

AGENTFLY: Towards Multi-Agent Technology in Free Flight Air Traffic Control

David Šišlák, Michal Pěchouček, Přemysl Volf, Dušan Pavlíček,
Jiří Samek, Vladimír Mařík and Paul Losiewicz

Abstract. Ever rising deployment of *Unmanned Aerial Assets* (UAAs) in complex military and rescue operations require novel and innovative methods for intelligent planning and collision avoidance among a high number of heterogeneous, semi-trusted flying assets in well specified and constrained areas [1]. We have studied the free flight concept as an alternative to the classical, centralized traffic control. In free flight the unmanned aerial assets are provided with flight trajectory that has been elaborated without consideration of other flying objects that may occupy the same air space. The collision threads are detected by each of the aircraft individually and the collisions are avoided by an asset-to-asset negotiation. Multi-agent technology is very well suited as a technological platform for supporting the free-flight concept among the heterogeneous UAAs. In this chapter we present AGENTFLY, multi-agent system for free-flight simulation and flexible collision avoidance.

1. Introduction

AGENTFLY is a software prototype of a multi-agent simulator of unmanned aerial vehicles air traffic control supporting the free flight concept. All aerial assets in AGENTFLY are modeled as asset containers hosting multiple intelligent software agents. Each container is responsible for its own flight operation. The operation of each vehicle is specified by an unlimited number of time-specific, geographical waypoints. The operation is tentatively planned before take-off without consideration of possible collisions with other flying objects. During the flight performance, the software agents hosted by the asset containers detect possible collisions and engage in peer-to-peer negotiation aimed at sophisticated re-planning in order to avoid

the collisions. The implemented simulator demonstrate readiness of the multi-agent technology for distributed, flexible, and collision-free coordination among heterogeneous, autonomous aerial assets (manned as well as unmanned) with a potential to (i) fly a higher number of aircrafts, (ii) decrease requirements for human operators and (iii) allow a flexible combination of cooperative and non-cooperative collision avoidance.

AGENTFLY is build on top of the *A-globe* multi-agent platform [12]. *A-globe* provides flexible middleware supporting seamless interaction among heterogeneous software, hardware and human actors. *A-globe* outperforms available multi-agent integration toolkits by its ability to model rich environments in which agents interact, by its support of full code migration and by its support for scalable experiments. For more information see Appendix A. Current AGENTFLY implementation provides a distributed model of flight simulation and control, time-constrained way-point flight planning algorithm avoiding specified no-flight zones and terrain obstacles, flexible collision avoidance architecture – cooperative and non-cooperative, connectors to external data sources (Landsat images, airports monitors, no-flight zones, cities), 2D/3D visualization including a web-client access component, and a multi access operator - a component facilitating real-time control of selected assets.

The present work mainly addresses the problem of distributed collision avoidance among autonomous aerial assets using multi-agent technology [11] – each UAA is represented by an agent container hosting different functional agents [15]. Each UAA is controlled by a single, dedicated agent. The presented collision avoidance architecture provides capability to integrate several different collision avoidance algorithms that plan the runtime trajectory of each individual UAA. Such architecture supports operation of the group of cooperative UAA within the environment hosting other non-cooperative flying objects (e.g., civilian air traffic or manned aircrafts in the same area).

Cooperative collision avoidance, the deconfliction process between two or more interacting and cooperating aerial assets, is based on using different collision metrics [6] and negotiation protocols. Recently, the centralized solution has been replaced by various distributed approaches facilitating deployment of e.g., principled negotiation [16], Monotonic Concession Protocol (MCP) [19, 9] for collision avoidance in between of two assets [17] or extensions towards groups of multiple UAAs [13]. Such approach can be slightly altered to optimize social welfare instead of individual goals in the group of UAAs. There are also various approaches based on the game theory (e.g., [5]) available in the research community.

Optimization of non-cooperative collision avoidance algorithms (deconfliction process of an individual aircraft facing a non-cooperative, possibly hostile object) [14, 3] allows optimal solving of the collision with a non-cooperative flying object (obstacle). These algorithms perform well when coping with a single alien flying object, but they cannot be extended to a situation with several flying objects, located nearby. Moreover they cannot be used simultaneously with other cooperative algorithms applied for the cooperative collisions at the same place. The research work

reported in this contribution was motivated by designing such a non-cooperative collision avoidance method that does not suffer from these weaknesses.

In this chapter we briefly present the architecture of the AGENTFLY system. The chief technical contribution of the presented work is, however, in the collection of agent-based collision avoidance methods and the flexible multi-layer deconfliction architecture allowing integration and run-time reconfiguration of the various collision avoidance approaches. We also discuss the properties of the presented algorithms on empirical data provided by large scale experiments and we present the testing scenarios.

2. AGENTFLY System Architecture

AGENTFLY is fully written in JAVA, AGENTFLY can be easily hosted on assets with different operating systems. The multi-agent system for flight modelling consists of several components, see Fig. 1. AGENTFLY system can be started on a single dedicated computer or distributed in computer clusters without any specific reconfiguration.

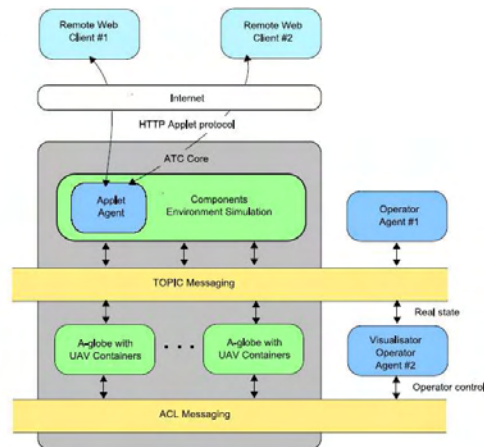


FIGURE 1. AGENTFLY System Structure Overview

Environment Simulation Components

The *components for environment simulation*, Fig. 2, of the AGENTFLY system is a sole central element of the system. It simulates positions of UAAs and other objects in the simulated world, aircraft hardware, weather condition, communication ranges given by the ranges of board data transmitters, etc. During the deployment of real UAA hardware these components will be removed and replaced by data

acquisition systems from real on board sensors hosted by UAA and the control link will be redirected to the UAA's actuators.

