AGENTFLY: NAS-WIDE SIMULATION FRAMEWORK INTEGRATING ALGORITHMS FOR AUTOMATED COLLISION AVOIDANCE

David Šišlák, Přemysl Volf, Štěpán Kopřiva, Michal Pěchouček,
Czech Technical University, Faculty of Electrical Engineering, Department of Cybernetics, Agent Technology Center, Czech Republic

Abstract

AgentFly is a software prototype providing a distributed architecture for large-scale NAS-wide simulation implemented as a multi-agent system. AgentFly is implemented on top of the Aglobe [1] platform which is both an implementation framework and a runtime engine for custom agents. It was selected over possible alternatives (e.g. JADE [2]) for its outstanding performance and scalability supporting seamless interaction among heterogeneous software, hardware and human actors. AgentFly system has been developed for over five years. It was initially built for simulation-based validation and comparison of various approaches for autonomous collision avoidance algorithms adopting the free-flight concept. Later, AgentFly has been extended with high-level control algorithms providing tactical control - i.e. coordination of several autonomous unmanned aerial vehicles (UAV). The same agents and algorithms integrated in AgentFly simulation are also deployed on real UAV platforms. Besides this UAV-related application, the U.S. Federal Aviation Administration (FAA) supports the application of the AgentFly system for simulation and evaluation of the future civilian air-traffic management system. AgentFly has been extended with high-fidelity computational models of civilian airplanes and a parallelization concept integrating dynamic load-balancing. The parallelized approach of AgentFly has been validated in simulation using data of a full civilian air-traffic touching NAS. Nowadays, AgentFly is being extended so that it provides a simulation of ATC functions for the NEXTGEN concept validation. There are being integrated ATC & NAS automation agents which are used to simulate human operation in ATM.

Agent – Based Architecture

The entities in AgentFly are implemented as software agents in the multi-agent platform Aglobe. Aglobe provides runtime environment for agents. The Aglobe platform has been selected as the platform for AgentFly project because it outperforms the other existing multi-agent platforms with its limited need for computational resources and efficient operation. Moreover, Aglobe facilitates modeling of communication inaccessibility and unreliability in ad-hoc networking environments.

In AgentFly there exist three different types of agents in AgentFly: (i) airplane agents, (ii) environmental simulation agents and (iii) visio agents. When AgentFly is started in the simulation mode, usually all three types of agents are used. On the other hand, when AgentFly is running directly on a real UAV platform, only airplane agents are running (one airplane agent per one UAV platform) and an actuator control and sensing perceptions are mapped to real hardware. The detailed description of the agents follows.

Airplane Agent

Each airplane in AgentFly is represented by one airplane Agent. This agent provides the unit control for the airplane in both modes – air traffic simulation and deployment on the real UAV. Intelligent algorithms for airplanes are integrated in this agent. Based on the configuration, the algorithms provide high-level functions like trajectory planning, collision avoidance see and avoid functionality and also autonomous coordination of a group of airplanes. AgentFly integrates algorithms providing a decentralized control approach. Thus, appropriate parts of algorithms are running in distributed manner within several airplane agents and they can utilize ACL messaging providing airplane-to-airplane communication channels.

Environment Simulation Agents

Environment simulation agents are used when AgentFly is started in the simulation mode. These agents are responsible for simulation of a virtual

978-1-4577-0592-2/11/$26.00 ©2011 IEEE

environment in which the UAVs operate. These agents simulate the actions which normally happen in the real world. They provide simulation of physical behaviors of virtual UAVs (non-real UAV platforms), mutual physical interactions (physical collisions of objects), atmospheric model (weather condition) influencing UAVs' behaviors, communication parameters based on the used wireless simulator and simulation of non-UAV entities in the scenario (e.g. humans, ground units). Through simulation infrastructure, these agents provide sensing perceptions for UAV agents. Besides simulation, there exist simulation control agents which are responsible for the scenario control (initialization of entities, parameter setups, etc.) and for data acquisition and analysis of configured properties which are studied in a scenario. These agents are created so that they support large-scale simulations which are started in distributed manner over several computers connected by a common LAN network.

