Planification distribuée par fusions incrémentales de graphes

Damien Pellier, Ilias Beliadi

Planification distribuée par fusions incrémentales de graphes

Damien Pellier[1], Ilias Belaidi[1]

Université Paris Descartes
Centre de Recherche en Informatique (Équipe IAD)
45, rue des Saints Pères
F-75270 Paris cedex 06
Damien.Pellier@univ-paris5.fr
Ilias.Belaidi@univ-paris5.fr

Résumé : Dans cet article, nous proposons un modèle générique et original pour la synthèse distribuée de plans par un groupe d'agents, appelé « planification distribuée par fusions incrémentales de graphes ». Ce modèle unifie de manière élégante les différentes phases de la planification distribuée au sein d'un même processus. Le modèle s'appuie sur les graphes de planification, utilisé en planification mono-agent, pour permettre aux agents de raisonner et sur une technique de satisfaction de contraintes pour l'extraction et la coordination des plans individuels. L'idée forte du modèle consiste à intégrer au plus tôt, *i.e.*, au sein du processus local de planification, la phase de coordination. L'unification de ces phases permet ainsi aux agents de limiter les interactions négatives entre leurs plans individuels, mais aussi, de prendre en compte leurs interactions positives, *i.e.*, d'aide ou d'assistance, lors de l'extraction de leurs plans individuels.

1 Introduction

La planification multi-agent est une problématique centrale de l'intelligence artificielle distribuée. Les applications qui peuvent bénéficier directement des avancées de ce domaine de recherche sont nombreuses, *e.g.*, la robotique Alami *et al.* (1998), les services web Wu *et al.* (2003), dans lesquelles les approches centralisées sont difficilement envisageables. Classiquement, la planification multi-agent se définit comme un processus distribué qui prend en entrée : une modélisation des actions réalisables par des agents ; une description de l'état du monde connu des agents et un objectif. Le processus doit alors retourner un ensemble organisé d'actions, *i.e.*, un plan, qui s'il est exécuté, permet d'atteindre l'objectif. Une telle définition permet facilement d'appréhender ce qu'est la planification, mais cache la complexité et le découpage des différentes phases du processus. La littérature distingue cinq phases :

1. *Une phase de décomposition.* Les agents raffinent l'objectif initial en sous objectifs atteignables individuellement par les agents. Les approches proposées reposent le plus souvent sur des techniques centralisées de planification, *e.g.*, HTN Younes & Simmons (2003) ou sur les graphes de planification Iwen & Mali (2002).
2. *Une phase de délégation.* Les agents s'assignent mutuellement les objectifs obtenus lors de la première phase. Cette étape s'appuie sur des techniques d'allocation de ressources dans un contexte centralisé Bar-Noy *et al.* (2001) ou dans un contexte distribué basées par exemple sur la négociation Zlotkin & Rosenschein (1990), l'argumentation Tambe & Jung (1999), ou encore la synchronisation Clement & Barrett (2003).
3. *Une phase de planification locale.* Au cours de cette phase, chaque agent essaie de trouver un plan individuel pour atteindre l'objectif qui lui est assigné à l'étape 2. Les techniques utilisées reposent sur les algorithmes classiques de planification mono-agent, *e.g.*, Blum & Furst (1997); Nau *et al.* (2003); Penberthy & Weld (1992).
4. *Une phase de coordination.* Lors de cette phase chaque agent vérifie que le plan local élaboré à l'étape précédente n'entre pas en conflit avec les plans locaux des autres agents. Contrairement à la planification mono-agent, cette étape est nécessaire pour garantir l'intégrité fonctionnelle du système,

i.e., assurer que le but de chaque agent reste atteignable dans le contexte global. La problématique de la coordination a été tout particulièrement étudiée dans le domaine des systèmes multi-agents notamment en la considérant comme un processus de fusion de plans Tonino *et al.* (2002) ou de résolution de conflits Cox & Durfee (2005).

5. *Une phase d'exécution.* Finalement, le plan exempt de conflit est exécuté par les agents.

Bien que cette décomposition soit conceptuellement correcte, l'expérience montre que les cinq phases sont souvent entrelacées : (*i*) les phases 1 et 2 sont souvent fusionnées à cause du lien fort qu'il existe entre les capacités des agents et la décomposition de l'objectif ; (*ii*) la dernière phase d'exécution conduit presque toujours à une replanification Fox *et al.* (2006) due à un événement non prévu lors de l'élaboration du plan et nécessite un retour aux phases 1, 2 et 3 ; (*iii*) la phase de coordination peut être réalisée avant Shoham & Tennenholtz (1995), après Tonino *et al.* (2002) ou pendant Lesser *et al.* (2004) l'étape de planification locale. Cette remarque fait clairement apparaître un manque de modèles capables d'embrasser les différentes phases du processus de planification multi-agent.

Pour tenter de répondre en partie à cette critique, nous présentons dans cet article un modèle de planification multi-agent dans lequel l'élaboration d'un plan partagé est vu comme un processus collaboratif et itératif, intégrant les quatre premières étapes décrites précédemment. L'idée forte du modèle proposé consiste à amalgamer au plus tôt, *i.e.*, au sein du processus local de planification, la phase de coordination en s'appuyant sur une structure classique dans le domaine de planification, les graphes de planification Blum & Furst (1997). Cette représentation possède trois principaux avantages : elle est calculable en temps polynomiale ; elle permet d'utiliser les nombreux travaux portant sur la recherche d'un plan solution dans les graphes de planification ; finalement, elle est assez générique pour représenter une certaine forme de concurrence entre les actions. Ce dernier point est un atout majeur dans le cadre de la planification multi-agent.

