le sofa et n'est pas pressé d'être servi. Dans ce cas de figure, les coûts associés aux actions de déplacement de Bob seront plus élevés que les coûts associés à ces mêmes actions pour Robot. Nous nous placerons ensuite dans une situation plus complexe où Bob souhaite être servi rapidement et ce même s'il doit agir et se déplacer pour cela. Dans ce nouveau cas de figure les coûts des actions de Bob et de Robot seront similaires. Enfin, nous illustrerons les capacités de HATP à prendre en compte les contraintes imposées par son partenaire humain en se plaçant dans des situations où ce dernier impose la façon dont les tâches doivent être réalisées.

Par souci de clarté, dans la suite de chapitre les *fonctions de contribution à la dégradation de la qualité du plan* qui sont associées à l'ensemble des règles sociales seront appelées de manière succincte *fonctions de contribution*. On rappelle que ces fonctions renvoient des nombres positifs qui sont d'autant plus grands que la baisse de qualité du plan est importante. Dans ce scénario, les règles sociales introduites dans le domaine de planification sont les suivantes :

➢ la porte du placard ne doit pas rester ouverte inutilement. Cette règle est exprimée sous la forme d'un *état indésirable*. Celui-ci est associé à une fonction de calcul de la contribution qui renvoie 0 si la durée est inférieure à une dizaine de secondes. Au-delà de ce délai la contribution augmente linéairement. Dans l'éventualité d'un plan où la porte du placard serait toujours ouverte à la fin de celui-ci, la contribution est mise directement à la valeur maximale.

➢ on ne souhaite pas voir d'objets posés sur le sofa et ce même pour une durée très courte. Cette règle est exprimée sous la forme d'un *état indésirable*. Celui-ci est associé à une fonction de contribution qui renvoie un nombre positif important dès que cet *état indésirable* est présent dans le plan et ce quelle que soit sa durée.

➢ pour des raisons de clarté on ne souhaite pas produire de plans dans lesquels on cascade successivement deux actions de déplacement pour un même agent. Cette règle est exprimée à l'aide d'une *séquence indésirable* associée à une fonction renvoyant un nombre positif important dès que cette combinaison apparaît dans le plan.

➢ lorsque le robot doit transmettre un objet à un être humain il est préférable qu'il le fasse en lui donnant l'objet plutôt qu'en le déposant quelque part obligeant alors son partenaire humain à le prendre. Cette règle est transcrite à l'aide de deux *mauvaises décompositions*. Les actions *DéposerDans(Ag1, Obj, Meub)* et *DéposerSur(Ag1, Obj, Meub)* permettent à l'agent $Ag1$ de déposer l'objet Obj dans/sur le meuble $Meub$. Si une de ces actions apparaît dans le plan et qu'elle est liée hiérarchiquement à la tâche de haut-niveau *TransmettreUnObjet(Ag1, Ag2, Obj)*, la fonction de contribution de la *mauvaise décomposition* associée renvoie un nombre positif important.

➢ on souhaite éviter, dans la mesure du possible, les dépendances des actions de l'homme vis-à-vis du robot. Cette règle se concrétise par la déclaration de la règle sur les *liens croisés* concernant Bob. Cette règle est associée à une fonction de contribution qui renvoie un nombre positif proportionnel au nombre de liens d'interdépendance existants dans le plan.

➢ on souhaite également éviter au maximum les temps d'attente qui rendent l'homme dépendant du robot. Le domaine inclut donc la déclaration de la règle des *temps d'inactivité* concernant Bob et y associe une fonction de contribution qui renvoie un nombre positif important au cas où le plan courant contient un *temps d'inactivité* trop long pour l'homme. La perte de qualité est un petit peu plus faible mais reste conséquente si le plan contient plusieurs *temps d'inactivité* de durées plus réduites. Si la première action de l'homme ne débute pas à l'instant initial de début du plan, le temps menant du début du plan à la première action de l'homme n'est pas comptabilisé comme un *temps d'inactivité*.

Les priorités données à chaque règle sociale sont globalement similaires et approximativement égales au critère de *gestion des efforts*.

5.1.2 Plans produits par HATP

Première situation : l'homme est patient.

Dans cette situation, Bob accepte de patienter pour être servi et préfère, dans la mesure du possible, rester assis sur le sofa. Ce cas de figure se traduit par le fait que les coûts associés aux actions de Bob sont plus élevés que ceux associés aux actions de Robot. Étant donné les objectifs à atteindre dans le cadre du scénario de simulation retenu, le problème qui sera soumis à HATP est la liste de tâches partiellement ordonnées illustrée par la figure 5.2. La première étape consiste à s'assurer que Bob obtienne la bouteille B et le verre V, pour cela le problème contient

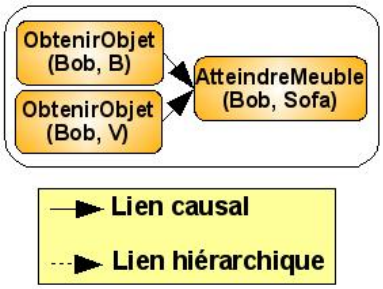

FIG. 5.2 – Premier problème de planification soumis à HATP. Dans ce problème aucune contrainte n'est imposée à HATP.

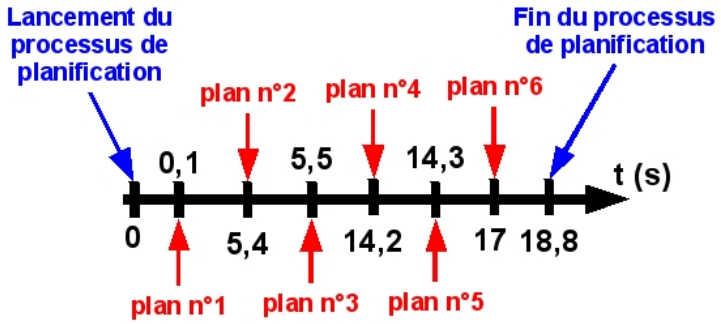

FIG. 5.3 – Évolution du plan de référence pour le premier problème où Bob est patient.

les deux tâches $ObtenirObjet(Bob, B)$ et $ObtenirObjet(Bob, B)$ sans contraintes de précédence entre elles. Une fois cette étape achevée, Bob devra atteindre le sofa s'il n'y est pas déjà. Pour cela, le problème inclut la tâche $AtteindreMeuble(Bob, Sofa)$ qui ne peut être réalisée que si les deux tâches précédentes ont été achevées. La tâche $AtteindreMeuble(Bob, Sofa)$ peut se décomposer de deux façons différentes : (1) Bob n'est pas au sofa à la fin de la première étape, dans ce cas une action de déplacement sera ajoutée au plan ou (2) Bob est déjà au sofa (i.e. les conditions décrivant le but associé sont vérifiées), dans ce cas cette tâche sera retirée de l'arbre de raffinement et n'apparaîtra donc pas dans le plan retenu.

Lors du lancement du processus de planification, HATP commence par établir un premier plan valide grâce à une recherche de type *profondeur d'abord*. Il poursuit ensuite l'exploration de l'espace de recherche en comparant le plan de référence avec le plan courant. Si ce dernier est préférable au plan de référence, il prend la place de celui-ci. L'échelle de temps de la figure 5.3 illustre l'évolution du plan de référence retenu par le processus de planification lorsque HATP reçoit le problème de la figure 5.2. Les figures 5.4 à 5.9 illustrent les différents plans de référence et la figure 5.10 fournit les scores associés aux règles sociales pour chacun de ces plans. Dans un souci de clarté, nous nous focaliserons uniquement sur les projections temporelles des plans retenus afin de limiter la quantité d'informations à analyser mais il faut conserver à l'esprit que tous les plans produits par HATP contiennent un arbre de raffinement. Dans toute la suite, il est important de noter que HATP peut produire des plans contenant des actions de Bob. Cela ne signifie pas que ces actions seront imposées au partenaire humain du robot mais que le plan produit permettra au robot de dialoguer sur la répartition des tâches entre les agents et/ou d'anticiper les actions de Bob et donc de contrôler le bon déroulement des évènements lors de l'exécution du plan.

Le premier plan de référence produit dans le cadre de cette variante du scénario est illustrée par la figure 5.4. On peut constater que, dans ce plan, c'est Bob qui réalise les différentes tâches du problème bien que les coûts de ses actions soient plus élevés. Cela s'explique par le fait que ce premier plan de référence est obtenu par une recherche en *profondeur d'abord*, la branche de l'espace de recherche à laquelle il appartient est donc la première explorée par HATP. Étant

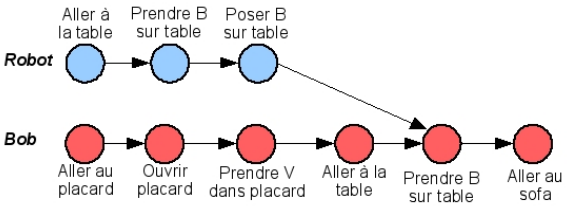

FIG. 5.4 – Premier plan de référence pour le premier problème où Bob est patient (plan n ° 1).

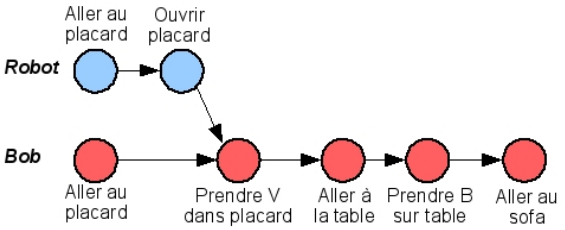

FIG. 5.5 – Second plan de référence pour le premier problème où Bob est patient (plan n ° 2).

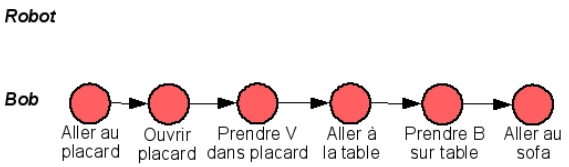

FIG. 5.6 – Troisième plan de référence pour le premier problème où Bob est patient (plan n ° 3).

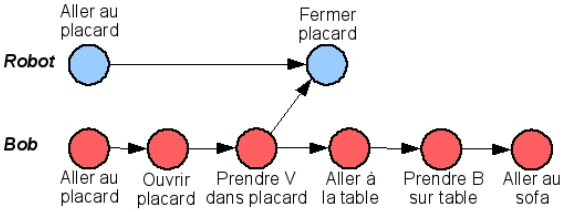

FIG. 5.7 – Quatrième plan de référence pour le premier problème où Bob est patient (plan n ° 4).

donné que dans ce plan Bob est amené à se déplacer, la quantité d'efforts à fournir est importante. De plus, on peut également noter que, dans ce plan, les actions réalisées par Robot sont inutiles et ne font qu'introduire une incompréhension de ses intentions. La conséquence directe de tout cela est un niveau élevé des efforts à fournir pour exécuter ce plan comme l'illustre la figure 5.10. Ce plan contient également la violation de certaines règles sociales :

✓ La présence d'un lien croisé ce qui provoque une dépendance des actions de Bob vis-à-vis de celles de Robot,

✓ Robot transmet un objet à Bob grâce à une combinaison d'actions « Déposer-Prendre » au lieu de le lui donner en mains propres,

✓ La porte du placard reste ouverte à la fin du plan.

Le second plan de référence produit par HATP est illustré par la figure 5.5. On peut noter que dans ce plan Bob est toujours amené à se déplacer, ceci s'explique par le fait que la partie de l'espace de recherche actuellement explorée par HATP correspond au fait que les décompositions retenues pour les deux tâches $Obtenir(Bob, V)$ et $Obtenir(Bob, B)$ font intervenir Bob de manière centrale et affecte un rôle « secondaire » à Robot. Ce nouveau plan de référence apporte une amélioration du point de vue des efforts à fournir pour l'exécuter. En effet, les actions de Robot ne sont plus inutiles ce qui supprime les efforts superflus qui étaient présents dans le plan précédent. Ce plan n'apporte toutefois aucune solution concernant la présence d'un lien croisé et le fait que le placard reste ouvert. On peut également déplorer l'apparition d'un temps d'inactivité important pour Bob qui se voit contraint de patienter près du placard en attendant que Robot l'ouvre.

Le planificateur étant toujours en cours d'exploration de la branche de l'espace de recherche

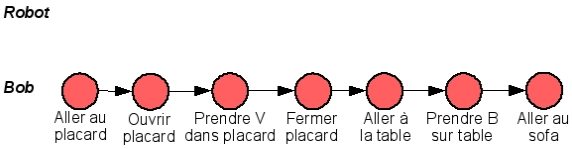

FIG. 5.8 – Cinquième plan de référence pour le premier problème où Bob est patient (plan n ° 5).