Visio Agents

Visio agents provide real-time visualization of the internal system state in a 3D or 2D environment. Based on a configuration, there can be displayed also many UAV-related information in various ways. In AgentFly, several Visio agents providing the same or different presentation layers from various perspectives can be connected simultaneously. The architecture of AgentFly automatically optimizes data collection and distribution so that the network infrastructure is optimally utilized. A Visio agent can be configured to provide HMI for the system, e.g. the user operator can amend the goal for algorithms.

Airplane Control Concept

The AgentFly control concept for both UAV control and real traffic simulation uses the layered control architecture, see Figure 1 and [3] The control method used in AgentFly is based on a complete flight trajectory description. The flight trajectory is the crucial structure which provides description of future airplane intentions covering also uncertainty while they are executed by UAVs. In AgentFly, it is supposed that the airplanes operate in a shared limited three-dimensional airspace called the operation space. Additional limits of the operation space are given by separation from the ground surface and by a set of no-flight areas which define a prohibited space. No-flight areas are also known as special use airspaces (SUAs) can be dynamically changed during the run-time. Besides the flight trajectory, there is used another crucial structure called the mission. The mission is an ordered sequence of way-points where each way-point can specify geographical and altitude constraints and combine optional constraints: time restrictions (e.g. not later than, not earlier than), the velocity restriction and an orientation restriction.

Flight Executor

The flight executor holds the current flight trajectory which is executed. The flight executor implements an autopilot function in order to track the request intentions in a flight trajectory as precise as possible. Such intelligent autopilot is known as the flight management system (FMS) in civilian airplanes.

Flight Trajectory Planner

The flight trajectory planner is a sole component in the control architecture which is responsible for creation of all flight trajectories. Planning can be viewed as the process of transformation of a way-point sequence to the detailed flight intent description considering UAV model restrictions (flight dynamics) and the airspace definition.

