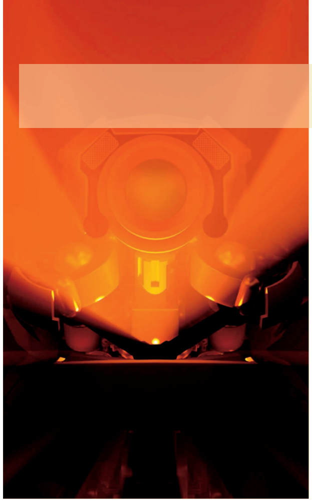

Editor: **Terry R. Payne,** University of Southhampton, trp@ecs.soton.ac.uk

Agent-Based Approach to Free-Flight Planning, Control, and Simulation

Michal Pechoucek and David Sislak, *Czech Technical University*

Air traffic is extremely busy and increasing daily. In fact, the aviation industry is planning for rapid growth worldwide. In the recent *2008 Current Market Outlook Report*, Boeing anticipates that cargo will triple over the next 20 years. Air traffic control faces yet another critical challenge: ever-increasing requirements for air traffic operation involving manned and unmanned aerial assets. Small unmanned aerial vehicles—often used, for example, for emergency surveillance and monitoring—must be able to operate near airports with heavy civilian air traffic. Clearly, current air traffic management systems can't efficiently support such requirements or handle much more than the current density. Sophisticated, intelligent technology is needed for further growth in the capacity and safety of worldwide, mixed-initiative air traffic.

The free-flight concept,[1] currently a hot paradigm, suggests moving away from centralized, predefined, prebooked flight corridors and recommends decentralizing air traffic control among multiple (manned or unmanned) flying assets. Decentralizing air traffic planning and control is expected to provide more efficient use of the available airspace and improve support for replanning and collision avoidance (CA), especially in the case of dynamic unmanned operations.

Intelligent-agent technology, supported by the research results from AI and from autonomous agents and multiagent systems, provides a set of mechanisms and protocols for negotiation, coordination, and decentralized decision making in communities of self-interested or partially cooperative actors. This functionality has the potential to directly support free-flight operations by modeling individual assets as agents and providing each asset with automated decision-making support aimed at coordinated, collision-free, efficient flight. At the same time, agent technology provides a valuable computational experimental environment that can be efficiently used for testing the properties of free-flight coordination algorithms prior to their deployment in real hardware and real air traffic.

AgentFly

In cooperation with the US Air Force, researchers in the Agent Technology Center at the Czech Technical University have developed AgentFly—a scalable, agent-based technology for free-flight simulation, planning, and CA. In AgentFly, each flying asset represents a specific software container hosting multiple intelligent software agents. Each agent either models a specific hardware functionality such as sensory capability, dynamic flight control, or communication, or encapsulates an intelligent decision-making technology that supports planning or CA. Such an architecture supports three principal AgentFly use cases:

- multiagent modeling and simulation of free flight,
- control of free-flight unmanned aerial platforms, and
- alternative approaches to planning, which supports civilian air traffic control.

We can use multiagent simulation of free-flight operations to empirically analyze various planning and CA algorithms before physically deploying them. If the simulation is realistic enough (AgentFly can provide all the tools and technology needed), an empirical analysis of the findings could provide valuable information about the properties of free flight in various circumstances (such as surveillance tracking, worsened weather conditions, dense civilian traffic, or emergency situations).

AgentFly is designed so that no centralized component is needed; all the planning and CA are based on the flying assets' sensory capability and distributed (peer-to-peer) decision-making capability. We can deploy the planning and CA agents directly onto hardware platforms and thus support real free-flight exercise of unmanned assets. The AgentFly system is based on Aglobe multiagent technology, which supports seamless migration from computational simulation to hardware deployment. Previously, researchers successfully migrated an Aglobe-based model of a ground-based robotic scenario to the RoboCup soccer environment.

1541-1672/09/$25.00 © 2009 IEEE
Published by the IEEE Computer Society

The most promising direction for applying AgentFly is in the area of air-traffic planning. The US Federal Aviation Authority (FAA) is interested in testing AgentFly's planning capacity for heavily overloaded civilian air traffic across the entire national air space. The idea is to relax the planning problem and perform multiagent flight simulations. Instead of planning a collision-free operation for numerous aircraft, we'd construct a flight plan for each individual craft without considering possible collisions. Subsequently, we simulate such an operation in the AgentFly environment, detect possible collisions, and solve them through either individual replanning or peer-to-peer negotiation.