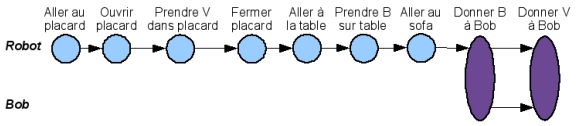

FIG. 5.9 – Sixième plan de référence pour le premier problème où Bob est patient (plan n ° 6).

où Bob joue le rôle principal et Robot un rôle secondaire le plan de référence suivant inclut uniquement des actions de Bob (voir figure 5.6). Comme l'illustre la figure 5.10 ce plan nécessite moins d'efforts que le plan précédent et ce malgré le fait qu'il contienne plus d'actions pour Bob. Cela s'explique par le fait que l'action d'ouverture du placard qui était préalablement réalisée par Robot nécessitait un déplacement de ce dernier. En effet, la combinaison d'actions « Déplacer-Ouvrir » du Robot requiert plus d'efforts que la seule action « Ouvrir » de Bob. De plus, le fait que toutes les actions soient attribuées à Bob permet de supprimer le lien croisé et le temps d'inactivité qui étaient présents dans le plan précédent. Toutefois, on peut constater que la règle sociale concernant l'ouverture de la porte du placard n'est toujours pas respectée.

Le plan de référence suivant produit par HATP est illustré par la figure 5.7. Celui-ci a pour base le plan précédent complété par deux actions de Robot qui permettent d'apporter une solution concernant la fermeture du placard. Les deux nouvelles actions introduites pour cela engendrent un « surplus » d'efforts à fournir. Cette augmentation est importante car le placard se situe en hauteur et que la fermeture de celui-ci par Robot est difficile. Concernant la fermeture du placard, on peut constater sur la figure 5.10 que le score associé à l'état indésirable de la « porte ouverte » est passé à 0. On obtient ainsi un plan dans lequel l'ensemble des règles sociales définies dans le domaine de planification sont respectées.

Le plan de référence suivant produit par HATP (voir figure 5.8) est obtenu selon le même « principe » que celui qui avait mené au plan de la figure 5.6. A la place des deux actions de Robot permettant de fermer le placard, il est moins « onéreux » du point de vue des efforts de faire fermer le placard à Bob puisque cela ne lui coûte qu'une action supplémentaire. Ainsi, on obtient un plan dans lequel aucune des règles sociales n'est violée et nécessitant moins d'efforts pour son exécution. Enfin, le planificateur HATP explore une branche de l'espace de recherche où Robot prend le « premier » rôle dans la réalisation des tâches $Obtenir(Bob, V)$ et $Obtenir(Bob, B)$ ce qui lui permet d'obtenir le nouveau plan de référence illustré par la figure 5.9. Dans ce plan, Robot se voit attribuer les actions qui étaient affectées à Bob dans le plan précédent ainsi que deux nouvelles actions « Donner » lui permettant de transmettre les objets qu'il a récupéré à Bob. On peut constater sur la figure 5.10 que la quantité d'efforts à fournir pour l'exécution de ce nouveau plan a fortement baissé et que celui-ci ne viole aucune des règles sociales définies dans le domaine.

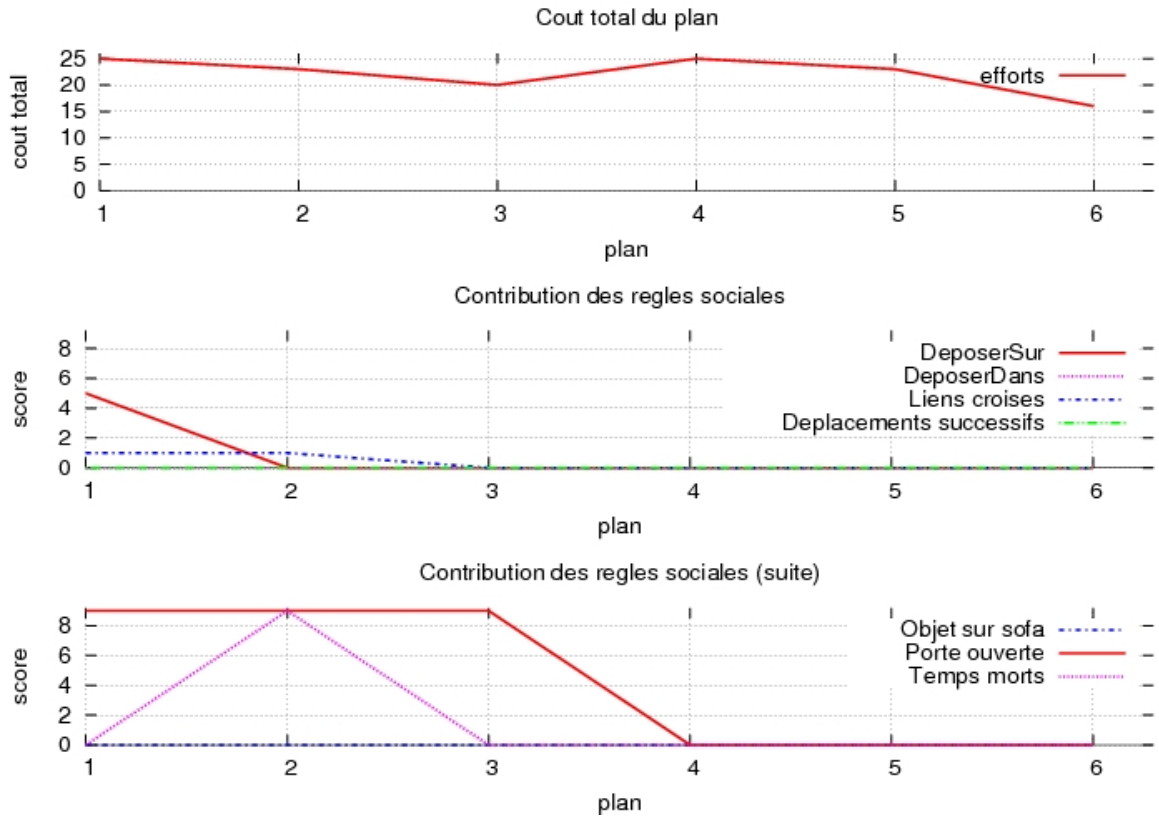

FIG. 5.10 – Scores des plans de référence pour le premier problème où Bob est patient.

On peut donc constater que pour ce problème HATP apporte une solution satisfaisante en environ 17 secondes. Ce temps peut être considéré comme acceptable étant donné que la réalisation des tâches formant le problème nécessitera quelques minutes. Nous justifierons plus rigoureusement cette durée ultérieurement lorsque nous nous intéresserons au dimensionnement de l'espace de recherche associé à ce scénario de simulation.

Seconde situation : l'homme accepte de participer.

Considérons maintenant la situation où Bob souhaite être servi relativement rapidement et qu'il accepte éventuellement de se déplacer pour participer à la réalisation des tâches. Dans ce cas de figure, les coûts associés aux actions de Bob et de Robot sont similaires ce qui rend la résolution plus complexe. En effet, dans cette situation, de nombreux plans vont voir leur « qualité » se rapprocher du meilleur plan possible formant ainsi un ensemble d'alternatives très proches les unes des autres. Ce cas de figure est artificiel, il correspond à une situation relativement peu vraisemblable mais qui montre la capacité de HATP à traiter des problèmes complexes.

Dans cette situation, le problème soumis à HATP reste le même (voir figure 5.2). L'évolution des plans de référence retenus par HATP est illustrée par la figure 5.11. On peut constater que l'évolution des plans retenus par HATP est très proche de celle associée à la situation précédente, la seule différence réside dans l'apparition de trois plans « intermédiaires » $i1$, $i2$

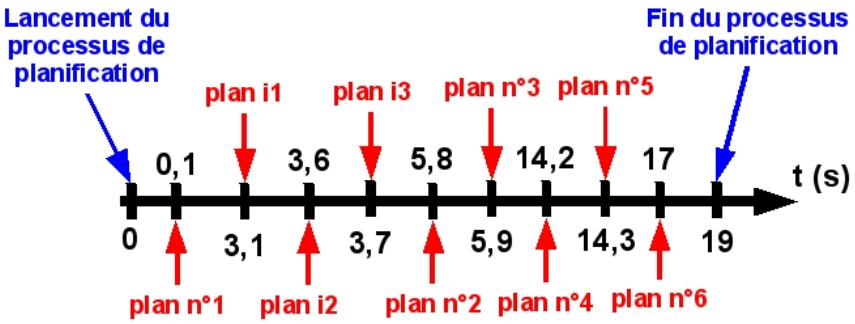

FIG. 5.11 – Évolution du plan de référence pour le premier problème où Bob accepte de participer.

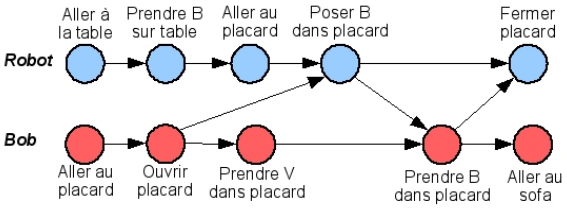

FIG. 5.12 – Premier plan intermédiaire pour le premier problème où Bob accepte de participer (plan $i1$).

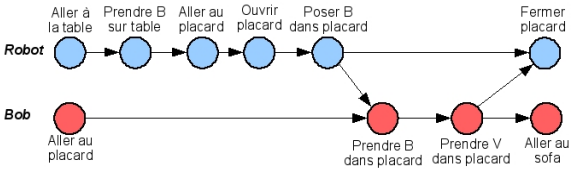

FIG. 5.13 – Second plan intermédiaire pour le premier problème où Bob accepte de participer (plan $i2$).

et $i3$. Les figures 5.12 à 5.14 illustrent ces plans intermédiaires. La figure 5.15 illustre les scores associés aux règles sociales des différents plans de références. La similarité des plans retenus peut s'expliquer par le caractère systématique de HATP dans l'exploration de l'espace de recherche. En effet, d'une exécution à l'autre l'espace de recherche sera exploré de la même façon, seule l'optimisation *branch-and-bound* peut influencer cela en supprimant certaines portions de cet espace. L'apparition des plans intermédiaires s'explique par la complexité de la situation. Nous nous proposons maintenant de détailler ceux-ci.

Le premier plan intermédiaire est illustré par la figure 5.12. Par rapport au premier plan de référence, on peut constater qu'il nécessite plus d'efforts et qu'il apporte des solutions à la violation de certaines règles sociales au détriment d'autres. Ainsi, on peut constater que, dans ce plan, la *mauvaise décomposition* décrivant une transmission d'objet par une combinaison «DéposerSur-PrendreSur» a un score nul mais que l'autre *mauvaise décomposition* est présente dans ce plan ce qui rétablit « l'équilibre ». Étant donné que ces deux règles sociales ont la même priorité dans la déclaration du domaine, le gain de qualité apporté par ce plan ne se situe pas à ce niveau. Par contre, on peut constater que ce plan permet de résoudre la problématique liée à la fermeture du placard mais qu'il provoque l'apparition d'un temps d'inactivité important pour Bob. Ainsi, le gain de qualité apporté par ce plan intermédiaire est minime.

Le second plan intermédiaire est illustré par la figure 5.13. On peut constater que, là aussi, le gain de qualité apporté par ce plan est relativement faible. En effet, ce nouveau plan nécessite moins d'efforts mais provoque une perte de qualité liée à l'augmentation du temps d'inactivité de Bob. On peut donc en conclure que ce plan est relativement similaire au plan précédent. On voit

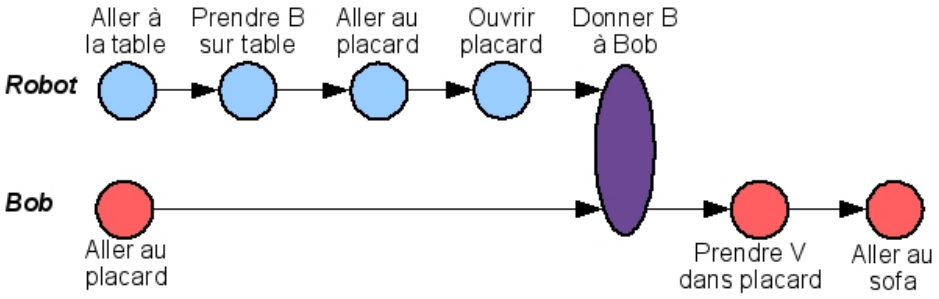

FIG. 5.14 – Troisième plan intermédiaire pour le premier problème où Bob accepte de participer (plan $i3$).

donc apparaître la complexité liée à la situation qui nous intéresse ici à savoir que l'ensemble des critères composant la métrique de qualité sociale des plans ont une importance similaire et que les différents plans solutions ont des estimations très proches les unes des autres. Enfin, le troisième et dernier plan intermédiaire produit par HATP est illustré par la figure 5.14. Là encore le plan produit est issu d'un compromis entre les différentes règles sociales. Les apports positifs de ce nouveau plan sont : (1) il nécessite moins d'efforts, (2) il supprime le lien croisé qui provoquait une dépendance de Bob vis-à-vis des actions de Robot et (3) il supprime la transmission d'un objet par une combinaison d'actions « DéposerDans-PrendreDans ». Cependant, ce plan porte un aspect négatif : la règle sociale portant sur la fermeture du placard est à nouveau violée. La suite du processus de planification va menée aux mêmes plans de références que ceux de la situation précédente. On aboutira donc à la même solution (voir figure 5.9) à savoir que Robot ira chercher le verre et la bouteille puis les donnera à Bob qui finalement n'aura pas besoin de se déplacer.