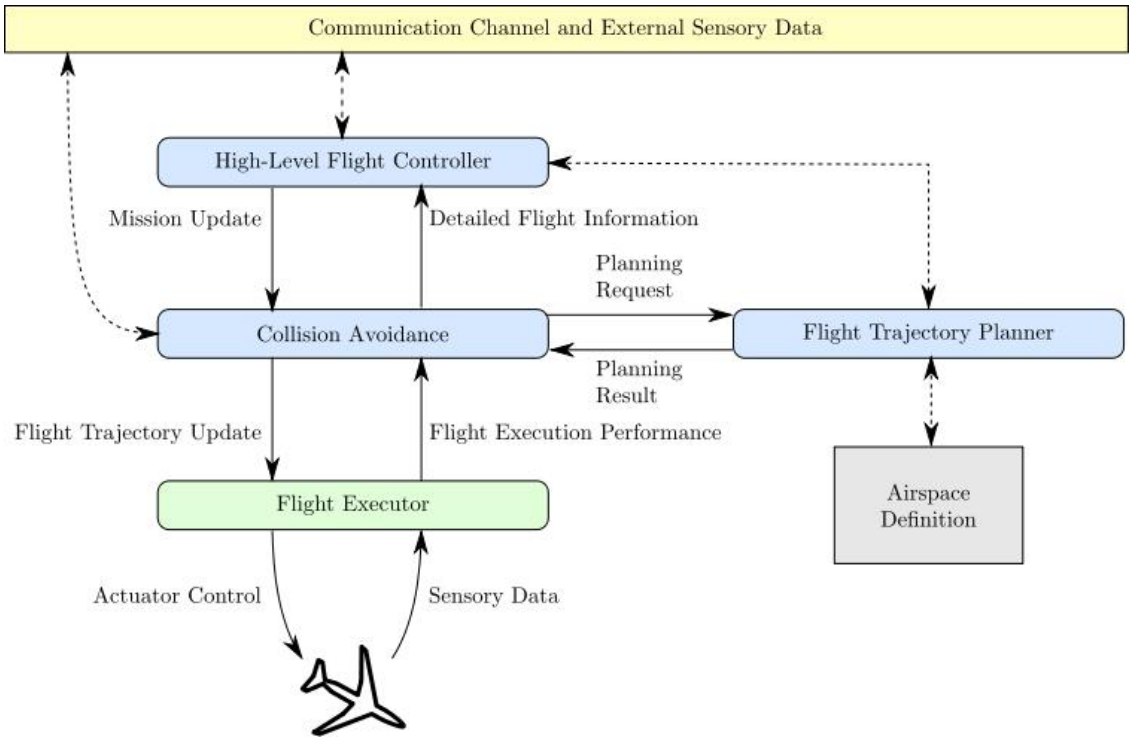

Figure 1. Airplane Control Concept

Collision Avoidance

The collision avoidance component is responsible for implementation of the sense and avoid function. In AgentFly, the method using additional control way-points which are inserted into the current mission way-point sequence is used. These control way-points are injected so that the final trajectory is collision-free with respect to other UAVs or piloted airplanes operating in the same airspace. The collision avoidance component chooses the appropriate collision modification with respect to the selected airplane preferences and optimization criterion. Each considered modification is transformed in the flight trajectory utilizing the flight trajectory planner.

All three algorithms for collision avoidance we have developed are utilizing the decentralized approach based on the free-flight concept – the aircrafts can fly freely according to its own priority but still respect the separation requirements from others in its neighbor. It means that there is no centralized element responsible for providing collision-free flight trajectories for UAVs operating in the same shared airspace. Intelligent algorithms integrated in this component can utilize a communication channel provided by on-board wireless data modems and sensory data providing information about objects in its surrounding (a large UAV platform can be equipped with an on-board radar system, a smaller one can utilize receivers of transponders' replies or receives radar-like data from a ground/AWACS radar system).

Collision avoidance implements the selected flight plan by passing the flight trajectory update to the flight executor. Collision avoidance utilizes the flight execution performance (uncertainty in the flight execution) for adjusting the algorithm separation used while searching for a collision-free trajectory for the aircraft. Collision avoidance monitors modification of the aircraft mission coming from upper layer and also detects the execution performance which is out of the predicted one in the currently executed flight trajectory. In such a case, collision avoidance invokes re-planning with new tolerances and conflict detection and separation processes are restarted with the new condition.

High-Level Flight Controller

The high-level flight controller provides a goal oriented control for the UAVs. This component includes intelligent algorithms for the group coordination and team action planning (assignment of

the specific task to the particular UAV). Depending on the configured algorithm, the high-level flight controller utilizes the communication channel, sensory perceptions (e.g. preprocessed camera inputs) and the flight trajectory planner to decide which tasks should be done by UAV. Tasks for UAV are then formulated as a mission which is then passed to the collision avoidance component. During the flight, the high-level flight controller receives updates with the currently executed flight trajectory including modification caused by collision avoidance. The high-level flight controller can identify that properties of the flight trajectory are no longer sufficient for the current tasks. In such a case, the high-level flight controller invokes algorithms to re-negotiate and adjust task allocations for UAVs and thus modify the current UAV mission.

If no high-level algorithm is used, there can be used only a simple implementation of this component which just provides one initial flight mission for UAV composed as a sequence of way-points from a departing position, flight fixes (where UAV should fly through) and a destination area where it has land.

Collision Detection and Resolution Algorithms

In this section we give description of major algorithms for autonomous collision avoidance in AgentFly. These algorithms adopt the decentralized autonomous approach based on the free-flight concept (R. Schulz, 1997). In the free-flight approach, the aircrafts fly freely according to their own priorities respecting the separation from others implementing the autonomous sense and avoid capability. In other words, there is no central element providing collision free flight paths for UAVs operating in the shared airspace. The collision avoidance algorithms in AgentFly consider the non-zero time required for the search for collision-free flight trajectories. While a collision avoidance algorithm is running or performs flight trajectory re-planning, UAV is still flying and the time for the flight trajectory change is limited.

