DISTRIBUTED PLANNING THROUGH GRAPH MERGING

Damien Pellier
Université Paris Descartes University, Laboratoire d'Informatique Paris Descartes, 45, rue des Saint Père, Paris, France

Keywords: Distributed problem solving, Cooperation, Coordination, Multi-agent planning, Planning graphs technics.

Abstract: In this paper, we introduce a generic and fresh model for distributed planning called "Distributed Planning Through Graph Merging" (DPGM). This model unifies the different steps of the distributed planning process into a single step. Our approach is based on a planning graph structure for the agent reasoning and a CSP mechanism for the individual plan extraction and the coordination. We assume that no agent can reach the global goal alone. Therefore the agents must cooperate, *i.e.,* take in into account potential positive interactions between their activities to reach their common shared goal. The originality of our model consists in considering as soon as possible, *i.e.,* in the individual planning process, the positive and the negative interactions between agents activities in order to reduce the search cost of a global coordinated solution plan.

1 INTRODUCTION

The problem of plan synthesis achieved by a group of autonomous agents in order to reach a common goal is one of the central problem of distributed artificial intelligence. Increasingly new application areas can benefit from this research domain: *e.g.*, robotics (Sariel et al., 2008), web services (Pistore et al., 2005) when considering actions as services and plans as composition schemes, decision making (Wilkins and desJardins, 2001), where planning process is viewed as an expert able to guide the search of a plan or a procedure. In all these applications, the implementation of a centralized approach of planning is often impossible due to technical constraints, *e.g.*, in robotics, no robot has enough computational resources to plan for the whole robots team, or organizational constraints, *e.g.*, two concurrent web services cannot share important business data.

Classically, multiagent planning (Durfee, 2000) is defined as a planning process involving a group of agents, *i.e.*, a process that takes as input the actions models of the agents, a description of the state of the world known by the agents, and some objectives and returns an organized collection of actions whose global effect, if they are carried out and if they are performed as modeled, achieves the objectives. Such a definition is correct, but hides the complexity of the different steps of a distributed planning process. Actually, a multiagent planning process can be divided into five separate steps: (i) a *task decomposition step* where the agents attempt to refine the initial task such that there is a matching with the set of the agents capabilities, (ii) a *subtask delegation step* where the agents attempt to assign sub-tasks to each other such that the capabilities offered are sufficient for the capabilities required, (iii) an *individual planning step* where each agent tries to find a plan to solve the task allocated in step 2, (iv) an *individual plans coordination step* where the agents coordinate their activities in order to conserve the functional integrity of the system, *i.e.*, ensure that the goal of each agent stay reachable in the global context, and finally (v) a *joint plan execution step* where the joint plan build by the agents is executed in a coordinated way. Although this decomposition is convenient to explain the problem of multiagent planning, experiences show that the five steps introduced are interlaced: (i) step 1 and 2 are often merged due to the existing link between the agents capabilities and the task decomposition, (*ii*) the joint plan execution involves replanning due to a failure and force into backtrack to step 1, 2 and 3, and (*iii*) the coordination can be executed either before, after or during the planning. This remark brings to light the lack of work to merge and articulate the different steps of multiagent planning.

In order to answer in part to this challenge, we introduce a multiagent planning model, called "Distributed Planning through Graphs Merging" (DPGM), that covers the four first steps of the multiagent planning process previously described. Our model focuses on generic and completely distributed mecha-

Pellier D. (2010).
DISTRIBUTED PLANNING THROUGH GRAPH MERGING.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 128-134
DOI: 10.5220/0002702601280134

nisms in order to allow a group of agents to jointly elaborate a global shared plan and perform a collective goal. We assume that no agent can reach the goal alone. By elaboration, we mean plan production and not instantiation of predefined global plan skeletons (D'Inverno et al., 2004) or distributed coordination of individual plans based on planning techniques (Iwen and Mali, 2002; Tonino et al., 2002; Cox and Durfee, 2005). This is achieved by composing agents capabilities, *i.e.*, the actions they can execute, for the benefit of the group. At the team's level, agents exchange constraints about their own activities based on the structure of planning graph (Blum and Furst, 1997). At the agent's level, each agent merges the constraints received from the others into its own planning graph and takes advantage of these constraints in order to extract an individual plan conflict free based on a CSP technique (Kambhampati, 2000).

