Combining Planning and Coordination to Improve Efficiency in Transport

TRAIL Research School, Delft, November 2002

Authors

Mathijs de Weerdt

Faculty of Information Technology and Systems, Delft University of Technology

Contents

Abstract

1 Introduction 1

2 Multi-agent planning in transport 2

3 Formal definition of multi-agent planning 4

3.1 Single-agent planning 4

3.2 Multi-agent planning 5

3.3 Forward heuristic cooperative planning 7

4 Discussion 12

Acknowledgments 14

References 15

Abstract

In transport, and especially public transport, many inefficiencies occur due to lack of coordination between companies. To reduce such inefficiencies, a multi-agent planning method is proposed that can be applied to coordinate the plans of both public transport and freight transport organizations.

The proposed algorithm integrates the coordination of actions into the plan construction phase of each company. Plans are constructed from start to end, and step by step. In each step a set of possible actions is evaluated using a heuristic. One of these actions is selected and included in the plan. at the end. Also, in each step, side-products and services are offered to other companies via a so-called blackboard.

Experiments for single-agent planning indicate that this polynomial approach can work rather well to solve the NP-hard (multi-agent) planning problem.

Keywords

coordination, planning, multi-agent systems, transport

1 Introduction

In transport, and especially public transport, many inefficiencies occur due to lack of coordination between companies.[1] For example, taxis and busses are often empty, many people do not use the car-pooling system because it is 'too cumbersome', and different modalities usually do not cooperate (in the Netherlands, except for 'treintaxi', i.e., a shared taxi to and from the railway station, and an ordinary, non-shared taxi). To improve the efficiency of public transport, and especially of those forms of transport that do not follow fixed schedules (different forms of dial-a-ride taxis and busses, and even charter planes), the companies involved should coordinate their actions. In this paper, a mechanism is introduced to support coordination of actions in general, and it is shown how this method can be applied to situations in transport.

The problem of coordinating the operations of a set of transport companies to achieve the goals of each individual company is called *multi-agent planning*, as defined by for example Von Martial (1991), who uses domain knowledge to create a hierarchy of actions, and uses the hierarchy to determine conflicts between agents' plans. The *agents* in multi-agent planning are the (transport) companies.

One possible solution to this multi-agent planning problem is to construct a plan for all companies centrally. However, most of these companies independently try to make profit. Therefore, such a central planner is not acceptable for those companies, and thus a coordinated planning method should be distributed, such that every agent retains its autonomy. This means that several (company) 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, possibly rewarded by some amount of money.

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, such as the approaches of Rosenschein (1982), Georgeff (1984), Stuart (1985), Foulser et al. (1992), and Tonino et al. (2002). 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, for example, when a company has to transport more passengers than it has capacity. In such a situation, actions of other companies should be incorporated in the planning. Therefore, we study the *interleaving* of planning and coordination.

In the next section the multi-agent planning problem is defined more precisely, and the general idea of the proposed distributed coordinated planning approach is given. In Section 3, a planning problem and its solution are described formally using a propositional planning language called STRIPS, introduced by Fikes and Nilsson (1971). In the discussion, we show how the coordinated planning technique can be applied to transport planning problems, and discuss some potential improvements.

[1] By coordination we mean 'the regulation of diverse elements into an integrated and harmonious operation'. In this paper the 'elements' to regulate are the actions of the transport companies. Such physical actions are regulated by regulating the representations of these actions in the plans of the companies.

2 Multi-agent planning in transport

Many planning problems, and especially concerning transportation, involve multiple interdependent agents (i.e., companies, people) and thus require coordination. The purpose of this paper is to find a method to deal with such planning and coordination problems. Each transport company usually has a set of orders and some resources to execute these orders. The goals of the agents representing such companies are to fulfill as much orders as possible. Often transport companies are not able to fulfill all their orders on their own. Most situations with a couple of such transport companies have some noticeable properties:

1. The agents (i.e., companies) 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, in other words the orders, are *different* for each agent.