L'intégration des phases de coordination et de planification locale, au sein d'un même processus que nous appelons *fusions incrémentales de graphes*, présente l'avantage de limiter les interactions négatives entre les plans individuels des agents, mais aussi et surtout, de prendre en compte les interactions positives, *i.e.*, d'aide ou d'assistance entre agents. Supposons un instant que le but d'un agent soit de prendre un objet dans une pièce dont la porte est fermée à clé. Même si l'agent possède un plan pour atteindre son but une fois la porte ouverte mais qu'il n'est pas capable d'ouvrir la porte, le processus de planification échoue. Supposons maintenant qu'un second agent soit capable d'ouvrir la porte, il existe alors un plan solution. Faut-il encore que le premier agent soit en mesure de considérer, au moment de la synthèse de son plan local, l'aide que peut lui apporter le second agent, et que ce dernier soit en mesure de proposer l'action requise.

Dans la suite de l'article, nous présentons tout d'abord les définitions préliminaires nécessaires à la formalisation de notre approche, puis nous détaillons les cinq phases de l'algorithme de planification distribuée par fusions incrémentales de graphes : la décomposition du but global en buts individuels ; l'expansion et la fusion des graphes de planification ; l'extraction d'un plan individuel et finalement la coordination des plans individuels solution.

2 Définitions préliminaires

La syntaxe et la sémantique utilisées dans notre approche s'appuient sur la logique du premier ordre, à savoir, les constantes, les prédicats et les variables. Le nombre des symboles de prédicats et de constantes est fini. Les propositions sont des tuples d'arguments, *i.e.*, des constantes ou des variables, négatives ou positives, *e.g.*, at(?x, paris), ou ¬at(?x,paris). Pour les prédicats et les constantes, nous utiliserons des chaînes de caractères alphanumériques comportant au moins deux caractères. Les variables seront définies par un seul caractère précédé par ?. Nous définissons la co-désignation comme la relation d'équivalence entre les constantes et les variables. Par extension, deux propositions se co-désignent si elles sont toutes deux négatives ou positives, si elles possèdent le même nombre d'arguments et si tous leurs arguments se co-désignent deux à deux, *e.g.*, at(?x, paris) et at(?y, paris) se co-désignent si $?x = ?y$.

Les opérateurs de planification sont définis comme des fonctions de transition au sens STRIPS. Nous nous plaçons ici dans le cadre de la planification classique : actions instantanées, statiques, déterministes et observabilité totale.

Définition 1 (Opérateur)
Un opérateur o *est un triplet de la forme* $(name(o), precond(o), effects(o))$ *ou* $name(o)$ *définit le nom de l'opérateur,* $precond(o)$ *l'ensemble des préconditions de* o *à satisfaire et* $effects(o)$ *l'ensemble de ses effets.*

Par la suite nous noterons respectivement $effects^+(o)$ *les effets positifs et* $effects^-(o)$ *les effets négatifs d'un opérateur* o.

Définition 2 (Action)
Une action *est une instance d'un opérateur. Si* a *est une action et* s_i *un état tel que* $precond(a) \subseteq s_i$ *alors l'action* a *est* applicable *dans* s_i *et le résultat de son application est un état :* $s_{i+1} = (s_i - effects^-(a)) \cup effects^+(a)$.

Définition 3 (Indépendance)
Deux actions a *et* b *sont* indépendantes[1] *ssi* $effects^-(a) \cap (precond(b) \cup effects^+(b)) = \emptyset$ *et* $effects^-(b) \cap (precond(a) \cup effects^+(a)) = \emptyset$. *Par extension un ensemble* A *d'actions est indépendant si toutes les paires d'actions de* A *sont indépendantes deux à deux.*

Définition 4 (Graphe de planification)
Un graphe de planification $\mathcal{G}$ *est un graphe orienté par niveaux*[2] *organisés en une séquence alternée de propositions et d'actions de la forme* $\langle P_0, A_0, P_1, \ldots, A_{i-1}, P_i \rangle$. P_0 *contient les propositions de l'état initial. Les niveaux* A_i *avec* $i > 0$ *sont composés de l'ensemble des actions applicables à partir des niveaux propositionnels* P_i *et les niveaux* P_{i+1} *sont constitués des propositions produites par les actions de* A_i. *Les arcs sont définis de la manière suivante : pour toutes les actions* $a \in A_i$, *un arc relie* a *avec toutes les propositions* $p \in P_i$ *qui représentent les préconditions de* a ; *pour toutes les propositions* $p \in P_{i+1}$ *un arc relie* p *aux actions* $a \in A_i$ *qui produisent ou suppriment* p. *À chaque niveau le maintien des propositions du niveau* i *au niveau* $i+1$ *est assuré par des actions appelées no-op qui ont une unique proposition comme précondition et effet positif.*

Définition 5 (Exclusion mutuelle)
Deux actions a *et* b *sont* mutuellement exclusives *ssi* a *et* b *sont dépendantes ou si une précondition de* a *est mutuellement exclusive avec une précondition de* b. *Deux propositions* p *et* q *d'un niveau propositionnel* P_i *d'un graphe de planification sont mutuellement exclusives si toutes les actions de* A_i *avec* p *comme un effet positif sont mutuellement exclusives avec toutes les actions qui produisent* q *au même niveau, et s'il n'existe aucune action de* A_i *qui produit à la fois* p *et* q. *Par la suite, nous noterons l'ensemble des exclusions mutuelles d'un niveau* A_i *et* P_i *respectivement* μA_i *et* μP_i.

Définition 6 (Point fixe)
Un graphe de planification $\mathcal{G}$ *de niveau* i *a atteint son* point fixe *si* $\forall i, i > j$, *le niveau* i *de* $\mathcal{G}$ *est identique au niveau* j, *i.e.*, $P_i = P_j$, $\mu P_i = \mu P_j$, $A_i = A_j$ *et* $\mu A_i = \mu A_i$.