The rest of this paper is organized as following: section 2 proposes an overview of the model and its primary definitions, and finally section 3 introduces the dynamic of the distributed planning through graph merging approach.

2 PRIMARY DEFINITIONS

Let us first define the primary definitions of DPGM based on (Blum and Furst, 1997). Operators are defined in STRIPS (Finke and Nilsson, 1971). We are dealing here only with classical planning assumption, *i.e.*, deterministic, static, implicit time and fully observable operators. We note $precond(o)$, the preconditions of an operator o, and respectively $effects^+(o)$ the positive effects of o and $effects^-(o)$ the negative effects of o. An action is a ground instance of an operator. We say that two actions a and b are *dependent* if: a deletes a precondition of b: the orderings $a \prec b$ will not be permitted; or a deletes a positive effect of b: the resulting state will dependent on their order; or symmetrically for negative effects of b with respect to a: b deletes a preconditions on a positive effect of a.

A *planning graph* G is a directed, leveled graph organized in an alternated sequence of propositions and actions $\langle P_0, A_0, P_1, \ldots, A_{i-1}, P_i \rangle$. The first level of the planning graph is a proposition level P_0 that defines the initial belief state of an agent. The levels A_i with $i > 0$ are action levels that define the set of actions applicable from the proposition levels P_i and the levels P_{i+1} define the proposition produced by the actions of A_i. Edges of planning graph explicitly represent relations between actions and propositions. Actions in an action level A_i are connected by precondition edges to their preconditions in level P_i, by add-edges to their positive effect in level P_i and by delete edges to their negative effects in level P_i. At each level P_i, every proposition $p \in P_i$ are propagated to the next level P_{i+1} by a dummy action no-op that has a single precondition and a single positive effect p.

Two actions a and b in level A_i are *mutex* if either a and b are dependent or if a precondition of a is mutex with a precondition of b. Two propositions p and q in P_i are mutex if every action in A_{i-1} that has p as a positive effect (including no-op actions) is mutex with every action that produces q. The set of mutual exclusion relations at a proposition level P_i and action level A_i respectively μP_i and μA_i.

A *fixed-point level* in a planning graph G is a level i such that for $\forall j > i$, level j of G is identical to level i, *i.e.*, $P_i = P_j$, $\mu P_i = \mu P_j$, $A_{i-1} = A_{j-1}$ and $\mu A_{i-1} = \mu A_{j-1}$.

An *agent* α is a couple $\alpha = (O, s_0)$ where O is the set of operators that describes the capabilities of α, and s_0 is its initial agent state.

Furthermore, we introduce $precond(\alpha)$ that defines the set of atoms used in the precondition of the agent operators of α, $effects^+(\alpha)$ and $effects^-(\alpha)$ that represent respectively the set of atoms used in the positive and negative operators effects of α. We assume that these sets of atoms constitute the public part of an agent, *i.e.*, agents can access anytime to this information from the others during the planning process.

An action a_i contained in the planning graph of an agent α at level i *threats* the activity of an agent β iff a proposition $p \in effects^-(a_i)$ is unifiable with a precondition or a positive effect of an operator $o \in O_\beta$.

An action a_i contained in the planning graph of an agent α at level i *promotes* the activity of β iff $p \in effects^+(a_i)$ is unifiable with a precondition or a positive effect of $o \in O_\beta$.

The threat definition expresses the classical negative interactions between agents. If an agent executes an action that deletes a property needed by another agent, their activities are confrontational. The promotion definition formulates the positive interactions between agents. If an agent produces a property useful to another agent, *i.e.*, that is needed to execute one of its own actions, their activities reinforce each other.

A *multiagent planning problem* $\mathcal{P}$ is a tuple $\mathcal{P} = (\mathcal{A}, g)$ where $\mathcal{A}$ is the set of agents that must solve the problem and g defines the goal, *i.e.*, a set of propositions that has to be satisfied by the agents (see Fig. 1).

