AN AGENT BASED FRAMEWORK FOR CONTROL OF MERGING AIR-TRAFFIC

Michael Heymann [*,1] **George Meyer** [**]
Stefan Resmerita [*]

[*] *Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel. E-mail: heymann@cs.technion.ac.il*
[**] *NASA Ames Research Center, Moffett Field, CA 94035. E-mail: George.Meyer-1@nasa.gov*

Abstract: The paper presents an agent based framework for control of conflict-free scheduling and spacing of airport arrival and approach traffic. The proposed tool incorporates a multi-agent conflict-resolution algorithm that guarantees to arriving aircraft safe approach trajectories, and a dynamic prioritization mechanism that aids in efficient utilization of the available airspace and arrivals runway capacity. The approach can be employed by a central authority or autonomously by the arriving aircraft.

Keywords: Air traffic control, Agents, Resource allocation.

1. INTRODUCTION

A recent trend in Air Traffic Management (ATM) research has been the development of various decision support tools for monitoring and routing en route air traffic as well as tools for sequencing and scheduling of traffic in the terminal area so as to increase overall capacity and efficiency of operation.

In the present paper we describe a new approach to the airport arrival-sequencing, scheduling, spacing and conflict resolution problems. The approach is based on a multi-agent model for air traffic management that was recently introduced in (Resmerita *et al.*, 2003), in which central elements are the multi-agent conflict detection and resolution algorithms introduced in (Resmerita and Heymann, 2003), (Resmerita, 2003). In that model, aircraft are viewed as agents and the airspace is partitioned into cells. The required separation between aircraft is satisfied by guaranteeing that at most one agent occupies a cell (henceforth called a *resource*) at any time. Thus, the resource system is modelled as an undirected graph, where a vertex corresponds to a cell and an edge represents an adjacency relation between two cells. An agent travels from an initial vertex to a destination vertex (both of which are specific to each agent). An agent trajectory is a timed directed path in the resource graph, starting at the initial vertex and ending at the destination vertex. Each edge (transition) in the path is labelled by the time of residence in the preceding vertex (representing the time of cell traversal). The number of agents in the system is not fixed and generally changes with time. An agent can enter the system at any arbitrary (integer) instant of time and exits the system upon arrival at its destination.

[1] The work of the first 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.

Each agent announces, upon entry, the finite set of all its optimal paths (by some criteria), called the agent's *model*. To satisfy safety (i.e., legal separation), an agent's movements are restricted to a *legal* subset of its model, called the agent's *legal plan*. Paths from legal plans of distinct agents are conflict-free. Each agent is then allowed to follow an arbitrary path in its legal plan.

At the heart of the framework proposed in (Resmerita *et al.*, 2003) is the mechanism by which agents select their legal plans (Resmerita and Heymann, 2003). The proposed methodology consists of two algorithmic phases, preceded by an initialization phase. First, each incoming agent determines the subset of its optimal paths that are conflict-free with the legal paths of all active agents (that are already in the system). Then, in the *conflict resolution* phase, the agent resolves potential conflicts with paths of all other incoming agents. The conflict resolution algorithm finds a *maximal* solution; that is, a set of legal plans for the incoming agents that satisfy the condition that none of these legal plans can be improved unilaterally (without creating conflicts with the legal plan(s) of other agents). Finally, in the *accommodation* phase, agents whose legal plan remained empty after the conflict resolution phase obtain resources (paths) from active agents if possible. (The accommodation algorithm is not relevant to the present paper.)

The conflict resolution problem is solved as a *non-cooperative* game, with greedy agents that compete for the conflicting resources. It is assumed that disputed resources are prioritized over the competing agents. The conflict resolution algorithm yields a Nash-equilibrium; that is, a set of agent strategies that always yield a maximal solution.

In adapting the agent approach to the arrival air-traffic control problem, two specific features of the arrival airspace structure play an important role. First, all aircraft are designated for arrival at a single final vertex. Secondly, the arrival airspace consists of a fixed geometric structure that includes all available control maneuvers that can be employed towards efficient and safe (i.e., conflict-free) scheduling, sequencing and spacing of the arriving aircraft. The operational airspace model, therefore, can be represented as a fixed resource graph that includes a specified set of entrance vertices (where traffic enters the controlled airspace under consideration) and a single target vertex. To compute the legal plans for all entering agents, the conflict resolution algorithm is then employed. To achieve efficient utilization of the arrival airspace and the available landing capacity, in addition to fair scheduling of all arriving traffic, a dynamic resource prioritization mechanism is employed, that constitutes the primary traffic-management control tool.