In AgentFly, there are two collision detection and resolution approaches: (i) cooperative and (ii) non-cooperative. The cooperative collision avoidance algorithm is a process of detection and finding a mutually acceptable collision avoidance manoeuver among two or more cooperating flying UAVs. It is supposed that UAVs are equipped with communication modems so that they can establish bi-directional data channels if they are close each other. UAVs don't have any common knowledge system like a shared blackboard architecture and they can utilize only information which they gather from own sensors of from negotiation with other UAVs. For the simplification of description in this chapter, we will suppose that UAVs provide fully trusted information. Based on the configuration, they are optimizing their own interests (e.g. fuel costs increase or delays) or the social welfare (the sum of costs of all involved parties) of the whole UAV group. On the other hand, the non-cooperative collision avoidance algorithm cannot rely on the bi-directional communication and any background information about algorithm used by other UAVs. Such algorithm is used when the communication channel cannot be established due to malfunction of communication modems or due to incompatible cooperative systems of considered UAVs. A non-cooperative algorithm can work only with information provided by sensors providing radar-like data.

Multi-Layer Collision Avoidance Architecture

In AgentFly the collision avoidance component is represented by the complex architecture called the Multi-layer collision avoidance framework, see Figure 2. The architecture is capable to detect and solve the future collisions by means of combination of variant collision avoidance methods. It provides robust collision avoidance functionality by the combination of algorithms having different temporal requirements and providing different quality of the solution. It considers the time aspect of algorithms. Based on the time remaining to the earliest collision, it chooses the appropriate method for its resolution. The architecture is modular and from its nature is domain independent. Therefore it can be used for deployment on different autonomous vehicles, e.g. UGV.

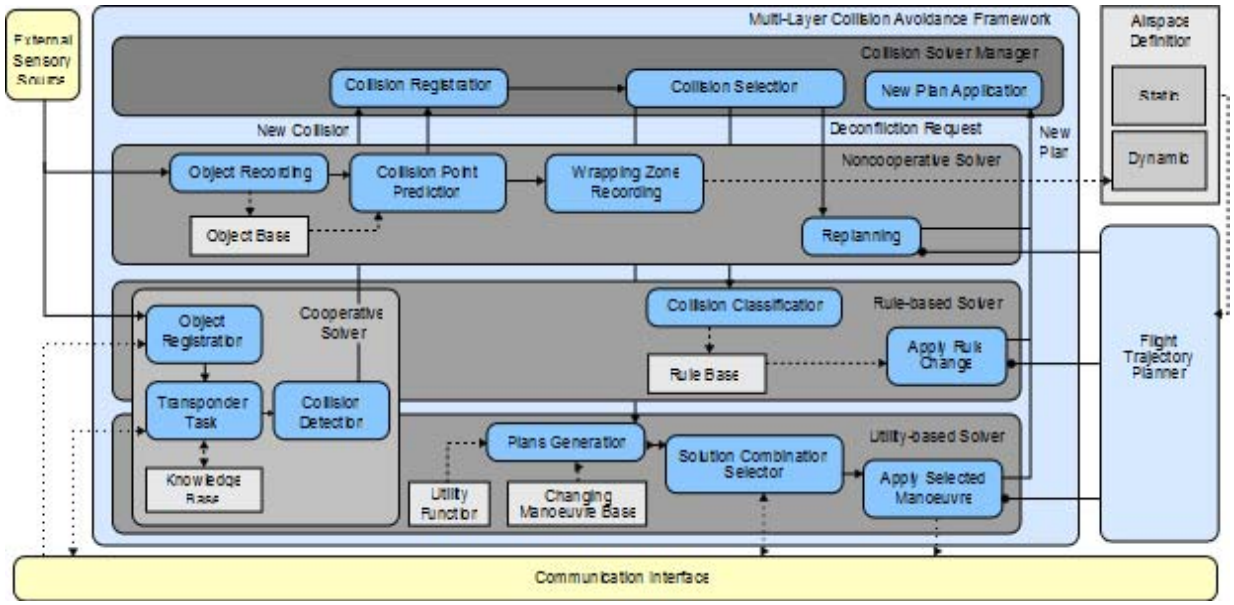

Figure 2. Collision Solver Manager Architecture

The collision solver manager (CSM) is the main controller responsible for the selection of a solver module that is used for the specific collision. Each solver module has a detection part which is responsible for detection of a potential future collision. In the case of cooperative solver module, this detector uses flight intent information which is shared among UAVs locally using data channels. In case of non-cooperative solver module, this detector is based on a prediction of the future trajectory based on the observation from radar-like data of its surrounding area. Each potential future collision is then registered within CSM. The same collision can be detected by one or several collision solvers.