The individual goal g_α of an agent α for a multiagent planning problem $\mathcal{P} = (\mathcal{A}, g)$ with $\alpha \in \mathcal{A}$ is defined by: $g_\alpha = \{p \in g \mid p \text{ is unifiable with an effect } q \in effects^+(\alpha)\}$. The goal decomposition relies on the fact that an agent can only reach predicates defined in the positive

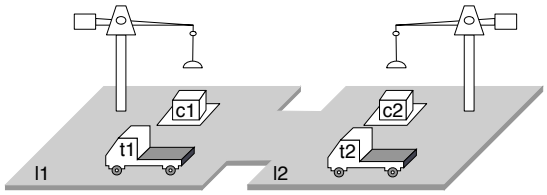


Figure 1: A simple logistic example with three agents: the agents ag1 and ag2 can load and unload containers and the agent ag3 can move containers. The global goal to reach is $g = \{\mathsf{at(c1,l2)}, \mathsf{at(c2,l1)}, \mathsf{at(t1,l2)}, \mathsf{at(t2,l1)}\}$. Initial states of the agents: $s_0^1 = \{\mathsf{at(c1,l1)}, \mathsf{at(t1,l1)}\}$, $s_0^2 = \{\mathsf{at(c2,l2)}, \mathsf{at(t2,l2)}\}$, $s_0^3 = \{\mathsf{at(t1,l1)}, \mathsf{at(t1,l2)}\}$.

effects of its operators.

A plan $\pi_\alpha = \langle A_0^\alpha, \ldots, A_n^\alpha \rangle$ is an *individual solution plan* to a planning problem $\mathcal{P} = (\mathcal{A}, g)$ for an agent $\alpha \in \mathcal{A}$ iff each $A_i^\alpha \in \pi_\alpha$ is independent, and each A_i^α is applicable to a state s_i^α defining a sequence $\langle s_0^\alpha, \ldots, s_n^\alpha \rangle$ such that $g_\alpha \subseteq s_n^\alpha$.

A plan $\Pi = \langle A_0, \ldots, A_k, \ldots, A_n \rangle$ is a *global solution* for a multiagent planning $\mathcal{P} = (\mathcal{A}, g)$ iff: (i) the union of the individual goal g_α with $\alpha \in \mathcal{A}$ is equal to g, (ii) each agent $\alpha \in \mathcal{A}$ has an individual plan $\pi_\alpha = \langle A_0^\alpha, \ldots, A_n^\alpha \rangle$ to reach its individual goal g_α, and (iii) each actions set $A_i = \bigcup A_i^\alpha$ with $\alpha \in \mathcal{A}$ is independent.

3 DPGM ALGORITHM

The DPGM procedure performs a distributed search close to iterative deepening, discovering a new part of the search at each iteration (see Fig. 2). First, each agent computes its individual goals and then enters in a loop where it iteratively: (i) expands its planning graph, (ii) merges threats and promotions from the others, and finally (iii) searches backward from the last level of its graph for an individual plan. If every agent succeeds in extracting an individual plan for its individual goal, then every agent tries to coordinate its individual plans to each others. Otherwise, each agent expands its graph one more time. The iterative loop of graph expansion, graph merging, individual plan search and coordination is pursued until either a global solution plan is found or a failure termination condition is met. Let us detail the algorithm and its properties.

3.1 Global Goal Decomposition

First of all, each agent computes its individual goal, *i.e.*, the subset of the global goal that is unifiable with an effect of its operators. If a part of the global goal of the planning problem does not belong to any individual goal of the agents, then the goal decomposition fails and the DPGM procedure ends. This failure means that the global goal cannot be reached, since a subset of the global goal cannot be produced by at least one agent. Otherwise, agents go to the expansion step. Consider our example, the individual goals of the agents $\mathsf{ag_1}$, $\mathsf{ag_2}$ and $\mathsf{ag_3}$ are as follow: $g_{\mathsf{ag_1}} = \{\mathsf{at(c2,l1)}\}$, $g_{\mathsf{ag_2}} = \{\mathsf{at(c1,l2)}\}$ and $g_{\mathsf{ag_3}} = \{\mathsf{at(t1,l2)}, \mathsf{at(t2,l1)}\}$. Each proposition of the global goal is assigned to at least one agent. Therefore, the agents can pursue the expansion step of the DPGM procedure.