One of the environment simulation components is responsible for acquisition and fusion of information about all airplanes with freely available geographical and tactical data sources. These are provided to both the *remote WEB client* and the *operator agents*. There is also a *simulation scenario player* that controls the simulation flow by e.g., creating new UAAs and providing them initial mission specification (sequence of time specific way-points). Detailed information about environment simulation components can be found in [8]. For the very extensive simulations with hundreds of UAAs these components simulating the environment can be split among several servers integrated by means of the **A-globe** *topic messaging* concept.

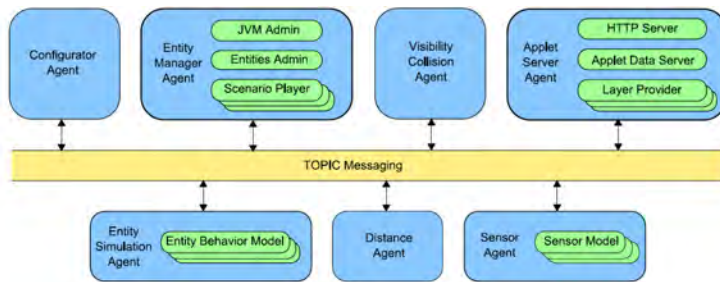


FIGURE 2. Components for environment simulation

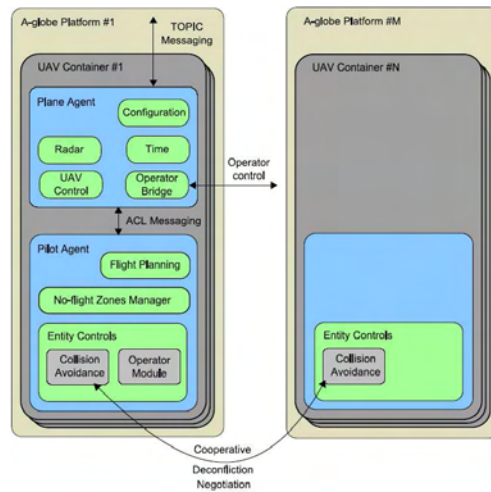
UAA Containers

Each airplane in the system is represented by one *UAA container*, Fig. 3. The container hosts two agents. The *plane agent* provides high-level airplane functions such as flight plan execution in cooperation with the simulator component, radar and detector readings, airplane configuration, time synchronization and operator bridge interface. The *pilot agent* is the main control unit of the UAA. It has a flight planning module a no-flight zones manager, a multi-layer collision avoidance module and a human-agent interface.

Within one **A-globe** instance (one JVM process running **A-globe**) there can be one or more such UAA containers. Such configuration allows to perform very large simulations (scalability experiments) using more host computers. The AGENTFLY simulation component responsible for the UAA startup performs also load balancing when deciding where the new UAA container will be created. One of the future versions of **A-globe** will provide also dynamic load balancing using the concept of container migration.

Remote WEB Client

The *remote WEB client* is an optional component of the AGENTFLY, Fig. 4.

FIGURE 3. *A-globe* with UAA containers

It allows a remote user to display requested information which she needs. There is a secured authentication of the user. So each user can have different levels of information enabled. AGENTFLY has been integrated with real world data. External data are taken from public databases with various GIS data for the area of the United States – landsat images, state boundaries, airports, cities, highways and real civil traffic. The client connection is optimized to provide only needed data so it can be operated using slow network connection.

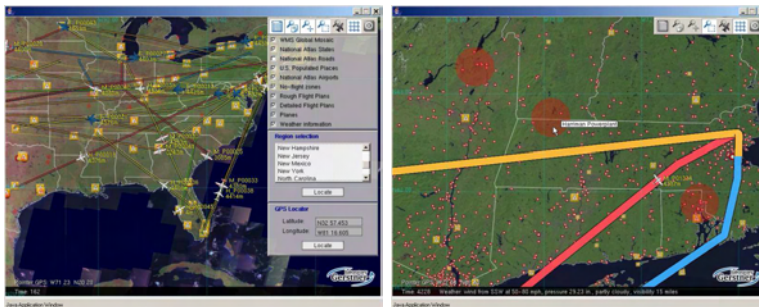


FIGURE 4. Web interface providing 2D view of the simulated area with integrated external data sources

Operator Agent

Real-time visualization of the internal system state in a 3D/2D environment is provided by the *operator agent*, Fig. 5. AGENTFLY allows running a number of such

agents who simultaneously provide different information with proper access level rights to different users. The operator agent is also able send the user commands back to the system or can be directly connected to the specific UAA container. The user can then manage the aircraft's way-point plan or change defined no-flight zones.



FIGURE 5. AGENTFLY system state provided in 2D (left) and 3D (right) view using operator agent.

3. Flight Planning in AGENTFLY

The inputs to the planner are (i) the list of waypoints (WP) (the coordinates that the airplane must visit in specific times), (ii) the velocity, which the airplane should have at the beginning of its route and (iii) repository with no-flight zones (NFZ), which the plane must not enter during its flight and plan the path to avoid them.

The planning of a flight path proceeds in two phases. In the first phase – *path planning*, the planner generates the shortest flight plan, which passes through all WPs avoiding the NFZs. In the next phase – *time planning*, some parameters of the segments (mostly speed, but sometimes also the trajectory) are adjusted in such a way that the modified flight plan satisfies the time constraints of the WPs.

NFZs represent an important concept for non-cooperative collision avoidance. The presented algorithms work with three types of no-flight zones used in the ATC system: (i) *world zones* – represent ground terrain and other static obstacles in the simulated world, (ii) *static zones* – encapsulate world areas where UAV cannot operate, e.g., enemy zones, and (iii) *dynamic zones* – hold zones which change frequently, they are mainly defined by NFZ-based non-cooperative deconfliction as described later. AGENTFLY supports various data structures for the no-flight zones: octant tree [2], height maps and primitive objects – sphere, cylinder and cube. All types can be combined together using grouping, scaling, translation and addition/subtraction operations on them.

The path planning problem has been solved by the original *manoeuvre-based path-finding* algorithm that is defined by two points (start, destination) and by two

vectors (initial direction, target direction). The manoeuvre-based algorithm incorporates a single A* progressive path planning [7] using basic flight plan elements with dynamic size of discrete steps.

Similarly to the algorithm of path planning, the time planning algorithm is based on particular elements' chaining. In case of path planning these elements were manoeuvres, in case of time planning these are time elements. In contrast to path planning, time elements are not expanded using algorithms for state space exploration (such as A*). Instead, they are only suitably chained one by one, with pre-calculated parameters. The time planning phases produce a plan for which the time of flight through each segment corresponds to the time constraints defined in the original planning problem specification.