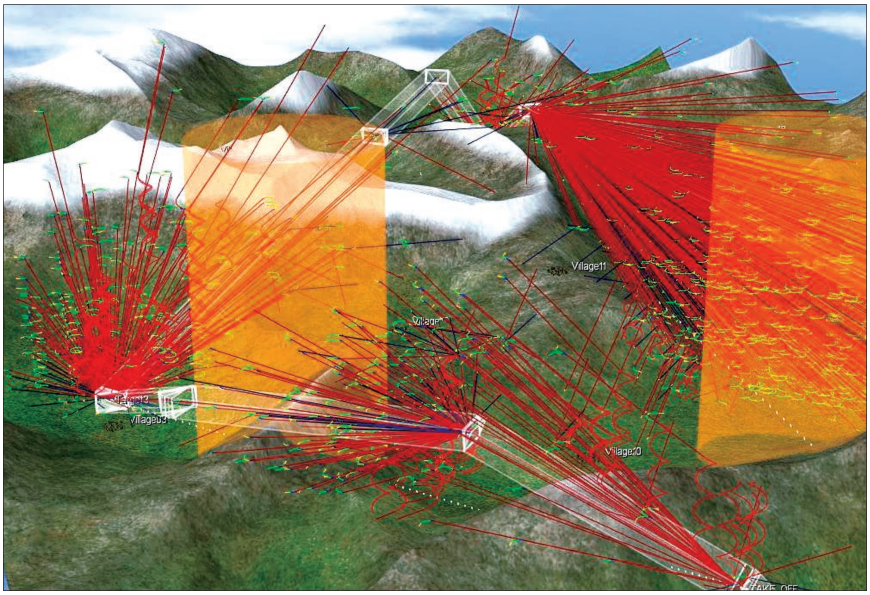

Figure 1. A path-planning example in a mountainous environment, with defined cylindrical no-flight zones. The white flight corridor highlights the final solution.

Planning

Each agent hosts its own path planner that provides a smooth flight plan trajectory, respecting all of the airplane model's constraints and goals (its mission). Each planner transforms these goals and requests for CA maneuvers into a sequence of waypoints, each with specified time and cruise-speed restrictions. The planners prepare detailed descriptions of the individual flight corridor—a geographical definition and cruise-speed changes over time. The given flight trajectory can be executed imprecisely by integrating the Required Navigation Performance (RNP) standard[2] into the planning process, thereby specifying permissible deviations in the plane's horizontal and vertical positions within its corridor. When an RNP level is no longer reachable, the planner finds another plan.

Internally, the path-planning process runs in two coupled phases: spatial and time planning. During the first phase, the planner prepares the spatial part of the flight plan, respecting the specified trajectory restrictions by means of the Accelerated A* (AA*) algorithm (see Figure 1). This algorithm is suitable for fast planning in large environments with numerous operational restrictions (such as dynamic no-flight zones, minimal flight levels, and noncooperative light operations). Defined airspace is kept in a tree structure, where each inner node is a composition or transformation operator and each leaf holds a zone definition with a geometrical description, a height map, and octant tree representations. Using such tree compositions, complex airspaces can be modeled.

In the second phase, the planner plans the cruise-speed changes, mapping them to the prepared spatial part of the trajectory. The goal here is to fit all given time and cruise-speed restrictions. The spatial and time phases are connected in a loop so that the system can handle extreme cases—for instance, if a plane must fly more slowly than its minimal cruise speed, its spatial plan must be lengthened.

Collision Avoidance

No matter whether AgentFly is used for simulation, planning, or real hardware control, its capability to autonomously avoid collisions is at the center of its research contribution and is critical for its commercial exploitation. AgentFly features four classes of CA mechanisms.

First, the Rule-Based CA (RBCA) algorithm is a domain-dependent algorithm based on the FAA's Visual Flight Rules (VFRs). Upon detecting a collision threat, the agent determines the collision type on the basis of the angle between the direction vectors of the aircrafts involved. Each collision type has a predefined fixed maneuver, which the agent applies in replanning. Each asset independently performs VFR-based changes to flight plans, relying on the counterpart asset detecting and avoiding the same collision from its point of view using the same algorithm. We implemented this RBCA in AgentFly as a reference mechanism for testing efficiency of the following three CA algorithms.