3.2 Planning Graph Expansion

Let an agent $\alpha = (O, s_0)$ such that O is a set of operators with no negated atom in their preconditions, and s_0 a set of propositions. Let g_α be the individual goal of α and A be the union of all ground instances of the operators in O and of all no-op actions a_p for every proposition p of A. A planning graph for α expanded up to level i is a sequence of levels and mutex pairs: $G = \langle P_0, \mu P_0, A_0, \mu A_0, \ldots, A_{i-1}, \mu A_{i-1}, P_i, \mu P_i \rangle$. Starting initially from $P_0 = s_0$ and $\mu P_0 = \emptyset$, each agent expands its planning graph G from level i to level $i+1$ as the expansion procedure of Graphplan. Then, it sends to the concerned agents the threats and the promotions contained in its planning graph. The figure 3 at level 1 shows the planning graphs of the agents $\mathsf{ag_1}$, $\mathsf{ag_2}$ and $\mathsf{ag_3}$ after the first expansion. For instance, the action move(t1,l1,l2) of $\mathsf{ag_3}$ at level A_0 threats the activity of $\mathsf{ag_1}$ since it deletes the precondition at(t1,l1) needed to apply load and unload of $\mathsf{ag_1}$, and promotes the activity of $\mathsf{ag_2}$ since it adds the effect at(t1,l2) needed to apply load and unload of $\mathsf{ag_2}$.

3.3 Planning Graph Merging

Each agent adds to its planning graph the threats and the promotions received from the others at the current expansion level. For instance, the action move(t1,l1,l2) that appears in the action level A_0 of the planning graph of $\mathsf{ag_3}$ is added as a threat at level A_0 of the planning graph of $\mathsf{ag_1}$. Take good note that only preconditions and effects of move(t1,l1,l2) that are unifiable with the atoms of the operators of $\mathsf{ag_1}$ are added in its planning graph. In fact, the other propositions have no meaning for $\mathsf{ag_1}$, unnecessarily grow the size of its planning graph and hinder the extraction of an individual plan. If an added action has an empty set of preconditions, we add a dummy precondition to keep link from action level to proposition level (see action load(c2,t2,l2) at level A_0 Fig. 3(a) and its precondition

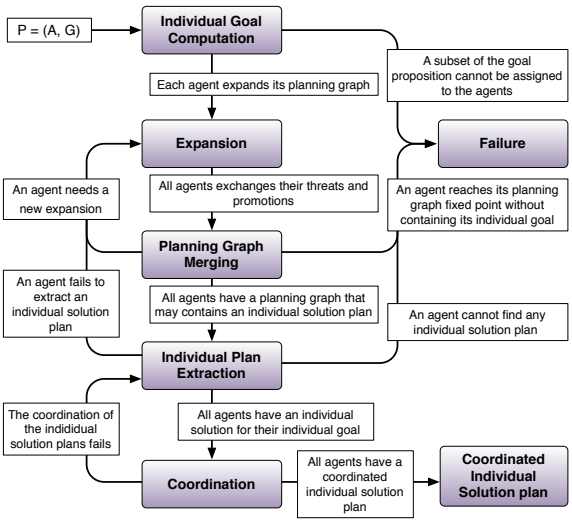
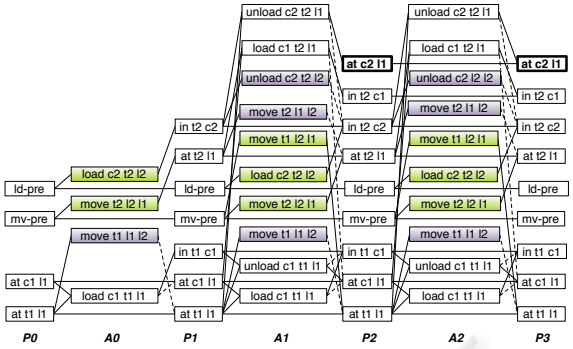


Figure 2: Organization of the DPGM procedure.

