

# Large-scale Agent-based Simulation of Air-traffic

David Šišlák, Přemysl Volf and Michal Pěchouček

Agent Technology Center, Department of Cybernetics

Faculty of Electrical Engineering

Czech Technical University in Prague

Czech Republic

sislakd@fel.cvut.cz, {volf,pechouc}@labe.felk.cvut.cz

## Abstract

We describe a multi-agent platform AGENTFLY used for large-scale high-detail simulation of air traffic. The platform introduces spatial and temporal planning within 3D space, multi-layer architecture using several collision avoidance algorithms. To be able to simulate a huge amount of airplanes, distributed simulation is introduced using spatial partitioning and dynamic load-balancing. The platform has been used to support simulation of an entire civilian air traffic touching National Air-Space of United States. Thorough evaluation of the system has been performed, confirming that it can scale up to a very high number of complex agents operating simultaneously (thousands of aircraft) with full detailed models.

## 1 Introduction

Agent-based simulations have recently become a popular way to model and study complex multi-actor systems. The complexity of the system is caused by a large number of agents and high complexity of their reasoning and/or high complexity of agent-environment interaction. In this paper, we propose a distributed agent-based platform AGENTFLY which is capable of simulating complex air-traffic systems, especially with the huge amount of simulated airplanes.

The key components for the simulation of a single aircraft is a 3D planner and a precise airplane model. Collision avoidance ability is needed to simulate multiple airplanes in the same airspace. Distributed simulation is necessary to simulate the huge amount of airplanes.

The trajectory path planner is able to find smooth three-dimensional spatial and temporary trajectory through given waypoints avoiding given obstacles and restricted areas. The field of the path planning problem has been studied by the research community for many decades. There exist very efficient (fast), but not optimal algorithms based on randomness, e.g. the randomized potential field [Carpin and Pilonetto, 2005] algorithms. There also exist algorithms providing an optimal solution for the given domains with

pre-built structures for the given environment definition, but they are slow, e.g. the 3D field D\* [Carsten *et al.*, 2006]. The *Accelerated A\** (AA\*) algorithm [Šišlák *et al.*, 2009] used in the AGENTFLY platform supplements existing optimization-based path planning methods for large-scale environments where the existing algorithms are unusable for their computational and memory requirements.

The trajectory planner, together with a high precision airplane model based on the BADA airplane performance models [Nuic *et al.*, 2005], makes the simulation of the single airplane computationally demanding.

The collision avoidance functionality is introduced to automatically ensure collision free trajectory for every airplane. The metrics used to evaluate collision avoidance are described in [Krozel *et al.*, 2001]. The collision detection and avoidance methods are widely addressed by the research community. Cooperative collision avoidance is based on communication and negotiation protocols, e.g. the negotiation protocol for two airplanes [Wollkind *et al.*, 2004] is based on classical agent-based negotiation protocols and there are also various approaches based on the game theory (e.g. [Hill *et al.*, 2005]). Optimization non-cooperative collision avoidance algorithm, e.g. [Tomlin *et al.*, 1998], allows optimal solving of the collision with the airplane that is not able or does not intent to communicate, negotiate or cooperate. The multi-layer collision avoidance architecture integrates several different collision avoidance algorithms together to provide a safe plan in every situation.

The approach designed for the simulation of air-traffic uses a combination of the analytic simulation and distributed virtual environments (DVE) approach [Fujimoto, 1999]. Agent's communication, internal processes and non-physical interaction with other agents is asynchronous, non-deterministic, possibly containing a human in the loop, and the implementation of these processes therefore uses the DVE approach. Simulation of the environment, situated entities and physical interactions in the virtual world is synchronous and deterministic, and an analytic approach providing maximal accuracy is therefore used. The simulation can run in as-fast-as-possible mode preserving the accuracy of the simulation together

with asynchronous and non-deterministic behavior of the agents in faster-than-wall-clock-paced time.