4. The agents are *benevolent*, i.e., they are prepared to help each other, when it does not conflict their own goals, because this way they can earn some extra money, and 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, e.g., try to drive the same car or pickup the same passenger.

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. This problem and the properties above are defined formally in the next section.

The fact that multiple agents act in the same environment at the same time introduces many complications, such as that other agents' actions may interfere with your own. 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 complicated, exactly because of the uncertain effects of actions. Another solution is 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. This way, reaching the goals is conflict-free (or impossible if the agent does not coordinate).

Problems that match the given properties are somewhat easier to deal with. Fortunately, as we have already seen, most transport planning problems fit into this category. In short, the idea behind a solution to these problems is as follows. When the agents are constructing plans to reach their own goals, they are offering each other some of the results of their partial plans at some kind of simulated market (see Figure 1). We define *products* as these results of a partial plan. In addition to this offering of products, agents

can also offer some *services*. A service is defined as the result of a partial plan that an agent is able to execute, but that has not been included in its plan (yet). When another agent requests a service, the corresponding partial plan is added to the plan. So services are more generally applicable products that can be 'produced' on demand.

Note that we speak of 'products' to denote any result of an action (or partial plan) of an agent. For transport companies products are for example *possibilities* to transport cargo or people from one place to another, or for example a truck at a certain location.

In the next section we describe the conflict-free benevolent multi-agent planning problem more formally, and we present the proposed solution in detail.

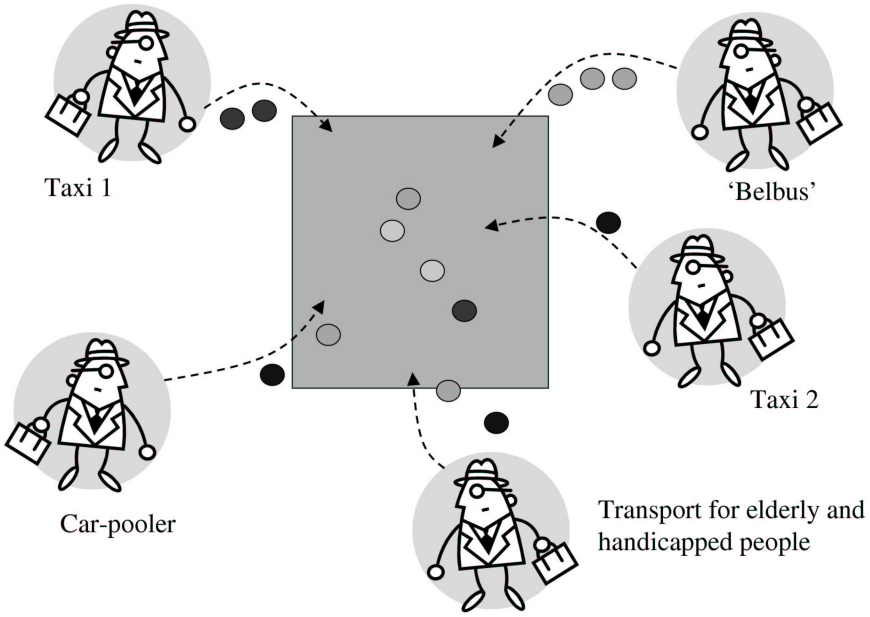

Figure 1: A number of agents, representing different companies offer some of their 'products' on a blackboard. These products are symbolized by the small balls.

3 Formal definition of multi-agent planning

We first briefly introduce single agent planning. Much of the following formal notation of a propositional STRIPS (Fikes and Nilsson, 1971) planning instance is based on work done by Nebel (2000).

3.1 Single-agent planning

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: $o = \langle pre, \mathit{eff} \rangle$. The formulas in the precondition of an operator must all be true in the state to which the operator is applied. For example, the operator "move from Amsterdam to Rotterdam" has as a precondition "being in Amsterdam" and as effect "being in Rotterdam" and "not being in Amsterdam" anymore.