Définition 7 (Agent)
Un agent *est un tuple* $\alpha = (name(\alpha), operators(\alpha), belief(\alpha))$ *où* $name(\alpha)$ *définit le nom de l'agent,* $operators(\alpha)$ *l'ensemble des opérateurs que peut réaliser l'agent, et* $belief(\alpha)$ *les croyances de l'agent.*

De plus, nous introduisons $precond(\alpha)$ qui définit l'ensemble des propositions utilisées dans la description des préconditions des opérateurs de α, $effects^+(\alpha)$ et $effects^-(\alpha)$ qui représentent respectivement l'ensemble des propositions utilisées dans la description des effets négatifs et positifs des opérateurs de α. Nous supposons que ces trois ensembles constituent la partie publique d'un agent, *i.e.*, les agents peuvent accéder à ces informations à n'importe quel moment du processus de planification. En d'autres termes, en rendant publique ces informations, un agent publie les propriétés du monde qu'il est en mesure de modifier.

Définition 8 (Interaction négative)
Une action a *d'un graphe de planification d'un agent* α menace *l'activité d'un agent* β *ssi une proposition* $p \in effects^-(a)$ *co-désigne une proposition* q *telle que* $q \in precond(\beta)$ *ou* $q \in effects^+(\beta)$.

Définition 9 (Interaction positive)
Une action a *d'un graphe de planification d'un agent* α assiste *l'activité d'un agent* β *ssi une proposition* $p \in effects^+(a)$ *co-désigne une proposition* q *telle que* $q \in precond(\beta)$ *ou* $q \in effects^+(\beta)$.

[1] L'indépendance entre actions n'est pas une propriété spécifique à un problème de planification donné. Elle découle uniquement de la description des opérateurs des agents.

[2] Un graphe dont les sommets sont partitionnés en ensembles disjoints tels que les arcs connectent seulement les sommets des niveaux adjacents.

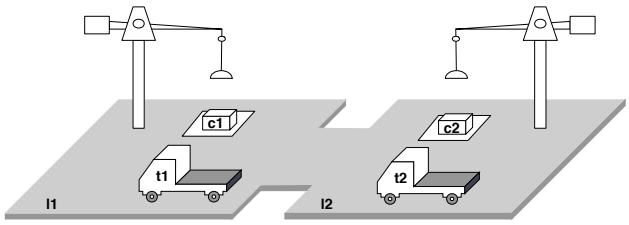


FIG. 1 – État initial de l'exemple :

$$\textit{belief}(\mathsf{ag1}) = \{\mathsf{at}(\mathsf{c1},\mathsf{l1}), \mathsf{at}(\mathsf{t1},\mathsf{l1})\}$$
$$\textit{belief}(\mathsf{ag2}) = \{\mathsf{at}(\mathsf{c2},\mathsf{l2}), \mathsf{at}(\mathsf{t2},\mathsf{l2})\}$$
$$\textit{belief}(\mathsf{ag3}) = \{\mathsf{at}(\mathsf{t1},\mathsf{l1}), \mathsf{at}(\mathsf{t2},\mathsf{l2})\}$$

Notons que ces deux définitions revêtent un caractère particulièrement important dans le cadre de notre modèle. C'est en effet, sur la base de ces interactions, que les agents seront capables de planifier en intégrant les activités des autres agents.

Définition 10 (Problème)
Un problème de planification multi-agent *est un tuple $\mathcal{P} = (\mathcal{A}, g)$ où $\mathcal{A}$ représente l'ensemble des agents devant résoudre le problème et g le but à atteindre, i.e., l'ensemble des propositions qui doivent être satisfaites par les agents après l'exécution du plan. Nous faisons l'hypothèse restrictive que l'union des croyances des agents d'un problème de planification est cohérente (cas classique de la planification mono-agent), i.e., pour deux agents α et $\beta \in \mathcal{A}$, si une proposition $p \in \textit{belief}(\alpha)$ alors $\neg p \notin \textit{belief}(\beta)$. Cependant, aucune hypothèse n'est faite sur le possible partage de croyances entre les agents en termes de faits ou d'opérateurs.*

Définition 11 (But individuel)
Le but individuel g_α d'un agent α pour un problème de planification multi-agent $\mathcal{P} = (\mathcal{A}, g)$ avec $\alpha \in \mathcal{A}$ est défini par $g_\alpha = \{p \in g \mid p$ co-designe une proposition $q \in \textit{effects}^+(\alpha)\}$. Autrement dit, un agent ne peut accepter comme but que les propositions qui sont définis comme des effets des actions qu'il peut exécuter.

Définition 12 (Plan)
Un plan *$\pi = \langle A_0, ..., A_n\rangle$ est une séquence finie d'ensembles d'actions avec $n \geq 0$. Si $n = 0$, alors le plan π est le plan vide, noté $\langle\rangle$. Si les ensembles d'actions composant le plan sont des singletons, i.e., $A_0 = \{a_0\}, \ldots, A_n = \{a_n\}$, nous utiliserons la notation simplifiée $\pi = \langle a_0, ..., a_n\rangle$.*

Définition 13 (Plan solution individuel)
Un plan $\pi_\alpha = \langle A_0^\alpha, ..., A_n^\alpha\rangle$ est un plan solution individuel *d'un problème de planification multi-agent $\mathcal{P} = (\mathcal{A}, g)$ pour un agent $\alpha \in \mathcal{A}$ ssi tous les ensembles d'actions A_i^α sont indépendants et l'application des A_i^α à partir de $\textit{belief}(\alpha)$ définit une séquence d'états $\langle s_0, \ldots, s_n\rangle$ telle que $g_\alpha \subseteq s_n$.*

Définition 14 (Plan solution global)
Un plan $\Pi = \langle A_0, ..., A_n\rangle$ est un plan solution global *d'un problème de planification multi-agent $\mathcal{P} = (\mathcal{A}, g)$ ssi il existe un sous-ensemble d'agents $\mathcal{A}'$ avec $\mathcal{A}' \subseteq \mathcal{A}$ tel que l'union des buts individuels g_α des agents $\alpha \in \mathcal{A}'$ est égale à g, que tous les agents $\alpha \in \mathcal{A}'$ possèdent un plan solution individuel $\pi_\alpha = \langle A_0^\alpha, ..., A_n^\alpha\rangle$ et que tous les ensembles d'actions $A_i = \bigcup A_i^\alpha$ sont indépendants.*

