A Method to Integrate Planning and Coordination

Mathijs de Weerdt and Roman van der Krogt
Delft University of Technology
Faculty of Information Technology and Systems
P.O.Box 5031, 2600 GA Delft, The Netherlands
{M.M.deWeerdt,R.P.J.vanderKrogt}@cs.tudelft.nl

Abstract

Multi-agent planning involves finding a plan for each agent in a group where the goals, the actions and the initial resources are distributed over these autonomous agents. An algorithm is given for multi-agent planning that constructs coordinated agent plans distributedly. Instead of designing a planning algorithm and a coordination method separately, this algorithm integrates planning and coordination.

The presented multi-agent planning algorithm is based on the idea of several forward heuristic planners that run in parallel. These planners are coordinated by communicating side-products and services via a blackboard.

Introduction

Multi-agent planning involves the problem of coordinating the operations of a set of autonomous agents to achieve the goals of each individual agent, as defined in for example (Georgeff 1984; Muscettola & Smith 1989; von Martial 1991). A solution to this problem would be to construct a plan for all agents centrally. However, since autonomy is intrinsic to multi-agent problems, it is evident that a solution to this problem should be obtained distributedly. This means that several agents construct their plans concurrently to attain their own goals. However, since they co-exist in the same environment, they have to coordinate their actions and are sometimes able to help each other. In this paper we focus on a cooperative form of multi-agent planning, where agents are sincere and willing to help other agents to achieve their goals.

A conventional model for cooperative multi-agent systems assumes that each agent makes its own plans and then (partly) shares them with other agents to detect helpful or harmful interactions (Rosenschein 1982; Stuart 1985; Foulser, Li, & Yang 1992; de Weerdt *et al.* 2002), for example by applying additional restrictions to the construction of plans (Yang, Nau, & Hendler 1992). These methods are called *multi-agent plan merging* methods. In general, however, it is not always possible for each agent to first construct its plan and then to coordinate. Therefore, we study the *interleaving* of planning and coordination.

(Generalized) Partial Global Planning (PGP) (Decker & Lesser 1992; Durfee & Lesser 1987) is a technique to build such systems where agents communicate parts of their local plans to build plans that are partially global. These partial global plans specify the relations among actions, and can be used by agents to adapt their local plans to other agents' actions. This approach provides a framework to exchange crucial information in specific domains to prevent conflicts and potentially exploit positive interactions.

Most solutions to multi-agent planning problems, such as PGP, consist in fact of two parts: on the one hand *planning* methods such that each agent can find a plan for itself, and on the other hand *coordination* methods for the plans of the agents. The purpose of this paper is to describe a distributed algorithm that integrates planning and coordination.

In the next section the multi-agent planning problem is defined more formally as an extension of single-agent planning. Then we give a forward planning algorithm for multi-agent planning based on a single-agent planning heuristic. In the discussion we will explore the benefits and the drawbacks of the proposed method.

Multi-agent planning problems

Many planning problems involve multiple agents and thus require coordination. The purpose of this paper is to find a method to deal with such problems. It is very hard to find an algorithm that finds good solutions in reasonable time for all these problems. Therefore we look for problems with specific properties, for example the planning of a set of transport companies. Each transport company usually has a set of orders and some resources to execute these orders. The goals of an agent representing such a company are to fulfill as much orders as possible. Often a transport company is not able to fulfill all its orders on its own. Most situations with a couple of such transport companies have some noticeable properties:

1. The agents are *self-interested*, i.e., they are primarily focused on attaining their own goals.

2. The (initial) states of the agents are consistent, i.e., all agents base their state specification on the current state of the world, and they are only concerned with their own resources and goals.
3. The goals of the agents are consistent, because they have (usually) different orders (*conflict-free*).
4. The agents are *benevolent*, i.e., they are prepared to help each other (sometimes), because by cooperating they are able to fulfill more of their own goals as well.
5. The agents don't change the same things in the environment at the same time.

Although for each application some exceptions to these properties exist, many applications, such as cooperating taxi companies, freight transportation companies, and even military forces, usually have these properties. A problem that has all of the above properties is called a *[distributed self-interested] conflict-free benevolent multi-agent planning* problem.