La figure 5.15 permet de constater que, malgré que le problème soit plus complexe que dans la situation précédente, le temps de calcul de HATP reste similaire. On peut même remarquer que la ressemblance entre les deux échelles de temps 5.3 et 5.11 est très marquée. En effet, on peut constater sur ces échelles que les changements de plans de référence sont « centrés » autour d'instants particuliers. Ainsi le premier plan de référence est obtenu en « temps constant » et l'exploration de l'espace de recherche se tourne vers de « meilleures zones » au bout d'environ 5.5, 14 et 17 secondes. La complexification apportée par la seconde situation provoque l'apparition d'une autre zone “clef” de l'espace de recherche autour d'environ 3.5 secondes. Ces similitudes s'expliquent par l'exploration systématique de l'espace de recherche par HATP. En effet, le planificateur effectue dans un premier temps un choix de décompositions pour les tâches de haut-niveau puis parcourt totalement le sous-espace de recherche lié à ce choix. Ainsi, il faut attendre que ce sous-espace ait été complètement exploré avant que HATP opte pour une autre décomposition des tâches de haut-niveau ouvrant ainsi la voie à des plans solutions d'une forme différente à ceux rencontrés jusque là. Il apparaît donc clairement que HATP gagnerait en rapidité s'il effectuait une exploration permettant une diversification rapide des plans rencontrés.

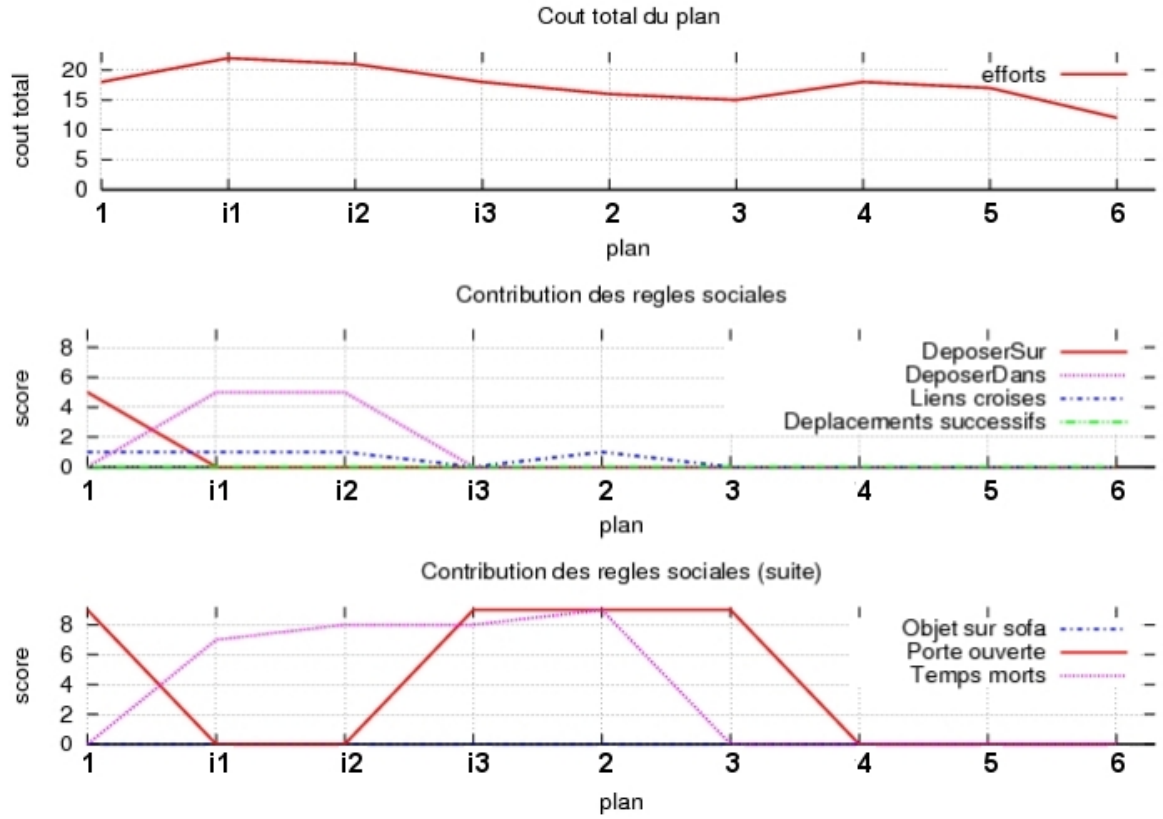

FIG. 5.15 – Scores des plans de référence pour le premier problème où Bob accepte de participer.

Troisième situation : l'homme impose des contraintes sur le plan.

Afin d'illustrer la capacité de HATP à prendre en considération les contraintes imposées par l'utilisateur sur la façon de réaliser les tâches de haut-niveau, considérons la situation où Bob indique la façon de procéder pour satisfaire les buts. Supposons que Bob exprime la requête suivante auprès de Robot : « J'ai soif, procure moi la bouteille et un verre ». Cette simple phrase peut être transcrite sous la forme d'un problème tel que celui illustré par la figure 5.16. Dans ce problème, chacune des tâches *ObtenirObjet* se décompose en une liste totalement ordonnée de deux tâches permettant à Robot de récupérer l'objet par lui-même dans un premier temps puis de le transmettre à Bob.

Lorsqu'on lance le processus de planification de HATP avec ce problème l'évolution du plan de référence retenu est illustré par l'échelle de temps de la figure 5.17. On peut constater que le temps nécessaire à HATP pour aboutir au meilleur plan possible est plus court que dans le cadre du premier problème. Ce gain de rapidité s'explique par le fait que les contraintes imposées par le partenaire humain du robot permettent de réduire fortement l'espace de recherche à parcourir par HATP. Cet aspect sera discuté ultérieurement. Les figures 5.18 à 5.21 illustrent les nouveaux plans de référence qui apparaissent et la figure 5.22 illustre l'évolution des scores composant l'évaluation sociale de ces plans.

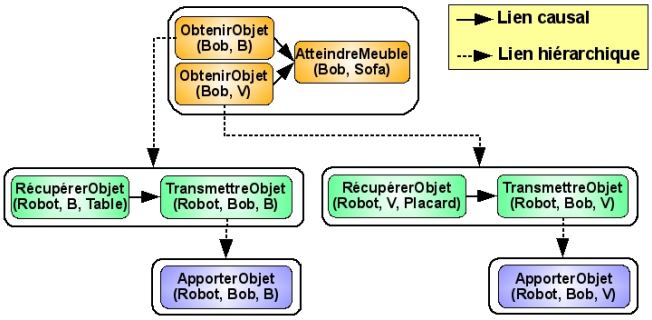

FIG. 5.16 – Second problème soumis à HATP.
Dans ce problème Bob impose des contraintes à HATP.

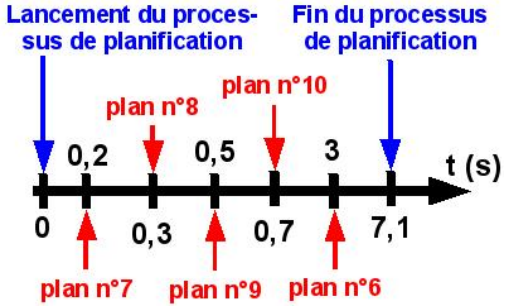

FIG. 5.17 – Évolution du plan de référence pour le second problème.

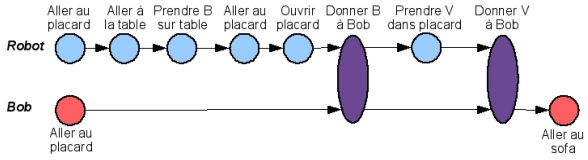

FIG. 5.18 – Premier plan de référence pour le second problème (plan n ° 7)

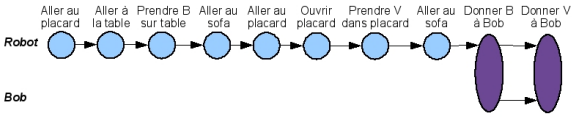

FIG. 5.19 – Second plan de référence pour le second problème (plan n ° 8)

Le premier plan de référence produit est illustré par la figure 5.18. On peut constater que ce plan est relativement proche du plan optimal trouvé avec le problème précédent. Toutefois, on peut noter que dans le cadre de la transmission des objets HATP a retenu une solution où Bob est amené à se déplacer jusqu'au placard pour recevoir la bouteille B et le verre V. Cette solution est relativement peu acceptable étant donné la requête émise par Bob. De plus, le déplacement de Bob provoque l'apparition d'un *temps d'inactivité* pour Bob qui va devoir attendre que Robot lui amène B et V. On peut également reprocher à ce plan la violation de la règle sociale portant sur l'état de la porte du placard et également la présence d'une combinaison de deux actions de déplacement successives. Le second plan de référence produit par HATP est illustré par la figure 5.19. L'amélioration principale apportée par ce plan est de supprimer le déplacement de Bob ce qui permet notamment d'annuler la perte de qualité qui était associée au *temps d'inactivité* de celui-ci. Toutefois, aucune solution n'est apportée pour le problème de la fermeture de la porte du placard et on peut déplorer l'apparition d'une deuxième combinaison de deux déplacements successifs.

Le plan de référence suivant produit par HATP est illustré par la figure 5.20. Ce plan permet de supprimer une des combinaisons d'actions provoquant deux déplacements successifs. Lors de la production du plan de référence suivant (voir figure 5.21) cette deuxième combinaison est supprimée ne laissant qu'une seule règle sociale encore violée : celle portant sur l'état de la porte du placard. Quelques secondes plus tard HATP détermine un nouveau plan de référence qui est le meilleur plan possible (voir figure 5.9) qui avait été trouvé avec le premier problème. On constate que le temps de planification est devenu très satisfaisant puisqu'il est maintenant d'environ 3s. Cet ordre de grandeur est largement acceptable car il permet une bonne réactivité du robot à un niveau symbolique en cas d'échec lors de l'exécution du plan.

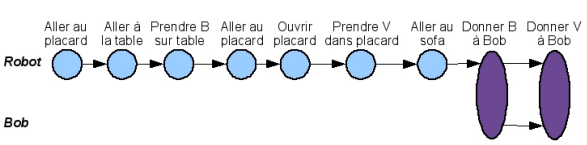

FIG. 5.20 – Troisième plan de référence pour le second problème (plan n ° 9)

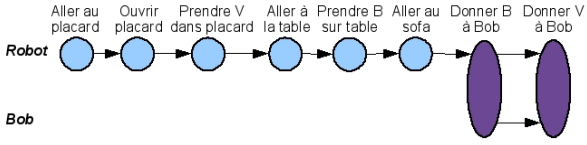

FIG. 5.21 – Quatrième plan de référence pour le second problème (plan n ° 10)

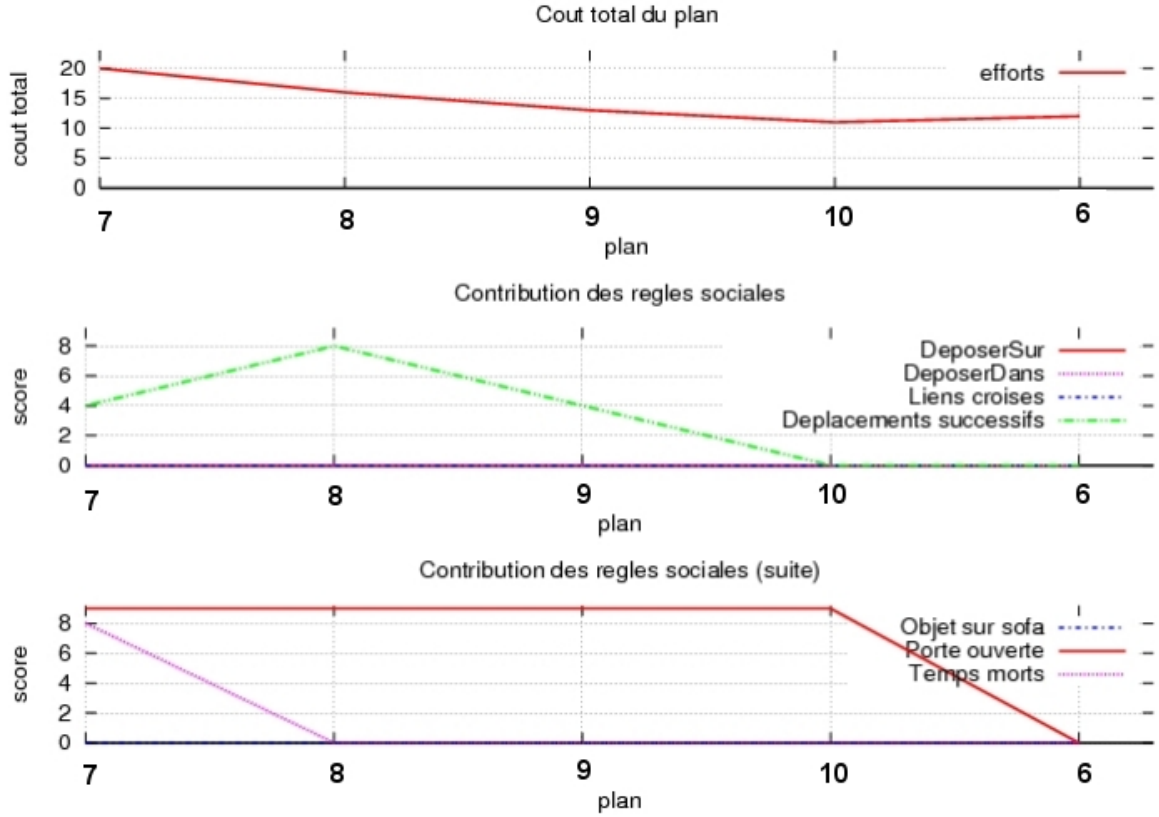

FIG. 5.22 – Scores des plans de référence pour le second problème.