Exemple 2.1 (Dockers) Considérons un exemple simple avec trois agents : deux agents capables de charger et de décharger des conteneurs et, un troisième qui assure le déplacement d'un conteneur d'un endroit à un autre. Le premier docker ag1 est à l'emplacement l1. Le second docker est à l'emplacement l2. Le but du problème est $g = \{\mathsf{at}(\mathsf{c1},\mathsf{l2}), \mathsf{at}(\mathsf{c2},\mathsf{l1}), \mathsf{at}(\mathsf{t1},\mathsf{l2}), \mathsf{at}(\mathsf{t2},\mathsf{l1})\}$. Nous donnons ci-dessous la définition des opérateurs du problème ainsi que son état initial (cf. *fig.* 1) :

;; Le docker ag1 charge un conteneur ?c dans le camion ?t à l'emplacement l1
load(l1,?c,?t)
precond: at(?t,l1), at(?c,l1)
effects: ¬at(?c,l1), in(?t,?c)

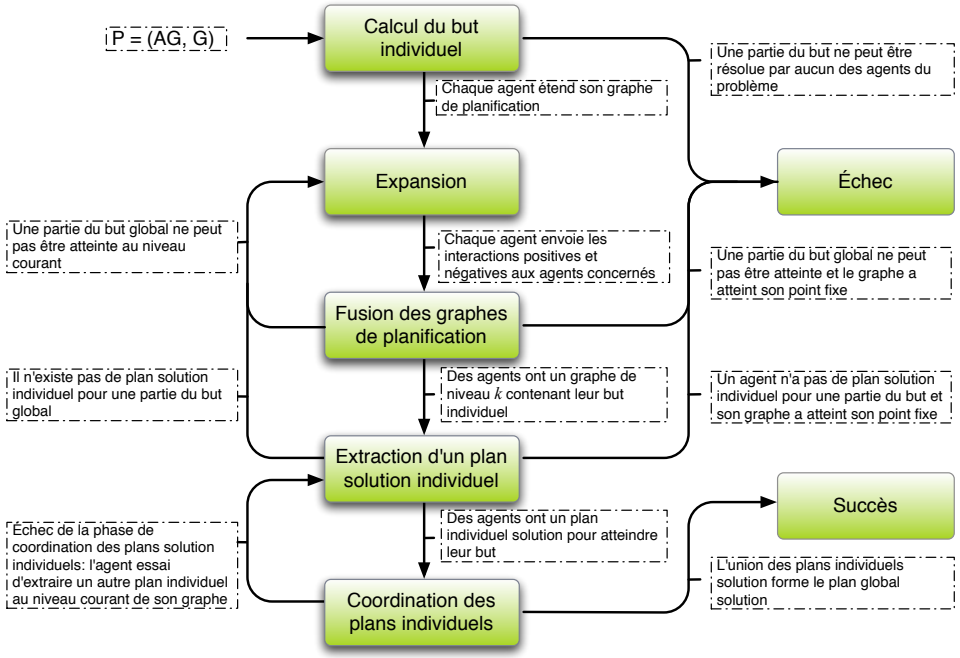


FIG. 2 – Organisation de la procédure de planification distribuée par fusions incrémentales de graphes.

;; Le docker ag1 décharge un conteneur ?c du camion ?t à l'emplacement l1
unload(l1,?c,?t)
precond: at(?t,l1), in(?t,?c)
effects: ¬in(?t,?c), at(?c,l1)

;; Le docker ag2 charge un conteneur ?c dans le camion ?t à l'emplacement l2
load(l2,?c,?t)
precond: at(?t,l2), at(?c,l2)
effects: ¬at(?c,l2), in(?t,?c)

;; Le docker ag2 décharge un conteneur ?c du camion ?t à l'emplacement l2
unload(l2,?c,?t)
precond: at(?t,l2), in(?t,?c)
effects: ¬in(?t,?c), at(?c,l2)

;; Le convoyeur déplace le camion ?t de l'emplacement ?from à l'emplacement ?to
move(?t,?from,?to)
precond: at(?t,?from)
effects: at(?t,?to), ¬at(?t,?from)

3 Dynamique du modèle

L'algorithme de planification distribuée par fusions incrémentales de graphes de planification (cf. *fig. 2*) s'articule autour de cinq phases. Il réalise une recherche distribuée en profondeur d'abord par niveau, découvrant une nouvelle partie de l'espace de recherche à chaque itération. Dans un premier temps, chaque agent calcule les buts individuels associés aux agents du problème puis entre dans une boucle dans laquelle il étend itérativement son graphe de planification, fusionne les interactions negatives et positives provenant de l'activité des autres agents et finalement, effectue une recherche locale d'un plan individuel solution. Si les agents réussissent à extraire un plan solution individuel à partir de leur graphe de planification, ils tentent alors de les coordonner. Dans le cas contraire, chaque agent étend une nouvelle fois son graphe de

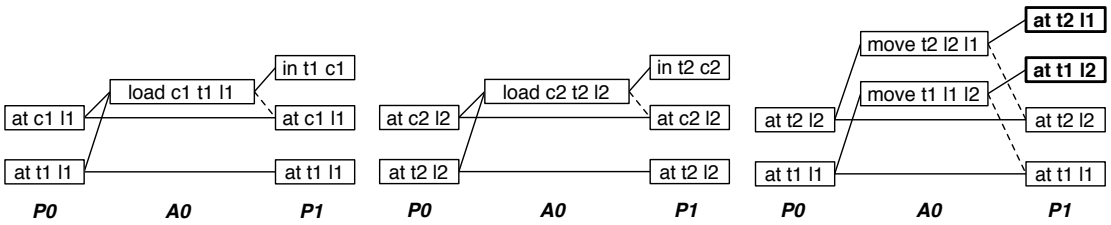
planification. La boucle d'expansion, de fusion, et de recherche de plans solution individuel se poursuit tant qu'aucun plan solution global n'est trouvé ou que la condition de terminaison sur échec n'est pas vérifiée.

