Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Conflict Resolution in Multi-Agent Systems[1]

Stefan Resmerita[2], Michael Heymann[3]

Abstract

We consider the problem of conflict resolution among multiple agents who navigate in a discrete shared resource environment. We present distributed algorithms that yield maximal solutions (Nash Equilibria) for the conflict resolution problem in the two- and multiple- (more than two) agent cases.

Keywords: Multi-agent systems, noncooperative conflict resolution

1 Introduction

This paper is concerned with multi-agent systems where agents navigate in a shared resource environment modelled as a resource-graph. Access to resources (vertices of the graph) is mutually exclusive: distinct agents are not permitted to simultaneously occupy the same vertex. Agents enter the system at arbitrary times. Each agent has a specific initial (entry) vertex, a specific final (exit) vertex, and a specified set of candidate trajectories (timed paths) from entry to exit, called the agent's (execution) *model*. Each transition in an agent path specifies the time of residence in the preceding vertex. The transition itself is instantaneous. The number of agents in the system is not specified and may change with time. This general setup is frequently encountered as a safety and liveness specification in applications like Robot Navigation [6], Traffic Control [5], Air Traffic Management [1] and the like.

Two trajectories of different agents are in conflict if they occupy the same resource at the same time. The problem addressed in this paper is how to resolve the conflicts; that is, how to find for each agent a subset of its model, called *legal plan*, so that no two legal plans are in conflict. A simple solution to this problem is for each agent to select its subset of trajectories that are conflict-free with all trajectories of other agents. But then all conflicting trajectories remain unclaimed and unutilized. Thus, we seek *maximal* solutions, where no agent can unilaterally improve its legal plan without creating a conflict with some other agent's legal plan.

We examine the *noncooperative* situation, where agents are greedy and no inter-agent communication takes place during the conflict resolution. We shall assume that disputed resources are prioritized over competing agents, and that distinct resources may have different prioritizations. We shall assume that the prioritization is given, and that each agent knows the models of the other agents, as well as the resource prioritizations. We seek an *optimal* solution to the conflict resolution problem. That is, a Nash-equilibrium that constitutes a set of noncooperative strategies that always yield a maximal solution. Obviously, we want the resultant solutions to assign resources to agents consistently with the resource prioritization (which, in the case of non-uniform prioritization, is far from being simple).

We shall consider first the case of two-agents and then the general (more than two agents) case. We shall present in this paper optimal algorithms, as well as non-optimal but computationally efficient approaches. Complete formal analysis for these algorithms can be found in [8].

Finding conflict-free paths by restricting given sets of paths is part of the approach called *roadmap planning* in the Robotics literature [6] [7]. Research in Artificial Intelligence concerning *multi-agent planning* for agents pursuing independent goals is primarily focused on cooperative planning [2] [3]. One of the works in AI that deals with planning under assumptions similar to ours is [4].

2 Preliminaries

Agent models and conflicts. An agent model can be compactly represented by a type of timed automaton called *agent automaton*. Since each agent knows the models of the other agents, it can individually detect all the conflicts in the system, by computing the parallel composition of the involved agent automata. The formal definition of the agent automaton, as well as an example of parallel composition of two automata can be found in [9].

Let V denote the (finite) set of resources (vertices of the resource graph). Suppose that n agents, referred

[1]The work of the second author was supported in part by the Technion Fund for Promotion of Research and was completed while he was visiting NASA Ames Research Center, Moffett Field, CA 94035, under a grant with San Jose State University.

[2]Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel. E-mail: stefan@cs.technion.ac.il

[3]Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel. E-mail: heymann@cs.technion.ac.il

0-7803-7924-1/03/$17.00 ©2003 IEEE

to as agent 1, ..., agent n, want to enter the system. We denote by $\mathcal{P}_i$ the model of agent i.

A *disputed resource*, or *conflict*, between paths p_i of agent i and p_j of agent j is a pair $(\tau, q) \in \mathbb{N} \times V$ such that both agents i and j would occupy q at time τ if they followed p_i and p_j respectively and at most one of i, j would occupy q at $\tau - 1$. Thus, τ is the instant of conflict occurrence between i and j at q. In the sequel, unless explicitly stated otherwise, by *resource*, we shall mean *disputed resource*. We use $\mathcal{D}_{(\mathcal{P}_1,\ldots,\mathcal{P}_n)}$ to denote the set of all disputed resources among the n agents.