4. Agent-Based Collision Avoidance Methods

The AGENTFLY system features a selection of different *cooperative collision avoidance* methods. Cooperative collision avoidance is a process of finding a mutually acceptable collision avoidance manoeuvre among two or more cooperating flying assets. The assets are capable of mutual interaction and provide each other fully trusted information. They are optimizing their own interests (e.g., fuel costs increase, delays) with consideration of the interests of the colliding assets. Even though the concept of cooperation in the field of multi-agent systems implies optimization of the social welfare (sum of costs and utilities of all the involved parties), we understand the concept of cooperation in broader sense. We have developed and studied three different cooperative collision avoidance algorithms: *Rule based collision avoidance* (RBCA), *Iterative peer-to-peer collision avoidance* (IPPCA) and *Multi-party collision avoidance* (MPCA).

Besides cooperative algorithms, the AGENTFLY system also features *non-cooperative collision avoidance* methods. The noncooperative collision avoidance algorithm operates an individual flying asset when facing a collision thread with a flying object that does not interact with the to-be-avoided asset or the asset is not trusted.

4.1. Rule-Based Collision Avoidance (RBCA)

RBCA is a domain dependent collision avoidance algorithm, which is based on the *Visual Flight Rules* defined by the Federal Aviation Authority (FAA)¹. Each flying asset performs one of the predefined collision avoidance maneuver by means of the following procedure. First, the type of the collision between the airplanes is identified. The collision type is determined on the basis of the angle between direction vectors of the concerned aircrafts projected to the ground plane. Depending on the collision classification each UAA applies the collision avoidance manoeuvre from the set of defined rules. The manoeuvres are parameterized that they uses

¹<http://www.faa.gov>

the information about collision and angle so the solution is fitted to the identified future conflict.

The above rule-based changes to the flight plan are done by both the assets independently because the second aircraft detects the possible collision with the first plane from its point of view. Substantial inefficiency of the RBCA algorithms is caused by the fact that the predefined visual flight rules perform collision avoidance without any altitude changes.

4.2. Iterative Peer-to-peer Collision Avoidance (IPPCA)

The *iterative peer-to-peer collision avoidance* algorithm is an extension of the pair optimization for multiple collisions among several UAVs based on utilities provided by themselves. The basic version provides a solution for a pair of colliding airplanes, see Fig. 6. The algorithm optimizes social welfare in that pair, thus the aircraft would like their flight plans to maximize the sum of their utilities, but still find collision-free paths.

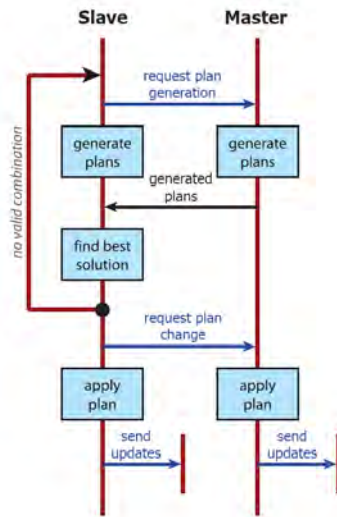


FIGURE 6. The negotiation during IPPCA

First, the participating airplanes in the colliding pair select the master and the slave part for the detected collision (usually the first entity which identifies a collision is regarded as a master entity). Each planning agent generates a set of new plans using the pre-defined parameterized collision avoidance manoeuvres. The flight plan modified by applying the manoeuvre includes its utility value which is composed as a weighted sum of several parts using the following equation:

$$u = \frac{\sum_i \alpha_i u_i}{\sum_i \alpha_i}, \quad (4.1)$$

where α_i denotes the weight for the i component of the utility function. The utility function is used for including the aircraft's intention in the proposed solution of the conflict and depending on the configuration it can contain different components to be taken into consideration, such as the total length of the flight plan, flight priority, fuel status and other factors. Seven parameterized changing manoeuvres can be used in the current version: *straight*, *turn left/right*, *turn up/down*, *speed up* and *slow down* changing manoeuvres. The parametrization is used during the generation process to obtain a wider range of solutions in a situation when the solution is not found using smaller changes.

The best possible solution is identified by the following algorithm. The master generates a combination of all the proposed plans (including the original one) and as a solution it selects a pair of plans for which the first collision occurs later than the collision currently being solved, and which has the best sum of the utility values of the plans used in the pair. When there are more pairs with the same sum value, the solution is selected randomly from these. Both sides (master and slave) then apply the selected solution. If there is no pair of plans fulfilling the condition that the first collision must occur later than the one being solved, it is necessary to generate more different flight plans using higher values of parameters describing the changing manoeuvres. The extension of the method for multi-collisions among several UAVs is in the iterative use of the described algorithm. To prevent infinite loops of iterations, the algorithm is restricted so that the UAV cannot generate such a change of the flight plan that would lead to a collision with an already de-conflicted aircraft occurring earlier than the currently solved collision. The second restriction is that an aircraft can apply only such changing manoeuvres that are not opposite² to those already applied within the same solving batch³.

The same algorithm has been used for finding the collision-free paths for self-interested UAAs. Such assets do not optimize the social welfare but they try to reduce the loss from collision avoidance. The best possible collision avoidance pair is identified by a variation of the *monotonic concession protocol* (MCP) [17]. The MCP is a simple protocol developed by Zlotkin and Resenschein for automated agent-to-agent negotiations [19, 9]. Instead of iterative concession on top of the negotiation set the algorithm uses the *extended Zeuthen strategy* [18, 4] providing negotiation equilibrium in one step and no agent has an incentive to deviate from the strategy. The implementation selects randomly one of the pareto-optimal solutions which maximize the product of the utilities.

²The opposite changing manoeuvres are defined in the three groups: turn left/right, turn up/down and speed up/slow down.

³Solving batch is a chain of consecutive algorithm runs.

4.3. Multi-Party Collision Avoidance (MPCA)

The multi-party collision avoidance approach removes the iteration known from the IPPCA algorithm during multi-collision situation – a situation when more than two UAAs have mutual future collision on their flight plans. MPCA introduces *multiparty coordinator* who is responsible for the state space expansion and searching for optimal solution of multi-collision. The multi-party coordinator is an agent whose role is to find a collision free set of flight plans for a possibly colliding group of UAAs - the *multi-party group*. The coordinator keeps information about the group, state space tree, chooses which airplane will be invited to the group, requests UAAs in the group for generating deconfliction proposals or sends the information about found non-colliding flight plans, see Fig. 7. Note that MPCA algorithm is running while planes are flying. Thus time for finding the solution is limited.

