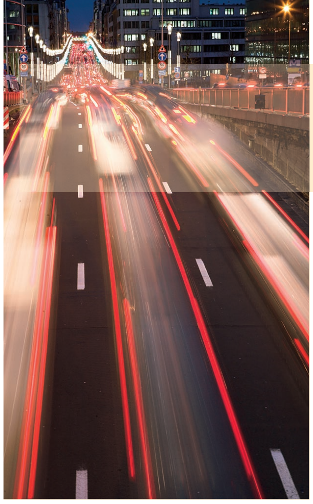
IEEE Intelligent Systems

www.computer.org/intelligent

Improving Air Traffic Management with a Learning Multiagent System

Kagan Tumer and Adrian Agogino

Vol. 24, No. 1
January/February 2009



Editor: **Daniel Zeng,** University of Arizona and Chinese Academy of Science, zeng@email.arizona.edu

Improving Air Traffic Management with a Learning Multiagent System

Kagan Tumer, *Oregon State University*
Adrian Agogino, *University of California, Santa Cruz*

A fundamental challenge facing the aerospace industry is efficient, safe, and reliable air traffic management (ATM). On a typical day, more than 40,000 commercial flights operate in US airspace, and the number of flights is increasing rapidly.[1] Current ATM systems rely on a centralized, hierarchical process, where decisions are based on flow projections ranging from one to six hours. Consequently, the system responds slowly to developing weather or airport conditions, potentially causing minor local delays to cascade into large regional congestion. The US Federal Aviation Administration (FAA) estimates that weather, routing decisions, and airport conditions caused 437,667 delays in 2007, accounting for 1,682,700 hours of delays.[2] According to a US Joint Economic Committee study, these delays cost more than US$41 billion.[3]

In addition, unlike other flow problems for which improved hardware has somewhat absorbed the increasing traffic (for example, more servers with larger memories and faster CPUs for Internet routing), the air traffic domain must find mainly algorithmic solutions because the infrastructure (for example, the number of airports) won't change enough to significantly alleviate the flow problem. So, there's a strong need to explore new, distributed, and adaptive solutions to ATM.

A learning multiagent approach in which intelligent agents either make decisions or make recommendations to human air traffic controllers is an ideal fit to this naturally distributed problem. In this article, we address three critical design decisions:

- What constitutes an agent in this system?
- What actions can those agents take to impact air traffic?
- How do those agents' actions get shaped—that is, what is their reward?

Results from a system of coordinated agents show that this approach can significantly improve ATM within current flow management procedures, without major policy shifts.

ATM

The continental US airspace comprises 20 regional centers (handling 200 to 300 flights in a busy hour) and 830 sectors (handling 10 to 40 flights). Flow control must address the integration of policies across these sectors and centers, account for the system's complexity (for example, more than 5,200 public-use airports and 16,000 air traffic controllers), and handle policy changes due to weather patterns. In addition, it must consider high-level criteria such as efficiency (for example, reducing delays), fairness (for example, dealing with different airlines), adaptability (for example, responding to developing weather patterns), reliability (for example, providing accurate predictions), and safety (for example, managing airports).

To address these issues, ATM occurs on four levels:

1. separation assurance (a 2- to 30-minute time horizon for decisions),
2. regional flow (20 minutes to 2 hours),
3. national flow (1 to 8 hours), and
4. dynamic airspace configuration (6 hours to 1 year).

Our multiagent approach focuses on regional and national flow, where agents look at time horizons between 20 minutes and 8 hours. The solution therefore isn't directly affected by separation-assurance guidelines and political