A *prioritization* of agents' access to the disputed resources is a map

$$\pi : \mathcal{D}_{(\mathcal{P}_1,\ldots,\mathcal{P}_n)} \times \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\} \longrightarrow \{1, 2, \ldots, n\},$$

given by

$$\pi(q, \mathcal{P}_i) = \begin{cases} k \in \{1, 2, \ldots, n\}, & \text{if } q \text{ is on a path in } \mathcal{P}_i; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

such that $\pi(q, \mathcal{P}_i) \neq \pi(q, \mathcal{P}_j)$ whenever they are defined and $i \neq j$. We assume that a larger number means a higher priority.

Consider, for example, two agents with models represented in Figure 1. The set of disputed resources is $\mathcal{D}_{(\mathcal{P}_R,\mathcal{P}_S)} = \{(1,j), (2,c), (3,i), (3,f), (4,d), (5,a), (5,g), (5,e), (6,b), (6,h), (7,k)\}$. The following rules have been used to prioritize the two agents, and priority is given to the agent that: (1) arrives first at the disputed resource; (2) has the shortest time to arrive at its destination from the arrival time at the disputed resource. If rule (1) does not yield a prioritization, rule (2) is applied. The priorities of the agents for the disputed resources are depicted in Figure 1 next to the corresponding vertices.

The basic conflict resolution problem is formulated as follows. Given a multi-agent system $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and a prioritization π, find for each agent a set of paths $LP_i \subseteq \mathcal{P}_i$ (a *legal plan*) such that $(LP_1, \ldots, LP_n)$ is conflict-free. That is, no two paths belonging to different legal plans have a disputed resource. A solution $(LP_1, \ldots, LP_n)$ is *less restrictive* than another solution $(LP'_1, \ldots, LP'_n)$ if $LP'_i \subseteq LP_i$ for all $i = 1, n$ and the inclusion is strict for at least one agent. A solution is *least restrictive*, or *maximal*, if no other less restrictive solution exists. While maximal solutions need not be unique, a maximal solution means that no agent can unilaterally improve its legal plan without creating a conflict with some other agent's legal plan (and thus violating the safety constraint). An algorithm that always finds a maximal solution is called *optimal*.

Resolution principles. Our approach to optimal conflict resolution is based on the following guidelines.

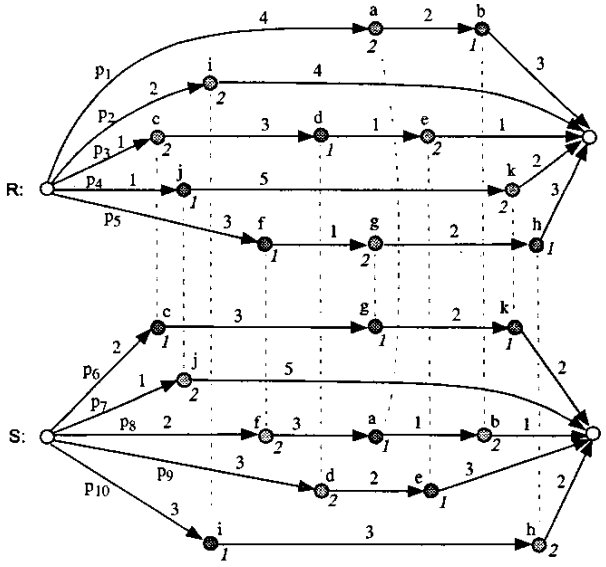

Figure 1: Agent models

An agent reserves a resource if and only if the following conditions are simultaneously met:
(*) The agent has highest priority for the resource among all agents that have legal access to it, and (**) The agent can make successful use of the resource, by completing a legal execution. This, in particular, implies that an agent is not permitted to reserve a resource (for which it may have priority) if the reservation cannot be applied toward a successful task completion.

We say that an agent *has legal access* to a resource q if it is not prevented by other agents from reaching q, even if it may be prevented from completing the execution (from q to the destination vertex). Thus, an agent does not have legal access to q only if each path p containing q also contains some other resource q'_p, preceding q, such that q' is reserved by some other agent. For example, agents R and S in Figure 1 have legal access to resource i. Since R has priority for i and it can complete a legal execution, it will reserve i. In particular, this means that h is inaccessible to S.