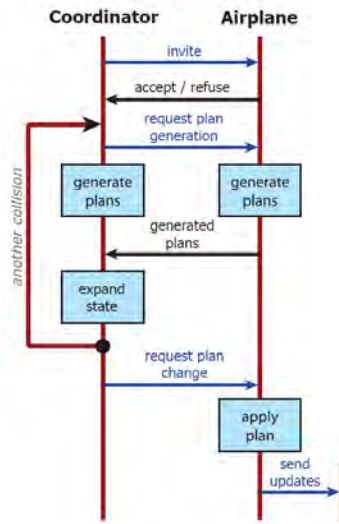


FIGURE 7. The negotiation during MPCA.

A coordinator agent is created by the master plane of the pair which detects future collision on their current flight plans. A master in the pair is determined according to alphabetical order of UAA's ids. When the coordinator is created it starts to search for a non-colliding set of flight plans. The collision which causes creation of a new coordinator is then treated as an *initial collision*. The searching algorithm proceeds in the three following steps which are repeated until a solution is found.

1. **Step 1** – The coordinator requests two planes from the group for possible flight plan changes to avoid their collision. Each of these planes individually

generates a set of changes and sends the partial plans back to the coordinator. Additionally, UAA checks for each avoiding manoeuvre, if it is colliding with other airplanes not included in the coordination group and adds a notification about it to the response.

2. **Step 2** – When the coordinator receives possible changes, it expands state space making their cartesian combination. Depending on the notification about external collisions the coordinator decides whether to invite (to enter the group) new UAA not already in its group which is colliding with the received changed flight plan. In the current version, the size of the coordination group is not restricted.
3. **Step 3** – The coordinator searches through combinations of generated flight plans. If the coordinator decides that more variants are needed, it continues with step 1. If it finds the final solution (a set of non-colliding flight plans), it sends message to planes from the group with their new flight plans.

The main part of communication during a session between the coordinator and a UAA is shown in the Fig. 7. The session starts with the `invite` message sent from the coordinator to an airplane. The UAA can respond with an `accept` or a `refuse` message. If the invitation is accepted, the UAA is added to the coordinator’s multi-party group and the coordinator can request for plan generation when needed. If the parts of the flight plan that are used for detecting a collision are too short (the first collision point is identified but the last collision point cannot be found due to the short flight plan), the coordinator sends a request to the UAA for a longer part of its plan. Additionally a `failure` message can be sent in both ways. Failure can occur when the flight plan is changed for reason unrelated to the group, e.g., an asset can be forced to switch to a different type of a conflict avoidance method which can make the change of its actual flight plan causing a removal of the plane from the group. Another situation when a failure can happen is when an asset contained in one multiparty-group changes its flight plan due to its involvement in another multiparty-group .

The coordinator assumes that the initial plans in the group are not changed during the search. If flight plans change for some of the planes, such a plane is removed from the group. This plane can be added to the group upon new collision detection. This relates to the problem of concurrent existence of several coordination multi-party groups.

By default, the state space generated in the MPCA is searched by the A* algorithm with zero heuristics which is only admissible one [10] in our case – due to the allowed changing manoeuvres which do not change the utility value but can find a non-colliding solution. For example in the scenario where two UAAs have collision on their perpendicular flight plans and the utility value depends only on the flight plan length, the best solution is when one plane speeds up and another slows down having the same utility values as the initial state. There is a defined condition (too large state space and not enough time for searching optimal solution) when the used heuristics need to be switched to the non-admissible heuristic preferring

expansion of states with less collisions. Such searching is then very fast, but its result is the less optimal solution.

4.4. Non-Cooperative Collision Avoidance Architecture

The path planning algorithms are used for planning individual flight plans (based on WP and NFZ) but also for non-cooperative collision avoidance replanning that is initiated once an aircraft detects a flying object. AGENTFLY uses the algorithm based on modeling the future possible trajectory of the opponent (the non-cooperative flying object) and dynamic encapsulation of its possible location by NFZ. NFZ is regularly updated after each radar update. Such implementation can be combined with another cooperative collision avoidance algorithm which uses the same identified dynamic NFZ.

The event that triggers the collision avoidance loop is information obtained from the radar describing the position of an unknown object in the area. This object is recorded in the base of non-cooperative objects, unless it's already present there. If the object is already known, its position is updated.

The next step is *prediction of the collision point*, an intersection of the flight plan and the non-cooperative object. If no such virtual collision point is found, the loop ends. In the opposite case, the collision point is wrapped by a dynamic no-flight zone. Such zone is then used for the test if the current flight plan intersects the zone and if the intersection is found, the path is re-planned.

The shape of dynamic no-flight zones of non-cooperative objects (Fig. 8) is derived from the possible future flight trajectory. The trajectory takes into account the minimal turning radius, maximal climbing and descending angle and the prediction time. We do experiments with dynamic NFZ with an ellipsoid shape which is not placed at the position of the observed flying object but at the place of the predicted collision and its size reflects the speed of the objects and the distance to the collision point.

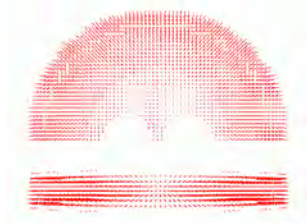


FIGURE 8. Shape of the dynamic no-flight zone

The described non-cooperative collision avoidance loop is executed for all objects found in the radar scan. This is done periodically for each radar scan.

5. Multi-layer Collision Avoidance Architecture

The listed collision avoidance methods are linked by the *multi-layer collision avoidance module* [13] that is a part of a special planning agent, hosted by each of the UAA platforms. This module is capable of solving the 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. The proposed modular architecture is domain independent. Therefore it is ready for deployment on autonomous vehicles like airplanes (UAA) or ground vehicles (UGV).

Multi-layer collision avoidance module hosts CSM (*Collision Solver Manager*), the main controller responsible for the selection of the CS (*Collision Solver*) that will be used for specific collision. CSM is able to combine all the available cooperative and noncooperative algorithms. The previously presented collision avoidance algorithms are implemented as plug-in solver modules and can be domain dependent or independent. Each collision solver is responsible for the collision detection (e.g., *Collision Point Prediction* in the non-cooperative CA or *Collision Detection* in cooperative CA) and collision registration with CSM. One collision can be detected by several collision solvers.