The fact that multiple agents act in the same environment at the same time introduces many complications. The research topic *non-deterministic multi-agent planning* deals with situations where the result of an action is not known in advance. However, this approach is very hard, exactly because of the uncertain effects of actions. Another solution would be to introduce a semaphore to guarantee that only one agent accesses a certain part of the world at a time. Such a semaphore can be seen as another agent controlling only this part of the world. This latter method matches property 5 above.

Before formally defining the multi-agent planning problem in detail, we first briefly recapitulate single agent planning. Much of the following formal notation of a propositional STRIPS (Fikes & Nilsson 1971) planning instance is based on work done by Nebel. In propositional STRIPS a *state* is described by a set of propositions. Such a state can be transformed into another state by operators. An *operator* is defined by a precondition and its effects: $o = \langle pre, eff \rangle$. The formulas in the precondition of an operator must all be true in a state to which the operator is applied.

Definition 1 *(Nebel 2000) The* application *of an operator o to a state T is defined as*

$$App(T,o) = \begin{cases} T \cup eff^+(o) - \neg eff^-(o) \\ \qquad \text{if } T \models pre(o) \text{ and } o \text{ is consistent} \\ \text{undefined otherwise} \end{cases}$$

□

That is, the positive terms in the effect clause (defined by $eff^+(o)$) will be added to the state and the negations of the negative effects (defined by $\neg eff^-(o)$) will be removed. Using this definition, the result $Res(T, \Delta)$ of applying a sequence of operators Δ to a state T can be defined.

Definition 2 *(Nebel 2000) The result $Res(T, \Delta)$ of* applying a sequence of operators Δ *to a state T is defined by*

$$Res(T, \langle\rangle) = T$$

$$Res(T, \langle o; \Delta \rangle) = \begin{cases} Res(App(T,o), \Delta) \\ \qquad \text{if } App(T,o) \text{ is defined} \\ \text{undefined otherwise} \end{cases}$$

□

Finally, Nebel defines a planning problem in propositional STRIPS as follows:

Definition 3 *(Nebel 2000) A* planning problem *in propositional STRIPS is a four-tuple $\Pi = (\Sigma, O, I, G)$ where*

- *Σ is the set of propositional atoms, called facts,*
- *O is the set of operators to describe state-changes, containing only atoms from Σ,*
- *$I \subseteq \Sigma$ is the initial state, and*
- *$G \subseteq \Sigma$ is the goal specification, a set of propositions that is to be satisfied .* □

A sequence of operators $\Delta = \langle o_1, \ldots, o_n \rangle$ is called a solution or a plan for a planning instance $\Pi = (\Sigma, O, I, G)$ if and only if $Res(I, \Delta) \models G$.

Besides planning, in a multi-agent setting, agents also need to coordinate. Possible conflicts are prevented, because each agent deals with propositions concerning its 'own' part of the world. However, sometimes an agent may need to act on something that is administered by another agent. In such a case the relevant propositions can be transfered to the agent that needs them. For example, in a blocks world with multiple agents, each having one gripper, propositions $holding(A)$ (denoting that the agent has a block A in its gripper) cannot be transfered, whereas $free(A)$ (denoting that no block is on top of A) can. Sometimes, only a special set of propositions can be exchanged, for example when they all denote attributes of the same physical object that agents can interchange. We will call such a set of propositions (possibly a singleton) that can be exchanged a *resource*. So a resource is in fact an object-oriented way to encapsulate a group of propositions that somehow belong together.

To represent the transfer of resources from an agent a_i to a_j, we introduce two actions:

- A get action: $\mathbf{get}(a_i, r_k) = \langle \emptyset, r_k \rangle$ to represent receiving resource r_k from agent a_i. This action requires a put action in the plan of agent a_i, but has no (other) precondition and produces the propositions of r_k.
- A put action: $\mathbf{put}(a_j, r_k) = \langle r_k, \neg r_k \rangle$ to represent giving resource r_k to agent a_j. This action has the propositions of r_k as a precondition and consumes all these propositions.

Note that it should not be possible to include a **put**-action without the corresponding **get**-action (or vice versa). Therefore, we will require that in the final plan for a multi-agent planning problem, these actions only come in pairs. This way, propositions that are exchanged are deleted from one agent's state, and added to another's state. This ensures that the same proposition cannot be used by two different agents at the same time (as required by property 5).