Il est également possible que le partenaire humain impose des contraintes au robot qui peuvent conduire à un plan sous-optimal. Dans ce cas de figure, HATP produira le meilleur plan possible parmi ceux qui intègre les contraintes imposées par le partenaire humain du robot. Prenons l'éventualité où Bob exprime la requête suivante : « J'ai soif. Je vais chercher la bouteille, procure moi un verre. ». Ce type de requête peut se traduire sous la forme du problème illustré par la figure 5.23. Dans ce problème, la tâche *ObtenirObjet(Bob,V)* est décomposée de la même façon que dans le premier problème avec contraintes mais la tâche *ObtenirObjet(Bob,B)* se décompose cette fois-ci en une seule tâche permettant à Bob de récupérer lui-même l'objet. L'évolution du plan de référence retenu par HATP est illustré par l'échelle de temps de la figure 5.24. On constate là aussi que le temps de planification est très rapide tout comme dans le cas du second problème. Cette rapidité s'explique là encore par le fait que les contraintes imposées par Bob permettent de réduire fortement l'espace de recherche à explorer. Les figures 5.25 à 5.29 illustrent les plans de référence produits par HATP pour le troisième problème et la figure 5.30 montre les

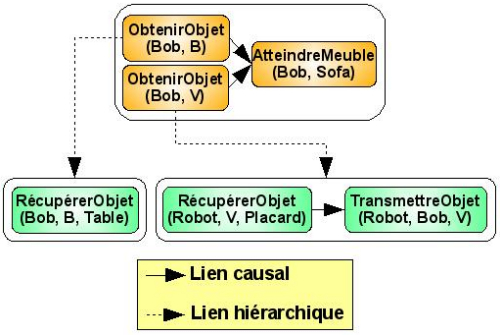

FIG. 5.23 – Troisième problème soumis à HATP. Ce problème contient des contraintes imposées par Bob à HATP. Ces contraintes vont conduire à explorer une partie différente de l'espace de recherche.

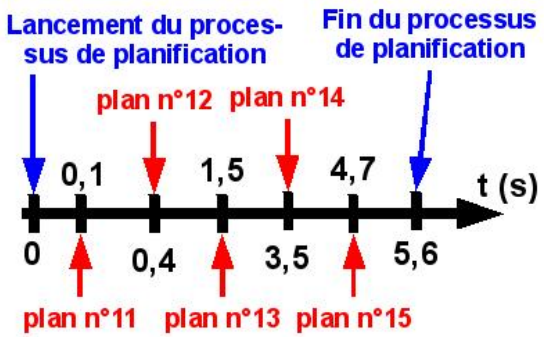

FIG. 5.24 – Évolution du plan de référence pour le troisième problème.

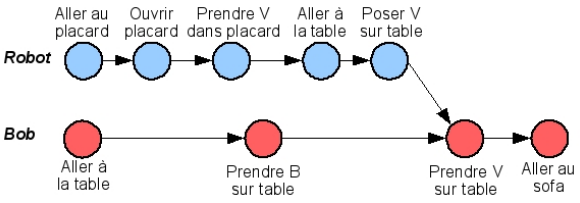

FIG. 5.25 – Premier plan de référence pour le troisième problème (plan n ° 11)

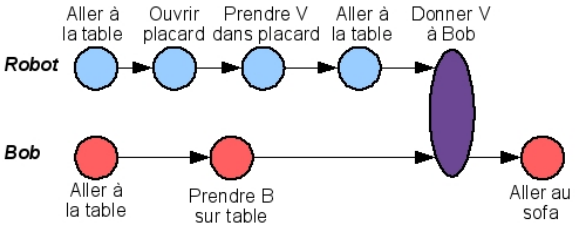

FIG. 5.26 – Deuxième plan de référence pour le troisième problème (plan n ° 12)

scores composant l'évaluation sociale de ces plans.

Le premier plan de référence retenu par HATP est illustré par la figure 5.25. On peut constater que ce plan prend bien en compte les contraintes imposées par Bob. On peut toutefois déplorer que la transmission de la bouteille B de Robot à Bob se fasse en déposant l'objet sur la table plutôt qu'en le donnant. De plus, ce plan viole la règle sociale sur l'état de la porte du placard, il présente des *temps d'inactivité* pour Bob et contient un *lien croisé*. Le plan de référence suivant est illustré par la figure 5.26. Ce nouveau plan permet d'améliorer la qualité du plan notamment en remplaçant la combinaison d'actions *Déposer-Prendre* par une action *Donner*. Ce remplacement permet de supprimer le *lien croisé* qui était présent dans le plan sans pour autant apporter de solutions aux deux autres règles sociales qui sont toujours violées.

Le plan de référence suivant retenu par HATP est illustré par la figure 5.27. Ce plan est considéré comme meilleur que le plan précédent parce qu'il annule la perte de qualité associée au critère des *temps d'inactivité*. En effet, le temps d'inactivité de Bob est considéré comme nul puisque toutes ses actions s'enchaînent les unes après les autres sans discontinuité et que le *temps d'inactivité* démarrant en début de plan n'est pas comptabilisé. Le plan de référence suivant est illustré par la figure 5.28. Ce nouveau plan est une version améliorée du plan de la figure 5.26 puisqu'il présente les mêmes avantages et qu'en plus il ne viole pas la règle sociale sur l'état de la porte du placard. Enfin, le dernier plan de référence produit par HATP est illustré par la figure 5.29. De la même manière que le cas précédent, ce nouveau plan est une version améliorée du

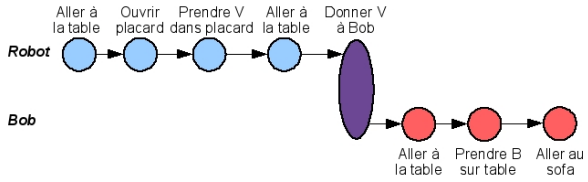

FIG. 5.27 – Troisième plan de référence pour le troisième problème (plan n ° 13)

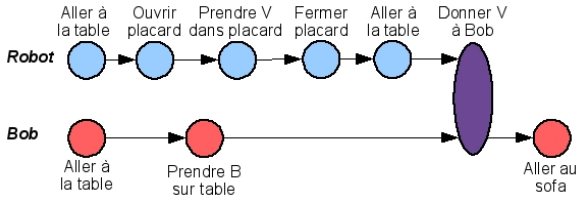

FIG. 5.28 – Quatrième plan de référence pour le troisième problème (plan n ° 14)

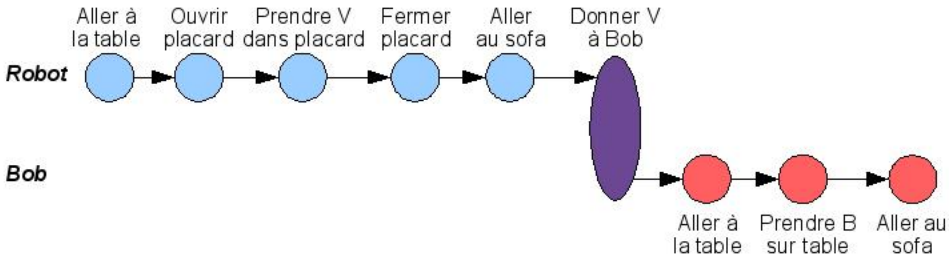

FIG. 5.29 – Cinquième plan de référence pour le troisième problème (plan n ° 15)

plan de la figure 5.27 puisqu'il conserve les avantages de celui-ci et qu'il ne viole pas la règle sur l'état de la porte du placard.

Analyse des dimensions des espaces de recherches.

Nous nous proposons de mesurer l'efficacité du planificateur HATP en se basant sur les dimensions des espaces de recherches associés aux trois problèmes de simulation auxquels il a été soumis. Pour obtenir ces informations nous avons désactivé l'optimisation *branch-and-bound* de HATP afin de lui permettre d'effectuer une exploration systématique de l'espace de recherche, nous l'avons ensuite soumis au trois problèmes précédents. Le tableau 5.1 récapitule les informations obtenues par ce procédé. Ce tableau contient :

- ✓ le nombre total de branches dans l'espace de recherche,
- ✓ le nombre de branches de l'espace de recherche menant à un plan invalide i.e. à un échec,
- ✓ le nombre de branches de l'espace de recherche menant à un plan valide i.e. à un succès,
- ✓ le nombre de plans valides *différents* présents dans l'espace de recherche. Deux plans sont considérés comme identiques s'ils ont la même projection temporelle *et* le même arbre de raffinement.

Ce tableau permet de mettre en évidence plusieurs points importants. Le premier point est le niveau de complexité associé au premier problème qui est un problème sans contraintes. En effet, on peut constater que dans cet espace de recherche le nombre de divergences est très important ce qui accroît considérablement sa dimension. Cela s'explique par le fait que si un problème inclut des tâches de haut-niveau sans contraintes de précédence entre elles, les divergences dues au parallélisme de ces tâches seront nombreuses provoquant ainsi un accroissement considérable de l'espace de recherche. Ce « phénomène » sera d'autant plus marqué que les tâches concernées auront un haut niveau dans la hiérarchie de l'HTN. En effet, plus une tâche est haute dans la

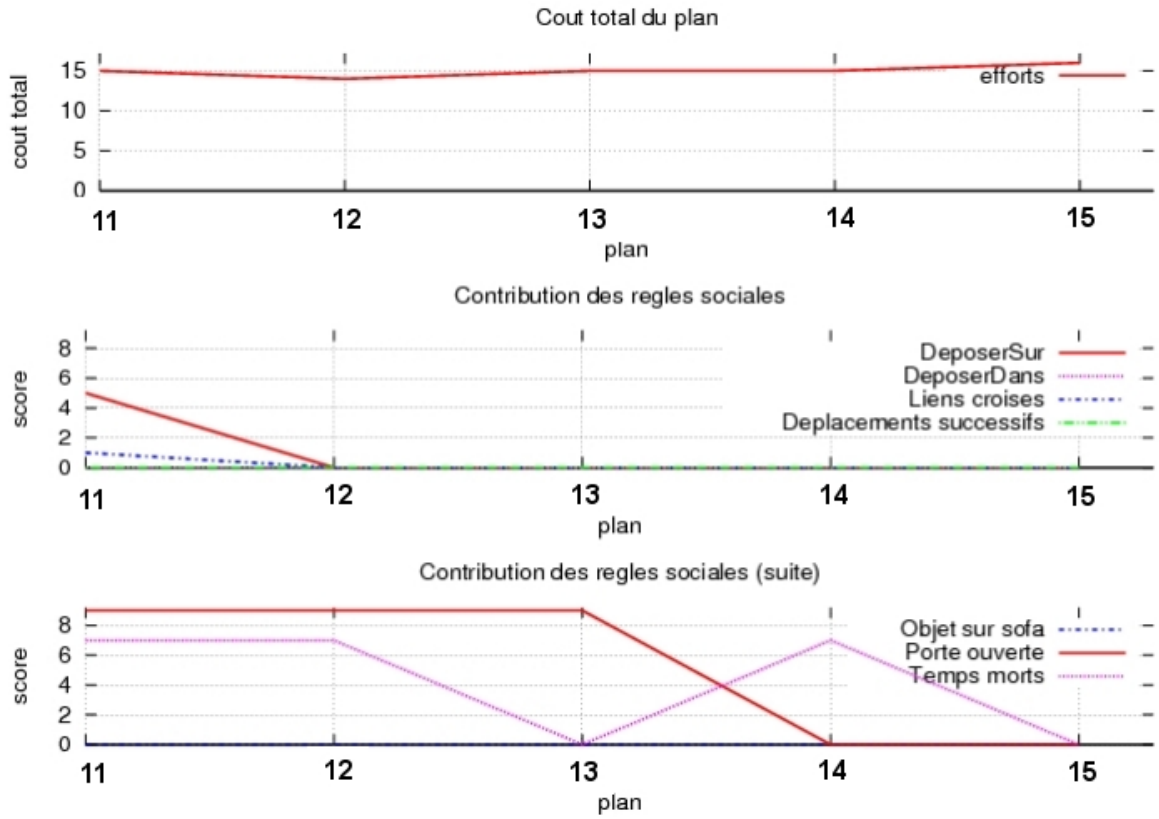

FIG. 5.30 – Scores des plans de référence pour le troisième problème.