Based on priority, CSM assigns each registered collision solver a time slot that can be used for solving by the specific CS. The priority of the solvers is preset, but can be altered during the runtime. Concatenation of these slots creates time axis providing a specific, time-oriented switching among the CS operation. Sophisticated switching of the collisions solvers is inevitable in our application as the solvers have different properties. Different solvers provide different quality of the collision-free solution, while they require different amount of time for finding such solution. Specifically, the negotiation oriented solvers may provide better solution than non-cooperative solvers, while they may be more time consuming (given by the multi-party interaction). As the time is a very critical factor in our collision avoidance domain, some solvers are not guaranteed to terminate prior a possible collision.

6. Deployment Scenarios and Selected Experimental Results

In this section, the selected deployment scenarios, where presented algorithms were tested, are listed. The main criteria that the algorithm is stable and converges in many testing setups to the final solution is fulfilled by all the tested collision avoidance methods.

In the first scenario, the UAAs are located in a circular formation at the same flight level (altitude, referred to as FL). All of them want to fly to the opposite side of the circle through its center. Therefore there is a multi-collision of all the planes at the same time located in the center of the circle. The results after using distributed rule-based (RBCA) and iterative peer-to-peer collision (IPPCA) avoidance methods are shown in the Fig. 9. The RBCA has defined rules which

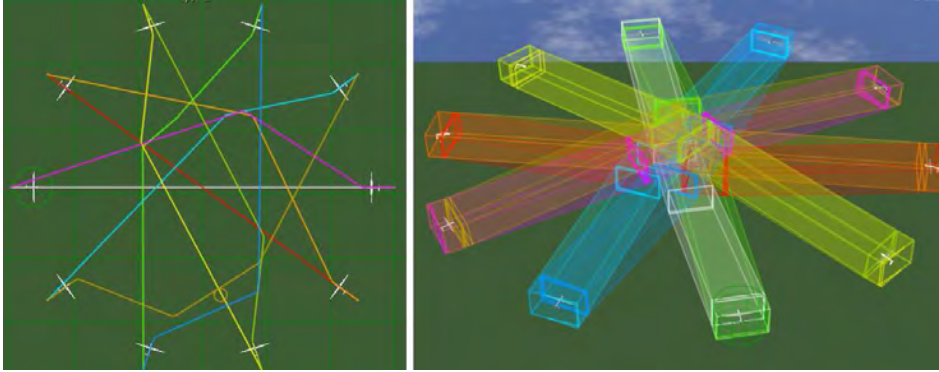


FIGURE 9. Scenario with 10 UAAs in the circle: The result for RBCA 2D view (left) and IPPCA 3D view (right).

change the flight plan only in the same flight level. The result for RBCA is still at the same FL. The IPPCA provides a solution that is substantially closer to the optimum (almost 100 times shorter additional flight trajectory for the scenario with 80 aerial assets - see Fig. 10. It uses six available avoiding manoeuvres (as defined in the section 4.2). The graphics in Fig. 9 demonstrate higher compactness of the IPPCA solution in comparison to the solution provided by RBCA.

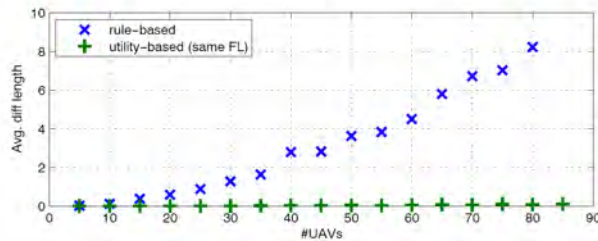


FIGURE 10. Scalability experiment: The comparison of the difference in the final flight plan length for the RBCA and IPPCA

We have defined an automatized experimental scenario setup which is used for scalability testing. During the experiment run several characteristic properties can be recorded and compared. The experimental environment uses the worst-case setup. The UAAs are randomly generated on one of the four sides of the limited experiment area. All of them need to fly to the opposite side. The entry point of each new UAA is generated on adjacent borders in clock-wise direction. Both the initial way-points entry and the exit are at the same FL. The setup provides the high number of the future collisions in the central part of the testing square. The plot in the Fig. 10 presents comparison of average (each setup was measured 50 times) sum of differences between the final deconflicted and the initial flight plan

for all the UAAs in the specific experiment run. For this experiment, IPPCA can use only four avoiding manoeuvres which do not change the flight altitude. The IPPCA provides solution much better than RBCA especially for the cases with more UAAs.

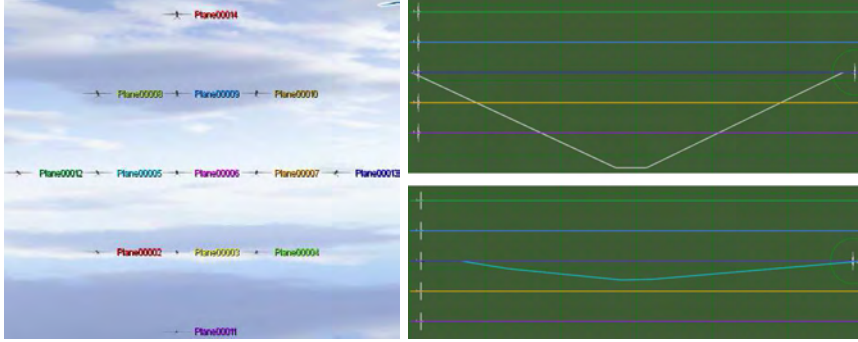


FIGURE 11. Wall scenario: Left: the setup of the experiment. Right: the result after IPPCA (top) and MPCA (bottom).

In the next scenario, there are 13 UAAs flying in a vertical plane in 3D. Their positions in the space are shown in the Fig. 11 left. All of them initially fly in the same direction at the same flight speed. There is another UAA which is flying in the opposite direction and has a potential to head collision with the UAA located in the middle of the first group. The final results comparing the IPPCA and MPCA algorithms are depicted on the right side of the same Figure. In the IPPCA version, only one plane performs a large detour of the group of UAAs and therefore no other planes participate in the solution. On the other hand, in the MPCA version, the middle UAA in the group performs a combination of several changing manoeuvres and creates a small hole in the middle of the group to let the opposite UAA fly through. Then the UAA goes back to its original relative (central) position within the group. The MPCA solution gives only 0.213 units longer solution than the initial plan while the IPPCA solution is 2.843 longer. The values were calculated as an average from 20 consecutive experiment runs.