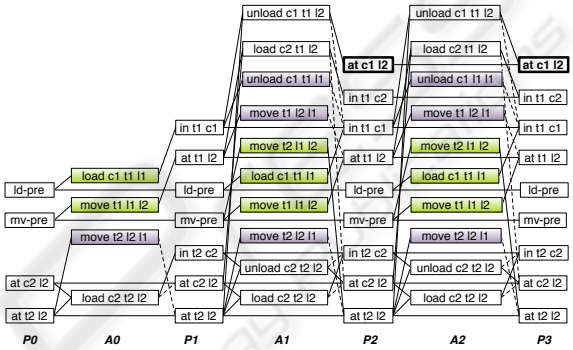
ld-pre). Then, each agent completes it graph expansion based on the new actions introduced at the current expansion level. From now on, the planning graphs of the agents may contain actions possibly executed by the other agents. The agents must now check if they are able to reached collectively the global goal and try to search for an individual plan or if they need to expand their planning graph one more time.

First, each agent α broadcasts the result of its graph merging. If its individual goal g_α is included in the proposition level P_i of its planning graph and g_α is mutex free at level i, *i.e.*, $g_\alpha \in P_i$ and $g_\alpha \cap \mu P_i = \emptyset$, then α has a planning graph that may contain an individual plan of length i to reach its goal g_α. If this condition is met, the merging is a success, otherwise a failure. Note that failure happens in two cases: (i) the ending condition of Graphplan is met (a new expansion to level $i+1$ will not allow α to find an individual plan) or (ii) its goals are not in P_i or not mutex free (a new expansion to level $i+1$ may allow α to find a solution).

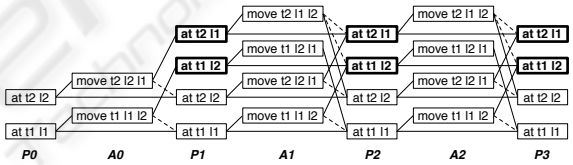
Second, each agent gathers merging results from the other agents. If no agent fails, each agent has a planning graph which may contain an individual plan and tries to extract an individual plan. Now suppose that an agent fails because its planning graph reaches its fixed-point without containing its individual goal. The DPGM procedure fails because a subset of the global goal is definitively unreachable. Finally, if no previous condition holds, the agents try to expand their planning graphs to level $i+1$. In order to illustrate the interlacing of expansion and merging, the figure 3 depicts the planning graphs of the agents ag_1, ag_2 and ag_3 obtained at level 3: gray boxes are threats



(a) Graph of ag_1: goal reached and mutex free at level 3.



(b) Graph of ag_2: goal reached and mutex free at level 3.



(c) Graph of ag_3: goal reached and mutex free at level 1.

Figure 3: Planning graphs of the agents at level 3: boxes at P_i are propositions and boxes at A_i are actions; to simplify, mutexes are not shown; solid lines are precondition-edges and add-edges; dashed lines are del-edges; bold boxes show goal propositions reached and mutex free.

and promotions; to simplify only relevant actions at action level A_2 are shown.

3.4 Individual Solution Plan Extraction

The search for a solution plan in a planning graph used in DPGM is based on a constraints satisfaction technique introduced by (Kambhampati, 2000). This technique has two main advantages for our approach: empirical results demonstrate that this technique improves classical Graphplan's performance on several benchmark problems and it can be easily modified to extract solution plans that respect coordination constraints as presented in the next section. The search consists in two steps out of scope of this paper: a plan-

ning graph encoding phase, where each agent encodes its planning graph into a CSP problem (propositions correspond to variables and actions to values), and a solving phase based on specific techniques adapted to Graphplan search such as explanation based learning, dynamic variable ordering and forward checking.

Now, consider what happens when an agent succeeds or fails to extract an individual solution. As in the merging step, the agents must check if they are always able to reach collectively the global goal. Thus, each agent broadcasts the result of the extraction of its individual plan. Then, based on this information, the agents decide if the global goal is unreachable, if a further expansion of their planning graph is needed, or finally if they must continue and coordinate their individual plans. Let consider the first case: an agent fails to extract its individual plan and no further expansion is possible. It means that a subset of the global goal becomes unreachable. DPGM ends and returns failure. Consider now the second case: an agent can still expand its planning graph. Each agent returns to the expansion step and tries to find a solution at the next level. Finally, if no previous case happens, DPGM assures that all agents have an individual plan to reach all the propositions of the global goal. Therefore, the agents must try to coordinate their individual plans.