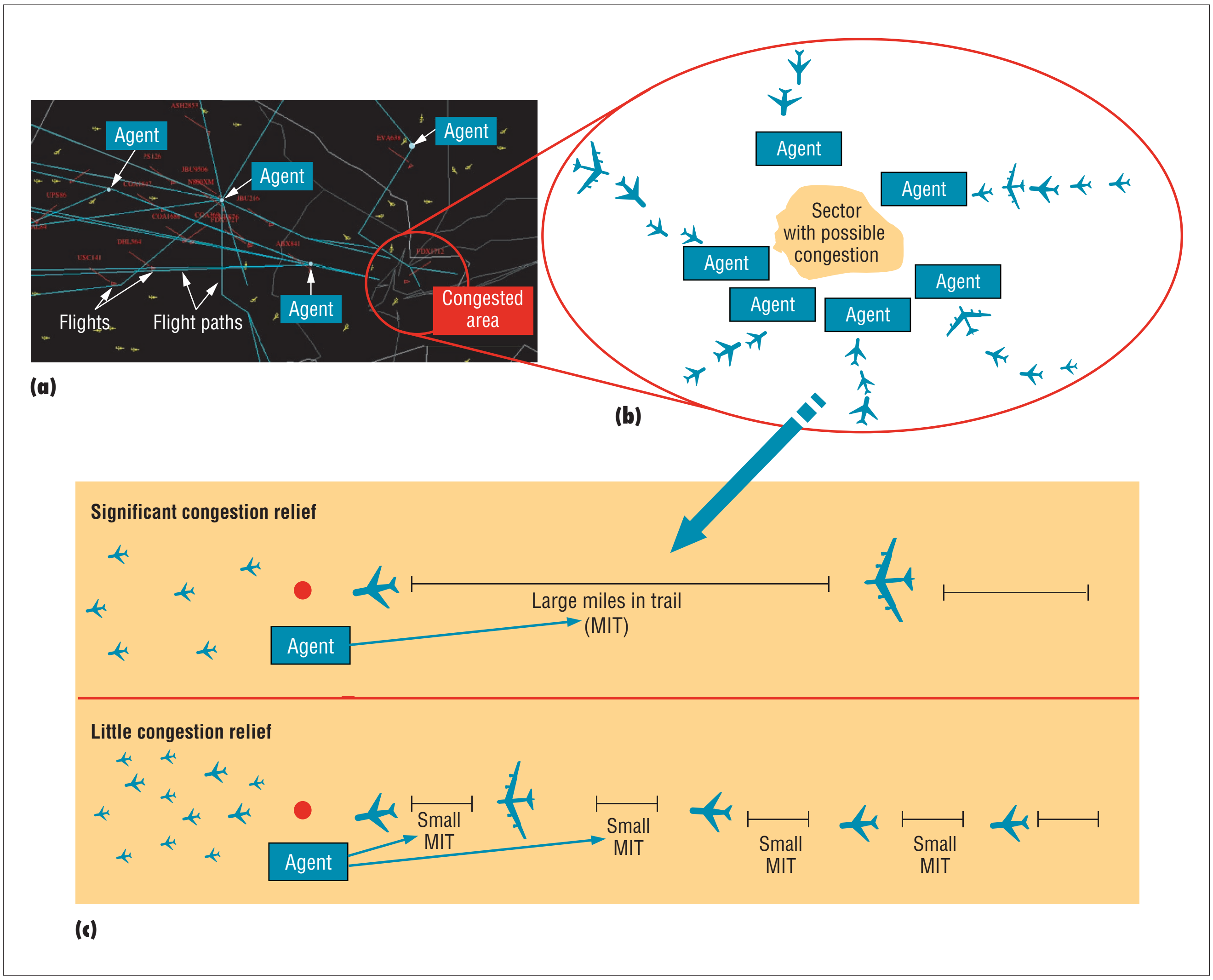


Figure 1. A multiagent system for air traffic management (ATM): (a) a screenshot of Facet (Future ATM Concepts Evaluation Tool) with agents superimposed, (b) agent activation around a congested area, and (c) the impact of agent actions (for example, setting miles-in-trail values), with the possible local-congestion implications of the agents' actions. The agent rewards aim to encourage agents to not only reduce local congestion but also take globally beneficial actions.

and business concerns for airspace configuration (long-term management).

We implemented our approach on Facet (Future ATM Concepts Evaluation Tool), a physics-based model of US airspace.[4] Facet is a reliable tool for simulating air traffic and is extensively used by the FAA, NASA, and industry (more than 40 organizations and 5,000 users).

A Multiagent ATM System

The agent approach to ATM is predicated on adaptive agents taking independent actions that provide good system-level behavior. To that end, the selection of agents, their actions, their learning algorithms, and their reward structure is critical.

Agent Selection

Perhaps the most obvious choice for an agent is the aircraft. With this approach, agent actions can be intuitive (for example, for changing a flight plan or increasing or decreasing speed or altitude) and offer a high level of granularity, in that each agent can have its own policy. However, this approach has two main drawbacks. First, on any given day, more than 40,000 aircraft are in US airspace, which would lead to a massively large multiagent system. Second, because the agents wouldn't be able to sample their state space sufficiently, learning would be prohibitively slow.

Instead, we assign agents to *fixes*—individual ground locations throughout the airspace. Each agent is then responsible for any aircraft going through its fix (see Figure 1).

Fixes offer four main advantages. First, their number can vary depending on need. The system can have as many agents as required for a given situation (for example, agents coming "live" around an area with developing weather conditions). Second, because fixes are stationary, agents can collect data and readily match behavior to reward. Third, because aircraft flight plans consist of fixes, agents can affect traffic flow patterns. Finally, fixes can be deployed within current air traffic routing procedures and can serve as tools to help air traffic controllers rather than compete with or replace them.