	Nombre Total de Branches	Branches menant à un échec	Branches menant à un succès	Nombre de plans différents
Premier problème (figure 5.2)	$>$ 285 678	$>$ 118 816	$>$ 166 842	$>$ 10 080
Second problème (figure 5.16)	827	80	747	232
Troisième problème (figure 5.23)	4 722	1 261	3 461	297

TAB. 5.1 – Dimensions des espaces de recherche associés aux problèmes soumis à HATP.

hiérarchie plus il est probable que l'espace de recherche associé à cette tâche soit de dimension importante. On peut d'ailleurs constater que les espaces de recherche associés aux problèmes où l'homme impose des contraintes (problèmes 2 et 3) ont un nombre de divergences très réduit car les tâches qui engendrent des divergences de parallélisme sont de plus bas niveau dans l'HTN. Cela permet également de mettre en avant que les contraintes imposées par l'homme permettent de réduire de manière significative l'espace de recherche ce qui explique pourquoi les temps de planification se sont considérablement réduits lorsque HATP a été soumis aux problèmes 2 et 3.

Ce tableau permet également de mettre en évidence un phénomène de duplication de branches similaires. En effet, pour l'ensemble des problèmes on peut constater que le nombre de branches de l'espace d'exploration menant à un plan valide est nettement plus important que le nombre de plans valides différents. Cela signifie donc que dans l'espace de recherche il existe plusieurs branches menant à des plans identiques. Cela s'explique, là encore, par la présence d'un grand nombre de divergences liées au parallélisme de plusieurs tâches comme nous l'avons vu dans la section 4.9.

	Nombre de branches explorées	Branches coupées par BAB[1]	Branches menant à un échec	Branches menant à un succès	Nombre de plans différents
Premier problème (figure 5.2)	7 629	5 520	629	1 480	225
Second problème (figure 5.16)	657	217	46	394	117
Troisième problème (figure 5.23)	2 202	1 508	84	610	66

TAB. 5.2 – Espaces de recherche explorés par HATP pendant la planification.

Le tableau 5.2 récapitule les informations liées aux espaces de recherche explorés par HATP pendant la planification (i.e. avec l'optimisation *branch-and-bound* activée). Une colonne de données a été ajoutée afin de quantifier le nombre de branches de l'espace de recherche qui ont été coupées grâce à l'optimisation *branch-and-bound*. Ces données nous permettent de mettre en avant l'efficacité de cette optimisation. En effet, on peut constater que le nombre de branches

coupées est d'autant plus important que l'espace de recherche associé est grand. De plus, on peut remarquer que, dans le cadre du premier problème, l'optimisation *branch-and-bound* permet de limiter l'exploration de l'espace de recherche à, au maximum, $7629/285678 \approx 2,7\%$ de celui-ci. L'efficacité de cette optimisation est moins marquée dans le cadre des problèmes avec contraintes puisque ce pourcentage passe à $657/827 \approx 79,4\%$ pour le problème 2 et à $2202/4722 \approx 46,6\%$ pour le problème 3. Toutefois, cette optimisation apporte un vrai gain de rapidité au système car elle permet de réduire de manière non négligeable les espaces de recherche parcourus par HATP notamment en cas d'absence de contraintes imposées par l'homme.

5.2 Utilisation de HATP sur un robot réel

La section précédente nous a permis de mettre en avant la validité de notre approche en illustrant les résultats qu'elle peut produire sur un exemple de simulation correspondant à une situation concrète à laquelle un robot assistant autonome pourrait être confronté. Cette section nous a également permis de mettre en avant les capacités du planificateur HATP notamment du point de vue des temps de planification et de la possibilité pour le partenaire humain du robot d'imposer certaines contraintes sur la structure arborescente des plans solutions à retenir. Nous allons maintenant nous intéresser aux résultats qui ont été obtenus avec le planificateur HATP lors de son utilisation sur un robot réel.

Ces résultats ont été obtenus dans le cadre du projet européen COGNIRON. Ce projet portait sur l'étude des différentes spécificités liées à la problématique des robots assistants. Parmi les différents aspects abordés par ce projet on peut citer de manière non exhaustive : la perception de l'homme par le robot, l'apprentissage de tâches par démonstration, la manipulation d'objets, etc... Un des axes majeurs du projet ciblait les capacités décisionnelles des robots assistants. Dans le cadre de cet axe de recherche nous avons développé conjointement le planificateur HATP et un superviseur appelé SHARY. Les résultats qui vont être présentés ici ont été obtenus grâce à l'implantation de nombreux travaux issus du projet COGNIRON sur un même robot.

5.2.1 Architecture logicielle de contrôle

Le robot utilisé pour les expérimentations réelles s'appelle Jido et est illustré par la figure 5.31. Ce robot est composé d'une plateforme mobile contrôlée par deux roues motrices ainsi que d'un bras articulé à six degrés de liberté. La perception de l'environnement se fait par l'intermédiaire de plusieurs capteurs dont notamment un scanner laser orientable à l'avant, un scanner laser fixe à l'arrière, des capteurs d'efforts sur la pince, un banc de stéréovision fixe pour la manipulation d'objets, un banc de stéréovision orientable et une caméra 3D. Jido est également équipé d'un écran tactile lui permettant de communiquer avec son partenaire humain via une interface.

L'utilisation d'un planificateur sur un robot exige que le robot soit doté d'une architecture

[1] Branch-And-Bound

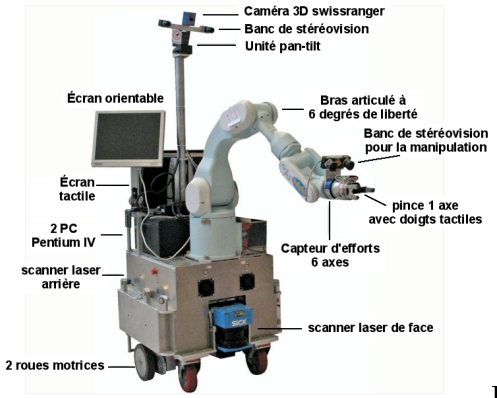

FIG. 5.31 – Le robot Jido.

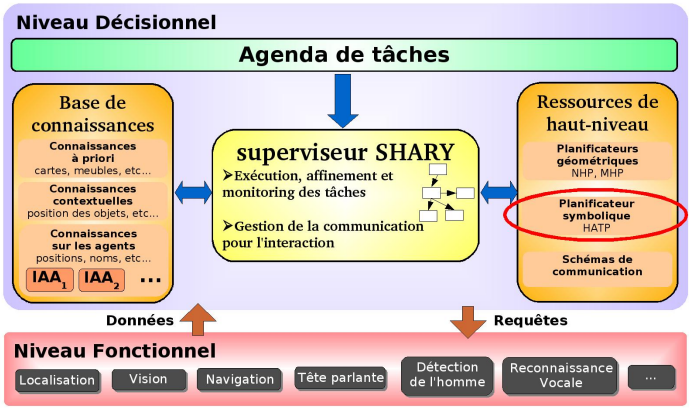

FIG. 5.32 – Architecture logicielle développée dans le cadre du projet COGNIRON.

logicielle adaptée permettant l'implantation de ce planificateur et son interopérabilité avec les autres éléments de l'architecture logicielle du robot. Nous avons donc développé une architecture logicielle [Clodic 05] pour le niveau délibératif de contrôle du robot (voir figure 5.32). Cette architecture permet une supervision adaptée à la problématique des robots assistants et un interfaçage avec un planificateur tel que HATP. Cette architecture se compose de deux niveaux : le niveau fonctionnel et le niveau délibératif. Le niveau fonctionnel est composé d'un ensemble de modules logiciels permettant de contrôler les actionneurs du robot et de transcrire les données fournies par les capteurs en informations exploitables par le niveau délibératif. Dans le cas de Jido, le niveau fonctionnel contient notamment un module de vision robotique permettant la détection et la reconnaissance de visages ainsi qu'un module de détection des positions des partenaires humains à l'aide des données fournies par les scanners lasers. Ces deux modules sont des atouts importants pour un robot interactif, leur fonctionnement est détaillé dans [Sisbot 06]. Le niveau délibératif est composé de quatre modules principaux :

1. *La base de connaissances* : elle est composée de connaissances de trois types. Les *connaissances a priori* représentent les données invariables concernant l'environnement dans lequel évolue le robot : la carte, les meubles en présence, les zones topologiques, les objets disponibles, etc... Ces connaissances sont des connaissances *fixes* dans le temps i.e. qui ne seront pas modifiées lors du déroulement de l'expérimentation avec le robot. Les *connaissances contextuelles* représentent les données portant sur l'environnement qui vont changées pendant le déroulement de l'expérimentation : position des objets, personnes en présence... Enfin, les connaissances portant sur les partenaires humains du robot sont représentées sous la forme d'IAAs (*InterAction Agents* en anglais). Pour chaque partenaire humain du robot, un IAA est créé et contient l'ensemble des informations portant sur le partenaire humain qu'il représente. Un IAA doit notamment contenir les attributs des entités de type *Agent* définies dans le domaine de HATP.
2. *L'agenda de tâches* : il est en charge de définir les tâches de haut-niveau qui doivent être réalisées. Ce module contient notamment le système de raisonnement permettant au robot de faire preuve d'initiative. En effet, ce module peut déterminer les tâches à accomplir

en fonction des requêtes émises par le partenaire humain du robot ou selon la situation courante. Ce module permet donc au robot de répondre aux ordres/suggestions de ses partenaires mais également de réagir à des situations spécifiques.

3. *Le superviseur* : le superviseur qui a été développé est appelé SHARY, son fonctionnement est détaillé dans [Clodic 07]. Il est en charge d'émettre les requêtes vers le niveau fonctionnel afin de permettre la réalisation des actions courantes. Il est également en charge de gérer la communication ciblant l'interaction homme-robot et de contrôler le bon déroulement de l'exécution des tâches courantes. Pour cela, SHARY reçoit le plan issu de HATP qu'il affine encore davantage afin de lui associer les requêtes destinées au niveau fonctionnel. Pendant l'exécution du plan, SHARY associe à chaque tâche de l'arbre de raffinement un ensemble de conditions de succès, d'échec et de pertinence conformément à la théorie de l'intention jointe (voir section 1.2.3.1). Ainsi le superviseur est capable de vérifier la pertinence de l'action courante à différents niveaux d'abstraction. Enfin, le dernier rôle du superviseur est de maintenir à jour la base de faits utilisée par HATP.
4. *Les ressources de haut-niveau* : cet ensemble contient les éléments de raisonnement du robot qui prennent explicitement en compte les partenaires humains du robot. Ces ressources incluent des planificateurs géométriques tels que le planificateur NHP (*Navigation in Human Presence* en anglais) [Sisbot 05] qui permet au robot de naviguer en prenant en compte explicitement la sécurité physique et le confort mental des partenaires humains du robot. Une autre ressource est le planificateur MHP (*Manipulation in Human Presence* en anglais) [Sisbot 08] qui permet au robot d'échanger des objets avec des partenaires humains de la façon la plus confortable possible. Les ressources de haut-niveau contiennent également les schémas de communication utilisés par le superviseur SHARY pour interpréter les données en provenance de son partenaire humain et les utiliser pour déterminer l'évolution de l'état de la tâche courante. Enfin, le planificateur symbolique utilisé par le robot est HATP.

Dans l'état actuel de l'implémentation de l'architecture, la base de connaissances et le superviseur SHARY sont écrits à l'aide du système de raisonnement procédural OpenPRS [Ingrand 96]. L'agenda n'est pas encore implanté mais son fonctionnement est simulé par l'injection manuelle de requêtes dans le système OpenPRS ce qui permet une émulation réaliste du module *Agenda de tâches*. Les schémas de communication sont également décrits sous une forme exploitable par OpenPRS. Les planificateurs géométriques NHP et MHP sont implantés sous forme de bibliothèques appelées par des modules fonctionnels. Le planificateur HATP est quant à lui implanté sous la forme d'un exécutable se connectant au système OpenPRS utilisé par SHARY et communiquant avec celui-ci par l'intermédiaire de messages sous forme de chaînes de caractères. Les requêtes échangées entre SHARY et HATP sont les suivantes :

✓ *Mise à jour d'un attribut* : cette requête est émise par SHARY vers HATP et permet de mettre à jour la valeur d'un attribut d'une entité de la base de faits de HATP.