Because most of the current planning domains are without an explicit time (other than a time step, which just counts the number of (parallel) actions that have been performed), it is very well possible for agents to get into a situation where there is a chain of agents where each agent is dependent on its successor, and where the last one is dependent on the first one via **get** and **put** actions. This situation is called a *circular dependency* and should not occur in any solution to a multi-agent planning problem, since such a set of plans cannot be executed.

Definition 4 *For a group of agents* $A = \{a_1, \ldots, a_n\}$ *a* multi-agent planning problem *is a tuple* $\Pi_A = (\Sigma, R, \{\Pi_a \mid a \in A\})$ *where*

- Σ *is the set of propositional atoms,*
- $R \subseteq 2^{\Sigma}$ *is the set of (disjunct) resources, and*
- $\Pi_a = (O_a, I_a, G_a)$, *with*
  - O_a *the set of operators to describe state-changes that can be done by agent* a*, containing only propositions from* Σ*, including a* ***get*** *and a* ***put*** *action for each resource/agent combination,*
  - $I_a \subseteq \Sigma$ *that part of the initial state that can be changed by agent* a*, and*
  - $G_a \subseteq \Sigma$ *the goal specification for agent* a*.*

Furthermore, we require that for two agents $a_i, a_j \in A$*: (i)* $I_{a_i} \cap I_{a_j} = \emptyset$*, i.e., each agent knows only about his own part of the world, and (ii)* $G_{a_i} \cap G_{a_j} = \emptyset$*, no two agents have the same goal (as this could lead to a conflict).* □

A sequence of operators $\Delta_a = \langle o_1, \ldots, o_n \rangle$ with $o_i \in O_a$, $1 \leq i \leq n$, is called a *solution* or a *plan* for a planning instance $(\Sigma, R, \Pi_a) = (\Sigma, R, (O_a, I_a, G_a))$ if and only if $Res(I_a, \Delta_a) \models G_a$.

A *solution* to a problem Π_A consists of a set of *consistent* solutions Δ_A to all problems in the problem set. A set of solutions $\Delta_A = \{\Delta_a \mid a \in A\}$ to a planning instance Π_A is consistent if and only if:

- every $\Delta_a \in \Delta_A$ is a solution to (Σ, R, Π_a),
- for every $\mathbf{get}\,(a_j, r_k) \in \Delta_{a_i}$ action with $i \neq j$, there is a $\mathbf{put}\,(a_i, r_k) \in \Delta_{a_j}$, and
- the exchange of resources does not introduce circular dependencies.

We define $out(P)$ as the set of propositions that is the final state reached by a plan. Such a plan P may be a *partial plan*, i.e., a plan that is not yet the solution to the particular planning problem at hand. We also define $in(P)$ to be the "precondition" of a plan: the smallest set of propositions that can form an initial state from which a plan can execute properly.

Finally, in some cases it is convenient for the agents to have a library of plan schemes PS, filled with plans for the services they can provide. These plan schemes are defined as formal plans (i.e., a sequence of actions), but have no associated initial state or goals. These can then be used to "advertise" services to other agents.

Complexity analysis The following result can be derived for a multi-agent planning problem:

Proposition 1 *The complexity class of the self-interested conflict-free benevolent multi-agent planning problem for* n *agents with at most* m *operations each is equal to that of single-agent planning with* $O(n \cdot m)$ *operations.* □

Proof First, note that multi-agent planning is at least as hard as single agent planning, since any single agent planning problem can be translated to a multi-agent planning problem with exactly one agent. To prove that it is not harder than single-agent planning, we show that it is possible to transform the multi-agent problem to a single-agent problem and that the solution of the single-agent problem can be translated back, all in polynomial time. We translate the problem $\Pi_A = (\Sigma, R, \{(O_a, I_a, G_a) \mid a \in A\})$ to a single-agent problem $\Pi = (\Sigma', O, I, G)$, where