Definition 1 *The* application *of an operator o to a state T is defined as*[2]

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

That is, the positive terms in the effect clause (defined by $\mathit{eff}^+(o)$) will be added to the state and the negations of the negative effects (defined by $\neg \mathit{eff}^-(o)$) 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 *The result $Res(T, \Delta)$ of* applying a sequence of operators $\Delta = \langle o_1, \ldots, o_n \rangle$ *to a state T is defined by*

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

$$Res\,(T, \Delta) = \begin{cases} Res\,(App(T, o_1), \langle o_2, \ldots, o_n \rangle) & \text{if } App(T, o_1) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Finally, Nebel (2000) defines a planning problem in propositional STRIPS as follows:

Definition 3 *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, and $\hat{\Sigma}$ is the set of literals based on Σ, i.e., $\hat{\Sigma} = \{p, \neg p \mid p \in \Sigma\}$.*
- *$O \subseteq 2^{\hat{\Sigma}} \times 2^{\hat{\Sigma}}$ is the set of operators to describe state-changes, containing only literals from Σ,*
- *$I \subseteq \hat{\Sigma}$ is the initial state, and*
- *$G \subseteq \Sigma$ is the goal specification, a set of propositional atoms 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 $\Pi = (\Sigma, O, I, G)$ such that for $1 \leq i \leq n$ holds that $o_i \in O$, if and only if $Res(I, \Delta) \models G$.

[2] In these definitions we use the standard entailment operator ($\models$) from propositional logic.

3.2 Multi-agent planning

Agents need to both plan and coordinate in a multi-agent setting. 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, a taxi-company can have a special bus equipped for wheel-chairs, denoted by the proposition $has(wheel-chair-bus)$. Transferring this proposition to another agent is the formal equivalent of the actual change of control of the bus to another agent. Sometimes, propositions can only be exchanged in specific combinations, for example when they all denote attributes of the same physical object. 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. It is important to realize that the definition of a resource is very domain dependent. Except that in any domain any resource is a set of propositions.

Example Suppose, in some city, we have three bakers that each can bake two of cookies, cake, and bread. For this they need sugar, milk, and flour. If one of the baker is ill and not able to bake cake, it can request some of the others. If a baker forgot to buy enough flour, it can try to get some of the others as well. In this 'bakery' domain, the resources are cookies, cake, bread, sugar, milk, and flour.

For planning we usually use propositional logic (like STRIPS), so each of these six resources has to be represented by one or more propositions. For example, if a baker has a 'bread' resource, the following propositions are included in its state: $has(bread_1)$, $date(bread_1, \text{26-11-2002})$, and $type(bread_1, sesame)$. In a similar way, each of the other resources is a set of propositions.

Example For a public transport domain, resources are *(i)* transportation devices, such as taxis, special busses, and trains, *(ii)* passengers, and *(iii)* travel capacities (rides). Each of these resources is represented by a set of propositions that are included in the state of the concerning agent. For example, the propositions that represent a taxi resource are $has(taxi_2)$, $capacity(taxi_2, 3)$, $free\_cap(taxi_2, 2)$, $fuel\_level(taxi_2, 30)$, and $location(taxi_2, \text{Amsterdam - Marnixstraat})$.

Transferring resources When a resource is transferred to another agent, all the representing propositions should be deleted from the state of the sending, and added to the state of the receiving agent.

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

- 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 in the previous section).

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,*
  - $I_a \subseteq \Sigma$ *the initial state of agent* a*, and*
  - $G_a \subseteq \Sigma$ *the goal specification for agent* a*.*

Furthermore, we require that for two agents $a_i, a_j \in A$ $(i \neq j)$*:* (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).*

We claim that any planning problem involving multiple agents that satisfies the 5 properties from the previous section, can be modeled as a multi-agent planning problem $\Pi_A = (\Sigma, R, \{\Pi_a \mid a \in A\})$ defined above.