The next part of the paper is focused on the distribution of the simulation to multiple hosts to be able to simulate higher amount of airplanes using division of the virtual world into a variable number of dynamically-sized partitions. Thanks to the prevailing locality of agent-to-agent and agent-to-environment interactions, such partitioning allows to effectively distribute the required computation between a number of machines while keeping the communication overhead low. Another approach to the multi agent simulation is described in [Viroli *et al.*, 2007] and [Mili *et al.*, 2006].

The proposed architecture has been applied to the simulation of the full civilian air-traffic touching National Air-Space of United States, requiring high-fidelity simulation involving the total of 52,799 airplanes with up to 6,500 simultaneous aircraft operating over the whole Earth – long flights departing, arriving or flying over the United States. The AGENT-FLY [Pěchouček and Šišlák, 2009] system implementing the proposed architecture is used for a thorough empirical evaluation.

## 2 Trajectory planning

The AA\* algorithm extends the original A\* algorithm to be usable in large-scale environments without forgetting about the search precision. The AA\* removes the trade-off between the speed and the precision by introducing of the *adaptive sampling*. During the expansion, child states are generated by applying vehicle elementary motion actions using elements' adaptive parametrization. A set of elementary motion actions is defined by the model of the non-holonomic vehicle movement dynamics. The adaptive parametrization varies and thus the algorithm makes larger steps when the current state is far from obstacles and restricted areas and smaller steps when it is closer. The Figure 1 shows the advantage of the adaptive sampling of the AA\*. The adaptive sampling can be seen as different size of the step (distance of the arc in the Figure) depending on the distance to the obstacle.

There is a defined *search precision* specifying the minimal sampling grid step which is used in the areas closest to obstacles. The search precision is defined so that the AA\* algorithm does not skip any existing gap between obstacles larger than this precision. Specifically, the AA\* algorithm uses the highest possible parametrization which ensures that the distance to the closest obstacle is not smaller than the distance corresponding to two respective sampling steps.

The adaptive sampling in the AA\* algorithm requires a different definition of identity tests when working with OPEN and CLOSE lists. The original equality implementation is replaced by a similarity check. Two states are similar if their Euclidean distance and their direction vector variation is less than a threshold derived from the respective sampling parametrization. Otherwise, the adaptive sampling of a non-holonomic vehicle trajectory causes an infinite

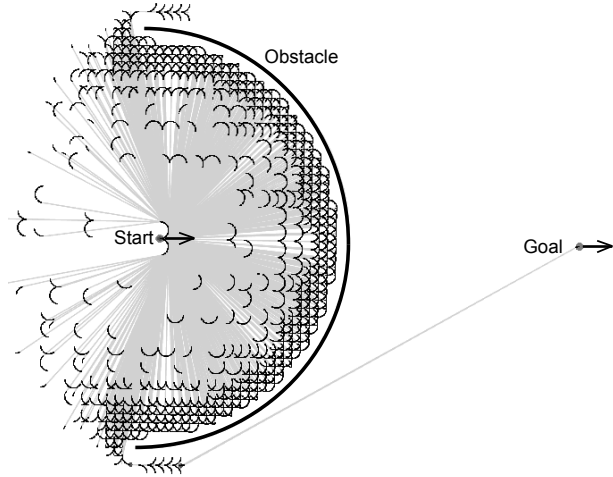


Figure 1: The adaptive sampling example in the two-dimensional setup.

state generation in the continuous space. To remove effects of varying sampling, each path candidate generated during the search is smoothed.

Properties of the AA\* concept were evaluated on a set of two and three-dimensional setups. The original A\* algorithm with a distance-to-target heuristics was chosen as a comparator because it is the only one which provides an optimal solution and does not require any pre-processing of the environment definition. The AA\* algorithm provides acceleration of the path planning up to 1400 times in comparison with the original A\* algorithm. Moreover, it was found that the AA\* algorithm also accelerates the result in case of failure (the path does not exist) due to the reduced number of all generated states.