3.1 Décomposition du but global

Tout d'abord, chaque agent calcule les buts individuels associés aux agents du problème (*def.* 11), *i.e.*, le sous-ensemble des propositions du but qui co-désignent un effet de ses opérateurs. Si une partie du but global n'apparaît dans aucun des buts individuels des agents, la phase de décomposition échoue, et la synthèse de plans se termine. Cet échec signifie qu'une partie du but global ne peut être atteinte, puisqu'aucun agent ne possède d'opérateurs capables de produire un sous-ensemble des propositions composant le but global. Dans le cas contraire, les agents passent dans la phase d'expansion. Considérons l'exemple 2.1, les buts individuels des agents sont définis de la manière suivante : $g_{\mathsf{ag1}} = \{\mathsf{at}(\mathsf{c2},\mathsf{l1})\}$; $g_{\mathsf{ag2}} = \{\mathsf{at}(\mathsf{c1},\mathsf{l2})\}$; $g_{\mathsf{ag3}} = \{\mathsf{at}(\mathsf{t1},\mathsf{l2}), \mathsf{at}(\mathsf{t2},\mathsf{l1})\}$. Il est important de noter qu'un agent peut ne pas avoir de but individuel, mais être nécessaire à la synthèse de plan. Dans ce cas, l'agent étend malgré tout son graphe pour lui permettre d'évaluer les actions d'aide qu'il peut proposer aux autres agents.

3.2 Expansion du graphe de planification

Cette phase débute par l'initialisation, par chaque agent, de son graphe de planification à partir de ses croyances. Puis, chaque agent étend son graphe en respectant la définition 4. Lorsque l'expansion de son graphe est terminée, l'agent envoie aux agents concernés les actions associées à son graphe qui peuvent interférer soit de manière positive (*def.* 9) soit de manière négative (*def.* 8). La figure 3 donne les graphes de planification associés aux agents ag_1 (a), ag_2 (b) and ag_3 (c) après la premier expansion de leur graphe : les boites au niveau P_i représentent des propositions et les boites au niveau A_i des actions ; pour simplifier, les relations d'exclusion mutuelles ne sont pas représentées ; les lignes pleines modélisent les préconditions et les effets positifs des actions tandis que les lignes pointillées modélisent les effets négatifs ; finalement, les boites en gras définissent les propositions but lorsqu'elles sont atteintes sans mutex. Le graphe de ag_3 contient son but individuel exempt d'exclusion mutuelle au niveau 1. En revanche, les graphes de planification de ag_1 et ag_2 ne contiennent par leur but individuel respectif. Considérons maintenant les interactions négatives et positives entre les actions des différents graphes. Par exemple, l'action move(t1,l1,l2) de ag_3 au niveau 0 menace l'activité de ag_1 car elle supprime la précondition at(t1,l1) qui peut être nécessaire à l'application de l'opérateur load et unload de ag_1. Par opposition, cette même action de ag_3 assiste l'activité de ag_2 puisqu'elle produit l'effet at(t1,l2) qui peut être nécessaire au déclenchement des opérateurs load et unload de ag_2. La liste complète des interactions entre les actions des agents est donnée par le tableau 1.



(a) Graphe de planification de l'agent 1 : but individuel non atteint.

(b) Graphe de planification de l'agent 2 : but individuel non atteint.

(c) Graphe de planification de l'agent 3 : but individuel atteint et exempt d'exclusion mutuelle.

FIG. 3 – Graphe de planification des agents de l'exemple 2.1.

Actions	*Agent*	*Négative*	*Positive*
load(c1,t1,l1)	ag_1	–	ag_2
load(c2,t2,l2)	ag_2	–	ag_1
move(t1,l1,l2)	ag_3	ag_1	ag_2
move(t2,l2,l1)	ag_3	ag_2	ag_1

TAB. 1 – Table des interactions entre agents au niveau 0.

3.3 Fusions des graphes de planification

Au cours de cette phase, l'agent fusionne les actions pouvant interférer avec son activité provenant des autres agents à son graphe de planification. Autrement dit, l'agent ajoute les actions à son graphe et achève son expansion *i.e.*, déclenche toutes actions possibles à partir des nouvelles propositions introduites au cours de l'expansion de son graphe. Le graphe contient désormais les actions provenant des autres agents pouvant interférer de manière positive ou négative avec sa propre activité. Deux cas sont à envisager :

1. Un agent α nécessaire à la réalisation du but global possède un graphe qui a atteint son point fixe au niveau i et $g_\alpha \not\subseteq P_i$ ou $g_\alpha \in P_i$ et $g_\alpha \cap \mu P_i \neq \emptyset$: la synthèse de plans échoue et l'algorithme se termine. En effet, un sous-ensemble des propositions du but global ne peut plus être atteint.
2. Un agent α nécessaire à la réalisation du but global possède un graphe qui n'a pas atteint son point fixe au niveau i mais $g_\alpha \not\subseteq P_i$ ou $g_\alpha \in P_i$ et $g_\alpha \cap \mu P_i \neq \emptyset$: tous les agents effectuent une nouvelle expansion de leur graphe de planification.

Finalement, si aucune des deux conditions n'est vérifiée, la phase de fusion se termine en garantissant qu'il existe un sous-ensemble des agents du problème capables d'atteindre le but global, et que chaque agent de ce sous-ensemble possède un graphe de planification de niveau i, tel que $g_\alpha \subseteq P_i$ et $g_\alpha \cup \mu P_i = \emptyset$.