The Fig. 12 displays the scalability experiment results comparing the IPPCA and MPCA algorithms. In this case both algorithms uses the same scenario setup and the same set of avoidance manoeuvres. Each run was measured 50 times to provide average result values for both the methods. The top plot in the Fig. 12 is the comparison of the final solution lengths. The MPCA algorithm gives a more optimal solution – depending on the number of UAAs, the results are improved by 10 to 50 percent compared to the IPPCA. The bottom chart is the comparison of total communication flow during the experiment run among all the UAAs. Both algorithms have almost the same amount of transmitted bytes, but we have identified that the flow distribution in time is different. The MPCA algorithm

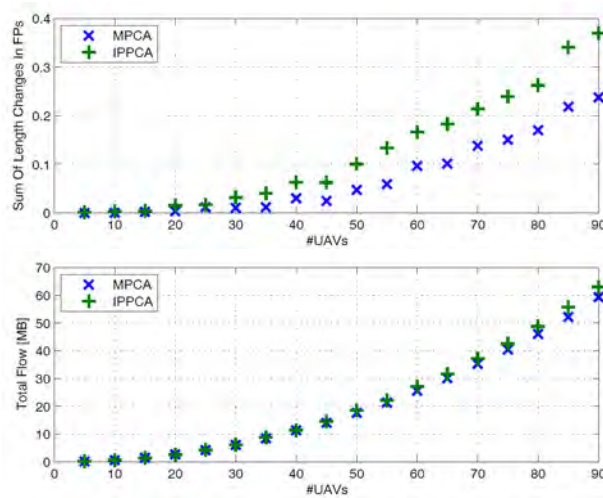


FIGURE 12. Scalability experiment IPPCA vs. MPCA: The sum of differences between final collision-free paths and the shortest regardless collisions (top). The total communication flow (bottom).

in the current version requires communication link bandwidth up to 600 kB per second and the IPPCA needs only 50 kB/s.

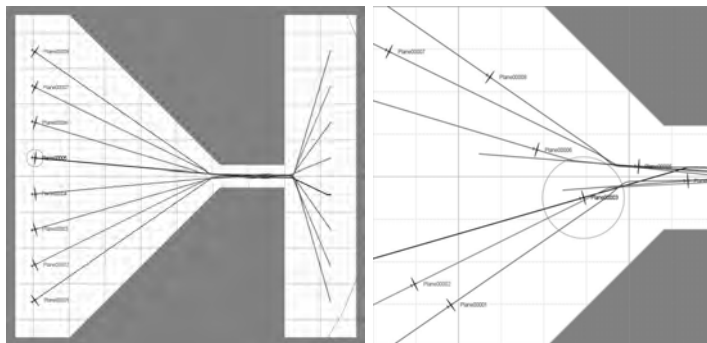


FIGURE 13. Tunnel scenario: Initial configuration (left) and the collision avoidance result for IPPCA (right).

Another high density scenario is shown in the Fig. 13. There are eight UAAs willing to fly from their starting positions through the small tunnel (the size of safety zone around UAA is the same as the size of the hole) to their destination on the right side. All UAAs fly at the same FL and they cannot manoeuvre and avoid collisions by changing this altitude. Both algorithms IPPCA and MPCA solve

this situation. UAAs change their flight speed to accelerate before the hole and then they adjust the flight speed that all of them have the same flight speed. The result is shown in Fig. 13 right screen. Although UAAs can use the right and left manoeuvres they do not apply them because there is not enough space there. This case can be compared to a very intensive landing scenario, where several airplanes want to land at the same runway in a very short time interval, e.g., landing at an aircraft carrier.

type	average trajectory length $l[u]$		
	proportional navigation	dynamic NFZ	no control
<i>perpendicular collision</i>	33,21	33,85	30,0
<i>slant collision</i>	31,17	31,52	30,0
<i>head-up collision</i>	30,81	30,62	30,0

TABLE 1. Non-cooperative experiments with one controlled and one uncontrolled UAA comparing proportional navigation with the dynamic no-flight zones algorithm.

The non-cooperative algorithm (Section 4.4) has been compared with the optimization proportional navigation (PN), see [13] for more details. The PN algorithm provides a very good result when coping with a single alien flying object, but it cannot be extended to the situation with several non-cooperative objects located nearby. Moreover such optimization algorithm cannot be used in a combination with other cooperative methods at the same place. In the Table 1, there are the results of non-cooperative experiments with one controlled and one uncontrolled UAA. The uncontrolled UAA always flies directly from the starting point to the destination and the controlled one is always heading north and it must avoid collision in the middle of the operation area. Again no altitude changes are allowed to provide relevant comparison (PN doesn't support such changes). The results for both algorithms are almost the same. The PN provides better results in two setups and in the third one the dynamic NFZ gives a more optimal solution.

The Fig. 14 displays the minimal separation among UAAs with safety zone size highlighted in the scenario with two uncontrolled UAAs (obstacles). The PN was configured to take into consideration the nearest obstacle first. The PN algorithm fails to avoid the collision while the dynamic NFZ works properly in this situation. We have performed several other experiments with more UAAs, all using the described non-cooperative algorithm within a worst-case scenario. The method handles all situations without any collision.

The multi-layer collision avoidance architecture allowing combination of cooperative and non-cooperative methods at the same time has been validated in the deployment where real civil traffic operates over Los Angeles International

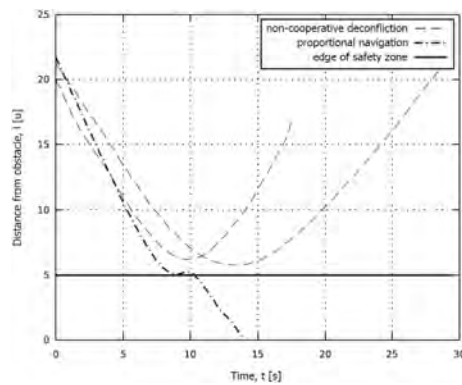


FIGURE 14. The distance from obstacle in the scenario with three UAAs, where only one uses active control non-cooperative avoidance method.