The Iterative Peer-to-Peer Collision Avoidance (IPPCA) algorithm deploys multiagent negotiations aimed at finding the pareto-optimal CA maneuver. Software agents hosted by each asset generate a set of viable CA maneuvers (by means of the planning mechanism described earlier) and compute the costs associated with each maneuver (on the basis, for example, of the flight plan's total length, time deviations for mission waypoints, altitude changes, curvature, flight priority, fuel status, possible damage, and type of load). Figure 2 (on the next page) shows an example of a pair negotiation that's searching for a combination of maneuvers that will minimize their joint cost associated with avoiding the collision.

The Multi-Party Collision Avoidance (MPCA) algorithm extends the first two algorithms by allowing several assets to negotiate their collective CA maneuvers. The order in which collision threats occur and are solved strongly affects the quality of the overall plan provided by IPPCA. The MPCA algorithm is designed to minimize CA maneuvers' ability to cause conflicts between future trajectories. This strategy requires substantially more computational and communication resources for solving a single encounter, but it provides more efficient free-flight collision-free trajectories in the long run.

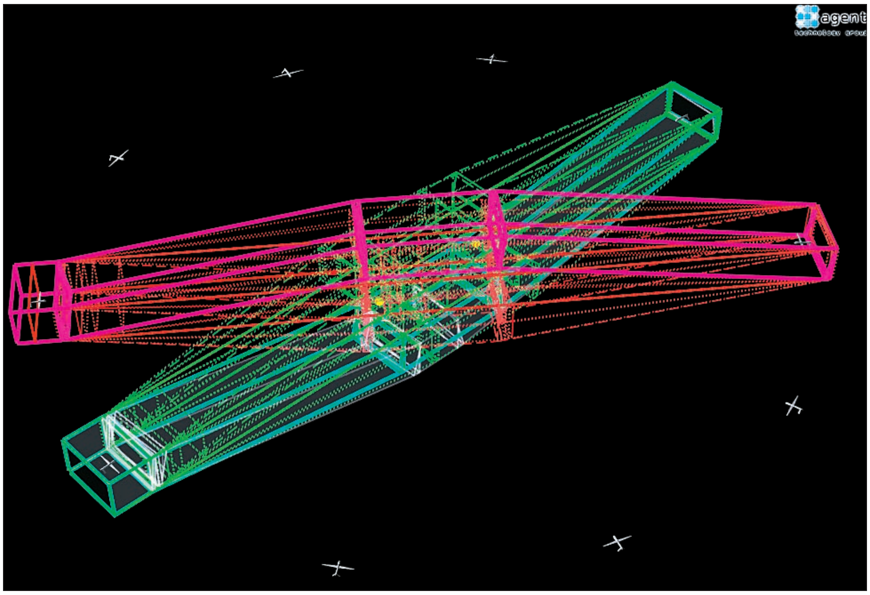

Figure 2. A state-space example of pair negotiation using the Iterative Peer-to-Peer Collision Avoidance (IPPCA) algorithm in the superconflict setup of 10 airplanes. Yellow points represent identified collision boundaries between the original flight plans of those two airplanes.