- $\Sigma' = \bigcup_{a \in A} \{p_a | p \in \Sigma\}$, all propositions in Σ are labeled with the names of the agents,
- $O = \bigcup_{a \in A} \{\langle pre_a, eff_a \rangle \mid \langle pre, eff \rangle \in O_a\} \cup \{\mathbf{putget} = \langle r_{k,a}, r_{k,b} \rangle | b \in (A - \{a\}), r_k \in R\}$, the possible actions are the actions of the individual agents, of which the pre- and post-conditions are labeled with the name of the agent, and actions **putget** are added that can change the label of the propositions in resources from one agent name to the other,
- $I = \bigcup_{a \in A} \{p_a | p \in I_a\}$, the initial state is the combination of the initial states of the agents, labeled with the name of that agent whose initial state it is, and
- $G = \bigcup_{a \in A} \{p_a | p \in G_a\}$, the goals of the new problem are the goals of the multi-agent problem, again labeled with the name of the agent that has that goal.

The solution (which is a sequence of operators Δ) to this problem can easily be translated back to a set of multi-agent plans Δ_a: first replace each $\mathbf{putget}(r_{k,a}, r_{k,b})$ action by a $\mathbf{put}(b, r_{k,a})$ and a $\mathbf{get}(a, r_{k,b})$ action. Then distribute the actions over the agents: agent a_i gets to execute all actions for which the pre- and postconditions are labeled with a_i. Now remove all labels and each agent has a plan Δ_a which can successfully execute from its initial state I_a to reach its goals G_a. □

Forward heuristic cooperative planning

This section describes an algorithm that facilitates self-interested cooperative conflict-free planning in a multi-agent system, based on a forward heuristic planner for each agent. This heuristic is similar to the one used in Fast Forward (FF, see (Hoffmann & Nebel 2001)). We assume that agents are able to communicate for cooperation, but each agent is free to decide to what extent. Communication is implemented by offering superfluous resources (*side-products*) to each other, and by offering resources that one agent is prepared to produce exclusively for another (called *services*). If the

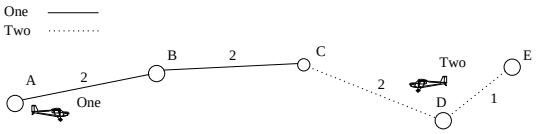

Figure 1: A situation with two agents, One and Two, that are ready to transport light weight goods and persons. Each agent has its own transport network.

agents are benevolent, many resources are offered this way. Conflicts concerning resources can now be solved by exchanging them.

Suppose that we have a set of agents, and that each agent a has a problem instance $\Pi_a = (O_a, I_a, G_a)$, and a set of plan schemes PS_a that represent services to produce resources for other agents. The problem to be solved is to find a plan for each agent to satisfy all its goals, starting from the initial state, and to determine additional goals for each agent to represent the transfer of a resource to another agent.

Heuristic Planning We propose the following method. Each plan is constructed bottom-up: starting with the initial state actions are added to the plan. Once an action has been added it may not be removed again. This form of planning is called *forward state space refinement planning* (Kambhampati 1997). Only actions whose preconditions have been fulfilled are added to a plan, so circular dependencies cannot be introduced.

The choice of which action to add to a fixed first part of a plan is done in the same way as in (Hoffmann & Nebel 2001): using a heuristic function. The heuristic function determines the cost of achieving the goal from the current state based on the cost of all actions in the future plan.[1] A state is represented by a set of propositions T, and usually is the result of a partial plan P, i.e., $T = out(P)$. To determine the heuristic value of a state, we assume actions do not have negative effects. Under this assumption the heuristic value (h) of a new state can be determined in polynomial time in $|O_a| + |I_a| + l$, where l is the length of the longest add list of an action in O_a. The sequence chosen within the heuristic is called a *heuristic plan.*

Definition 5 *Given a partial plan P, a set of goals G, and a heuristic plan P^h, the set of propositions $free\,(P, P^h, G)$ is defined as all propositions occurring in the final state of the heuristic plan except those that are used to fulfill the goals: $out\,(P^h) - G$.* □

Blackboard Each agent offers resources to others on a blackboard (Hayes-Roth 1985). We allow an agent to decide itself which resources it would like to offer, although in the algorithm below we describe how an agent offers *all* its superfluous resources (*side-products*). These offers are updated (by UPDATEBB) each time actions have been added (using the enforced hill-climbing method given in (Hoffmann & Nebel 2001), called PLANSTEP here). We allow agents to offer two kinds of resources: firstly, resources in the current state that they don't expect to need themselves, that is, consisting of propositions that are not used in their own heuristic plan, and secondly, resources that represent certain services. The agent should be able to attain such resources by using one of its plan schemes.