A recent review of the literature on conflict resolution in ATM can be found in (Kuchar, 2000). Collision avoidance strategies have been studied in (Tomlin *et al.*, 1996) (Menon, 1999). A token-based framework was described in (Devasia *et al.*, 2002). More recent work on scheduling and flow regulation of airport arrival traffic can be found in (Bayen *et al.*, 2003), (Bayen and Tomlin, 2003).

2. AGENT MODELS AND CONFLICT DETECTION

Agent models are represented as timed automata (with discrete time) whose vertices and edges are derived from the underlying resource graph. For example, consider two agents, hereby denoted by R and S, with models in Figure 1. The different paths in the two agents' models are denoted $p_1, ...p_{10}$.

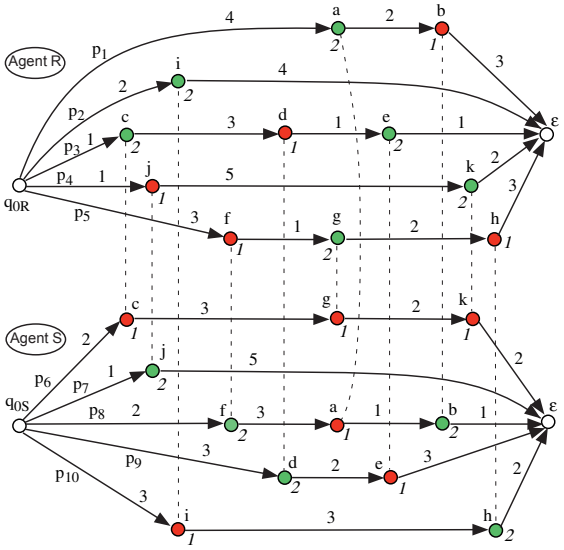

Fig. 1. Agent models

If agent R follows the path p_1, it stays 4 units of time at its initial resource q_{0R} before moving to a. It stays 2 units of time at a and then moves to b at which it resides 3 units of time before leaving the system. The label ϵ denotes task (path) termination (ϵ is not actually a vertex of the resource graph). The numbers below the vertices represent priorities, to be explained later.

Let V denote the set of vertices of the resource graph. 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. For example, in Figure 1, $(2, c)$ is a disputed resource between the path p_6 of S and the path p_3 of R.

Conflicts are detected by computing the parallel composition of the involved automata. For a detailed discussion of efficient computation of the conflict detection see (Resmerita, 2003).

For the airport-arrival problem, the arrival geometry is fixed and so is the associated resource graph. To illustrate the creation of a resource graph for a representative case, consider Figure 2 that shows the typical arrival (and departure) traffic pattern at San Francisco airport as recorded on the air-traffic-controllers' screen. The coordinates are in nautical miles.

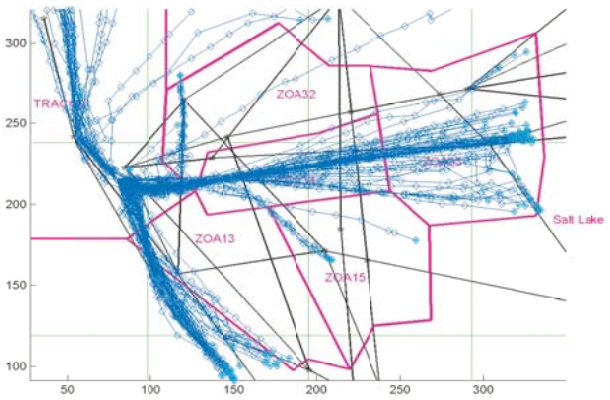

Fig. 2. San Francisco Airport Arrivals Airspace

The corresponding resource graph is depicted in Figure 3. Notice the explicit representations of certain delay maneuvers and holding patterns that can be followed by the aircraft in order to achieve various alternate arrival times (delays) that can be employed towards conflict resolution, sequencing and spacing. In practice, more extensive maneuvers are generally available and these must all be accounted for in the resource graph so as to obtain efficient airspace management.