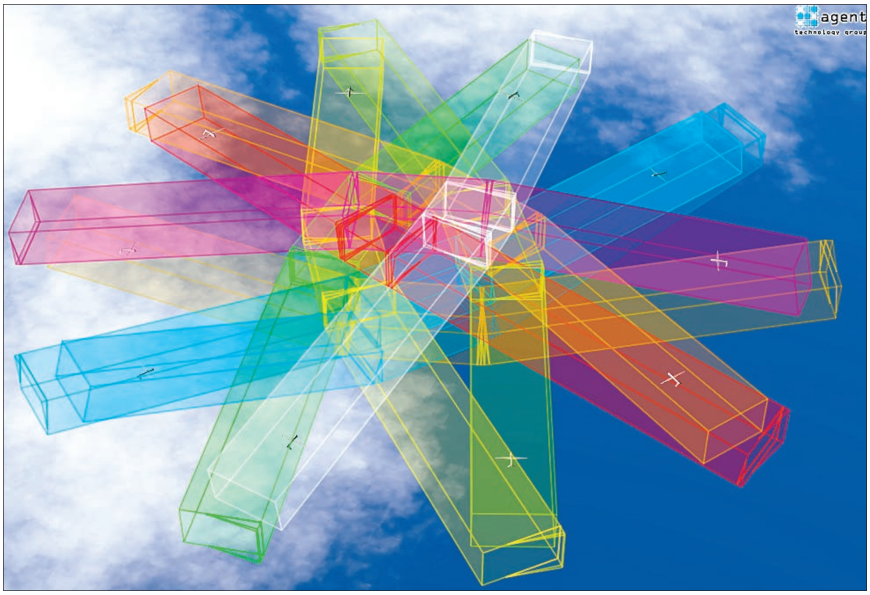

Figure 3. A 3D solution after several IPPCA iterations for the superconflict setup of 10 airplanes. Each experiment compared the final trajectory addition to the optimal (shortest) path.

The Noncooperative Collision Avoidance (NCCA) algorithm supports CA when communication between assets is impossible. This situation can arise, for example, when on-board communication devices are temporarily unavailable or when an asset avoids a hostile flying object. This class of algorithms is based on modeling and predicting the noncooperative object's future airspace occupancy and representing its possible future positions in terms of dynamic no-flight zones. On the basis of this information, the algorithm performs continuous replanning using the previously described planning algorithm.

Even though AgentFly can compare the effectiveness of various CA methods in different scenarios, the free-flight dynamic environments are rarely suited for using a single CA algorithm at all times. So, AgentFly features an efficient multilayer CA architecture that provides sophisticated mechanisms for the flexible selection of an appropriate CA algorithm in various situations. This architecture features a meta-reasoning process that analyzes time-to-collision and estimated time requirements for each CA method with respect to efficiency of the process needed. The multilayer module works in a fully decentralized manner and doesn't use any central planner. Its architecture is domain independent and therefore ready for deployment in autonomous vehicles such as airplanes, robots, cars, and submarines.

Deployment Scenarios and Selected Experimental Results

We used AgentFly to validate and test decentralized CA algorithms mainly in two ways: we compared selected properties (such as quality of solution and required computational and communication resources) of given algorithms under the same configuration, and we validated algorithms in mixed cooperative and noncooperative configurations. Algorithms are often benchmarked in complicated collision cases—for example, in a superconflict scenario in which airplanes are located in a circle and are all flying to opposite sides of the circle, implying the multicollision of all the planes in the center of the circle (see Figure 3). Figure 4 presents comparisons of the RBCA, IPPCA, and MPCA algorithms regarding the quality of the final solution and the number of algorithm invocations when airplanes are randomly generated in the restricted area.[3] Although MPCA provided the most efficient flight plans (in terms of the total flight distance), it required frequent, voluminous communication among the assets as well as substantial computational requirements. So, IPPCA is regarded as the most suitable for existing flight scenarios.

The second group of deployment scenarios validates algorithms in the mixed mode, in which airplanes operate in the airspace shared with others (noncooperative objects) with which they can't communicate. The noncooperation mode can be caused by malfunctions in a communication transceiver, an incompatible system, or a manned airplane. In such situations, we can validate the solution if the tested algorithm can simultaneously take advantage of a communication-based solution for cooperating airplanes and avoid noncooperative objects. For example, a team of autonomous assets might fulfill their missions near the airport where manned traffic can't be suspended.

Although we'd be happy to support AgentFly deployment on flying hardware platforms, we're currently investigating AgentFly's scalability toward modeling tens of thousands of flying assets. For example, we're modeling the weather using various commercial airplanes' physical properties (on the basis of Base of Aircraft Data models provided by Eurocontrol). Meanwhile, a project funded by the US Army's Communications-Electronics Research, Development, and Engineering Center is using AgentFly to investigate the agent-based approach in collaborative, dynamic planning of tactical surveillance operations. Here, the CA capability is complemented with negotiation-based algorithms aimed at coordinating collective flight and planning coordinated surveillance. BAE Systems used AgentFly as a testbed for their MOD-funded research effort studying probability-based CA mechanisms. ■