1. Self-interestedness ensures that each agent a has its own set of goals G_a.
2. The facts that the inital states are consistent (and based on the real world), and that the agents are only concerned with their own resources, ensure the requirement $I_{a_i} \cap I_{a_j} = \emptyset$ in Definition 4.
3. The goals are different, so $G_{a_i} \cap G_{a_j} = \emptyset$.
4. That the agents are benevolent is not really required for a problem being a multi-agent planning problem, but if a problem does not have this property the algorithm presented in the next section will not be very successful.
5. The property that agents only perform actions on their 'own' objects in the world, ensures that not only for the initial states, but also for any intermediate state $T_A = \{T_a \mid a \in A\}$ it holds that the agent states T_{a_i} and T_{a_j} for two agents $a_i, a_j \in A$ $(i \neq j)$ are disjunct, in other words, it holds that $T_{a_i} \cap T_{a_j} = \emptyset$.

Multi-agent plan 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
- there is no 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. Such situations are called *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. These plan schemes are defined as plans (i.e., a sequence of actions), but without associated initial state or goals. The output resources of these plan schemes are the services that can be offered to other agents, whenever the input resources are available.

3.3 Forward heuristic cooperative planning

To solve problems that match the description of self-interested cooperative conflict-free planning in a multi-agent system, we use a single-agent planning technique called *forward heuristic planning*, and combine this with communication via a blackboard.

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 for another (the *services*, as described in the previous section). If the 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, a multi-agent planning problem instance Π_A, and a set of plan schemes PS_a that describe the partial plans to produce resources for other agents (services). 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 end of 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 Hoffmann and Nebel (2001) did: 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.[3] 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 $\left(P, P^h, G\right)$ is defined as all propositions occurring in the final state of the heuristic plan except those that are used to fulfill the goals: $out\left(P^h\right) - G$.*

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 by Hoffmann and 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. When an agent a_j sees an interesting resource r on the blackboard, it invokes REQUEST remotely on the serving agent a_i, which runs the following code.

REQUEST (a_j, r)

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

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

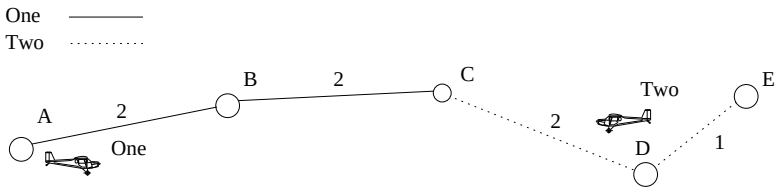

Figure 2: A situation with two agents, One and Two, that are ready to transport light weight goods and persons. Each agent has its own transport network.

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.

This algorithm consists of two phases: in the first phase a plan is constructed for the agent itself, and the blackboard is updated. Once the agent has completed its plan, some of the other agents may still be looking for some resources. Therefore, the algorithm contains a second phase in which the agent is only updating the blackboard.

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

The updates of the blackboard are done by the function UPDATEBB in the COOPERATIVEPLANNING algorithm. This function updates the offers of free resources and of the output resources of plan schemes which input resources can be found.