## 3 Collision Avoidance

The multi-layer collision avoidance (MLCA) component is capable of solving future collisions by means of combination of different avoidance methods. There is no central planner providing a collision-free flight plan, hence the individual plans are provided by the planning agents<sup>1</sup>. Based on priority, MLCA selects an appropriate collision solver for specific possible future collision. Sophisticated switching of the collision avoidance algorithms is inevitable as the algorithms have different properties. Different algorithms provide different quality of the collision avoidance solution, while they require different amount of time for finding such a solution. Specifically, the negotiation-oriented algorithms (cooperative) may provide better collision avoidance solution than non-cooperative algorithms, while they may be more time-consuming (due to multi-party interactions).

There are several collision avoidance algorithms implemented in current system. The cooperative Rule-Based CA (RBCA) algorithm is a domain-dependent

<sup>1</sup>The proposed modular architecture is domain independent. Therefore it is ready for deployment on autonomous vehicles like airplanes or ground vehicles.

algorithm based on the FAA's Visual Flight Rules (VFRs). The implementation in AGENTFLY is used as a reference mechanism for testing efficiency of the following three CA algorithms. The Iterative Peer-to-Peer Collision Avoidance (IPPCA) algorithm employs pair-wise negotiation to solve the collision. The Multi-Party Collision Avoidance (MPCA) algorithm extends the first two algorithms by allowing several assets to negotiate over their collective CA maneuvers. The Noncooperative Collision Avoidance (NCCA) algorithm supports CA when communication between assets is impossible. This class of algorithms is based on modeling and predicting the non-cooperative objects future airspace occupancy and representing its possible future positions in terms of dynamic no-flight zones.

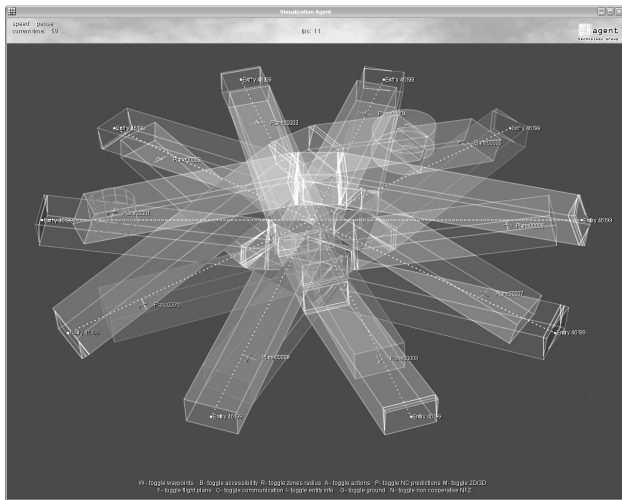


Figure 2: A 3D solution after several IPPCA iterations for the superconflict setup of 10 airplanes.

The IPPCA algorithm is used in the experiments for collision avoidance. It 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 (based on, for example, the flight plans total length, time deviations in mission waypoints, altitude changes, curvature, flight priority, fuel status, possible damage, and type of load).

## 4 System distribution

The simulation involves large numbers of *situated entities*<sup>2</sup> (aircraft) operating and interacting in a realistically modeled large-scale Earth-like 3D environment limited by a maximum altitude *virtual world*. The entities are not present during the whole simulation but are dynamically introduced or removed based on the simulation requirements.

Each situated entity in the simulation carries a *state*, components of which can be either (i) *observ-*

<sup>2</sup>By situated entities we mean that they are embedded in a synthetic model of a 3D physical space

*able* i.e. public and external or (ii) *hidden* i.e. non-observable, private and internal. The fundamental component of the entity's observable state present in the simulation is its position and orientation in the virtual world.

The evolution of the entity's state is governed by the defined entity's *physics*. Physics can be divided to *intra-entity* and *inter-entity*. The intra-entity physics captures those aspects of physical dynamics that can be fully ascribed to a single entity. Although the equations of the intra-entity physics can refer to any states in the simulated world, they only govern the states carried by its respective entity. In contrast, the inter-entity physics captures the dynamics that affects multiple entities simultaneously and which cannot be fully decomposed between them (consider e.g. the effects of a physical collision of two entities).