Cooperative Collision Avoidance

AgentFly integrates three core cooperative negotiation-based collision avoidance algorithms: (i) rule-based (RBCA), (ii) iterative peer-to-peer (IPPCA)and (iii) multi-party (MPCA) collision avoidance. All these three methods are decentralized and the collisions are solved as localized collision avoidance problems. The solution of the artificial scenario using the algorithms is presented in Figure 3. At the same time, there can be running several (also different) algorithms resolving various conflicts in different local areas. The local problem is restricted by the defined time horizon. In many our applications, we are using the 15 minutes time horizon which correlates with the mid-term collision detection and resolution known from the civil air-traffic management. Theoretically, this time horizon can be set to a very large value which implies that algorithms search for a global collision avoidance solution. All three methods use the same detector part which is based on sharing of UAVs' local intentions. These intentions are shared using subscribe-advertise protocol and are formed as limited parts of their trajectories. UAVs share flight trajectories from the current time moment for the defined time horizon. By acceptance of the subscribe protocol, each UAV makes a commitment that it will provide the update of this limited part of its flight trajectory once its current flight trajectory is modified (e.g. due to other collision or change in its mission tasks) or the already provided part is not sufficient to cover the defined time horizon. This shared information about flight intention is used only to describe the flight trajectory for that time horizon and doesn't contain any detail about future way-points and their constraints. So, even though UAV cooperatively shares its flight intention, it doesn't disclose its mission to others. Using this mechanism, UAVs build knowledge-base with local intentions of other UAVs in its surrounding. Every flight intention contains information about the flight execution performance (uncertainty) which is then used along with the separation requirement for the detection of conflicting situations (positions of any two UAVs are not satisfying the required separation distance in any time moment). This mechanism is robust, as cross conflicts are at least checked by two UAVs.