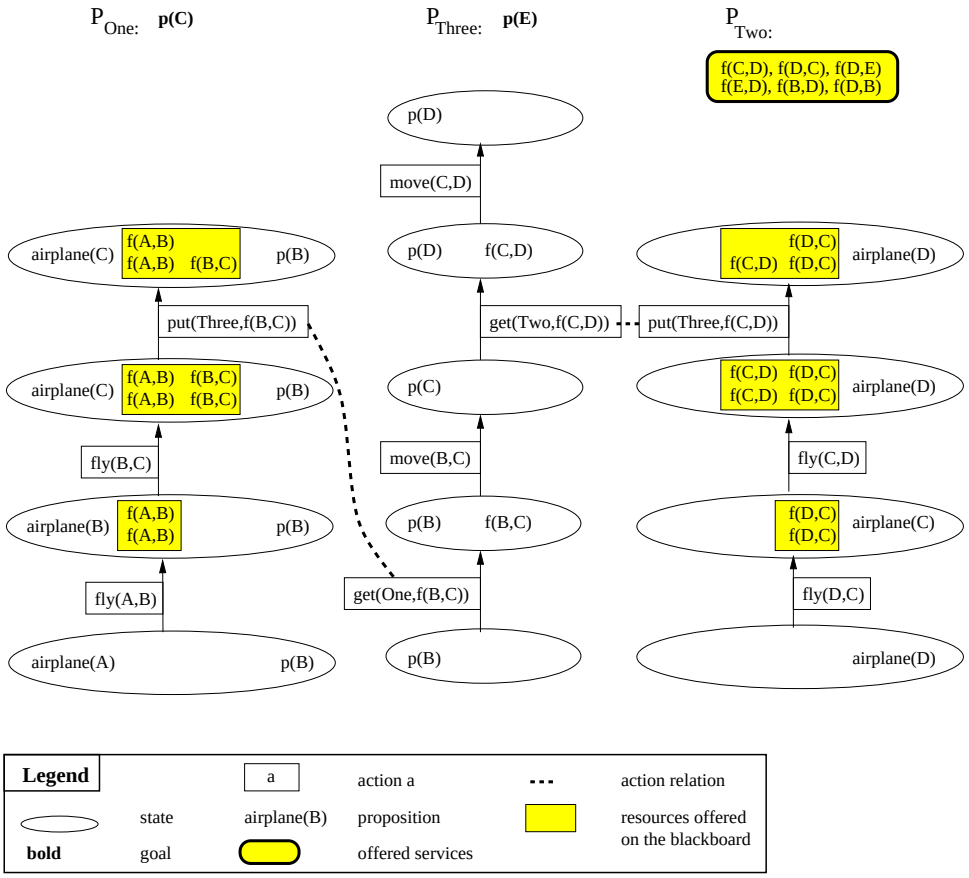

Figure 3: The partial plans (read from bottom to top) of agents One, Two and Three during the planning process. Furthermore, this figure shows the resources and services that are offered by each agent on the blackboard in each step.

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 2) 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

$$R = \left\{ \begin{array}{cccc} \{\mathrm{f}_i(A,B)\}, & \{\mathrm{f}_i(B,A)\}, & \{\mathrm{f}_i(B,C)\}, & \{\mathrm{f}_i(C,B)\}, \\ \{\mathrm{f}_i(C,D)\}, & \{\mathrm{f}_i(D,C)\}, & \{\mathrm{f}_i(D,E)\}, & \{\mathrm{f}_i(E,D)\} \end{array} \middle| \; i = 1,2 \right\}.$$

The propositions $\mathrm{f}_i(x,y)$ represent the possibility for one person or package to fly from x to y (for a fixed $x, y \in \{A, B, C, D, E\}$). Each airplane has a capacity of 2.

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 3. As can be seen from this example, agent Two is activated by agent Three, that otherwise would not be able to attain its goals.

The same method can be applied to for example the planning of taxi companies, busses, and car-pooling.

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 can fail. However, the AIPS planning competition in 2000 (Hoffmann and Nebel, 2001, cf.) has shown that a smart heuristic can ensure that in most cases, this is no problem.

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 proliferation of problems where one agent's problem causes all other agents to start all over again.[4]

The multi-agent planning problem is as hard as a single-agent planning problem, which is PSPACE-hard, as proven by Bylander (1994). The good news is that the dead ends for single agent planning using this heuristic seldom occur and that the algorithm performs extremely well (Hoffmann and Nebel, 2001). This (single-agent) planner performed also very well for the "logistics" domain, one of the problem domains for the AI Planning competition, where the planner has to allocate objects to airplanes and trucks such that all objects reach their destinations.

The algorithm uses an enforced hill-climbing technique, that works well in practice, but, theoretically, it may have to search the whole state tree for an improvement of the heuristic.

[4]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.