By the above reservation guidelines, a path is considered legal whenever all disputed resources on the path are legally accessible and the agent has priority for the last disputed resource on the path.

Theorem 1 A solution based on the above principles is conflict-free and maximal.

Proof

The solution is clearly conflict-free: different agents cannot reserve the same resource, since each agent has a unique priority for that resource.

If the solution is not maximal, then some agent has an illegal (i.e., unreserved) path which is conflict-free with the legal plans of the other agents. This means that every resource on this path is legally accessible and the agent has the highest priority for the last disputed resource on the path (among all the agents for which that resource is legally accessible). Hence, the path must be legal (i.e., reserved) a contradiction. ∎

In the sequel, we shall present algorithms which implement the above principles, first for the two-agent case, and then for the general case of more than two agents.

3 The Case of Two Agents

Consider two agents, denoted by R (with model $\mathcal{P}_R$) and S (with model $\mathcal{P}_S$), and a prioritization π of all disputed resources in $\mathcal{D}_{(\mathcal{P}_R,\mathcal{P}_S)}$ over R and S.

Outline of the procedure. The resolution algorithm, henceforth referred to as *DOR2*, works on two sets of "unresolved" paths, initialized with the paths of the models of the two agents. We shall denote these by MP ("my paths") and OP ("opponent's paths"). At each iteration, paths which are determined as legal are moved from the unresolved set to a "legal" set. Legal sets are denoted by MLP ("my legal paths"), and OLP ("opponent's legal paths"). Paths which are determined as illegal are eliminated from the unresolved sets. The algorithm stops when the unresolved sets are empty, i.e., when each path in the initial models is marked as either legal or illegal.

At each iteration, the algorithm determines a set of resources which are legally accessible in MP. A path p in MP is then considered as legal if all disputed resources on it are legally accessible and MP has priority for the last disputed resource on p. In this case, all the paths in OP having disputed resources with p are illegal, and therefore they are eliminated from OP. Such paths may contain resources that are disputed with other paths (beside p) in MP. These resources become now undisputed, and this may give legal access to resources in MP that were previously inaccessible. Hence, in the next iteration, new legal paths may be found. Similar operations are executed by reversing MP with OP (at each iteration). Thus, the algorithm will yield the agent's legal plan as well as the opponent's legal plan.

Example. Let us illustrate the algorithm by resolving the conflicts in the example of Figure 1. Consider the viewpoint of agent R. Initially, MP is R's model and OP is S's model (as depicted in the figure). In this example, for simplicity, we shall omit the time component from a disputed resource (e.g., we shall use a instead of $(5, a)$). At the beginning of the first iteration, the set of resources that are legally accessible in MP is $\{a, b, i, c, d, j, f\}$. Of these, R has priority only for a, c, and i. Since a legal execution can be completed from i, it follows that p_2 is legal. Consequently p_{10} is illegal and therefore resource h is available in MP. The path p_2 is moved from MP to MLP and p_{10} is removed from OP. Each of the remaining paths in MP (i.e., p_1, p_3, p_4, p_5) contains a disputed resource for which R does not have priority. Therefore, we turn now to check legality of paths in OP. Since S has priority for j, and it can complete a legal execution from j, it follows that p_7 is legal. Consequently, p_4 is illegal which means that resource k is available in OP. Now each path in OP has a resource for which S does not have priority. The algorithm proceeds to check legal accessibility of resources in MP and then in OP. Since a can be claimed in MP, it follows that b is inaccessible in OP, which means that b is available in MP. Hence, a and b can be reserved in MP. Resource c can also be claimed in MP (it is legally accessible and R has priority for it), causing g to be inaccessible in OP and therefore to be available in MP. However, in contrast to the case of b, g is inaccessible in MP at this step (due to f). No new legally accessible resource can be determined now in MP, hence we turn to OP. Since e is inaccessible in MP, it will be reserved in OP (together with d). The sets MP and OP at the beginning of the second iteration are given in Figure 2 (where the prioritization of inaccessible resources is zero).