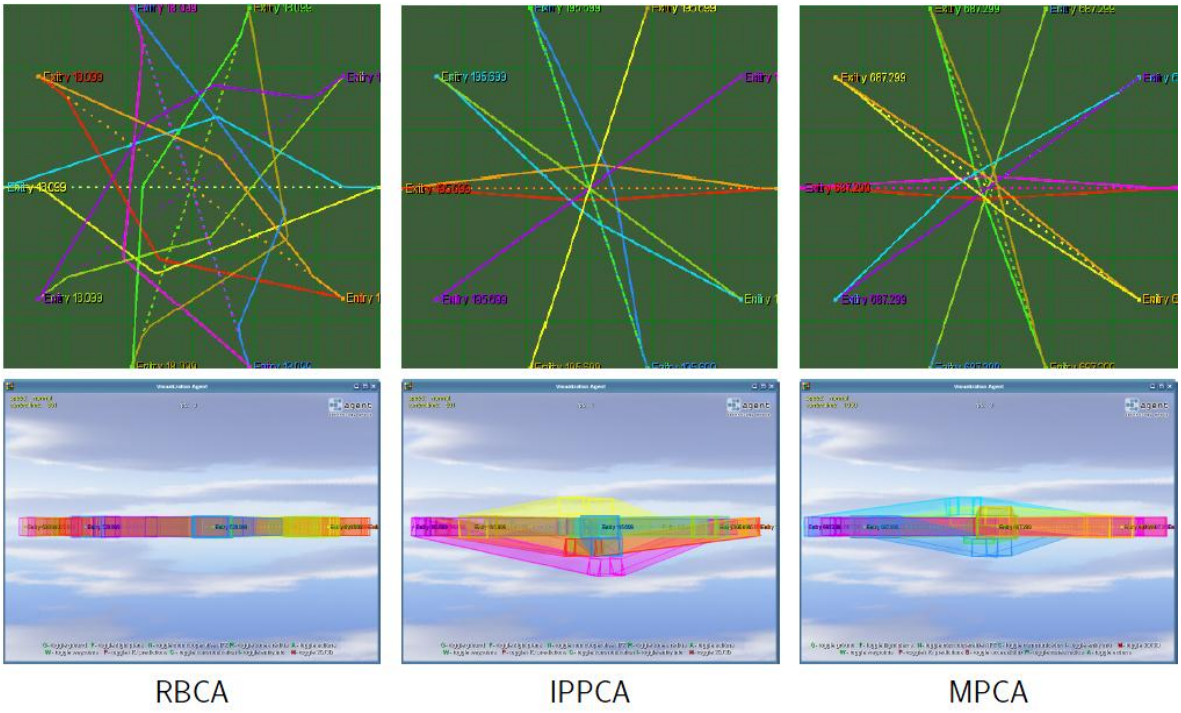

Figure 3. Comparison of Collision Avoidance Method Solutions

All three algorithms modify the flight path using the trajectory planner by definition of a re-planning task. The modification of the flight trajectory is given by a modification of way-point sequence in the request. There can be inserted new, modified existing or removed control way-points. Please note that collision avoidance cannot alter any way-point which is defined in the UAV mission defined by the high-level flight controller. There is defined a set of evasion maneuvers which can be applied to the specified location with the specified parameterization. Usually, an evasion maneuver is positioned using the time moment in the flight trajectory and its parameterization defines the strength of the applied maneuver. We define seven basic evasion maneuvers: left, right climb-up, descend-down, fly-faster, fly-slower and leave-plan-as-it-is evasion maneuvers. Maneuvers can be applied sequentially on the same place so that this basic set can produce all changes. Each UAV can be configured to use only sub-set of these maneuvers or can prioritize some of them. For example, UAV can be configured to solve conflicts only by horizontal changes (no altitude changes). The leave-plan-as-it-is maneuver is included in the set so that it simplifies all algorithms as they can easily consider the unmodified flight trajectory as one of the option.

Rule-Based Collision Avoidance (RBCA)

RBCA is motivated by the Visual Flight Rules defined by ICAO. Each UAV applies one of predefined collision avoidance maneuvers by means of the following procedure. First, the type of the collision between UAVs is determined. The collision type is identified on the basis of the angle between direction vectors of UAVs in the conflict time projected to the ground plane. Depending on the collision classification and defined rule for that collision type, each UAV applies the appropriate conflict resolution. Maneuvers are parameterized so that they use the information about collision and angle so that the solution is fitted to the identified conflict. Application of the resolution maneuver is done by both involved UAVs. This algorithm is very simple and doesn't require any negotiation during the process of the selection and application of the appropriate collision type solution. The algorithm uses only indirect communication via the updated flight trajectory.

Iterative Peer-to-Peer Collision Avoidance (IPPCA)

IPPCA extends the pair-wise conflict resolution with negotiation over a set of combinations

considering the cost values computed by involved UAVs. First, participating UAVs generate a set of new flight trajectories using configured evasion maneuvers and their lowest parameterization. Only those which do not cause an earlier collision with any known flight trajectory are generated. Each flight trajectory is evaluated and marked with the cost for its application. Then, a generated set of variants are exchanged (there are included also original unmodified trajectories). During this exchange, UAV provides only limited future parts of trajectories considering the configured time horizon which is the same as is used in the conflict detector. Then, the own set of variants and a received set is used to build combinations of trajectories which can be used to resolve the conflict. It is possible that no such combination is found as some variants are removed because they cause earlier conflicts with others and the rest do not solve conflict. In such a case, UAVs extend their sets of new flight trajectories with modifications with higher parameterizations until some combinations are found. The condition removing variants causing earlier collisions is crucial. Without this criterion the algorithm could iterate in the infinite loop and can also generate new conflicts which are so close that they cannot be resolved.