4 Discussion

Not all transport companies are prepared to combine their resources and commit to a centrally coordinated and computed planning. However, coordination is indispensible for efficient transportation. So, we may conclude that there is a need for a more distributed approach to coordinate such that each participant can retain its autonomy.

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 autonomous agents to construct coordinated plans for a large class of multi-agent planning problems. Unfortunately, the algorithm is incomplete (i.e., it does not always find a solution when there is one), but this is inevitable in the view of agents that are fully autonomous, and given the fact that the multi-agent planning problem is PSPACE-complete.

Applications to transport Using this planning algorithm, we can have many autonomous transport companies efficiently cooperate without a central planning authority. This could lead to a system where many public transport methods are coupled. Modalities such as the 'treintaxi' (a taxi from and to a railway station), car-pooling, the 'belbus' (dial-a-ride bus), special transport facilities for elderly and disabled people, and shared (air) taxis could take advantage of a shared capacity.

Such a system has some prerequisites that are hard to establish, but not impossible, such as a fair cost model, and an information-infrastructure to each of the (mobile) participants. Every company and person should have some sort of device, for example a mobile phone, to communicate with the others and offer or request capacity. All travelers then can connect to this system and decide for themselves whether they use offered resources (trips), go by car and thus supply some capacity by themselves, or go by car without offering a trip to others.

Likewise, such a system can be used to share capacity among freight transport companies without explicitly constructing a central plan. An example where the latter approach is taken can be found elsewhere in these proceedings (Aronson et al., 2002). In some situations (e.g., in airplanes), we may even combine freight transport capacity with the transportation of people. Moreover, almost any planning problem involving multiple companies (agents) can use the proposed method to coordinate their actions.

Future work Concerning the more technical issues, the blackboard that is used in our forward heuristic cooperative planning implements a market mechanism without any realistic payments. A real computational market, as introduced by Wellman (1998) 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.

In this paper we have omitted the details of the algorithm. However, most of these

details are implemented in C++ and soon we will be able to verify the practical applicability of this method using realistic data.

New ideas and algorithms from the AI Planning community can have a great impact on planning in a multi-agent context. Particularly *non-deterministic planning* may be very promising, since especially in a multi-agent context the effect of actions is uncertain, because other agents also execute actions. Another important extension, especially in the transport sector, is working with time, for example as proposed by Do and Kambhampati (2001). Finally, in the action-resource planning formalism, as proposed by De Weerdt et al. (2003), 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. Such new techniques may lead to a more efficient planning of public and freight transport.

Acknowledgments

I’d like to thank Hans Tonino and Roman van der Krogt for their useful comments, and the TRAIL research school for Transport, Infrastructure and Logistics for their support.

This research is carried out within the Seamless Multi-modal Mobility (SMM) research program, and is part of the Collective Agent-Based Systems (CABS) project, supervised by Cees Witteveen.

References

Aronson, L., van der Krogt, R., and Zutt, J. (2002). Incident management in transport planning. In *Proceedings of the Seventh Congress on Transport, Infrastructure and Logistics (TRAIL-02)*.

Bond, A. H. and Gasser, L., editors (1988). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA.

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

de Weerdt, M. M., Bos, A., Tonino, J., and Witteveen, C. (2003). A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence, special issue on Computational Logic on Multi-Agent Systems*, 37(1–2):93–130.

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)*, pages 109–120.

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)*, pages 121–125. AAAI Press, Menlo Park, CA. Also published in Bond and Gasser (1988), pages 205–209.

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.

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)*, pages 115–119. AAAI Press, Menlo Park, CA. Also published in Bond and Gasser (1988), pages 187–191.

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)*, pages 1031–1033. Morgan Kaufmann Publishers, San Mateo, CA. Also published in Bond and Gasser (1988), pages 216–219.

Tonino, J., Bos, A., de Weerdt, M. M., and Witteveen, C. (2002). Plan coordination by revision in collective agent-based systems. *Artificial Intelligence Journal*, 142(2):121–145.

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

Wellman, M. P. (1998). Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125.