3.5 Distributed Plans Coordination

Coordination Constraints. Remember that DPGM merging step adds new actions in the planning graphs of the agents. These actions define threats and promotions between their activities. Thus, individual plans may contain actions that must be executed by the other agents involved in the planning problem. In other words, an individual plan can be viewed as a conditional plan, *i.e.*, a plan which could be executed if certain conditions are met. In our case, these conditions are defined as constraints of the form (a,i) where a is an action and i the level where a must be executed. Consider our example and the planning graphs of $\mathsf{ag_1}$, $\mathsf{ag_2}$ and $\mathsf{ag_3}$ (see Fig. 3). The agents $\mathsf{ag_1}$, $\mathsf{ag_2}$ can respectively extract one individual plan at level 3:

$\pi_{\mathsf{ag1}} = \langle \mathsf{load(c2,t2,l2)}, \mathsf{move(t2,l2,l1)}, \mathsf{unload(c2,t2,l1)} \rangle$

$\pi_{\mathsf{ag2}} = \langle \mathsf{load(c1,t1,l1)}, \mathsf{move(t1,l1,l2)}, \mathsf{unload(c1,t1,l2)} \rangle$

More precisely, π_{ag1} is executable if π_{ag2} matches the constraint $(\mathsf{load(c2,t2,l2)},0)$ and π_{ag3} the constraint $(\mathsf{move(t2,l2,l1)},1)$. Symmetrically, π_{ag2} is executable if π_{ag1} matched the constraint $(\mathsf{load(c1,t1,t1)},0)$ and π_{ag3} the constraint $(\mathsf{move(t1,l1,l2)},1)$. As regards $\mathsf{ag_3}$, it may extract several individual plans, but no one needs the help of $\mathsf{ag_1}$ or $\mathsf{ag_2}$. Thus, $\mathsf{ag_3}$ individual plans imply no constraint for the other agents:

$\pi_{\mathsf{ag3}} = \langle \{\mathsf{move(t2,l2,l1)}, \mathsf{move(t1,l1,l2)}\}, \mathsf{no\text{-}op}, \mathsf{no\text{-}op} \rangle$

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

$\pi''_{\mathsf{ag3}} = \ldots$

We call this first kind of constraints *requirement constraints* because they express that an agent needs the execution of some other actions at a specified time step to execute its own plan.

Is this first kind of constraints enough to guarantee the correctness of the individual plans in a distributed context ? Not quite. Because, individual plans may contain commitment actions, *e.g.*, $\mathsf{ag_1}$ is committing itself to execute action unload(c2,t2,l1) at time step 3. No constraint says that the mutex actions of unload(c2,t2,l1) must not be executed at the same level by the other agents. Hence, we have to consider a second kind of constraints called *commitment constraints* of the form (a,i) that express that the mutex actions of a specified action a must not be executed at a given level i.

Coordination Mechanisms. In order to coordinate their individual plans, each agent broadcasts the requirement and the commitment constraints of its individual plan. Then it tries to integrate the received constraints into its own individual plan. This integration follows a least commitment principle based on two coordination mechanisms.

First, each agent checks if the received constraints can be directly integrated in its individual plans previously computed. This checkout is quite simple and does not need any replanning mechanism. Indeed, the threats, the promotions and the mutexes are already in the agents planning graphs. Thus, agents have all the needed information to decide if the constraints can be satisfied. Consider the constraints related to π_{ag1}: the requirement constraints are $\{(\mathsf{load(c2,t2,l2)},0),(\mathsf{move(t2,l2,l1)},1)\}$ and the commitment constraints are $\{(\mathsf{unload(c2,t2,l1)},2)\}$. Let the individual plan of $\mathsf{ag_2}$:

$\pi_{\mathsf{ag2}} = \langle \mathsf{load(c1,t1,l1)}, \mathsf{move(t1,l1,l2)}, \mathsf{unload(c1,t1,l2)} \rangle$

and let us study how π_{ag2} is refined to integrate the constraints set from π_{ag1}. The requirement constraint $(\mathsf{load(c2,t2,l2)},0)$ of $\mathsf{ag_1}$ can be satisfied because no action of π_{ag2} at level 0 is mutex with load(c2,t2,l2). Hence, load(c2,t2,l2) is added to π_{ag2} at level 0 to check the requirement constraint of $\mathsf{ag_1}$. Consider now the commitment constraint of $\mathsf{ag_1}$. Is unload(c2,t2,l1) mutex with an action of π_{ag2} at level 2 ? In our example the answer is no. It means that π_{ag2} holds the commitment constraint of $\mathsf{ag_1}$: (unload(c2,t2,l1),2). But what about the commitment constraint (move(t1,l1,l2), 1) of $\mathsf{ag_1}$ and (move(t2,l2,l1), 1) of $\mathsf{ag_2}$ not yet considered ?

No previous individual plan of $\mathsf{ag_3}$ holds these constraints.

The second coordination mechanism consists in adding coordination constraints as CSP constraints in the CSP problem resulting from the planning graph encoding. The key idea of this mechanism is to extract an individual plan that takes into account the interactions between the activities of the agents. Remember that each agent encodes its planning graph in a CSP where propositions correspond to variables and actions to values. Thus, the question is how to encode requirement and commitment constraints into CSP constraints ? Let a coordination requirement constraint (a,i), this constraint cannot be directly added to the CSP. Suppose that an agent must find an individual plan satisfying the constraint (a,i) at level i. It means that at least one proposition or variable at level i must be supported by the action a. This constraint can be encoded as $(p_1 = a \vee, \ldots, \vee\ p_n = a)$ where $p_1, \ldots, p_n$ corresponds to propositions at level $i+1$. Let now consider a commitment constraint (a,i), it says that no action mutex with a can be executed at level i. Thus, the proposition variable at level $i+1$ cannot be supported by an action mutex with a. This constraints is encoded as $(p_1 \neq \mu A_i(a) \wedge, \ldots, \wedge\ p_n \neq \mu A_i(a)$ where $p_1, \ldots, p_n$ corresponds to propositions at level $i+1$ and $\mu A_i(a)$ the set of mutex actions of a at level i. Of course this coordination mechanism is more time consuming than the first one because it needs to extract a new individual plan. In our example, this mechanism will be used by $\mathsf{ag_3}$ to extract a coordinated individual plan satisfying the constraints $(\mathsf{move(t1,l1,l2)},1)$ from $\mathsf{ag_1}$ and $(\mathsf{move(t2,l2,l1)},1)$ from $\mathsf{ag_2}$.

To conclude with our example, the final coordinated individual plans obtained after the success of the coordination step of DPGM are as follow (actions executed by the other agents are not shown):

$$\pi_{\mathsf{ag1}} = \langle \mathsf{load(c1,t1,l1)}, \emptyset, \mathsf{unload(c2,t2,l1)} \rangle$$
$$\pi_{\mathsf{ag2}} = \langle \mathsf{load(c2,t2,l2)}, \emptyset, \mathsf{unload(c1,t1,l2)} \rangle$$
$$\pi_{\mathsf{ag3}} = \langle \emptyset, \{\mathsf{move(t1,l1,l2)}, \mathsf{move(t2,l2,l1)}\}, \emptyset \rangle$$

Coordination-Extraction Loop. Consider the case where both previous coordination mechanisms fail to coordinate individual plans. It does not mean that there is no solution at the specified level. Indeed, planning graphs may contain several individual plans at a same level. Thus, the agents must return in the extraction step to attempt to extract the other individual plans and try again to coordinate the new ones until no more individual plan can be proposed at the specified level (otherwise until all individual plans on a given planning graph length are exhausted).

First, let's deal with the termination case of this loop. If no other individual plan can be proposed at the considered level, the agents can either return to the expansion and try again to find a solution at the next level, or fail if no further expansion is possible (failure condition of the extraction step see previous section for more details).