From the set of suitable combinations of flight trajectories, UAVs select the best combination based on the configured strategy. UAVs can be configured to optimize the global cost (minimize the sum of costs from both UAVs). Or they can be configured as self-interested UAVs. In such a case, they try to reduce the loss caused by their collision. Then, the best possible collision avoidance pair is identified by a variation of the monotonic concession protocol- the protocol for automated agent-to-agent negotiations.

Multi-Party Collision Avoidance (MPCA)

MPCA removes iterations from IPPCA for multi-collision situations -- more than two UAVs have mutual future collisions on their flight trajectories. MPCA introduces the multi-party coordinator who is responsible for the state space expansion and the search for the optimal solution of the multi-collision situation. The multi-party coordinator is an agent whose role is to find a collision-free set of flight trajectories for the whole group of colliding UAVs considering the limited time horizon. The coordinator keeps information about the group and state space. It chooses which UAV will be invited to the group and requests UAVs to generate modifications of their trajectories. Each state contains one flight trajectory for every UAV in the multi-party group. Initially, the group is constructed from two UAVs which have identified a conflict on their future trajectories. Later, the group is extended with UAVs which have conflicts with them and also which have potential conflicts with any considered flight trajectory in the state space. Similarly to IPPCA, UAVs provide the cost value for the considered collision avoidance maneuver along with its partial future flight trajectory.

The coordinator searches until it finds a state which does not have any collision considering the limited time horizon. Its internal operation can be described as a search loop based on the $OPEN$ list and states. States in $OPEN$ are ordered depending on the used optimization criterion (e.g. the sum of costs, the lowest is the first). In each loop, the coordinator takes the first state and checks if there are some collisions. If there is no collision, trajectories in this state is the solution. If any trajectory has a conflict with other UAV not included in the multi-party group, the coordinator invites this UAV and extends all states with the original flight trajectory of this invited UAV. Then, it selects a pair of UAVs from that state which have the earliest mutual collision and asks them to provide modifications of flight trajectories so that the collision can be eliminated. This step is similar to IPPCA. From the resulting set of options, new children states are generated.

As described above, the participation of UAVs in one multi-party algorithm is determined dynamically by identified conflicts on their trajectories. Thus, there can run two independent multi-party algorithms over disjoint sets of UAVs. UAV already participating in one multi-party run can be invited into other one. In such a case, UAV decides which one has higher priority and where it will be active based on the earliest collision which is solved by appropriate multi-party runs. The second one is paused until the first one is resolved. Please note that also the run-time of the multi-party algorithm is monitored by the multi-layer collision avoidance framework. The framework can terminate the participation due to lack of time and can select other solver to resolve the collision.

Non-Cooperative Collision Avoidance

AgentFly also integrates the collision avoidance algorithm which doesn't require a bi-directional communication channel for conflict detection and resolution. In such a case, only radar-like information about objects in its surrounding is used. The method used in AgentFly is based on the dynamic creation of no-flight areas which are then used by the flight trajectory planner to avoid a potential conflict. Such approach allows combining both cooperative and non-cooperative approaches in the same time. Such dynamically created no-flight areas are then taken into account also when the planner is used by cooperative solver for application of an evasion maneuver.

The detection part of the non-cooperative algorithm is active permanently. It receives information about positions of objects in the surrounding area. The observation is used for the update of the algorithm knowledge base. If there is enough history available, the prediction of a potential collision point process is started. The collision point is identified as an intersection of the current airplanes flight trajectory and the predicted flight trajectory of the object for which the algorithm receives the radar update. There can be integrated various prediction methods in this component: the simple linear prediction estimating the future object trajectory including the current velocity which requires two last positions with time information or a more complex tracking and prediction method based on longer history which is able to track also a curved trajectory. The result of the collision point prediction is a set of potential conflict points with a probability of conflict. For many cases it is possible that is no such collision point is found. In such a case, the previously registered conflict within the collision solver manager is canceled.