À titre d'illustration, nous donnons à la figure 4 les graphes de planification des trois agents de l'exemple 2.1 après trois itérations d'expansion et de fusion. Afin de limiter l'ajout de propositions inutiles au niveau du graphe de planification pour ne pas pénaliser la recherche de plans, seules les propositions apparaissant dans la description des opérateurs d'un agent sont ajoutées. Par exemple, l'ajout de l'action move(t2,l2,l1) de l'agent ag3 dans le graphe de planification de l'agent ag1 ne produira qu'un seul effet, at(t2,l1), au niveau 1 du graphe de ag1. Le même principe s'applique également aux préconditions des actions ajoutées. Toutefois, si l'action n'est pas liée par au moins une précondition au niveau propositionnel (c'est le cas de l'action move(t2,l2,l1)), nous ajoutons une précondition fictive pour garantir l'existence d'un chemin de P_0 à P_i, et ainsi permettre l'extraction d'un plan individuel solution.

3.4 Extraction d'un plan individuel

L'extraction d'un plan solution est effectuée en s'appuyant sur une technique de satisfaction de contraintes proposée par Kambhampati (2000). Cette technique présente deux principaux avantages : d'une part, elle améliore de manière significative les temps d'extraction des plans solution et d'autre part, elle peut être facilement modifiée pour extraire un plan solution individuel intégrant les contraintes provenant des autres agents. L'extraction débute par l'encodage du graphe de planification sous la forme d'un CSP. Chaque proposition p à un niveau i du graphe de planification est assimilée à une variable CSP. L'ensemble des actions qui produisent p au niveau n constitue son domaine auquel on ajoute une valeur $\perp$ nécessaire à l'activation des actions dans le graphe. Les relations d'exclusion correspondent aux contraintes du CSP. Lorsque deux actions a_1 et a_2 sont mutuellement exclusives, alors pour toutes les paires de propositions p_1 et p_2 telles que a_1 peut produire p_1 et a_2 respectivement p_2, la contrainte $(p_1 = a_1) \Rightarrow (p_2 \neq a_2)$ est ajoutée au CSP. Lorsque deux propositions p_1 et p_2 sont mutuellement exclusives, on le traduit par une contrainte $\neg(p_1 \neq \perp) \wedge (p_2 \neq \perp)$. L'assignation des valeurs aux variables est dynamique car chaque assignation, à un niveau donné, active des variables au niveau précédent en respectant la procédure classique d'extraction de graphplan. Initialement, seules les variables représentant le but individuel de l'agent sont actives. L'activation dynamique se traduit par des contraintes d'activation au niveau du CSP. Autrement dit, lorsque une variable, représentant une proposition, prend une certaine valeur, *i.e.*, est produite par une action, alors d'autres variables, qui correspondent aux préconditions de l'action choisie, deviennent actives. L'extraction d'un plan solution individuel à partir du graphe de planification de l'agent est réalisée en résolvant le CSP. En partant des variables actives initialement, *i.e.*, le but individuel de l'agent, la procédure d'extraction cherche à assigner une valeur ou une action, à chaque variable ou proposition, pour satisfaire l'ensemble des contraintes, *i.e.*, des exclusions mutuelles. Si la résolution du CSP réussie, l'agent a extrait un plan solution individuel. Dans le cas contraire, l'agent n'est pas capable de produire un plan individuel solution. Deux cas doivent être considérés :

1. Un agent nécessaire à la réalisation du but global ne parvient pas à extraire un plan solution individuel même en poursuivant l'expansion de son graphe, l'algorithme se termine. Une partie du but global ne peut être démontrée.