In addition to physical interactions, autonomous entities can also interact via communication, i.e. exchange electronic messages, and by using *sensors* to perceive their surrounding virtual environment.

Any situated entity in the virtual world consists of up to three subcomponents:

- *Body* – The entity's body encapsulates entity's intra-physics which governs the evolution of entity's state as a function of other, possibly external states. Typically, the body defines motion dynamics for the entity.
- *Reactive control* – The entity's reactive control contains real-time loop-back control for the entity. The loop-back control affects entity's states based on the observation of other states. Reactive control e.g. handle urgent reactive behaviors (such as avoiding an obstacle) triggered in emergency situations.
- *Deliberative control* – The entity's deliberative control contains complex algorithms employing planning, prediction and communication with other entities. The output of deliberative control are typically fed back into the entity's reactive control module and provides new control for the entity. Typically, complex deliberative algorithms providing high-level entity control are invoked based on the sensing perceptions.

Due to their principal similarity in the loop-back approach, the body and the reactive control are collectively referred to as entity's *state updater*.

Components presented in this section are implemented as software agents in a multi-agent middleware supporting the architecture with effective agent-to-agent communication, yellow pages services, agent full migration and also agent life-cycle management. All components in the simulator architecture are divided into two groups, see Figure 3:

- *Environment simulation agents* (ES agents) – These agents are responsible for application of all entities' state updaters and also for the updates resulting from the inter-physics.
- *Deliberative control agents* – The deliberative control part of each situated entity is encapsu-

lated in a separate deliberative control agent. Deliberative parts can communicate via agent-to-agent message transmission if this is allowed by inter-physics governing the communication medium.

Even though the deliberative control part and the state updater are decoupled in the simulator architecture, the deliberative control and the state updater of each entity in the simulation is connected by a signal channel through which sensing perceptions (one way) and control commands to the reactive control (the other way) are transmitted.

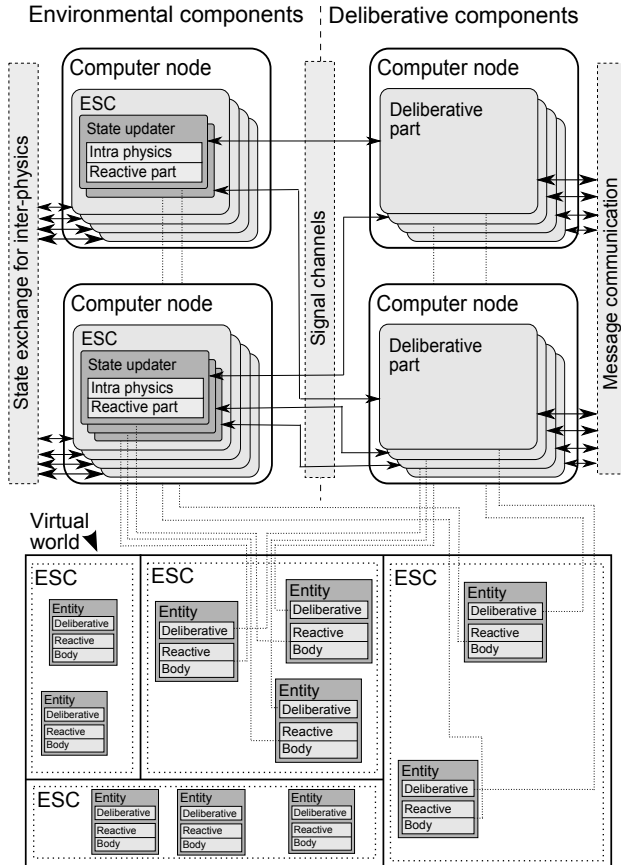


Figure 3: Architecture overview of the distributed simulation approach

Both the environment simulation and deliberative control agents can be distributed over multiple computer nodes during the simulation. The whole virtual simulation world is spatially divided into a number of partitions. The partitions are mutually disjoint except for small overlapping areas around partitions' boundaries. Each partition has a uniquely assigned environment simulation agent (ES agent) which is responsible for updating states corresponding to all entities located within its assigned partition. Figure 4 presents an example of air-space partitioning from the air-traffic simulation domain.