✓ *Définition du problème* : cette requête est émise par SHARY vers HATP et permet de définir le problème de planification soumis à HATP.

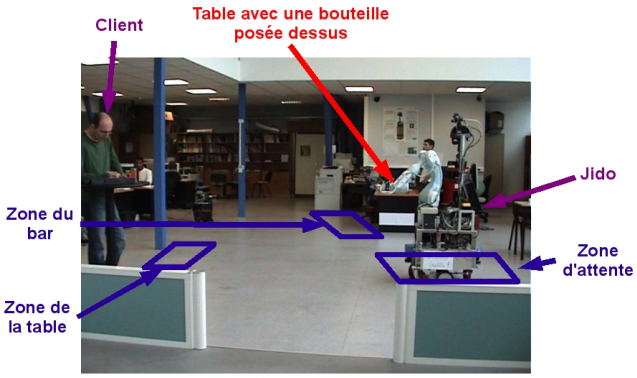

FIG. 5.33 – Scénario réel : situation de départ

FIG. 5.34 – Scénario réel : Jido détecte la présence d'un client.

FIG. 5.35 – Scénario réel : Jido propose à boire au client.

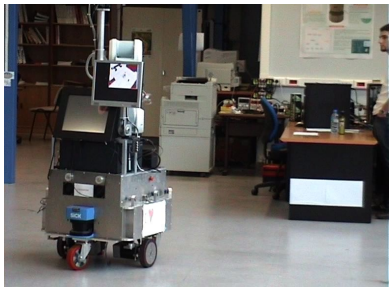

FIG. 5.36 – Scénario réel : Jido va chercher la bouteille au bar.

✓ *Lancement du processus de planification* : cette requête est émise par SHARY vers HATP et permet de lancer le processus de planification i.e. de déclencher le fonctionnement des threads de raffinement et d'évaluation de HATP.

✓ *Échec de la planification* : cette requête est émise par HATP vers SHARY quand aucun plan n'a pu être trouvé lors du processus de planification.

✓ *Retour du plan* : cette requête est émise par HATP vers SHARY et permet de transmettre le plan retenu par le processus de planification. Cette requête est composée d'un ensemble de messages permettant de transmettre la totalité de l'arbre de raffinement ainsi que les contraintes temporelles existantes entre les actions présentes dans la projection du plan.

On peut noter que dans les requêtes détaillées ci-dessus aucune ne permet de régler la fenêtre temporelle allouée au planificateur. Cette fonctionnalité n'a pas été utilisée dans cette phase expérimentale. En effet, lors de ces expérimentations les espaces de recherche explorés par HATP étaient suffisamment « petits » pour que leur exploration totale s'effectue en quelques dixièmes de seconde.

5.2.2 Scénario de l'expérimentation

La situation de départ du scénario réalisé est illustré par la figure 5.33. Dans cette situation l'objectif de Jido est d'apporter à boire aux clients qui en émettent le souhait. D'un point de vue symbolique, l'environnement est composé de trois zones : la zone d'attente dans laquelle Jido attend l'arrivée des clients, la zone de la table où peuvent apparaître les clients et la zone

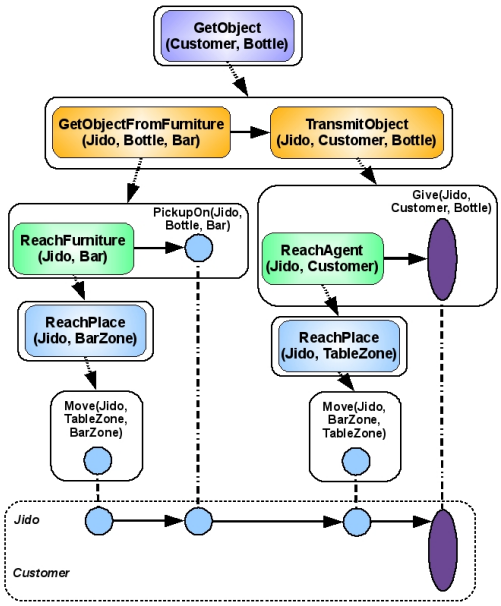

FIG. 5.37 – Scénario réel premier cas : plan solution produit par HATP

du bar où Jido peut se procurer les bouteilles à apporter aux clients. Les bouteilles que devra apporter Jido aux clients sont posées sur le bar. Dans ce scénario Jido attend qu'un client éventuel apparaisse aux environs de la table (figure 5.34), lorsque cela se produit il se déplace vers cette table pour proposer à boire au client qui s'y trouve (figure 5.35). Si ce dernier répond par l'affirmative Jido devra lui apporter la bouteille. Dans un premier temps, ce scénario a d'abord été élaboré en utilisant une bibliothèque de recettes représentant les plans de haut-niveau. Cela a permis de valider le fonctionnement des modules de perception ainsi que le fonctionnement du superviseur. Dans un second temps, afin de conférer à Jido une plus grande autonomie décisionnelle, la bibliothèque de recettes a été remplacée par le planificateur HATP. C'est cette partie de l'expérimentation que nous allons maintenant détailler.

Lors de la réalisation de ce scénario certaines fonctionnalités de contrôle du niveau fonctionnel n'étaient pas suffisamment robustes pour permettre une exécution purement autonome du scénario. Concrètement ce sont les actions de manipulation d'objets qui ont été émulées artificiellement. Ce scénario reste donc totalement utilisable pour valider le planificateur HATP étant donné que celui-ci fonctionne en *conditions réelles*.

5.2.3 Résultats obtenus

Le domaine utilisé par HATP pour la réalisation de ce scénario est celui qui a été utilisé dans le cadre du scénario de simulation de la section 5.1. Les différences principales résident dans les entités présentes dans la base de faits et dans les fonctions de calcul de coûts associées aux actions. Dans le cadre de ce scénario le coût associé au déplacement du client sera beaucoup plus important que celui associé à un déplacement du robot. Hormis ces différences la structure de l'HTN et l'ensemble des règles sociales sont identiques à celles utilisées dans le scénario de

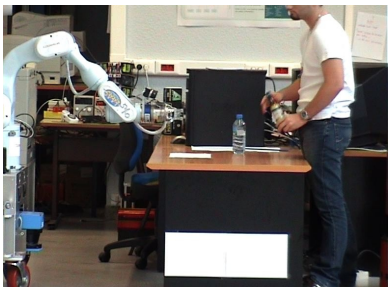

FIG. 5.38 – Scénario réel : Jido est en position d'attente pour la saisie de la bouteille.

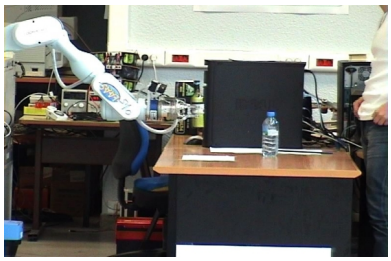

FIG. 5.39 – Scénario réel : Jido a saisi la bouteille.

FIG. 5.40 – Scénario réel : Jido apporte la bouteille au client.

FIG. 5.41 – Scénario réel : Jido tend la bouteille au client.

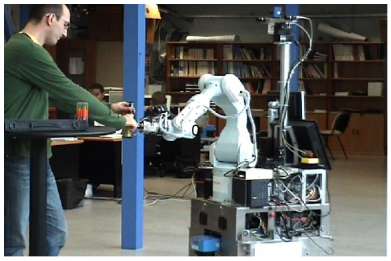

FIG. 5.42 – Scénario réel : Jido attend que le client se saisisse de la bouteille.

FIG. 5.43 – Scénario réel : Jido retourne à sa zone d'attente.

simulation.

Lorsque le client a confirmé à Jido qu'il souhaite boire. Une requête est émise du superviseur SHARY vers le planificateur HATP, cette requête contient un problème composé d'une seule tâche *GetObject(Customer, Bottle)*. Étant donné que l'environnement est très restreint l'espace de recherche est relativement faible et HATP produit le plan illustré par la figure 5.37 en moins de 0,2s. On peut constater que le plan retenu n'inclut aucun déplacement du client, que l'échange de la bouteille entre Jido et le client se fait par une action du type *Donner* plutôt que par une combinaison de type *Déposer-Prendre*. Ce plan est ensuite exécuté action après action. Le robot se déplace en direction de la table (figure 5.36) puis se saisit de la bouteille. Étant donné que le module fonctionnel de saisi d'objets par le bras robotisé n'était pas suffisamment robuste cette action a été simulée par la mise en position d'attente de Jido pendant une durée déterminée (figure 5.38), une fois cette durée écoulée Jido referme sa pince pour saisir la bouteille qui était auparavant tenue par un homme (figure 5.39). Il est important de préciser que l'homme tenant la bouteille n'est pas pris en compte par le robot. Grâce aux capteurs d'efforts disponibles sur sa pince Jido détecte la présence de la bouteille, le superviseur SHARY en déduit donc que l'action *PickupOn(Jido, Bottle, Bar)* est accomplie, il passe donc à l'action suivante présente dans le plan et se lance donc dans un déplacement (figure 5.40) le menant vers la zone de la table avant de donner la bouteille au client. Là encore le manque de robustesse de certains modules fonctionnels nous a conduit a simulé l'action *Give(Jido, Customer, Bottle)* par une prise de position spécifique du bras pendant une durée déterminée (figure 5.41), si pendant cette durée les capteurs d'efforts de la pince détecte des vibrations dues au fait que le client saisit la bouteille (figure 5.42) alors

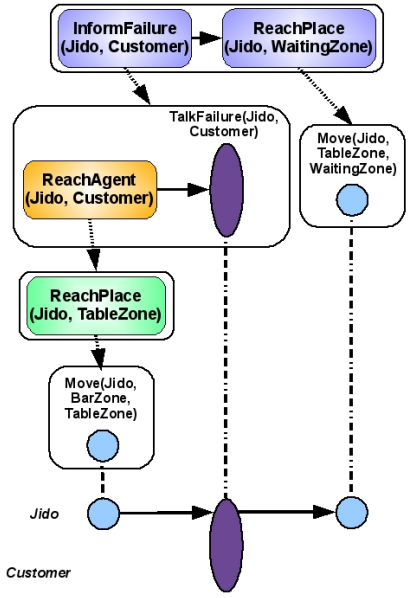

FIG. 5.44 – Scénario réel second cas : plan produit par HATP.

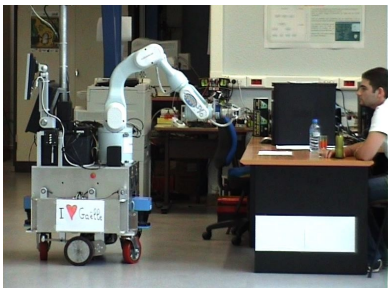

FIG. 5.45 – Scénario réel : Jido repart du bar sans la bouteille.

FIG. 5.46 – Scénario réel : Jido informe le client qu'il ne peut pas lui apporter à boire.

FIG. 5.47 – Scénario réel : Jido retourne à sa zone d'attente.

Jido ouvre sa pince. Son but courant étant achevé, Jido retourne dans la zone d'attente (figure 5.43).

Le scénario a également été utilisé pour tester la réactivité du robot en cas d'échec dans la réalisation d'une action. Ainsi, si lors de l'exécution de l'action *PickupOn(Jido, Bottle, Table)* la pince se referme mais ne détecte aucun objet le superviseur interprète cela comme un échec de l'action et donc une remise en cause du plan. Étant donné l'engagement pris auprès du client par Jido, ce dernier se doit de l'avertir (i.e. Jido doit établir la croyance mutuelle que le plan a échoué). Cela se traduit par l'émission d'un problème composé de deux tâches totalement ordonnées *InformFailure(Jido, Customer)* puis *ReachPlace(Jido, WaitingZone)*, dans ce cas le plan produit est très simple (voir figure 5.44) et est fourni de manière quasi instantanée. Jido exécute alors le nouveau plan en quittant la zone du bar (figure 5.45) pour se diriger vers la table afin de parler au client, il informe celui-ci de son incapacité à lui apporter une bouteille (figure 5.46) puis retourne dans sa zone d'attente (figure 5.47).

Ce scénario démontre l'aptitude du planificateur HATP à fonctionner en ligne de façon embarquée sur un robot interactif. Étant donné la complexité des différents modules fonctionnels à utiliser pour un tel scénario, certaines actions ont été simulées au niveau du superviseur mais d'une façon transparente pour HATP. Ce scénario démontre donc la possibilité pour HATP de maintenir sa base de faits à jour en fonction des informations transmises par le superviseur ainsi que sa capacité à générer des plans correspondants aux problèmes qui lui sont soumis. Les plans produits par HATP prennent explicitement en compte les règles sociales définies dans son domaine et permettent donc au robot de produire des plans adaptés à une situation où l'interaction homme-robot est un point clef de l'application.