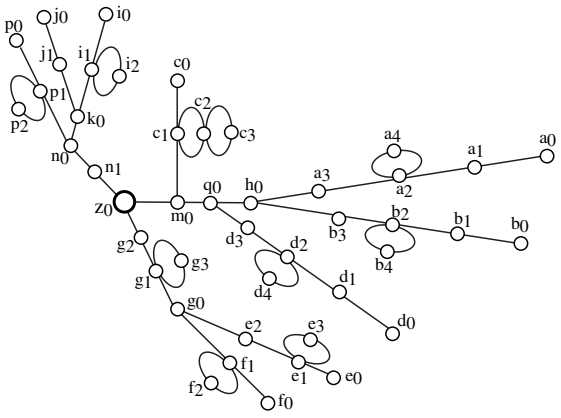

Fig. 3. San Francisco arrivals resource graph

Each aircraft enters the system at one of the entry ports (vertices in the graph) and its available arrival maneuvers are completely specified by its entrance port. It follows that the agent model is completely determined by the entrance port and entrance time. To illustrate, Figure 4 shows three agent models for aircraft entering the San Francisco airspace (of Figure 3). Here the labels, e.g., $3a0$, etc., list for each vertex the time of entry followed by its name. Notice that each of the agent models displays the direct path and also delayed paths. Obviously, more delayed paths could be displayed for each agent, if so desired. Disputed resources are circled. The numbers above them will be explained shortly.

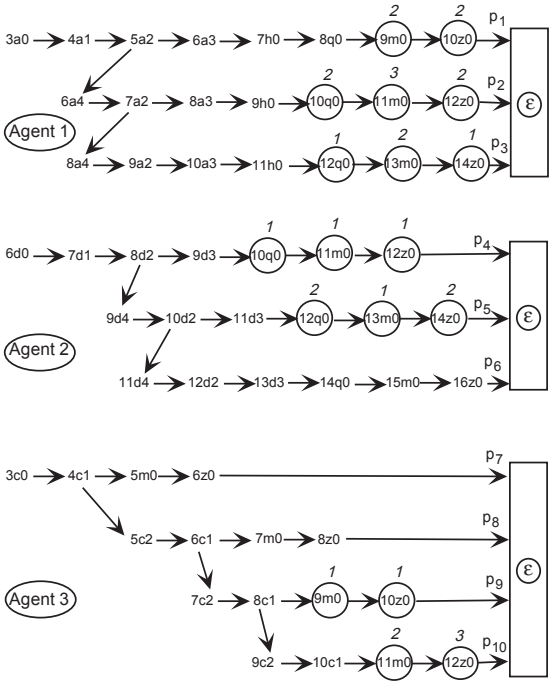

Fig. 4. Agent models for San Francisco Arrivals

3. CONFLICT RESOLUTION

Suppose that n agents, referred to as agent $1, \ldots,$ agent n, want to enter the system. We denote by $\mathcal{P}_i$ the model of agent i. Let $\mathcal{D}$ denote the set of all disputed resources among the n agents. In the sequel, by *resource*, we shall frequently mean *disputed resource*.

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

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

given by

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

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.

The basic conflict resolution problem can be 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 distinct 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*.

Our approach to optimal conflict resolution is based on the following guidelines. An agent can acquire 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 acquire a resource (for which it may have priority) if the acquisition cannot be applied towards a successful task completion.

One can prove (Resmerita and Heymann, 2003) that a solution based on the above principles is conflict-free and maximal.

3.1 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 over R and S.

Outline of the procedure. The resolution algorithm for the two-agent case, henceforth referred to as *DOR2*, works on two sets of "unresolved" paths, initialized with the paths of the models of the two agents. These will be denoted 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.

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'_p is acquired by some other agent. For example, both 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 acquire i. In particular, this means that h is inaccessible to S.

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. All resources are disputed. The priorities for the resources are as listed in the figure. 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 acquired 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 acquired in OP (together

with d). At the beginning of the second iteration $MP = \{p_1, p_3, p_5\}$ and $OP = \{p_6, p_8, p_9\}$. 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 algorithm stops. The maximal solution obtained by agent R is $MLP = \{p_1, p_2, p_5\}$ and $OLP = \{p_7, p_9\}$. 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 $OLP_S = MLP_R$).