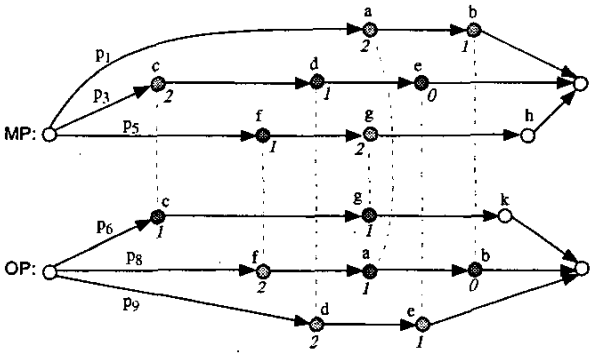

Figure 2: After the first iteration

The second iteration begins with moving p_1 from MP to MLP, and deleting p_8 from OP. Consequently, f becomes undisputed in MP, which makes g legally accessible. Therefore, p_5 is legal, which implies that p_6 is illegal. Now MP contains only p_3 and OP only p_9. Clearly, p_9 is legal and hence p_3 is illegal. The algorithms stops. The maximal solution obtained by agent R is depicted in Figure 3. Agent S executes the same algorithm (this time MP is initialized with the model of S and OP with the model of R) and obtains the same solution (where $MLP_S = OLP_R$ and $OLP_S = MLP_R$).

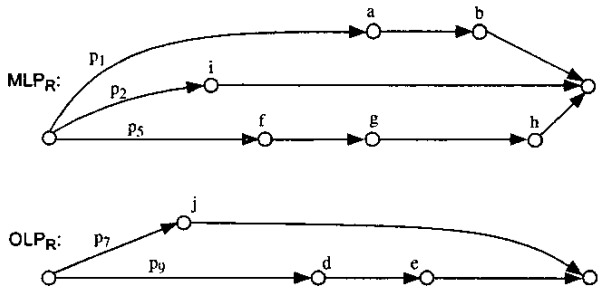

Figure 3: The solution obtained by R

Monotonicity. An interesting property of *DOR2* is monotonicity with respect to the input models, which guarantees that the algorithm can be safely used when the knowledge of the opponent's model is not exact, but is *conservative*, as follows. Let $\mathcal{P}_R$ and $\mathcal{P}_S$ be the real models of agents R and S, respectively. Let $\mathcal{P}_{S^\bullet} \supseteq \mathcal{P}_S$ be the model that R knows about S and $\mathcal{P}_{R^\bullet} \supseteq \mathcal{P}_R$ be the model that S knows about R. Agent R is given a prioritization π^*_R defined over $\mathcal{D}_{(\mathcal{P}_R,\mathcal{P}_{S^\bullet})}$. Agent S is given a prioritization π^*_S defined over $\mathcal{D}_{(\mathcal{P}_S,\mathcal{P}_{R^\bullet})}$. We assume that the restrictions of π^*_R and π^*_S to $\mathcal{D}_{(\mathcal{P}_R,\mathcal{P}_S)}$ are identical. Let π denote this restriction. Let $(MLP^*_R, OLP^*_R) = DOR2(\mathcal{P}_R, \mathcal{P}_{S^\bullet}, \pi^*_R)$ and $(MLP^*_S, OLP^*_S) = DOR2(\mathcal{P}_S, \mathcal{P}_{R^\bullet}, \pi^*_S)$.

Theorem 2 Under the above assumptions, the pair (MLP^*_R, MLP^*_S) is conflict-free.

4 The General Case

In the multiple (more than two) agent scenario, we consider two extreme situations: (1) The most that an agent can do; that is, to take into consideration all the agents and conflicts in the system, including those in which it is not directly involved (and assuming it knows the prioritization for all of them). For this case, we present an optimal resolution algorithm. (2) The least that an agent must do; that is, to take into consideration only the agents with which it has conflicts, and to ignore all the others. Conflicts are resolved with each of the competing agents, pairwise, the final result being the intersection of the partial results. Since the number of agents conflicting with a given one is usually much lower than the number of all agents in the system, the pairwise approach is computationally more efficient, but the result is, in general, not maximal.

In the sequel, we shall use the following example to illustrate the two situations. Consider the agents with models and prioritization given in Figure 4 (transition times are omitted for simplicity).