5.3 Conclusion

Ce chapitre nous a permis de valider notre approche sur un scénario de simulation en mettant en avant l'aptitude du planificateur HATP à produire des plans respectant les règles sociales définies dans le domaine de planification. Ce chapitre a également permis de mettre en avant la difficulté de produire des plans socialement acceptables en temps réel dans le cadre d'applications de robots assistants. En effet, nous avons vu que les temps de planification pour des problèmes apparemment simples peuvent être relativement importants. Ceci est principalement dû à la dimension des espaces de recherche associés dont la taille peut exploser si le problème contient plusieurs tâches n'ayant aucune contrainte de précédence entre elles. Ce chapitre nous a également permis de montrer l'efficacité de l'optimisation *branch-and-bound* de l'algorithme HATP. Enfin, l'utilisabilité du planificateur HATP sur un robot réel a été illustrée dans le cadre d'un scénario réaliste. Même si ce scénario est relativement réduit il nous a permis d'expérimenter la possibilité pour HATP de recevoir des problèmes en temps réel et de produire des plans appropriés.

Conclusion et perspectives

Les travaux de cette thèse ont pour cadre les robots assistants. Cette problématique soulève de nombreux défis pour tous les aspects de la robotique et notamment pour les composantes décisionnelles. En effet, un robot assistant se doit d'agir de manière acceptable et prévisible de façon à ce que ses partenaires humains puissent comprendre ses « intentions ». Dans ce contexte, nous avons essayé de montrer l'intérêt de la planification de tâches. Cet outil confère au robot une grande autonomie décisionnelle en lui permettant de déterminer par lui-même les actions à réaliser pour satisfaire ses buts courants. Il participe donc directement à la « lisibilité » et à « l'acceptabilité » du comportement du robot.

Bien que cette thèse soit centrée sur la planification de tâches il est important de préciser que les travaux présentés ici ont été réalisés dans une démarche globale. Le planificateur HATP détaillé dans ce manuscrit a été développé conjointement avec le superviseur SHARY. L'objectif global était de fournir un framework complet facilitant la prise en compte explicite de l'homme dans les composantes décisionnelles des robots. Cette démarche nous a conduit à effectuer des choix spécifiques concernant notamment la forme des problèmes de planification et la structure des plans que doit produire le planificateur.

Tout au long de cette démarche nous avons essayé de donner un sens à la "lisibilité" et à "l'acceptabilité" du comportement du robot. Du point de vue du planificateur ces deux caractéristiques se traduisent par la forme des plans qu'il génère. L'approche proposée dans cette thèse consiste à introduire un ensemble de règles dans le processus de planification afin d'influencer la forme des solutions produites. Nous avons tenté d'établir un ensemble de règles/critères génériques permettant d'influer sur les comportements du robot à un haut niveau.

Le développement de notre approche s'est principalement porté sur les aspects symboliques de la planification. Les aspects temporels qui y sont liés n'ont été que peu abordés. Nous avons tenté d'améliorer le pouvoir expressif des planificateurs « traditionnels » en développant un langage permettant de décrire facilement les règles que nous avons définies. Nous avons également proposé une méthode permettant de comparer des plans en agrégeant les évaluations liées à ces différentes règles. Cette méthode permet notamment d'associer à chaque règle une priorité définissant son importance. Il devient alors possible d'adapter le comportement du robot à haut-niveau en renforçant certaines règles et en atténuant d'autres.

Dans une première étape, notre approche a été validée par l'intermédiaire de scénarios de simulation. Cela nous a permis de montrer la capacité de HATP à traiter des problèmes complexes en un temps acceptable tout en produisant des plans solutions « conformes » aux règles sociales introduites dans le domaine de planification. Dans un second temps, nous avons montré que HATP peut être utilisé en ligne sur un robot réel. Cela nous a notamment permis de mettre en avant sa capacité à fournir des structures de données exploitables par un superviseur dédié aux applications robotiques interactives.

Les principales perspectives de recherche qui apparaissent à l'issue de cette thèse portent sur l'utilisabilité de notre approche sur un robot réel et sur l'introduction d'un système d'apprentissage au sein du planificateur. Concernant le premier point, la rapidité avec laquelle HATP trouve un plan de bonne qualité est d'une grande importance puisqu'elle définit la réactivité du robot. Afin d'améliorer cette rapidité deux aspects peuvent être approfondis. Le premier concerne un problème inhérent à la planification de tâches, il s'agit de la limitation de l'explosion de l'espace de recherche. Nous avons vu dans le chapitre 4 que les plans produits par HATP peuvent contenir des tâches « parallèles » i.e. des tâches qui ne possèdent pas de liens de précédence entre elles, qui n'affectent pas les mêmes faits et qui ne concernent pas les mêmes agents. Or l'algorithme principal de HATP traite actuellement les tâches de manière séquentielle ce qui provoque une expansion importante de l'espace de recherche comme nous l'avons illustré dans la section 4.9. Pour pallier cet inconvénient il serait judicieux de doter le planificateur d'une procédure permettant de détecter les tâches qui sont complètement « indépendantes » les unes des autres. Une fois ces tâches identifiées, elles pourraient être traitées de manière totalement ordonnée par le processus de planification sans pour autant réduire la cardinalité de l'espace des plans solutions.

Le second aspect qui permettrait d'améliorer la rapidité du processus de planification est la conception d'une heuristique efficace. En effet, avec une heuristique adaptée HATP pourrait explorer les parties de l'espace de recherche les plus prometteuses en premier lieu et ainsi obtenir un plan de référence proche du meilleur plan possible très rapidement. Cela présenterait deux avantages importants : (1) dans le cas d'une planification avec une fenêtre temporelle limitée, HATP pourrait produire rapidement des plans de bonne qualité (socialement parlant) et (2) obtenir rapidement un plan de référence proche du meilleur plan ce qui permettrait à HATP d'appliquer plus efficacement l'optimisation *branch-and-bound* et donc de réduire le temps total nécessaire à l'exploration de l'espace de recherche.

La seconde perspective qui apparaît à l'issue de cette thèse est l'introduction d'un système d'apprentissage dans le planificateur HATP. En effet, l'approche que nous avons développée se base sur un ensemble de règles sociales prédéfinies combinées à des fonctions numériques spécifiques permettant leur évaluation. Dans le cadre d'une utilisation en situation réelle, au fur et à mesure des contacts avec ses partenaires humains, le robot devra être capable d'apprendre

et d'intégrer dans le processus de planification de HATP les éléments suivants :

✓ *les règles sociales.* Le robot devra être capable d'apprendre de nouvelles règles sociales sous une forme utilisable par HATP (*état indésirable*, *séquence indésirable*, etc...). Il s'agit d'un problème complexe. En effet, le système d'apprentissage devra être capable de déterminer par lui-même la forme la plus adaptée pour exprimer les différentes règles apprises tout en faisant évoluer leur description. Par exemple, l'apprentissage d'une *séquence indésirable* nécessite de déterminer les actions inhérentes à cette règle mais également la nature des liens causaux reliant ces actions sans oublier les contraintes qui doivent exister entre leurs paramètres.

✓ *les règles à appliquer et leurs priorités.* Selon son(ses) partenaire(s) courant(s) le robot devra déterminer par lui-même le sous-ensemble de règles sociales à utiliser et les priorités qui doivent y être associées afin d'adapter son comportement à la situation courante.

✓ *les fonctions numériques d'évaluation.* Les fonctions numériques d'évaluation des règles sociales et des coûts des actions devront également être apprises par le robot. Cet aspect de l'apprentissage est probablement le plus complexe puisqu'il nécessite de déterminer les attributs de la base de faits participant à chacune des évaluations mais également la façon dont ces attributs doivent être « combinés » pour produire cette évaluation.

Bibliographie

[Alami 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy.* Internatonal Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming, vol. 17, no. 4, 1998.

[Beynier 06] A. Beynier. *Une contribution à la résolution de Processus Décisionnels de Markov Décentralisés avec contraintes temporelles.* PhD thesis, 2006.

[Botelho 99] S. C. Botelho & R. Alami. *M+ : a Scheme for Multi-Robot Cooperation Through Negotiated Task Allocation and Achievement.* Dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 2, pages 1234 – 1239, 1999.

[Breazeal 04] C. Breazeal, G. Hoffman & A. Lockerd. *Teaching and Working with Robots as a Collaboration.* International Conference of Autonomous Agents and MultiAgent Systems (AAMAS), 2004.

[Clodic 05] A. Clodic, V. Montreuil, R. Alami & R. Chatila. *A Decisional Framework for Autonomous Robots Interacting with Humans.* Dans Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), 2005.

[Clodic 06] A. Clodic, S. Fleury, R. Alami, R. Chatila, G. Bailly, L. Brèthes, M. Cottret, P. Danès, X. Dollat, F. Eliseï, I. Ferrané, M. Herrb, G. Infantes, C. Lemaire, F. Lerasle, J. Manhes, P. Marcoul, P. Menezes & V. Montreuil. *Rackham : An interactive Robot-Guide.* Dans Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Septembre 2006.

[Clodic 07] A. Clodic. *Supervision pour un robot interactif : action et interaction pour un robot autonome en environnement humain.* PhD thesis, 2007.

[Cohen 90] P. R. Cohen & H. J. Levesque. *Intention is Choice with Commitment.* Artificial Intelligence, vol. 42, no. 2-3, pages 213–361, 1990.

[Cohen 91] P. R. Cohen & H. J. Levesque. *Teamwork.* Nous, vol. 25(4), pages 487–512, 1991.

[Dauthenhahn 05] K. Dauthenhahn, S. Woods, C. Kaouri, M. Walters, K.L. Koay & I. Werry. *What is a Robot Companion - Friend, Assistant or Butler ?* Dans Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), 2005.

[Doniec 06] A. Doniec, S. Espié, R. Mandiau & S. Piechowiak. *Multi-Agent Coordination and Anticipation Model to Design a Road Traffic Simulation Tool.* Dans Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS), Decembre 2006.

[Dorais 98] G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell & D. Schreckenghost. *Adjustable Autonomy for Human-Centered Autonomous Systems on Mars.* Dans Proceedings of the 1st International Conference of the Mars Society, pages 397–420, 1998.

[Fong 01a] T. Fong & C. Thorpe. *Advanced Interfaces for Vehicule Teleoperation : Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools.* Autonomous Robots, vol. 11, no. 1, pages 77–85, Juillet 2001.

[Fong 01b] T. Fong, C. Thorpe & C. Baur. *Collaboration, Dialogue, and Human-Robot Interaction.* Dans Proceedings of the 10th International Symposium of Robotics Research, 2001.

[Fong 06] T. Fong, C. Kunz, L. M. Hiatt & M. Bugajska. *The Human-Robot Interaction Operating System.* Dans Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-Robot Interaction (HRI), pages 41–48, New York, NY, USA, 2006. ACM Press.

[Forman 01] E. Forman & M. A. Selly. *Decision by Objectives (How to convince others that you are right).* World Scientific, 2001.

[Fox 03] M. Fox & D. Long. *PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains.* Journal of Artificial Intelligence Research (JAIR), vol. 20, pages 61–124, 2003.

[Galindo 04] C. Galindo, J. Gonzalez & J.A. Fernandez-Madrigal. *Interactive Task Planning in Assistant Robotics.* Dans Proceedings of International IFAC Symposium of Intelligent Autonomous Vehicules (IAV), Juillet 2004.

[Gallien 06] M. Gallien & F. Ingrand. *Controlability and Makespan Issues with Robot Action Planning and Execution.* Dans Proceedings of the International Workshop on Planning and Scheduling For Space (IWPSS), Octobre 2006.

[Gerevini 06] A. Gerevini & D. Long. *Preferences and Soft Constraints in PDDL3.* Dans Proceedings of ICAPS Workshop on Planning with Preferences and Soft Constraints, pages 46–53, 2006.

[Ghallab 94] M. Ghallab & H. Laruelle. *Representation and Control in IxTeT, a Temporal Planner.* Dans Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS), pages 61–67, 1994.

[Ghallab 98] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld & D. Wilkins. *PDDL - The Planning Domain Definition Language.* AIPS-98 Planning Competition Commitee, 1998.

[Ghallab 04] M. Ghallab, D. Nau & P. Traverso. *Automated Planning - Theory and Practice.* Morgan Kaufmann Publishers, ed. Elvesier, 2004.

[Gockley 04] R. Gockley, R. Simmons, J. Wang, D. Busquets, C. DiSalvo, K. Caffrey, S. Rosenthal, J. Mink, S. Thomas, W. Adams, T. Lauducci, M. Bugajska, D. Perzanowski & A. Schultz. *Grace and George : Social Robots at AAAI.* AAAI 2004 Mobile Robot Competition Workshop, Août 2004.

[Gockley 05] R. Gockley, A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, R. Simmons, K. Snipes, A. Schultz & J. Wang. *Designing Robots for Long-Term Social Interaction.* Dans Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Août 2005.

[Gockley 06] R. Gockley, J. Forlizzi & R. Simmons. *Interactions with a Moody Robot.* Dans Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI), pages 186–193. ACM Press, 2006.