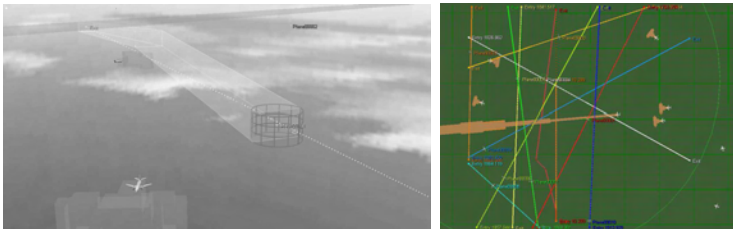


FIGURE 15. Operation of agent-controlled UAAs over LA with imported real air-traffic

Airport. There are two types of aircrafts in the setup. There are randomly operating agent-controlled UAAs which are configured to use the IPPCA algorithm with other UAAs controlled by agents. If there is another flying object identified they will use non-cooperative avoidance methods to solve the identified collision with it.

The simulated air-traffic is extended by real civil airplanes of which the dynamic positions are imported from publicly available internet sources. These airplanes are detected by on-board radars of the agent-controlled UAAs. The Fig. 15 provides both the 3D and the 2D view of the simulated area.

7. Conclusion

This chapter describes a possible use of agent technologies in the free-flight collision avoidance domain. We present a complex multi-agent system that provides a realistic air traffic simulation and a practical collection of different collision avoidance methods. A sophisticated switching architecture allows autonomous, decentralized,

run-time selection of an appropriate deconfliction method based on current distances and velocities, traffic density and level of trust between the to-be-avoided closing objects. Extensive experiments demonstrated viability, efficiency and scalability of this approach.

The experimental verification of the efficiency of the negotiations among highly distributed autonomous assets did show clear convergence of all the proposed algorithms. Given the results shown in the previous section, it is clear that the rule based approaches are obsolete given the demonstrated performance of the iterative peer-to-peer negotiation protocols. However, the difference in performance between these and the multi-party collision avoidance is less evident.

All the scenarios presented as test cases here represent just the starting point of investigation of more complex cases where more UAAs and civil aircraft will be engaged, with a wider variety of permitted maneuvers, with sub-optimal communications and with more dramatic changes in the environment. This approach is inherently less constrained, and may allow for unanticipated emergent behaviors to arise, hence our emphasis on empirical testing. However, we feel we have clearly documented that our approach is robust and efficient, and definitely worthy of continued development.

The main potentials for further extensions of the AGENTFLY technology include e.g., collision avoidance among the collaborating assets with limited communication capabilities (so that the location of the asset is kept undisclosed), introduction of the mobile stand-in agents in the collision avoidance process (in order to minimize the required communication traffic), use of the collision avoidance architecture for implementation of various collective flight models or theoretical analysis of the convergence of the listed communication protocols.

Acknowledgement

AGENTFLY development has been sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3073. *A-globe* has been mainly developed under the grant numbers FA8655-04-1-3044 and FA8655-02-M-4057. 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 Air Force Office of Scientific Research or the U.S. Government.

The authors wish to acknowledge help, assistance and inspiration of their collaborators Victor Skormin and John Beyerle.

Appendix A. *A-globe* Multi-Agent Platform

AGLOBE⁴ is a flexible and open multi-agent environment, which supports integration of heterogeneous distributed computational processes (agents). It is distributed under CPL licence⁵. *A-globe* is different from existing multi-agent platforms' by:

- SCALABILITY - its emphasis on high efficiency of the computational process allowing scalability of multi-agent simulations,
- SIMULATION - support for modeling and simulation in of the environment in which the agents have been designed to operate,
- MIGRATION - full support of migration of the agents and computational processes in distributed environment, and
- VIZUALIZATION - *A-globe* also provides sophisticated visualization support for design and testing of complex multi-agent systems.

A-globe is a fast and lightweight platform with agent mobility and inaccessibility support. Besides the functions common to most of agent platforms it provides Geographical Information System-like service to the user. Therefore, the platform is ideally suited for testing experimental scenarios featuring agents' position, position dependent environment simulation and communication inaccessibility. The platform provides support for permanent and mobile agents, such as communication infrastructure, storage, directory services, agent migration (including library migration with version handling), service deployment, etc. *A-globe* is optimized to consume just a limited amount of resources.

A-globe platform is not fully compliant with the FIPA specifications; still it implements most protocols and respects the spirit of the specification. It does not support communication between different agent platforms (e.g., with JADE, JACK, etc.). For large scale scenarios the problems with system performance that interoperability brings (memory requirements, communication speed) outweigh any advantages, as heterogeneous environment is of limited interest for simulations. The *A-globe*'s operation is based on several core components, see Fig. 16:

- Agent Platform that provides the basic components necessary to run one or more agent containers, the container manager and the library manager.
- Agent Container that is a skeleton entity that provides basic functions such as communication, storage and management for agents.
- Services that provide shared functions for all agents in one container
- Environment Simulator Agents that simulates the real-world environment and controls visibility among other agent containers
- Agents who integrate various computational processes, user interface or hardware components and represent basic functional entities in a specific simulation or control scenario.

⁴<http://agents.felk.cvut.cz/aglobe/>

⁵Common Public License Version 1.0 – <http://www.opensource.org/licenses/cpl1.0.php>

The agents have various means of communication. They can interact either via (i) standard ACL (Agent Communication Language) message passing, (ii) by topic messaging - specific simulation oriented messaging among containers, (iii) service sharing - where the agents can use or provide each other with various specific services.

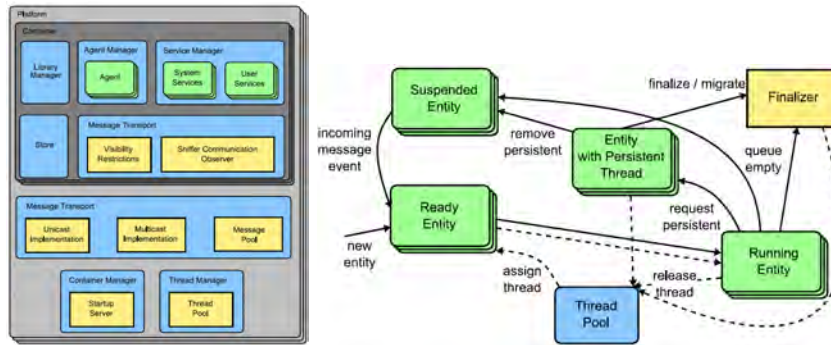


FIGURE 16. *A-globe* platform architecture (left). Agent lifecycle (right).