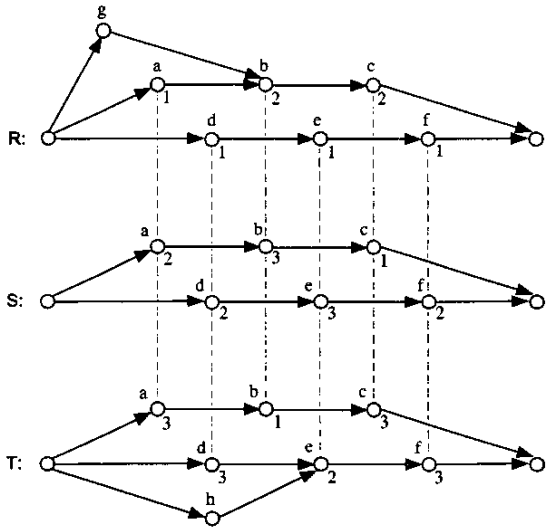

Figure 4: Three agents

(1) All-agent resolution. When taking all the agents in the system into account, one must deal with the complex interactions among the agent models, due to the (generally non-uniform) prioritization of agents over disputed resources. In our example, we present two possible reasonings. Since resource d is available to T, it will be blocked in R and S. Now e is inaccessible to S, which means that it is legally accessible to T. Hence, d, e, and f are deployed by T, together with h (which is not disputed) leading to the modified triple in Figure 5. Then, a can be tentatively claimed in model T', thereby being blocked in S' and R'. R' can go to g, and it has priority over T' for resource b (to which S' has no access). Hence, R' can block T' at b if T' acquires a. And now consider two possible

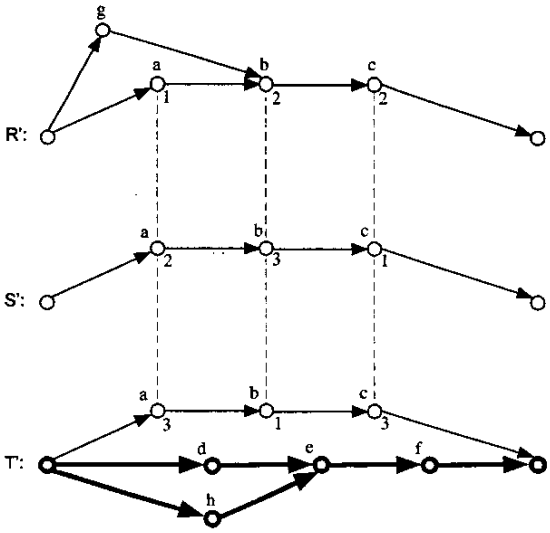

Figure 5: The marked resources are already reserved

continuations:

1. Since the acquisition of a by T' does not enable T' to successfully complete its task, T' cannot reserve a. Next in line for resource a is S' which can go to a (knowing that T' cannot) - and S' also has priority over R for b. Consequently, S' gets a legal path and R' is completely blocked. The solution is given in Figure 6. Notice that the solution is maximal. However this argument does

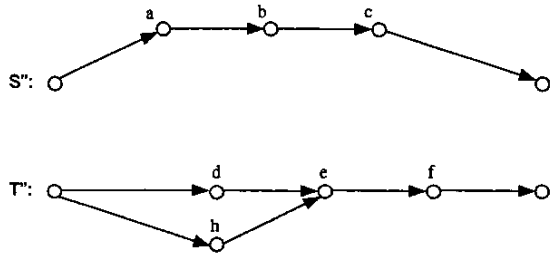

Figure 6: Solution 1

not fully comply with the guiding principles outlined in Section 2 in two respects: (i) T' should not give up resource a just because b is legally accessible to R'. Rather, it should do that only if (and when) R' (or S') can *reserve* b. (ii) The reasoning on behalf of R' has not been completed. By completing it, we obtain another solution below.

2. Since b is legally accessible to R', and R' has priority for b (due to T' blocking S' at a), it follows that c is also legally accessible to R'. Moreover, T' is blocked at b, and therefore R' has priority for c. Hence, the path containing g, b, and c is legal. This means that a cannot be used by S' or T' and consequently the path containing a becomes also legal for R'. The maximal solution is shown in Figure 7.