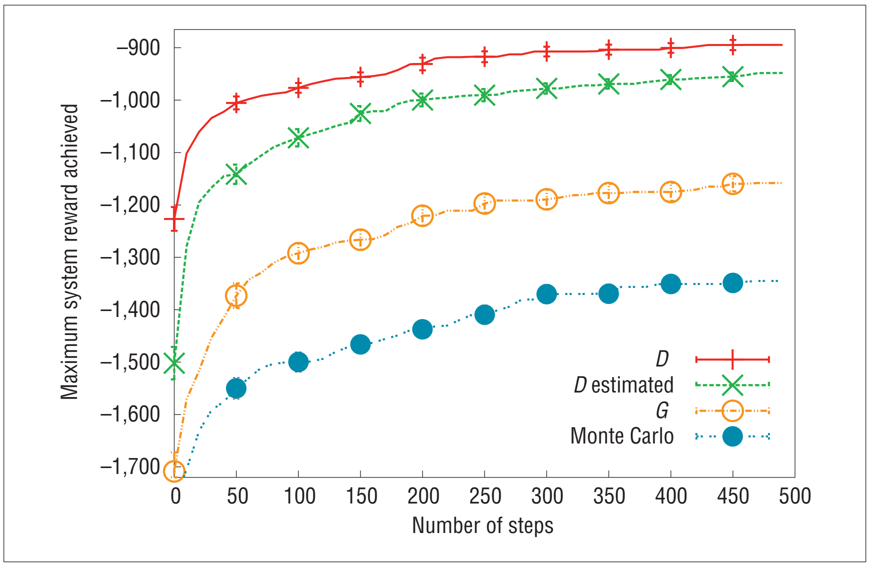


Figure 2. The performance of 40 agents controlling miles in trail (MIT) for a system with 300 aircraft. *D* stands for the difference reward, and *G* stands for the system performance as a reward. All agent-based algorithms outperformed the state of the art, and agents using *D* significantly outperformed "traditional" agents using *G*. The estimate for *D* provides a good balance between performance and computational cost.

Agent Actions

To control the flow, an agent assigned to a fix can take three kinds of actions. First, it can set the *miles in trail* (MIT)—the distance aircraft must keep from each other while approaching the fix. With a higher MIT, fewer aircraft will be able to go through that fix during congested periods because they'll be slowing down to maintain their spacing. So, the agent can set high MIT values to reduce congestion downstream of its fix.

Second, an agent can order *ground delays*, causing aircraft headed toward its fix to wait on the ground. Consequently, aircraft will arrive later at the fix. If some agents choose this action and others don't, congestion will decrease because it will be spread out. However, if all the agents choose the same ground delay, the congestion will simply occur later.

Third, an agent can *reroute* aircraft going through its fix. By diverting certain aircraft away from particular regions, this action can prevent specific instances of congestion and thus reduce congestion overall.

Learning Algorithms

In an adaptive multiagent approach, each agent aims to learn the actions that provide the best returns. Here, we focus on agents using reinforcement learning to maximize their rewards[5] (although alternatives such as evolving neurocontrollers are also effective[6]). At each time step, an agent takes an action and receives a reward evaluating that action. It uses this reward to update its action policy such that it will try to take actions leading to higher reward.[7] The simple reinforcement learners are equivalent to ε-greedy learners with a discount rate of zero,[5] where the value associated with each action $Q(s, a)$ after taking action a from state s and receiving reward R is updated by

$$Q'(s, a) = (1 - \lambda)Q(s, a) + \lambda R, \qquad (1)$$

where λ is the learning rate. To promote exploration of new actions, at each time step the agent chooses the action with the highest table value with probability $1 - \varepsilon$ and chooses a random action with probability ε.

Reward Structure

The most direct approach is to let each agent receive the system performance as its reward. Although this form of reward has been successful for small multiagent reinforcement learning problems, it doesn't scale well because an agent's actions have a relatively small impact on the system reward.

To alleviate this problem, we developed an additional reward that aims to be both sensitive to an agent's actions and aligned with the overall system reward.[8,9] Given a system state z and system-level reward function $G(z)$—for example, the system objective of reducing total delays and ensuring all sectors remain below their critical congestion limits—we derive each agent's *difference reward*:

$$D_i \equiv G(z) - G(z_{-i}), \qquad (2)$$

where z_{-i} is the system's state without agent i. In practice, we set all the state components of i to a constant, arbitrary value.