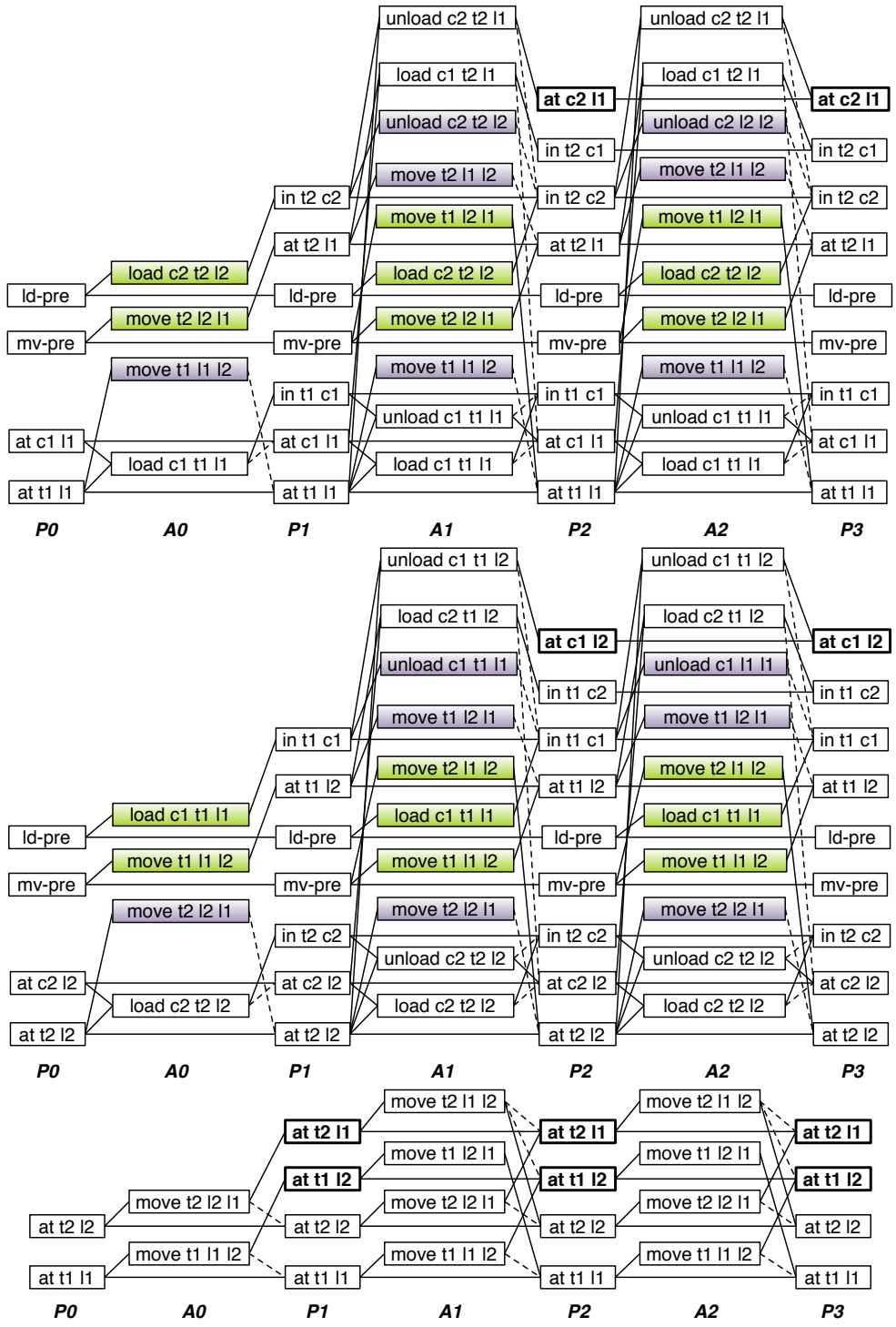


FIG. 4 – Graphes de planification des agents de l'exemple des dockers après la phase d'expansion et de fusion.

2. Un agent nécessaire à la réalisation du but global ne parvient pas à extraire un plan solution individuel mais peut espérer en trouver un en effectuant une nouvelle expansion de son graphe. Tous les agents effectuent une nouvelle expansion de leur graphe de planification.

Finalement, la phase d'extraction se termine en garantissant qu'il existe un sous-ensemble des agents du problème capables d'atteindre le but global, et possédant un plan individuel solution.

3.5 Coordination des plans individuels

Il s'agit maintenant de vérifier la compatibilité des plans individuels dans le contexte multi-agent. Rappelons que lors de la phase de fusion, les agents ont ajouté dans leur graphe les actions provenant d'autres agents et pouvant interférer avec leur propre activité. Par conséquent, un plan individuel solution peut contenir des actions qui doivent être exécutées par d'autres agents. Autrement dit, un plan solution individuel est un plan conditionnel *i.e.*, un plan exécutable si certaines contraintes sont vérifiées. Dans notre cas, ces contraintes sont définis par des couples (a, i) ou a représente une action et i le niveau où elle doit être exécutée. Considérons l'exemple des dockers et leur graphe de planification (cf. *fig.* 4). Les agents ag1 et ag2 peuvent extraire respectivement un plan solution individuel au niveau 2 :

$$\begin{aligned}\pi_{\mathsf{ag1}} &= \langle \mathsf{load(c2,t2,l2)}, \mathsf{move(t2,l2,l1)}, \mathsf{unload(c2,t2,l1)}\rangle \\ \pi_{\mathsf{ag2}} &= \langle \mathsf{load(c1,t1,l1)}, \mathsf{move(t1,l1,l2)}, \mathsf{unload(c1,t1,l2)}\rangle\end{aligned}$$

Le plan π_{ag1} est valide si le plan individuel de ag2 respecte la contrainte $(\mathsf{load(c2,t2,l2)}, 0)$ et celui de ag3 la contrainte $(\mathsf{move(t2,l2,l1)}, 1)$. De manière symétrique, π_{ag2} est valide si le plan individuel de ag2 respecte $(\mathsf{load(c1,t1,l1)}, 0)$ et celui de ag3 la contrainte $(\mathsf{move(t1,l1,l2)}, 1)$. De son côté, l'agent convoyeur ag3 peut potentiellement extraire plusieurs plans individuels solution dont aucun n'implique des actions provenant des autres agents :

$$\begin{aligned}\pi_{\mathsf{ag3}} &= \langle \{\mathsf{move(t2,l2,l1)}, \mathsf{move(t1,l1,l2)}\}, \mathsf{no\text{-}op}, \mathsf{no\text{-}op}\rangle \\ \pi'_{\mathsf{ag3}} &= \langle \mathsf{move(t2,l2,l1)}, \mathsf{move(t1,l1,l2)}, \mathsf{no\text{-}op}\rangle \\ \pi''_{\mathsf{ag3}} &= \ldots\end{aligned}$$

Les agents débutent la phase de coordination par l'échange des contraintes de leur plan individuel, et tentent ensuite d'intégrer les contraintes reçues à leur plan. L'intégration des contraintes repose sur le principe de moindre engagement. Tout d'abord, l'agent teste si les contraintes peuvent être ajoutées directement à son plan individuel solution. Ce teste ne nécessite aucune replanification, puisqu'il suffit de vérifier que l'action à ajouter n'est pas mutuellement exclusive avec une action déjà présente dans le graphe de planification de l'agent au même niveau. Dans notre exemple, ce mécanisme de coordination sera utilisé par les agents ag1 et ag2 pour prendre respectivement en compte les contraintes $(\mathsf{load(c2,t2,l2)}, 0)$ et $(\mathsf{load(c1,t1,l1)}, 0)$. Si ce premier mécanisme échoue, l'agent tente alors d'extraire un nouveau plan individuel solution en incluant les contraintes. En d'autres termes, il propage les contraintes dans le graphe de contraintes représentant son graphe de planification, et effectue l'extraction d'un nouveau plan individuel solution. Ce mécanisme sera utilisé par l'agent ag3 pour intégrer les contraintes $(\mathsf{move(t1,l1,l2)}, 1)$ et $(\mathsf{move(t2,l2,l1)}, 1)$ provenant des agents ag1 et ag2. Au final, les plans individuels solutions de l'exemple 2.1 obtenus après la phase de coordination sont les suivants :

$$\begin{aligned}\pi_{\mathsf{ag1}} &= \langle \mathsf{load(c1,t1,l1)}, \mathsf{no\text{-}op}, \mathsf{unload(c2,t2,l1)}\rangle \\ \pi_{\mathsf{ag2}} &= \langle \mathsf{load(c2,t2,l2)}, \mathsf{no\text{-}op}, \mathsf{unload(c1,t1,l2)}\rangle \\ \pi_{\mathsf{ag3}} &= \langle \mathsf{no\text{-}op}, \{\mathsf{move(t1,l1,l2)}, \mathsf{move(t2,l2,l1)}\}, \mathsf{no\text{-}op}\rangle\end{aligned}$$

Finalement, si ce second mécanisme de coordination échoue, les agents retournent dans la phase d'extraction de plans individuel. L'algorithme de fusions incrémentales de plans peut alors se terminer (si aucun autre plan individuel ne peut être extrait) ou encore nécessité une nouvelle expansion des graphes de planification des agents.