3.2 The General Case

In the multiple (more than two) agent scenario, there are two extreme situations: (1) The optimal resolution, where all agents in the system are considered simultaneously, but where the computational complexity is maximal as well. (2) The simplest resolution methodology where conflicts are resolved pairwise, between all competing agents, employing algorithm *DOR2*, the final result being the intersection of the partial results. While the pairwise approach is computationally more efficient, the result is, in general, not maximal.

The optimal resolution algorithm is presented in (Resmerita and Heymann, 2003), (Resmerita, 2003). The next example will be used to illustrate the pairwise resolution. Consider the agents with models and prioritization given in Figure 5 (transition times are omitted for simplicity). The algorithm *DOR2* is employed as follows. R versus T: since R has access to b, it will block b at T, therefore R can acquire b and c. Consequently, the uppermost path of T is illegal and R can also acquire a. T will acquire d, e, and f. The solution is: $MLP_R^1 = \{p_1, p_2\}$, $MLP_T^1 = \{p_7, p_8\}$. R versus S: $MLP_R^2 = \emptyset$ and $MLP_S^2 = \{p_4, p_5\}$ (because S has priority for a, b, and d over R). S versus T: $MLP_S^3 = \emptyset$ and $MLP_T^3 = \{p_6, p_7, p_8\}$. The outcome is $MLP_R = MLP_R^1 \cap MLP_R^2 = \emptyset$, $MLP_S = \emptyset$, and $MLP_T = \{p_7, p_8\}$, which is not a maximal solution.

4. THE RESOURCE PRIORITIZATION CONTROL TOOL

The employment of the conflict resolution algorithm for the control of inbound air traffic, insures that the legal plans for the entering agents will be conflict free, for any resource prioritization. Thus, agents that are allowed to enter the controlled airspace, that is, agents that are assigned a nonempty legal plan, will be sequenced and scheduled for orderly and safe terminal arrival. However, using the conflict resolution algorithm

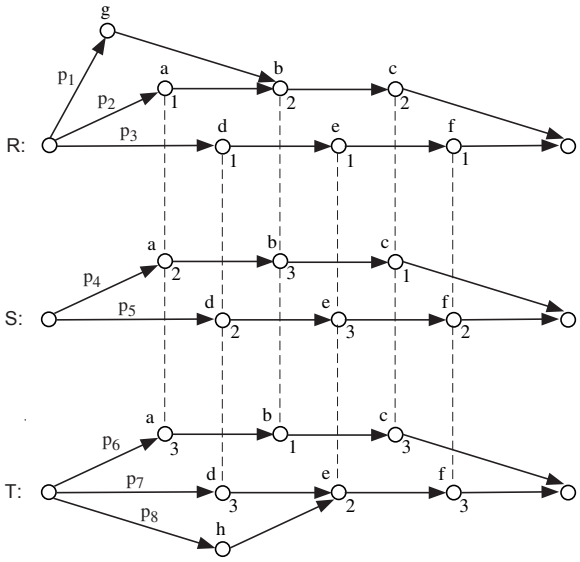

Fig. 5. Three agents

does not, in itself, insure efficient utilization of the airspace, of the available airport arrival capacity, and delay-free agent arrivals. To achieve efficient operation, a dynamic resource prioritization mechanism is employed as delineated below.

Each entering agent has a nominal terminal arrival time which is the time of arrival if the agent follows its nominal direct flight-path (without any controllers' interference). All other flight paths available to an entering agent in its agent-model have delayed arrivals. Thus, each non-direct flight path includes certain delay maneuvers so that there are alternate paths that achieve, say, $1, 2, 3, ...$ units of delay, respectively. The total accumulated delay for a set of entering agents, is the sum of all their individual arrival delays (relative to their nominal arrival times). Clearly one objective of an efficient operation of the arrival system is to minimize the total delay.

However, there may be situations where the total delay will be minimized at the expense of the individual delay of certain specific agents. Thus, an additional (or alternate) requirement of optimal operation is to minimize the maximal agent arrival delay. In our agent approach to the traffic management problem we wish to develop a mechanism that has some of the advantages of both approaches. Specifically, we wish to reduce as much as possible the total arrival delay while insuring that individual agents are not unduly penalized in achieving this goal.

Since our approach is based on a dynamic priority assignment mechanism which is at present based on heuristics, we cannot as yet insure that it will achieve provable optimality. This begs the question as to whether there exist priority assignments for which an arrival planning methodology based on the conflict resolution mechanism, will achieve

optimal safe arrival scheduling and planning. The following elementary but important theorem supports the basic validity of our proposed approach.