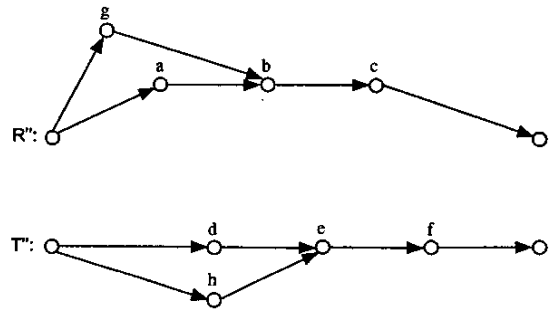

Figure 7: Solution 2

The conflict resolution algorithm for multiple agents, henceforth referred to as *DOR*, implements the principles presented in Section 2, by executing the same operations as *DOR2* on two aggregate models, as follows.

The *joint model* of agents $i_1, \ldots, i_k$ is $\mathcal{JP}(\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k}) = \bigcup_{j=1,k} \mathcal{P}_{i_j}$, where $i_j \in \{1, \ldots, n\}$, and $k \leq n$. We say that q is a *disputed resource between two joint models* $\mathcal{JP}_R = \mathcal{JP}(\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k})$ and $\mathcal{JP}_S = \mathcal{JP}(\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_m})$ if q is a disputed resource between two agent models $\mathcal{P}_{i_h} \subseteq \mathcal{JP}_R$ and $\mathcal{P}_{j_l} \subseteq \mathcal{JP}_S$ for some $h \in \{1, \ldots, k\}$ and $l \in \{1, \ldots, m\}$ with $i_h \neq j_l$. Of special interest is the case when $\mathcal{JP}_R$ and $\mathcal{JP}_S$ are identical and contain the models of all agents: $\mathcal{JP}_R = \mathcal{JP}_S := \mathcal{JP}(\mathcal{P}_1, \ldots, \mathcal{P}_n)$. One can see that the set of all disputed resources between $\mathcal{JP}(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ and itself is the same as the set of all disputed resources among the agent models $\mathcal{P}_1, \ldots, \mathcal{P}_n$: $\mathcal{D}_{(\mathcal{JP}_R, \mathcal{JP}_S)} = \mathcal{D}_{(\mathcal{P}_1, \ldots, \mathcal{P}_n)}$. The algorithm *DOR* works on two sets of paths $MP \subseteq \mathcal{JP}_R$ and $OP \subseteq \mathcal{JP}_S$, initialized with $\mathcal{JP}_R$, respectively $\mathcal{JP}_S$. Notice that MP and OP can be in conflict only at paths belonging to different agents. The operations on MP, OP are the same as those executed by *DOR2*. For instance, the two identical joint models corresponding to the agent models depicted in Figure 4 are shown in Figure 8.

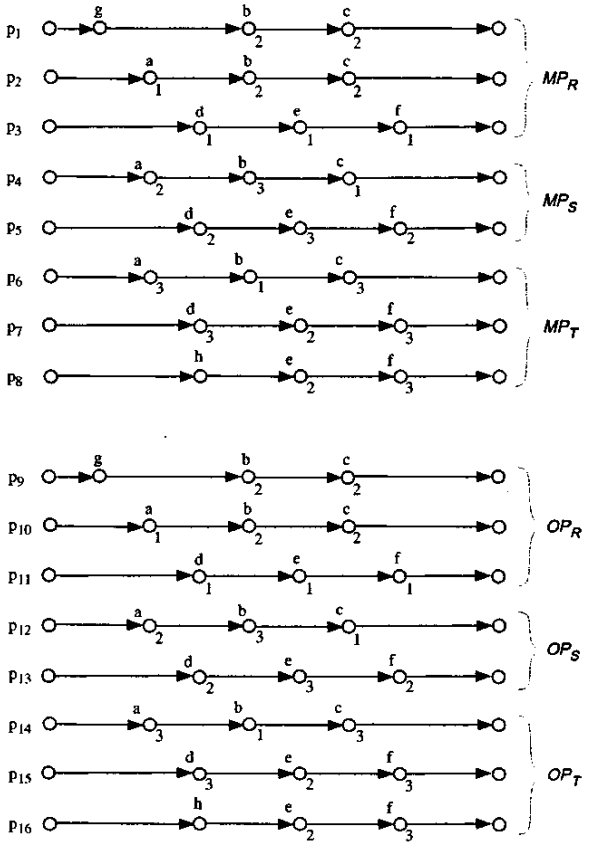

Figure 8: Joint models

Proposition 1 Under the assumptions of perfect knowledge of models and prioritization, if each agent runs *DOR*, then it obtains the correct legal plans of all agents.