The *A-globe* platform is primarily aimed at large scale, real world simulations with fully fledged agents. To support this goal, it includes a special infrastructure for environmental simulation. Actor agents play roles in the simulated world, while Environment Simulation (ES) agents implement the simulated world itself. ES agents only rarely use messages to communicate with actor agents. Instead, they communicate via topic messaging. Topic messaging implements container to container messaging reserved for easy environmental simulation. Topic messaging is built on top of standard messages and is managed by the Geographic Information System (GIS) Services - server and clients. GIS services provide distribution and subscription mechanism for the agents. ES agents can be responsible for nearly any simulation layer, depending on the wishes of the developers. The accessibility agent, which controls the availability of communication links between containers holding the actor agents, is one of the most important of ES agents. *A-globe* messaging layers use the information provided by the accessibility agent to prevent sending messages between inaccessible nodes. The accessibility simulated by the system can depend on many factors, typically including the distance and simulated link reliability.

A-globe has been developed in the Gerstner Laboratory, Czech Technical University. Since its initial development in 2001 *A-globe* has been successfully deployed in various industrial domains. Besides its chief deployment in Air Traffic control of unmanned aerial vehicles, described in this chapter, *A-globe* has been also used for the simulation of underwater minesweeping operations, funded by the Office of Naval Research. Agents perform collaborative decision making aimed at intelligent surface search and sharing communication bandwidth when

streaming high resolution images to the control base. The simulation has been successfully transformed to the robotic environment in order to prove its versatility. The robocup soccer robots were used for the hardware simulation of **A-globe** based underwater minesweeping operation. Based on **A-globe** the Gerstner Laboratory developed in cooperation with DENSO Automotive, GmbH an agent based system for distributed diagnostics of on-board car electronics. The model has been used for an operation failure root-cause-detection while also for the process of graceful degradation of the systems operation (for which safe regions of car electronics need to be dynamically identified). US ARMY CERDEC are currently supporting agent-based modeling of large scale computer networks, based on **A-globe** multi-agent environment. Besides modeling, **A-globe** is here used as an integration platform for collaborative intrusion detection and prevention application. The company CADENCE Design Systems are using **A-globe** CE (CADENCE EDITION) for simulation and modeling of the chip design process. They use multi-agent simulation for analysis and measurement of the performance and efficiency of their production processes.

Besides the listed industrial companies, **A-globe** is used also by several academic institutions: University of Edinburgh, Florida Institute for Human and Machine Cognition and Masaryk University.

References

- [1] DOD. Unmanned aircraft systems roadmap 2005-2030, 2005.
- [2] S. Frisken and R. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, 7(3), 2002.
- [3] S. C. Han and H. Bang, "Proportional navigation-based optimal collision avoidance for uavs," in S. C. Mukhopadhyay and G. Sen Gupta, editors, *Second International Conference on Autonomous Robots and Agents*, pages 76–81. Massey University, New Zealand, 2004.
- [4] J.C. Harsanyi, "Approaches to the bargaining problem before and after the theory of games: a critical discussion of zeuthen's, hick's, and nash's theories," *Econometrica*, (24):144–157, 1956.
- [5] J. C. Hill, F. R. Johnson, J. K. Archibald, R. L. Frost, and W. 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.
- [6] J. Krozel, M. Peters, K. D. Bilimoria, C. Lee, and J. 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.
- [7] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., 1971.
- [8] M. Pěchouček, D. Šišlák, D. Pavlíček, and M. Uller, "Autonomous agents for air-traffic deconfliction," in Peter Stone and Gerhard Weiss, editors, *Proceedings of the*

- Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1498–1505. ACM, 2006.
- [9] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter*. The MIT Press, Cambridge, Massachusetts, 1994.
 - [10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence, Englewood Cliffs, New Jersey, 1995.
 - [11] T. Sandholm, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter Distributed Rational Decision Making, pages 201–258. MIT Press, Cambridge, MA., 1999.
 - [12] D. Šišlák, M. Reháč, M. Pěchouček, M. Rollo, and D. Pavlíček, “**A-globe**: Agent development platform with inaccessibility and mobility support,” in Rainer Unland, Matthias Klusch, and Monique Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.
 - [13] D. Šišlák, P. Volf, A. Komenda, J. Samek, and M. Pěchouček, “Agent-based multi-layer collision avoidance to unmanned aerial vehicles,” In James Lawton, editor, *Proceedings of 2007 International Conference on Integration of Knowledge Intensive Multi Agent Systems*, volume KSCO 2007. IEEE, IEEE, 2007.
 - [14] C. Tomlin, G. Pappas, and S. Sastry, “Conflict resolution for air traffic management : A study in multi-agent hybrid systems,” 1998.
 - [15] D. Šišlák, M. Reháč, M. Pěchouček, D. Pavlíček, and M. Uller, “Negotiation-based approach to unmanned aerial vehicles,” In *DIS '06: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 279–284, Washington, DC, USA, 2006. IEEE Computer Society.
 - [16] J. P. Wangermann and R. F. Stengel, “Optimization and coordination of multiagent systems using principled negotiation,” *Journal of Guidance, Control, and Dynamics*, 22(1):43–50, 1999.
 - [17] S. Wollkind, J. Valasek, and T. R. Ioerger, “Automated conflict resolution for air traffic management using cooperative multiagent negotiation,” in *Proc. of the American Inst. of Aeronautics and Astronautics Conference on Guidance, Navigation, and Control*, Providence, RI, 2004.
 - [18] F. L. B. Zeuthen, *Problems of monopoly and economic warfare*. Routledge and Sons, London, UK, 1930.
 - [19] G. Zlotkin and J. S. Rosenschein, “Negotiation and task sharing among autonomous agents in cooperative domains,” in N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 912–917, San Mateo, CA, 1989. Morgan Kaufmann.

David Šišlák
Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics
e-mail: sislak@fel.cvut.cz

Michal Pěchouček
Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics

Přemysl Volf

Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics

Dušan Pavlíček

Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics

Jiří Samek

Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics

Vladimír Mařík

Gerstner Laboratory, Agent Technology Group
Czech Technical University, Department of Cybernetics

Paul Losiewicz

European Office of Aerospace Research and Development
Office for Scientific Research, US Air Force Research Laboratory
e-mail: paul.losiewicz@london.af.mil