In contrast with the state updaters, entity's deliber-

ative control parts are deployed to the computer nodes irrespective of the partitions and corresponding entity's position in the virtual world.

The above virtual world partitioning implies that whenever the location of an entity changes in such a way that the entity moves (spatially) to a new partition, the entity's state updater needs to be migrated to ES agent responsible for the new partition. In contrast, the relatively heavy-weight deliberative control parts are never moved between computer nodes.

In order to provide maximum performance throughout the whole simulation, the proposed architecture implements load-balancing mechanisms. Load-balancing is applied to both environment simulation agents (through dynamic re-partitioning) and deliberative control agents (through balanced startup deployment).

The virtual world decomposition to partitions is dynamically updated depending on the load of computer nodes hosting environment simulation agents. Based on the time required for processing of the state updaters and inter-physics during the simulation cycle by each ES agent, the world is re-partitioned so that the number of entities belonging to partitions is proportional to the measured times.

The load of computer nodes running deliberative control agents is balanced only when new deliberative agents are instantiated. Deliberative parts of newly introduced entities are spread proportionally among computer nodes according to individual nodes' average load in several previous simulation cycles, expressed as the sum of each node's idle time over those simulation cycles.

## 5 Evaluation

We have evaluated the proposed distributed simulation architecture on a large-scale simulation of civilian air-traffic. The AGENTFLY platform has been deployed for the simulation of civilian air-traffic within U.S. National Airspace (U.S. NAS). The resulting airspace simulator is used for studying concepts for the Next Generation Air Transportation Systems (NGATS) in cooperation with the Federal Aviation Administration (FAA).

Each civilian flight operated under Instrumented Flight Rules (IFR) is represented as a situated entity agent within the simulation. One simulation run covers one day and simulates operation of all civilian IFR flights which traverse, fully or partially, U.S. NAS. Even though the area of the study is limited to U.S. NAS, some effects external to U.S. NAS are also studied, covering almost the whole Earth for long-distance flights. The overall number of simulated airplanes is 52,799 (up to 6,500 are simultaneous) corresponding to real air-traffic for a particular day in 2007, see Figure 4. Depending on the airplane type, each entity uses a precise flight model based on the BADA airplane performance models [Nuic *et al.*, 2005].

The distribution of the number of airplanes during the simulation day time is depicted in Figure 5. Note that there are significant peaks every half an hour of

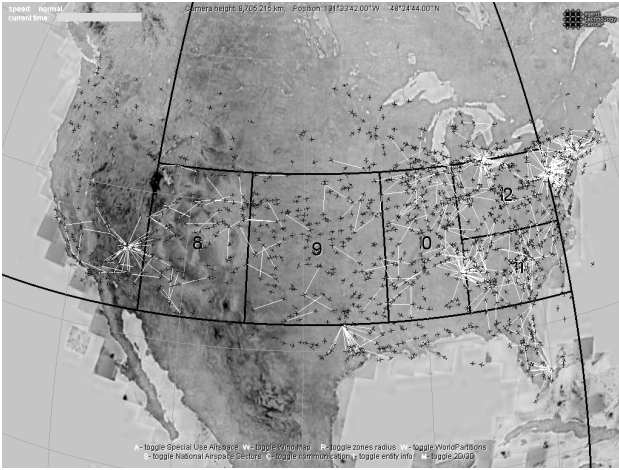


Figure 4: Airplanes over U.S. NAS and world partitioning

the simulation time; these peaks lead to extreme load of the system with more than 300 airplanes created at one simulation cycle in the highest peak of the day.

The granularity of the simulation is 12 seconds time step, when states of all airplanes are recomputed. The simulation of each airplane goes to the level of drag, lift and engine thrust forces in any moment to be able reach maximal precision in fuel consumption computation. All airplanes avoid the special use airspaces (provided by FAA) as well as avoiding each other (collision avoidance).