Theorem 3 The algorithm *DOR* is optimal.

One should emphasize that, while each agent executes the same algorithm on the same input information, the exact executions of distinct agents may be different. This is because the property of legal accessibility is independent of the order in which resources are checked by the algorithm. This makes *DOR* suitable for being utilized in a distributed setup.

(2) Pairwise resolution In general, an agent may not have full knowledge of all conflicts and prioritizations. However, each agent must know at least the models of and prioritization with the other agents with which it has direct conflicts. Even when an agent has complete knowledge, executing *DOR* may be computationally prohibitive. An alternative to all-agent resolution is pairwise resolution, where an agent executes *DOR2* sequentially against each other agent with which it has conflicts. The legal plan obtained from the pairwise resolution is the intersection of the legal plans of the individual (pairwise) executions.

Let us see how the pairwise approach applies to the agents in Figure 4. R versus T: since R has access to b, it will block b at T, therefore R can reserve b and c. Consequently, the uppermost path of T is illegal and R can also reserve a. T will reserve d, e, and f. The solution is the same as the one in Figure 7. R versus S: R obtains nothing and S gets both paths (because S has priority for a, b, and d over R). The legal plan of R is then the empty set. S versus T: S gets nothing and T gets everything. Therefore, the legal plan of S is empty and the legal plan of T is given by T'' in Figure 7. Notice that this solution is not maximal.

Another pairwise approach is to execute *DOR2* successively against each other model by starting with the own model and using at each execution the result of the previous execution (as opposed to the case above, where the own model is used at each iteration). This is computationally more efficient, since the own intermediate legal model is smaller and smaller at each new execution. One can show, by using the monotonicity of *DOR2* (Theorem 2), that the solution obtained in this case is always at least as restrictive as the one given by the first pairwise approach.

Remarks:

(1) Clearly, *DOR* works also for the case of two agents only, where it is not identical to *DOR2*, which is tailored for that case (and executes less operations than *DOR*). In contrast with *DOR2*, the algorithm *DOR* is in general not monotonic with respect to input models. In fact, *DOR* may yield an unsafe output when the knowledge of other agents' models is conservative. On the other hand, the pairwise algorithms are clearly monotonic and therefore they can be safely used in such circumstances.

(2) Both algorithms work work correctly when the agent models are replaced by simplified agent models, in which undisputed resources are not presented. Since the disputed resources are in general significantly fewer than the total number of employed resources, the computational complexity is considerably reduced.

(3) Dealing first with early conflicts and then with late conflicts makes our approach suitable to the case of limited lookahead, where an agent may consider only partial models of other agents, with a certain time depth.

References

[1] S. Devasia, M. Heymann and G. Meyer, 2002. Automation procedures for Air Traffic Management: a token-based approach. Proceedings of the American Control Conference.

[2] M. E. desJardins, E. H. Durfee, C. L. Ortiz, Jr., and M. J. Wolverton, 1999. A Survey of Research in Distributed, Continual Planning. *AI Magazine 20(4)*, pp. 13-22.

[3] E. Ephrati and J. S. Rosenschein, 1997. A Heuristic Technique for Multiagent Planning. *Annals of Mathematics and Artificial Intelligence, 20*, pp. 13-67.

[4] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein, 1988. Cooperation without communication. In A.H. Bond and L. Gasser, Eds., *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, pp. 220-226.

[5] J. K. Kuchar and L. C. Yang, 2000. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems 1(4)*, pp. 179-189.

[6] J. C. Latombe, 1991. *Robot Motion Planning.* Kluwer Academic Publishers, Boston.

[7] S. M. LaValle and S. A. Hutchinson, 1998. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation, 14(6)*, pp. 912-924.

[8] S. Resmerita, 2003. A multi-agent approach to control of multi-robotic systems. PhD thesis, Department of Computer Science, Technion - Israel Institute of Technology.

[9] S. Resmerita, M. Heymann and G. Meyer, 2003. A framework for conflict resolution in Air Traffic Management. CDC 2003.