If another agent requests a resource r and if these propositions are available, i.e., $r \subseteq free(P_a, P_a^h, G_a)$, a **put** action can be added to the plan. Otherwise a plan scheme ps that produces this resource is added to the plan if the precondition of ps is fulfilled, i.e., $in(ps) \subseteq free(P_a, P_a^h, G_a)$. The requesting agent then has to wait for the resource until all propositions are available. If none of these activities is possible, the requesting agent receives a failure. If several agents request the same resource, they are served on a first come first served basis. So, in this case all except the first one will get a negative response. This method is implemented by the REQUEST function that runs concurrently with the COOPERATIVEPLANNING function. A REQUEST is usually remotely invoked by the requesting agent a_j on the serving agent a_i.

REQUEST (a_j, r)

1. **if** $r \subseteq free(P_{a_i}, P_{a_i}^h, G_{a_i})$ **then**
   add **put**(a_j, r) *to* P_{a_i}
   add r *to* G_{a_i}*; reply success*
2. **elseif** $\exists ps_{a_i} \in PS_{a_i} : r \subseteq out(ps_{a_i})$ *and* $in(ps_{a_i}) \subseteq free(P_{a_i}, P_{a_i}^h, G_{a_i})$ **then**
   add ps_{a_i} *and* **put**(a_j, r) *to* P_{a_i}
   add r *to* G_{a_i}*; reply success*
3. **else**
   reply fail

On the receiving side, agents can use resources offered by others by the introduction of a **get** action for each resource that is offered. These actions can be used in the planning process and even in the heuristic in the same way as the usual actions. A more precise specification of the method described above can be found in the COOPERATIVEPLANNING algorithm.

Algorithm COOPERATIVEPLANNING (a_i)

Input: *For agent a_i: its goals G_{a_i}, its initial resources I_{a_i}, its possible actions O_{a_i}, the blackboard BB, a set of plan schemes PS_{a_i}, and a set of resources R.*

Output: *A plan P_{a_i} to achieve G_{a_i}.*

begin

1. $P_{a_i} := \langle\rangle$; $T_{a_i} := I_{a_i}$; $h_{a_i} := \infty$; $P_{a_i}^h := \langle\rangle$
2. **repeat**
   $O'_{a_i} := O_{a_i} \cup \{\mathbf{get}(a_j, r) \mid (a_j, r) \in BB\}$
   PLANSTEP$(P_{a_i}, O'_{a_i}, T_{a_i}, G_{a_i}, P_{a_i}^h, h_{a_i})$

[1] Currently we assume uniform costs, e.g., $cost(o) = 1$ for each $o \in O$.

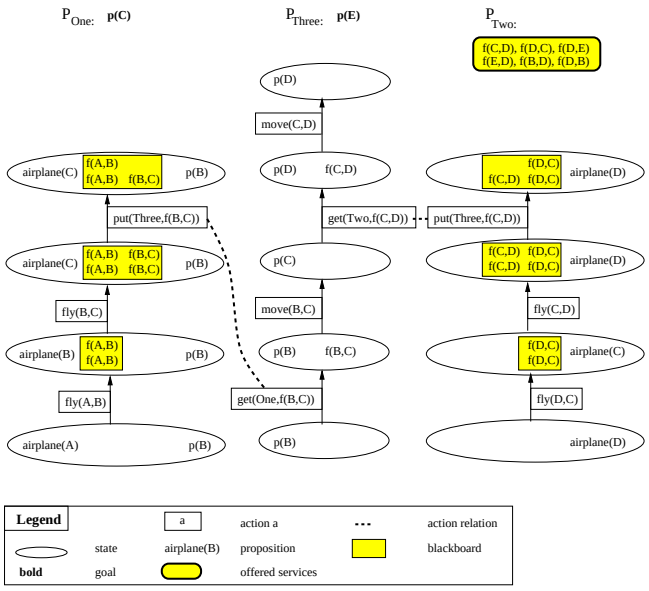

Figure 2: The partial plans of agents One, Two and Three during the planning process. Furthermore, this figure shows the resources and services that are offered by each agent.