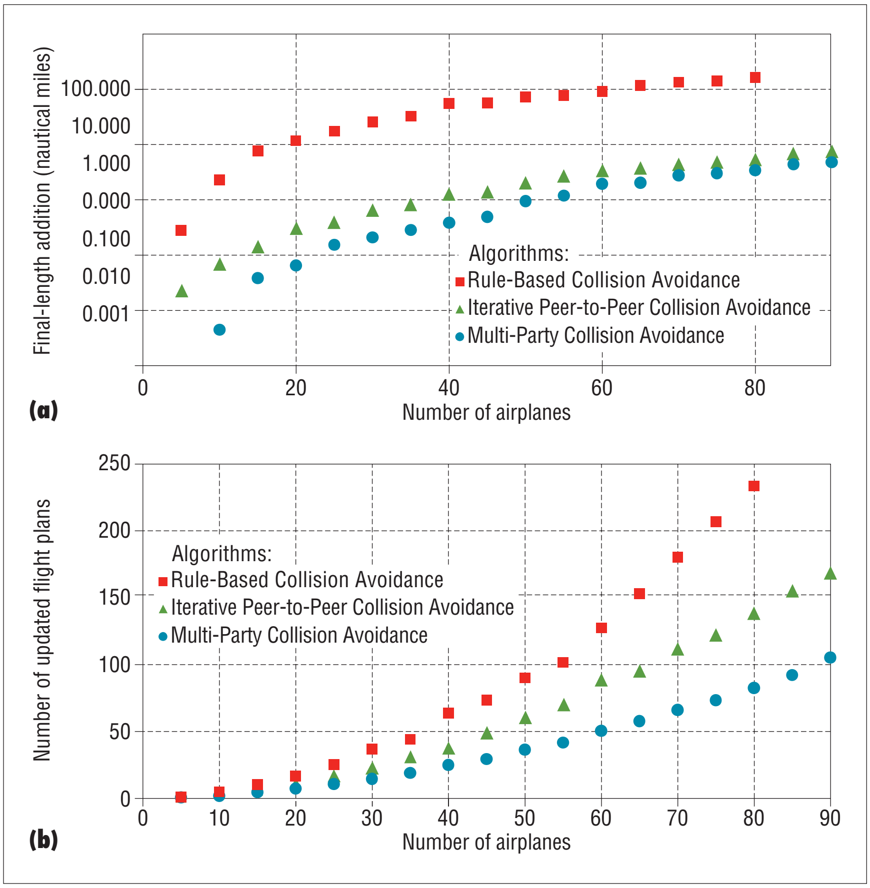

Figure 4. Comparisons of the RBCA, IPPCA, and MPCA algorithms. Using all three algorithms, we calculated (a) the final trajectory addition, in nautical miles, to the optimal (shortest) total flight distance for each experimental setting—that is, the length of all flight trajectories after all iterations minus the length of all shortest trajectories, divided by the number of planes. We also calculated (b) the number of flight plan iterations necessary to remove all collisions, averaged across 50 repeated experiments. The smallest blue value means that for simple scenarios MPCA avoids collision by one change only. So, averaging it across 10 airplanes results in these small values.

Acknowledgments

The US Air Force partially supported AgentFly development through grants FA8655-04-1-3044-P00001 and FA8655-06-1-3073, and the Czech Republic Ministry of Education partially supported development of the planning algorithms through its Research Program MSM6840770038. The US Army funds adaptation to tactical operations through grant W911NF-08-1-0521. The US Federal Aviation Administration supports adaptation of AgentFly to civilian traffic planning under contract DTFACT-08-C-00.

References

1. R. Schulz, D. Shaner, and Y. Zhao, "Free-Flight Concept," *Proc. AIAA Guidance, Navigation and Control Conf.*, AIAA Press, 1997.
2. *RTCA 236B, Minimum Aviation System Performance Standards: Required Navigation Performance for Area Navigation*, Radio Tech. Commission for Aeronautics, 2003.
3. D. Sislak, J. Samek, and M. Pechoucek, "Decentralized Algorithms for Collision Avoidance in Airspace," *Proc. 7th Int'l Conf. Autonomous Agents and Multi-Agent Systems*, AAMAS Press, 2008, pp. 543–550.

Michal Pechoucek is head of the Agent Technology Center and vice-chairman of the Department of Cybernetics, both at the Czech Technical University. Contact him at michal.pechoucek@agents.felk.cvut.cz.

David Sislak is a PhD student and a researcher in the Agent Technology Center, Department of Cybernetics, at the Czech Technical University. Contact him at david.sislak@agents.felk.cvut.cz.