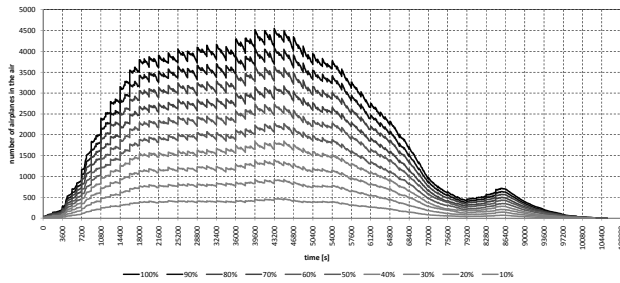


Figure 5: The number of simultaneous airplanes during the simulation.

The configuration contained a set of several heterogeneous (both in terms of the operating system and hardware) computer nodes. During experiments, nodes were dedicated for ESCs or deliberative controllers. The testing configuration consisted of 9 cores with 9GB RAM for ESCs and 11 cores with 15GB RAM for the flight control integrated in deliberative controllers. All computers were interconnected through a 1 Gbit Ethernet network.

The *run-time* is the total real (wall clock) time needed for the simulation multiplied by the number of processor cores used during the simulation.

The *normalized airplane* is a fictional airplane that is flying during the whole simulation from its start to the end. The number of airplanes is changing during the simulation, thus the mean of the number of air-

planes is used as the number of normalized airplanes running during the whole simulation. This number is used to normalize run-time to study requirements per each situated entity.

The objective of the experiment was to explore how the speed of the simulation changes with the changing level of system load. The experiment was performed using all computer nodes. The variations in system load were introduced by varying the number of airplanes simulated.

The aggregated results of the experiment are depicted in Figure 6 showing the correspondence between the total duration of the simulation and the number of flights. Chart A shows that the maximum duration is less than 13 minutes for 100% of flights. Chart B shows run-time needed to simulate one normalized airplane.

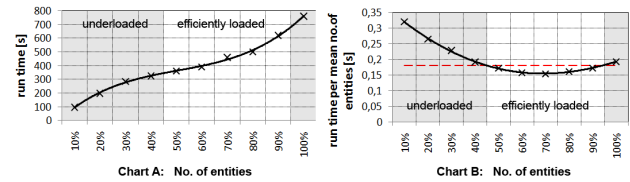


Figure 6: Effect of the increasing number of flights simulated with fixed hardware resources, measured in terms of the total duration of the simulation (Chart A) and run-time per normalized airplane (Chart B).

The experiment demonstrates properties and effectiveness of the distributed simulation. The measured characteristics can be divided into three regions depending on the system load (the number of airplanes): (i) underloaded, (ii) optimally loaded and (ii) overloaded simulation. The regions can be determined by the specified threshold, a dashed horizontal line at 0.18 s in Figure 6, Chart B.

The upper-left part of the plot above the threshold corresponds to the underloaded simulation, the part below the threshold is the optimally loaded simulation and the upper-right part above the plot is the overloaded simulation.

At low levels of system load (few situated entities), the simulation is underloaded and thus inefficient. The hardware resources cannot be effectively used because the load of the simulation is too low (e.g. the simulation running 5 airplanes cannot effectively use 20 processor cores), but there is an overhead given by the maintenance of distributed architecture which depends both on the number of processors cores and the number of airplanes in the system.

In the optimally loaded configuration, the simulation is able to use all available hardware resources effectively. This section begins at the moment when all hardware can be used and ends when some components start to be overloaded. Within the optimally loaded region, the increasing load of the simulation is distributed regularly among all resources, leading to almost linear increase of the simulation run-time.

Moving towards the highest load of the fixed computer configuration, the hardware resources become

overloaded. The overload has three main reasons.

CPU overload – The simulation management is designed to run the simulation as fast as possible considering the CPU load. If the CPU performance is not sufficient, the simulation continuously slows down until the whole simulation almost stops.

Memory overload – Once the simulation requires more than available physical memory, this results in higher probability of page faults and causes swapping which leads to decrease of the simulation performance.