UPDATEBB$(P_{a_i}, O'_{a_i}, T_{a_i}, G_{a_i}, P^h_{a_i}, R, BB)$
until $P_{a_i} \models G_{a_i}$

3. *announce 'finished'*
4. **repeat**
   UPDATEBB$(P_{a_i}, O'_{a_i}, P^h_{a_i}, R, BB)$
   until *all agents are finished*
5. **return** P_{a_i}

end

UPDATEBB $(P_{a_i}, O'_{a_i}, T_{a_i}, G_{a_i}, P^h_{a_i}, R, BB)$

1. $R_1 := \mathit{free}(P_{a_i}, P^h_{a_i}, G_{a_i})$
2. $PS'_{a_i} := \{ps \in PS_{a_i} \mid \mathit{in}(ps) \subseteq \mathit{free}(P_{a_i}, P^h_{a_i}, G_{a_i})\}$
3. $R_2 := \bigcup_{ps_{a_i} \in PS'_{a_i}} \mathit{out}(ps_{a_i})$
4. **for each** $r \in 2^{R_1 \cup R_2} \cap R$ *update* (a_i, r) *on* BB
5. *remove all other offers by this agent*

Example The following example illustrates the COOPERATIVEPLANNING algorithm. Assume we have three companies: One and Two each have a small airplane (with capacity 2) and rights to land at certain airports (see Figure 1) and Three is a travel agency without any airplanes, but it has many customers that wish to travel. We suppose that One has a goal to get a passenger from B to C, and Three has also one goal: to get a passenger from B to E. Two has nothing to do, but offers some services.

Since we would like companies to exchange capacity, we model this problem by defining the following resources: $R = R_1 \cup R_2$, where

$$R_i = \{\{\mathrm{f}_i(A,B)\}, \{\mathrm{f}_i(B,A)\}, \{\mathrm{f}_i(B,C)\}, \{\mathrm{f}_i(C,B)\}, \{\mathrm{f}_i(C,D)\}, \{\mathrm{f}_i(D,C)\}, \{\mathrm{f}_i(D,E)\}, \{\mathrm{f}_i(E,D)\}\}$$

The propositions $\mathrm{f}_i(x,y)$ represent the possibility for one person or package to fly from x to y (for a fixed $x, y \in \{A,B,C,D,E\}$). The subscript i indicates the seat number.

Agent One is perfectly able to reach its goal by having its airplane fly from A to C. By the exchange of free resources, Three can buy capacity from B to C, but the passenger won't be able to reach E by this exchange alone. Fortunately, company Two offers many services, such as $\mathrm{f}_1(C,D)$, $\mathrm{f}_1(D,C)$, $\mathrm{f}_1(E,D)$ and $\mathrm{f}_1(D,E)$. By receiving for example $\mathrm{f}_1(C,D)$ and $\mathrm{f}_1(D,E)$ even Three is able to attain its goal. Intermediate states of the plans during the planning of agents One, Two and Three can be found in Figure 2. As can be seen from this example, agent Two is activated by agent Three, that otherwise would not be able to attain its goals.

Analysis Analyzing the algorithm we see that in two cases an agent may not be able to reach its goals anymore, because we do not back-track. Firstly, an agent may add an erroneous, irreversible action to its plan, because the used heuristic is can fail. However, the AIPS planning competition in 2000 (Hoffmann & Nebel 2001) has shown that a smart heuristic can ensure that this is no problem in most cases.

Secondly, in the multi-agent setting an agent may need propositions it has already given away. However, this has also the advantage that agents do not have to (and are not allowed to) withdraw previously made commitments as long as these commitments are based on the fixed first part of agents' plans. This prevents escalations of problems where one agent's problem causes all other agents to start all over again.[2]

Concerning the time-complexity, in the worst case, this algorithm is not polynomial. It uses an enforced hill-climbing technique, that worked well in practice, but, theoretically, it may have to search the whole state tree for an improvement of the heuristic.

These rather negative results can be explained by the difficulty of the problem that is being solved. The multi-agent planning problem is at least as hard as a single-agent planning problem (see Proposition 1), which is PSPACE-hard. (Bylander 1994). The good news is that the dead ends for single agent planning using this heuristic seldomly occur and the algorithm performs extremely well (Hoffmann & Nebel 2001).