Second, assume that one or several agents can compute new individual plans at the same level, the agents return to the coordination step and coordinate their new individual plans. In order to illustrate this specific loop of extraction and coordination, consider a generic example with three agents $\mathsf{ag_1}$, $\mathsf{ag_2}$ and $\mathsf{ag_3}$ and pick up at the point where the agents are about to coordinate their new individual plans at the same level for the second time, having respectively proposed the plans π_{ag1}, π_{ag2} and π_{ag3} at iteration 1 and π'_{ag1}, π'_{ag2} and π'_{ag3} at iteration 2. Each agent broadcasts the coordination constraints of their individual plans and stores the received constraints (requirements and commitments) into a table at each iteration. A possible constraints table is depicted Tab. 1. For instance, the coordination process could succeed if the individual plan π'_{ag1} asserted by $\mathsf{ag_1}$ at iteration 2 matches the constraints C_{12} from $\mathsf{ag_2}$ and C_{13} from $\mathsf{ag_3}$ or if π'_{ag1} matches the constraints C_{12} from $\mathsf{ag_2}$ and C_{23} from $\mathsf{ag_3}$. In other words, π'_{ag1} is a coordinated individual plan if it matches a combination of the coordination constraints broadcasts by the other agents. In the first case, π'_{ag1}, π_{ag2} and π_{ag3} are the solution to the planning problem and in the second case, π'_{ag1}, π_{ag2} and π'_{ag3}.

Table 1: Example of constraints table at a specified level.

Iterations	$\mathsf{ag_1}$	$\mathsf{ag_2}$	$\mathsf{ag_3}$
1	C_{11}	C_{12}	C_{13}
2	C_{21}	C_{22}	C_{23}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	C_{n1}	C_{n2}	C_{n3}

As a general rule, the coordination phase consists in testing all combinations of constraint sets. In the worst case, the number of combinations is exponential to the number of individual solution plans produced at a specified level. In practice, this complexity is tractable because many set of constraints can be pruned. Indeed, a constraint set can be pruned if it contains two constraints (a_1,i) and (a_2,i) such that a_1 and a_2 are mutex. Moreover, the coordination procedure can record failed constraint sets into a table as a distributed constraints nogood table, and checks each constraints sets with respect to the recorded constraints. The efficiency of this mechanism can be very

likely needed for some classes of problems and would require a complexity analysis of the cost of nogood caching. Finally, note that a constraints set is added to the table only if the second coordination mechanism fails to integrate a constraints set.

4 SUMMARY

In this paper, we introduce a fresh model for distributed planning called "Distributed Planning Through Graph Merging" (DPGM). This sound and complete model unifies the different steps of the distributed planning process into a single one based on planning graph structure used in centralized planning for agent reasoning and CSP mechanisms for individual plan extraction and coordination. The key idea is to incorporate as soon as possible, *i.e.*, in the local process of planning, the coordination step. The underlying reason for that consists in considering as soon as possible, *i.e.*, in the individual planning process, the interactions between agents activities in order to reduce the search cost of a global coordinated solution plan.

REFERENCES

Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artif. Intell.*, 90(1-2):281–300.

Cox, J. and Durfee, E. H. (2005). An efficient algorithm for multiagent plan coordination. In *AAMAS*, pages 828–835.

D'Inverno, M., Luck, M., Georgeff, M., Kinny, D., and Wooldridge, M. (2004). The DMARS architecture: A specification of the distributed multi-agent reasoning system. *JAAMAS*, 9(1-2):5–53.

Durfee, E. H. (2000). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, chapter Distributed Problem Solving and Planning. MIT Press.

Finke, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 3-4(2):189–208.

Iwen, M. and Mali, A. D. (2002). Distributed graphplan. In *ICTAI*, pages 138–145.

Kambhampati, S. (2000). Planning graph as a (dynamic) CSP: Exploiting EBL, DDB and other CSP search techniques in graphplan. *JAIR*, 12(1):1–34.

Pistore, M., Traverso, P., and Bertoli, P. (2005). Automated composition of web services by planning in asynchronous domains. In *ICAPS*, pages 2–11.

Sariel, S., Balch, T., and Erdogan, N. (2008). Naval mine countermeasure missions. *Robotics & Automation Magazine, IEEE*, 15(1):45–52.

Tonino, H., Bos, A., de Weerdt, M., and Witteveen, C. (2002). Plan coordination by revision in collective agent-based systems. *Artif. Intell.I*, 142(2):121–145.

Wilkins, D. and desJardins, M. (2001). A call for knowledge-based planning. *AI Magazine*, 22:99–115.