Network overload – Despite the distributed design which inherently aims to minimize the network traffic, the network can get overloaded. If the network is not able to transmit all traffic required by the simulation especially related to the environment simulation, the simulation is paused and waits for their delivery.

In the experiments, usually both CPU and memory resources were overloaded. Thus the simulation gradually slows down at the beginning of the overload and with higher load it slows down quickly.

## 6 Conclusion

We have investigated the problem of the efficient simulation of a huge number of high-detailed aircraft models in a large-scale world. A combination of spatial and temporal 3D planner, multi-layer collision avoidance architecture and distribution of the simulation across several computers allows to process such simulation efficiently. The approach has been implemented using an agent-based simulation platform AGENTFLY, and evaluated on the simulation of the complete civilian air-traffic touching the U.S. NAS involving the total of 52,799 flights with up to 6,500 airplanes simulated at one time. The evaluation confirms the ability of the approach to handle complex high-detailed large-scale agent-based simulations.

## Acknowledgments

The work has been supported by the Federal Aviation Administration (FAA) under project number DTFAC-08-C-00033 and by Czech Ministry of Education under grant number 6840770038. The underlying AGENTFLY system was supported by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA8655-06-1-3073. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Federal Aviation Administration, the Air Force Office of Scientific Research or the U.S. Government.

## References

[Carpin and Pilonetto, 2005] S. Carpin and G. Pilonetto. Merging the adaptive random walks planner with the randomized potential field planner. In *Proceedings of IEEE International Workshop on Robot Motion and Control*, pages 151–156, 2005.

[Carsten *et al.*, 2006] Joseph Carsten, Dave Ferguson, and Antony Stentz. 3D Field D\*: Improved Path Planning and Replanning in Three Dimensions. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3381–3386, 2006.

[Fujimoto, 1999] R.M. Fujimoto. *Parallel and Distribution Simulation Systems*. John Wiley & Sons, Inc. New York, NY, USA, 1999.

[Hill *et al.*, 2005] Jared C. Hill, F. Ryan Johnson, James K. Archibald, Richard L. Frost, and Wynn C. Stirling. A cooperative multi-agent approach to free flight. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1083–1090, New York, NY, USA, 2005. ACM Press.

[Krozel *et al.*, 2001] Jimmy Krozel, Mark Peters, Karl D. Bilimoria, Changkil Lee, and Joseph S.B. Mitchel. System performance characteristics of centralized and decentralized air traffic separation strategies. In *4th USA/Europe Air Traffic Management R & D Seminar*, Stanta Fe, NM, December 2001.

[Mili *et al.*, 2006] R.Z. Mili, R. Steiner, and E. Oladimeji. DIVAs: Illustrating an abstract architecture for agent-environment simulation systems. *Multiagent and Grid Systems*, 2(4):505–525, 2006.

[Nuic *et al.*, 2005] A. Nuic, C. Poinot, M.G. Iagaru, E. Gallo, F.A. Navarro, and C. Querejeta. Advanced Aircraft Performance Modelling for ATM: Enhancements to the BADA Model. In *Beitrag zur 24th Digital Avionics System Conference. Washington: DASC*, 2005.

[Pěchouček and Šišlák, 2009] Michal Pěchouček and David Šišlák. Agent-based approach to free-flight planning, control, and simulation. *IEEE Intelligent Systems*, 24(1), 2009.

[Tomlin *et al.*, 1998] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management : A study in multi-agent hybrid systems, 1998.

[Viroli *et al.*, 2007] M. Viroli, T. Holvoet, A. Ricci, K. Schelfhout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, 2007.

[Šišlák *et al.*, 2009] David Šišlák, Přemysl Volf, and Michal Pěchouček. Accelerated a\* path planning. In *The Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary, May 2009.

[Wollkind *et al.*, 2004] Steven Wollkind, John Valasek, and Thomas R. Ioerger. Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *Proceedings of the American Inst. of Aeronautics and Astronautics Conference on Guidance, Navigation, and Control*, Providence, RI, 2004.