Theorem 1. Suppose there are k agents in the arrivals airspace. Let $\mathcal{P} = (p_1, ..., p_k)$ be any conflict-free path assignment to the k agents. Then there exists a resource prioritization for which the solution of the conflict resolution algorithm will include for each agent i, $i = 1, ..., k$, the path p_i in its legal-plan. ■

Proof. For each i let the resources along path p_i be given highest priority to agent i. This can be done since the paths are conflict-free. Then, a conflict resolution algorithm based on the principles in Section 3 will assign p_i to agent i. ■

To achieve our stated objectives, the proposed prioritization mechanism is designed to have the following properties.

1. If an available airport arrival resource (landing time slot) is prioritized to a given agent, then all resources of at least one path to this resource are prioritized to this agent, whenever possible.
2. Each arrival resource, along with exactly one fully prioritized path is removed from the dynamic prioritization and resolution system if such a path to the arrival resource exists.
3. Delayed aircraft receive "arrival prioritization enhancement" that increases monotonically with increasing delay.

Property 1) insures that the agent that was prioritized for arrival at a specific time, will be allocated a conflict-free path to realize this arrival. Property 2) is geared for the implementation of recursive control laws aimed at (cummulative) conflict-free arrival-delay minimization. Property 3) is aimed at insuring that individual agents be treated fairly without undue arrival-delay.

5. EXAMPLE OF ARRIVALS RESOLUTION

Consider the case of the three agents shown in Figure 4, where an assignment of priorities is given by the numbers above the disputed resources. In this example, the assignment is quite arbitrary and is only aimed at demonstrating the procedure and not at optimizing the outcome. The results of the pairwise resolution are shown in the table below.

Agent pairs	Agent 1	Agent 2	Agent 3
$\{1,2\}$	$\{p_1, p_2\}$	$\{p_5, p_6\}$	
$\{1,3\}$	$\{p_1, p_2, p_3\}$		$\{p_7, p_8\}$
$\{2,3\}$		$\{p_5, p_6\}$	$\{p_7, p_8, p_9, p_{10}\}$

The final results are obtained as the intersections of the individual agents' pairwise resolutions as $LP_1 = \{p_1, p_2\} \bigcap \{p_1, p_2, p_3\} = \{p_1, p_2\}$, $LP_2 = \{p_5, p_6\}$, $LP_3 = \{p_7, p_8\}$. It is noteworthy that in this case, although the resolution was performed by the generally non-optimal pairwise algorithm, the obtained solution is maximal. It is further noteworthy that the solution does not minimize the total delay. In particular, agent 2 did not get its nominal trajectory with arrival time 12, an arrival time that was allocated to agent 1. Had agent 1 been given only its nominal trajectory 1 (with arrival time 10), agent 2 could have been given its nominal trajectory without creating a conflict.

In conclusion, we have seen that a particular assignment π of priorities to the conflicting resources resulted in particular conflict-free plans for all agents. We are currently exploring ways to recursively construct various control laws π for implementing a variety of desired overall behaviors of the air traffic.

REFERENCES

Bayen A., P. Grieder and C. Tomlin (2003). Lagrangian delay predictive model for sector based air traffic flow, *AIAA Journal on Guidance, Control and Dynamics*, to appear.

Bayen A. and C. Tomlin (2003). Real time discrete control law synthesis for hybrid systems using MILP: application to congested airspace. Proceedings of the 2003 American Control Conference, Denver, Colorado.

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

Kuchar J. K. and L. C. Yang (2000). A review of conflict detection and resolution modeling methods. *IEEE Trans. on Intel. Transp. Syst.* 1(4), 179-189.

Menon P. K., G. D. Sweriduk, and B. Sridhar (1999). Optimal strategies for free-flight air traffic conflict resolution. *Journal of Guidance, Control, and Dynamics* 22(2), 202-211.

Resmerita S., M. Heymann, and G. Meyer (2003). A multi-agent approach to air traffic management. Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii.

Resmerita S. and M. Heymann (2003). Conflict resolution in multi-agent systems. Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii.

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

Tomlin C., G. Pappas, J. Lygeros, D. Godbole, and S. Sastry (1996). A Next Generation Architecture for Air Traffic Management Systems. *Lecture Notes in Computer Science* 1271, Springer Verlag.