[Goodrich 01] M. Goodrich, D. Olsen, J. Crandall & T. Palmer. *Experiments in Adjustable Autonomy.* Dans Proceedings of IJCAI Workshop on Autonomy, Delegation and Control : Interacting with Intelligent Agents, 2001.

[Gravot 01] F. Gravot & R. Alami. *An Extension of the Plan-Merging Paradigm for Multi-Robot Coordination.* Dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 3, pages 2929 – 2934, 2001.

[Grosz 96] B. J. Grosz & S. Kraus. *Collaborative Plans for Complex Group Action.* Artificial Intelligence, vol. 86, pages 269–358, 1996.

[Grosz 99] B. Grosz & S. Kraus. *The Evolution of SharedPlans.* Foundations and Theories of Rational Agencies, 1999.

[Hall 66] E. T. Hall. *The Hidden Dimension.* New York : Doubleday, 1966.

[Hoffman 06] G. Hoffman & C. Breazeal. *What Lies Ahead ? Expectation Management in Human-Robot Collaboration.* Dans To Boldly Go Where No Human-Robot Team Has Gone Before : Papers from the 2006 AAAI Spring Symposium, 2006.

[Huttenrauch 02] H. Huttenrauch & K. Severinson-Eklund. *Fetch-and-carry with CERO : Observations from a Long-Term User Study.* Dans Proceedings of the IEEE International Workshop on Robot and Human Communication (RO-MAN), 2002.

[Ilghami 03] O. Ilghami & D. Nau. *A General Approach to Synthesize Problem-Specific Planners.* Rapport technique CS-TR-4597, Department of Computer Science, University of Maryland, Octobre 2003.

[Ingrand 96] F. F. Ingrand, R. Chatila, R. Alami & F. Robert. *PRS : A High Level Supervision and Control Language for Autonomous Mobile Robots.* Dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 43–49, 1996.

[Jensen 02] B. Jensen, G. Froidevaux, X. Greppin, A. Lorotte, L. Mayor, M. Meisser, G. Ramel & R. Siegwart. *Visitor Flow Management using Human-Robot Interaction at Expo.02.* Dans Proceedings of the Workshop Robots and Exhibitions during IEEE International Conference of Robots and Systems (IROS), 2002.

[Kanda 05] T. Kanda & H. Ishiguro. *Communication Robots for Elementary Schools.* Dans Proceedings of Workshop on Robot Companion at AISB, 2005.

[Kawamura 03] K. Kawamura, P. Nilas & K. Mugumura. *An Agent-Based Architecture for an Adaptive Human-Robot Interface.* Proceedings of the 36th Hawaii International Conference on System Sciences (HCISS), Janvier 2003.

[Kennedy 07] W.G. Kennedy, M. D. Bugajska, M. Marge, W. Adams, B. R. Fransen, D. Perzanowski, A. C. Schultz & J. G. Trafton. *Spatial Representation and Reasoning for Human-Robot Collaboration.* Dans Proceedings of AAAI Conference, pages 1554–1559, 2007.

[Klein 04] G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman & P. J. Feltovich. *Ten Challenges for Making Automation a "Team Player" in Joint Human-Agent Activity.* IEEE Intelligent Systems, vol. 19, no. 6, pages 91–95, 2004.

[Kvarnstrom 01] J. Kvarnstrom & P. Doherty. *TALplanner : A Temporal Logic Based Forward Chaining Planner.* Annals of Mathematics and Artificial Intelligence, vol. 30, pages 119–169, 2001.

[Levesque 90] H. J. Levesque, P. R. Cohen & J. H. T. Nunes. *On Acting Together.* Proceedings of the Annual Meeting of the American Association for Artificial Intelligence (AAAI), 1990.

[Lochbaum 98] K. E. Lochbaum. *A Collaborative Planning Model of Intentional Structure.* Computational Linguistics, vol. 24, no. 4, pages 525–572, 1998.

[Lund 03] H.H. Lund. *Adaptive Robotics in the Entertainment Industry.* Dans Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), 2003.

[Malfaz 04] M. Malfaz & M.A. Salichs. *A New Architecture for Autonomous Robots Based on Emotions.* Dans Proceedings of 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), Juillet 2004.

[Mateas 02a] M. Mateas. *Interactive Drama, Art, and Artificial Intelligence.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, December 2002.

[Mateas 02b] M. Mateas & A. Stern. *A Behavior Language for Story-based Believable Agents.* Dans Artificial Intelligence and Interactive Entertainment, AAAI symposium, Mars 2002.

[Mateas 04] M. Mateas & A. Stern. *A Behavior Language : Joint Action and Behavioral Idioms.* H. Prendinger and M. Ishizuka (Eds), Life-like Characters. Tools,

Affective Functions and Applications, Springer, 2004.

[Montreuil 05] V. Montreuil & R. Alami. *Report on Paradigms for Decisional Interaction.* Rapport technique, LAAS-CNRS, 2005.

[Myers 97] K. Myers. *Abductive Completion of Plan Sketches.* Dans Proceedings of the 14th National Conference on Artificial Intelligence. AAAI Press, 1997.

[Myers 02] K. L. Myers, W. M. Tyson, M. J. Wolverton, P. A. Jarvis, T. J. Lee & M. Desjardins. *PASSAT : A User-centric Planning Framework.* Dans Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space, Octobre 2002.

[Myers 03] K. L. Myers, P. Jarvis, W. M. Tyson & M. Wolverton. *A Mixed-initiative Framework for Robust Plan Sketching.* Dans Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pages 256–266. AAAI, 2003.

[Nau 03] D. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu & F. Yaman. *SHOP2 : An HTN Planning System.* Journal of Artificial Intelligence Research, pages 379–404, Decembre 2003.

[Nourbakhsh 99] I. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer & A. Soto. *An Affective Mobile Robot Educator with a Full-Time Job.* Artificial Intelligence, vol. 114, no. 1–2, pages 95–124, 1999.

[Pacchierotti 05] E. Pacchierotti, H. I. Christensen & P. Jensfelt. *Human-Robot Embodied Interaction in Hallway Settings : a Pilot User Study.* Dans Proceedings of the 14th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), 2005.

[Pollack 02] M. Pollack, C. McCarthy, S. Ramakrishnan, I. Tsamardinos, L. Brown, S. Carrion, D. Colbry, C. Orosz & B. Peintner. *Autominder : A Planning, Monitoring, and Reminding Assistive Agent.* Dans Proceedings of the 7th International Conference on Intelligent Autonomous Systems (IAS), 2002.

[Pollack 03] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan & I. Tsamardinos. *Autominder : An Intelligent Cognitive Orthotic System for People with Memory Impairment.* Robotics and Autonomous Systems (RAS), 2003.

[Rich 97] C. Rich & C. L. Sidner. *COLLAGEN : when Agents Collaborate with People.* Dans Proceedings of the 1st International Conference on Autonomous Agents (AGENTS), pages 284–291. ACM, 1997.

[Rich 98] C. Rich & C. L. Sidner. *COLLAGEN : A Collaboration Manager for Software Interface Agents.* User Modeling and User-Adapted Interaction, vol. 8, no. 3-4, pages 315–350, 1998.

[Saaty 99] T. L. Saaty. *Decision Making for Leaders : The Analytic Hierarchy Process for Decisions in a Complex World.* RWS Publications, 1999.

[Saaty 00] T.L. Saaty. *Fundamentals of the Analytic Hierarchy Process.* RWS Publications, 4922 Ellsworth Avenue, Pittsburgh, PA 15413, 2000.

[Salichs 06] M.A. Salichs, R. Barber, A.M. Khamis, M. Malfaz, J.F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales & E. Delgado. *Maggie : A Robotic Platform for Human-Robot Social Interaction.* Dans Proceedings of IEEE International Conference on Robotics, Automation and Mechatronics (RAM), Juin 2006.

[Scerri 03] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si & M. Tambe. *A Prototype Infrastructure for Distributed Robot-Agent-Person Teams.* Dans Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2003.

[Scerri 04] P. Scerri, D. Pynadath, N. Schurr, A. Farinelli, S. Gandhe & M. Tambe. *Team Oriented Programming and Proxy Agents : The Next Generation.* Dans Proceedings of 1st International Workshop on Programming Multiagent Systems, 2004.

[Sidner 03a] C. L. Sidner & C. Lee. *An Architecture for Engagement in Collaborative Conversations between a Robot and Humans.* Rapport technique TR2003-13, Mitsubishi Electric Research Labs, Avril 2003.

[Sidner 03b] C. L. Sidner, C. Lee & N. Lesh. *Engagement Rules for Human-Robot Collaborative Interaction.* Dans Proceedings of the International Conference on Systems, Man and Cybernetics (SMC), volume 4, pages 3957–3962, Octobre 2003.

[Simmons 03] R. Simmons, D. Goldberg, A. Goode, M. Montemerlo, N. Roy, B. Sellner, C. Urmson, A. Schultz, M. Abramson, W. Adams, A. Atrash, M. Bugajska, M. Coblenz, M. MacMahon, D. Perzanowski, I. Horswill, R. Zubek,

D. Kortenkamp, B. Wolfe, T. Milam & B. Maxwell. *GRACE : An Autonomous Robot for the AAAI Robot Challenge.* vol. 24, no. 2, pages 51–72, 2003.

[Sisbot 05] E. A. Sisbot, R. Alami, T. Simeon, K. Dautenhahn, M. Walters, S. Woods, K. L. Koay & C. Nehaniv. *Navigation In Presence of Humans.* Dans Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2005.

[Sisbot 06] E. A. Sisbot, A. Clodic, L. F. Marin Urias, M. Fontmarty, L. Brethes & R. Alami. *Implementing a Human-Aware Robot System.* Dans Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Septembre 2006.

[Sisbot 07] E. A. Sisbot, L. F. Marin & R. Alami. *Spatial Reasoning for Human Robot Interaction.* Dans Proceedings of International Conference on Robots and Systems (IROS), 2007.

[Sisbot 08] E.A. Sisbot, A. Clodic, R. Alami & M. Ransan. *Supervision and Motion Planning for a Mobile Manipulator interacting with Humans.* Dans Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction (HRI), 2008.

[Tambe 97] M. Tambe. *Agent Architectures for Flexible, Practical Teamwork.* Proceedings of the National Conference on Artificial Intelligence (AAAI), 1997.

[Tapus 06] A. Tapus & M. J. Mataric'. *User Personality Matching with Hands-Off Robot for Post-Stroke Rehabilitation Therapy.* Dans Proceedings of the 10th International Symposium on Experimental Robotics (ISER), Juillet 2006.

[Trafton 05] J. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz & A. C. Schultz. *Enabling Effective Human-Robot Interaction using Perspective-Taking in Robots.* Dans Proceedings of IEEE transactions on Systems, Man and Cybernetics, pages 460–470, Novembre 2005.

[Trinquart 01] R. Trinquart & M. Ghallab. *An Extended Functional Representation in Temporal Planning : Towards Continuous Change.* Dans Proceedings of the European Conference on Planning (ECP), 2001.

TITLE : Human-robot decisional interaction : task planning used for robot sociability.

Summary :
This thesis deals with a number of challenges for an assistive robot and more especially decisional issues linked to it. An assistive robot has to interact with humans which implies that it must integrate in its high-level decisional process some social constraints inherent in a behaviour acceptable by its human partner(s). We propose an approach allowing to describe, in a generic way, a set of social rules introduced in the robot planning process in order to evaluate social quality of solution plans and, thus, keep the most appropriate. We also describe the implementation of this approach in the form of a task planner called HATP (Human Aware Task Planner). Finally, we present and discuss a validation of the developed approach with a simulation scenario and an implementation on a real robot.

AUTEUR : Vincent MONTREUIL

TITRE : Interaction Décisionnelle Homme-Robot : la planification de tâches au service de la sociabilité du robot

DIRECTEUR DE THESE : Rachid ALAMI

LIEU ET DATE DE SOUTENANCE : LAAS-CNRS, le 07 novembre 2008

Résumé :
Cette thèse aborde la problématique du robot assistant et plus particulièrement les aspects décisionnels qui y sont liés. Un robot assistant est amené à interargir avec des hommes ce qui impose qu'il doit intègrer dans son processus décisionnel de haut-niveau les contraintes sociales inhérentes à un comportement acceptable par son(ses) partenaire(s) humain(s). Cette thèse propose une approche permettant de décrire de manière générique diverses règles sociales qui sont introduites dans le processus de planification du robot afin d'évaluer la qualité sociale des plans solutions et de ne retenir que le(s) plus approprié(s). Cette thèse décrit également l'implémentation de cette approche sous la forme d'un planificateur de tâches appelé HATP (Human Aware Task Planner en anglais). Enfin, cette thèse propose une validation de l'approche développée grâce à un scénario de simulation et à une mise en oeuvre sur un robot réel.

MOTS-CLEFS :
Interaction homme-robot, robotique assistive, planification de tâches, comportement social d'un robot.

DISCIPLINE ADMINISTRATIVE : Systèmes Informatiques

LABORATOIRE :

LAAS-CNRS
7, avenue du Colonel Roche 31077 TOULOUSE Cedex 4