Conclusion

We have seen that even a rather simple market mechanism to support the exchange of resources, i.e., partial results of plans, can help to coordinate multiple agents, each trying to construct plans to attain their own goals. The proposed algorithm provides a distributed way for

[2]The autonomy of the agents can also prevent agents from attaining their goals, for example, when one of the agents keeps some resources that other agents need to fulfill their goals.

autonomous agents to construct coordinated plans for a large class of multi-agent planning problems. Unfortunately the algorithm is incomplete, but we have seen that this is inevitable in the view of agents that are fully autonomous, and given that the multi-agent planning problem is PSPACE-complete.

By constructing the plans bottom-up, the problem of introducing circular dependencies of agents' states was circumvented. The blackboard that is used in our forward heuristic cooperative planning implements a market mechanism without any payments. A real computational market, as introduced by Wellman *et al.* (Wellman 1997) would add enormously to the potential applications of our approach. Instead of requiring each agent to achieve its goals, these goals should represent certain profits and agents would have the freedom to choose between them. So we would not need the assumption about the goals of all agents being 'conflict-free' anymore. Furthermore, using such a price mechanism it is far more naturally to have agents constructing plans on request. This functionality is now only included in a very basic way by assuming the availability of a set of plan schemes.

New ideas and algorithms from the AI Planning community can have also a great impact on planning in a multi-agent context. Particularly *non-deterministic planning* may be very interesting, since especially in a multi-agent context the effect of actions is unsure, because other agents also execute actions. Other interesting extensions are also possible, for example working with time and quantitative resources (Do & Kambhampati 2001). Finally, in the action-resource planning formalism, as proposed in (de Weerdt *et al.* 2002), a state is not modeled by a set of propositions, but by a set of resources. Each resource is labeled with a unique identifier. Such a slightly different formalism would make the interpretation of 'exchangeable propositions' alias resources easier.

Acknowledgments Mathijs de Weerdt is supported by the Seamless Multi-modal Mobility (SMM) research program and Roman van der Krogt is supported by the Freight Transport Automation and Multi-modality (FTAM) research program. Both programs are carried out within the TRAIL research school for Transport, Infrastructure and Logistics.

References

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1–2):165–204.

de Weerdt, M. M.; Bos, A.; Tonino, H.; and Witteveen, C. 2002. A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence, special issue on Computational Logic on Multi-Agent Systems, to appear.*

Decker, K. S., and Lesser, V. R. 1992. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems* 1(2):319–346.

Do, M., and Kambhampati, S. 2001. Sapa: A domain-independent heuristic metric temporal planner. In *Proceedings of the Sixth European Conference on Planning (ECP-01)*, 109–120.

Durfee, E. H., and Lesser, V. R. 1987. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, 875–883.

Fikes, R. E., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 5(2):189–208.

Foulser, D.; Li, M.; and Yang, Q. 1992. Theory and algorithms for plan merging. *Artificial Intelligence Journal* 57(2–3):143–182.

Georgeff, M. P. 1984. A theory of action for multiagent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, 121–125.

Hayes-Roth, B. 1985. A blackboard architecture for control. *Artificial Intelligence* 26:251–321.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research* 14:253–302.

Kambhampati, S. 1997. Refinement planning as a unifying framework for plan synthesis. *AI Magazine* 18(2):67–97.

Muscettola, N., and Smith, S. F. 1989. A probabilistic framework for resource-constrained multi-agent planning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, 1063–1066.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *Journal of AI Research* 12:271–315.

Rosenschein, J. S. 1982. Synchronization of multi-agent plans. In *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, 115–119.

Stuart, C. J. 1985. An implementation of a multi-agent plan synchronizer. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, 1031–1033.

von Martial, F. 1991. *Coordinating Plans of Autonomous Agents via Relationship Resolution and Communication.* Ph.D. Dissertation, Universität des Saarlandes.

Wellman, M. P. 1997. Market-aware agents for a multiagent world. In *Lecture Notes on Computer Science*, volume 1237. Springer Verlag.

Yang, Q.; Nau, D. S.; and Hendler, J. 1992. Merging separately generated plans with restricted interactions. *Computational Intelligence* 8(4).