D has two beneficial properties. First, it has the same derivative as *G* with respect to the states of *i* because the second term on the right doesn't depend on *i*'s actions. So, any changes to *i*'s state that benefit *i* also benefit the entire system. Second, the second term on the right acts as a "noise" remover, providing a cleaner "signal" to *i*. This approach has proven effective in many domains, including ATM,[7] congestion games,[10] robot coordination,[6] and data routing.[11]

Although the difference reward effectively lets an agent see its actions' impact, it could be plagued by computational cost. Because *D* relies on the computation of the counterfactual term $G(z_{-i})$—that is, system performance without agent *i*—it might be difficult or impossible to compute, particularly when we don't know *G*'s exact mathematical form. So, we developed a third reward: an estimate of *D* that's computationally tractable and requires far fewer calls to Facet (one per time step, rather than one per agent).

Representative Results

We've applied our approach to artificial data with a single point or dual points of congestion as well as to limited real-world data related to historical congestion. In all cases, we compared the results to those from a basic Monte Carlo estimate, which is the state of the art for ATM.

Figure 2 shows the results for 40 agents controlling the MIT. The agents set the MIT at 0, 25, or 50 miles. Setting the MIT at 0 produces no effect, whereas setting it at high values forces aircraft to slow down to maintain their separation distance. So, setting high MIT values upstream of congestion can alleviate congestion but increase delay.

The results are based on a simulation with two independent instances of congestion with a total of 300 aircraft over five hours of flight time. The first instance involves relatively light congestion with 75 aircraft; the second involves heavy congestion with 225 aircraft. The system reward consists of a function of both delay and congestion (a detailed mathematical formulation appears elsewhere[7]). All results are based on 30 runs. Although we plotted the error bars, they're smaller than the symbols for the rewards in most cases. These results are representative of the relative performance of the agent-based approaches, where the full difference reward performs best but the estimated difference reward still performs better than directly using the system reward. In addition, all the learning-based agents outperformed the Monte Carlo approach.

Assigning an agent to a fix and providing those agents with local reward functions is a first step in bridging the technological gap between the current ATM process and the process needed to accommodate future air traffic.

We're extending this approach in five directions. First, to further speed up the simulations, we're exploring new methods of estimating agent rewards. Second, we're exploring how agent coupling affects system performance. In this situation, one agent's actions restrict another agent's actions; for example, setting ground delays might impact a reroute, or a reroute might impact the MIT. Third, we're investigating the performance of agents that provide only recommendations to air traffic controllers. This approach presents interesting problems in that the reward an agent receives might not be based on the action it recommended (for example, the air traffic controller might have ignored the recommendation). Fourth, we're integrating real data (from Chicago and New York) to extend the simulation results to real historical congestion. Finally, we're investigating broad definitions of system performance ranging from delays to more complex criteria including fairness, timeliness, and economic concerns.

References

1. B. Sridhar et al., "Aggregate Flow Model for Air-Traffic Management," *J. Guidance, Control, and Dynamics*, vol. 29, no. 4, 2006, pp. 992–997.
2. "Airline On-Time Statistics and Delay Causes," US Bureau of Transportation Statistics, 2007; www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp.
3. *Your Flight Has Been Delayed Again*, US Joint Economic Committee, May 2008, pp. 1–12.
4. K.D. Bilimoria et al., "Facet: Future ATM Concepts Evaluation Tool," *Air Traffic Control Quarterly*, vol. 9, no. 1, 2001, pp. 1–20.
5. R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
6. A.K. Agogino and K. Tumer, "Efficient Evaluation Functions for Evolving Coordination," *Evolutionary Computation*, vol. 16, no. 2, 2008, pp. 257–288.
7. K. Tumer and A. Agogino, "Distributed Agent-Based Air Traffic Flow Management," *Proc. 6th Int'l Joint Conf. Autonomous Agents and Multiagent Systems* (AAMAS 07), Int'l Foundation for Autonomous Agents and Multiagent Systems, 2007, pp. 330–337.
8. K. Tumer, "Designing Agent Utilities for Coordinated, Scalable and Robust Multi-agent Systems," *Challenges in the Coordination of Large Scale Multiagent Systems*, P. Scerri, R. Mailler, and R. Vincent, eds., Springer, 2005, pp. 173–188.
9. K. Tumer and D. Wolpert, "A Survey of Collectives," *Collectives and the Design of Complex Systems*, Springer, 2004, pp. 1–42.
10. A.K. Agogino and K. Tumer, "Handling Communication Restrictions and Team Formation in Congestion Games," *J. Autonomous Agents and Multi-agent Systems*, vol. 13, no. 1, 2006, pp. 97–115.
11. D.H. Wolpert and K. Tumer, "Collective Intelligence, Data Routing and Braess' Paradox," *J. Artificial Intelligence Research*, vol. 16, 2002, pp. 359–387.

Kagan Tumer is an associate professor at Oregon State University. Contact him at kagan.tumer@oregonstate.edu.

Adrian Agogino is a research scientist at the University of California, Santa Cruz and the NASA Ames Research Center. Contact him at adrian@email.arc.nasa.gov.