4 Conclusion

Dans cet article, nous avons proposé un modèle générique et original pour la synthèse distribuée de plans par un groupe d'agents, appelé *planification distribuée par fusions incrémentales de graphes*. Le modèle semble correct et complet, et unifie de manière élégante les différentes phases de la planification distribuée au sein d'un même processus. En outre, il permet aux agents de limiter les interactions négatives entre leur plan individuel, mais également de prendre en compte leurs interactions positives, *i.e.*, d'aide ou d'assistance, au plus tôt, *i.e.*, avant l'extraction des plans individuels. Notre approche est actuellement en cours d'implantation en s'appuyant sur les librairies PDDL4J[3] et Choco[4]. Chose intéressante, les premiers

[3] http://sourceforge.net/projects/pdd4j/

[4] http://choco-solver.net/

résultats expérimentaux semblent indiquer que la complexité n'est pas dépendante du nombre d'agents présents dans le problème, mais uniquement du couplage entre les activités des agents, *i.e.*, des interactions positives. Plusieurs pistes pour la poursuite de ce travail sont envisagées :

– la prise en compte de la robustesse des plans. Par robustesse, nous entendons la capacité d'un plan à tolérer plus ou moins des aléas d'exécution ou des croyances erronées sur l'état du monde. Supposons que nous soyons capables de construire un faisceau de plans solutions, *i.e.*, un plan contenant différentes alternatives. La question qui se pose alors est la suivante : comment choisir les plans individuels les plus robustes minimisant ainsi le risque d'échec à l'exécution ?
– l'extraction de plans temporels flottants, *i.e.*, de plans dans lesquels les actions sont représentées par des intervalles pouvant être décalés dans le temps. Bien que la causalité soit suffisante pour un grand nombre d'applications, la prise en compte des aspects temporels est parfois nécessaire. La notion de plans flottants apparaît particulièrement intéressante dans un contexte multi-agent, car ils possèdent une grande flexibilité qui peut être mise à profit au cours de la phase de coordination et d'exécution.

Références

ALAMI R., FLEURY S., HERRB M., INGRAND F. & ROBERT F. (1998). Multi robot cooperation in the MARTHA project. *IEEE Robotics and Automation Magazine*, **5**(1), 36–47.

BAR-NOY A., BAR-YEHUDA R., FREUND A., NAOR J. & SCHIEBER B. (2001). A unified approach to approximating resource allocation and scheduling. *Journal of Association for Computing Machinery*, **48**(5), 1069–1090.

BLUM A. & FURST M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, **90**(1-2), 281–300.

CLEMENT B. & BARRETT A. (2003). Continual coordination through shared activities. In *Proceedings of the International Conference on Autonomous Agent and Muti-Agent Systems*, p. 57–67.

COX J. & DURFEE E. H. (2005). An efficient algorithm for multiagent plan coordination. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, p. 828–835, New York, NY, USA : ACM Press.

FOX M., GEREVINI A., LONG D. & SERINA I. (2006). Plan stability : Replanning versus plan repair. In *Proceedings of the International Conference on Planning and Scheduling*, California, USA : AAAI Press.

IWEN M. & MALI A. D. (2002). Automatic problem decomposition for distributed planning. In *Proceedings of the International Conference on Artificial Intelligence*, p. 411–417.

KAMBHAMPATI S. (2000). Planning graph as a (dynamic) CSP : Exploiting EBL, DDB and other CSP search techniques in graphplan. *Journal of Artificial Intelligence Research*, **12**(1), 1–34.

LESSER V., DECKER K., WAGNER T., CARVER N., GARVEY A., HORLING B., NEIMAN D., PODOROZHNY R., NAGENDRAPRASAD M., RAJA A., VINCENT R., XUAN P. & ZHANG X. (2004). Evolution of the gpgp/taems domain-independent coordination framework. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, p. 87–143 : Kluwer Academic Publishers.

NAU D., AU T., ILGHAMI O., KUTER U., MURDOCK W., WU D. & YAMAN Y. (2003). Shop2 : An HTN planning system. *Journal of Artificial Intelligence Research*, **20**(1), 379–404.

PENBERTHY J. & WELD D. (1992). UCPO : A sound, complete, partial order planner for ADL. In C. R. B. NEBEL & W. SWARTOUT, Eds., *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, p. 103–114 : Morgan Kaufmann Publishers.

SHOHAM Y. & TENNENHOLTZ M. (1995). On social laws for artificial agents societies : off-line design. *Artificial Intelligence*, **73**(1–2), 231–252.

TAMBE M. & JUNG H. (1999). The benefits of arguing in a team. *Artificial Intelligence Magazine*, **20**(4), 85–92.

TONINO H., BOS A., DE WEERDT M. & WITTEVEEN C. (2002). Plan coordination by revision in collective agent-based systems. *Artificial Intelligence*, **142**(2), 121–145.

WU D., PARSIA B., SIRIN E J. & NAU D. (2003). Automating daml-s web services composition using shop2. In *Proceedings of International Semantic Web Conference*.

YOUNES H. & SIMMONS R. (2003). VHPOP : Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, **20**(1), 405–430.

ZLOTKIN G. & ROSENSCHEIN J. (1990). Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the American National Conference on Artificial Intelligence*, p. 100–105, Boston, Massachusetts.