In opposite case, the collision points with higher probability than the configured threshold are wrapped by a dynamic no-flight area. The shape of a dynamic no-flight area is derived from the set of possible collision positions. The predicted dynamic no-flight area is put to the UAV's airspace definition database. Such areas are then used by the trajectory planner while it is called by any other component (cooperative collision avoidance algorithm or high-level flight controller). Information about the predicted future collision is also registered within the collision avoidance manager that will decide when the collision will be solved considering the time to collision. Once the non-cooperative solver is asked to solve the collision, the re-planning of the current flight trajectory is invoked by calling the flight trajectory planner which keeps the flight trajectory only within the operation airspace excluding all no-flight areas. The modified flight trajectory is then applied for its execution.

Experimental Evaluation

The provided concept was tested on The United States National Airspace System (US NAS) domain, see (Nolan, 2004). The goal of the experiment is to model the air traffic in the US NAS using real data provided by Federal Aviation Administration (FAA). We simulate the real flights by planning the trajectory between each two waypoints of the specific flight using the trajectory planner. In some scenarios we are also using the conflict resolution module (collision avoidance) to solve the potential collisions. We use a spherical model of Earth using Earth-centered Earth-fixed coordinate frame (GPS).

The simulated data set consists of following:

- Description of flights starting and landing in the USA, specifically from 30th Aug 2007 09:00 AM to 31st Aug 2007 08:59 AM. For each flight, the data file defines an initial configuration point c_I , the goal configuration point c_G and the velocity of the airplane. For each flight there are also several mid-waypoints the airplane has to pass when flying from c_I to c_G. The data cover only the en-route part of the flight.
- Definition of the Special Use Airspaces (SUA). The SUAs are modeled as three-dimensional shapes derived from polygons projected on a spherical surface and extruded either towards or away from the center of the Earth.

We have designed several measurement scenarios:

- *Middle waypoints, no SUA zones, no CA* – Initial setup using middle waypoints between starting and final waypoint provided by FAA, not considering any special use airspace (thus the waypoints

are connected by smooth connectors) and not using collision avoidance.

- *Middle waypoints, SUA zones, no CA* – Setup using middle waypoints between starting and final waypoint provided by FAA, considering all special use airspace (thus AA* algorithm is used in case, when smooth connector intersect some SUA) and not using collision avoidance.
- *No middle waypoints, SUA zones, no CA* – Setup using only starting and final waypoint, considering all special use airspace (thus AA* algorithm is used in case, when smooth connector intersect some SUA) and not using collision avoidance.
- *No middle waypoints, SUA zones, CA* – Setup using only starting and final waypoint, considering all special use airspace (thus AA* algorithm is used in case, when smooth connector intersect some SUA) and using collision avoidance.

The provided data set contains several airplanes with identical departure time and airport, that cause the immediate loss of separation (both safety zone – 5 nm and crash range – 300 ft). To avoid initial violation, departures of selected airplanes are postponed by inserting 120 second time gap between the aircrafts that violate the separation range.

The experiments were run in a distributed manner using the underlying high-scalable distributed simulation platform. The simulation contains one simulation controller, at least one node to simulate environment and at least one node to compute flight control. For all experiments the following computers were connected in LAN:

- Intel Core2 6700 (2x2.6GHz), 4GB RAM, Windows Vista 64-bit – *simulation controller* (2 cores, 3GB RAM)
- Intel Xeon E5530 (4x2.4GHz), 24GB RAM, Windows Vista 64-bit – *environment simulation* (8 cores, 22GB RAM)
- Intel Xeon E5530 (4x2.4GHz), 24GB RAM, Windows Vista 64-bit – *flight control* (4 cores, 8GB RAM), (4 cores, 8GB RAM)
- 2x Intel Xeon E5420 (8x2.5GHz), 8GB RAM, Windows Vista 64-bit – *flight control* (4 cores, 6.5GB RAM)
- Intel Core2Quad Q6600 (4x2.4GHz), 4GB RAM, Windows Vista 64-bit – *flight control* (2 cores, 3GB RAM)
- Intel Core2 6300 (2x1.86GHz), 3GB RAM, Windows 7 64-bit – *flight control* (1 core, 2GB RAM)

This configuration used 8 cores with 22GB RAM for the environment simulation and 15 cores with 26.5GB RAM for the flight control simulation. All computers were connected by 1Gbit Ethernet network (connection through a single switch) using TCP/IP connections. All systems were running Sun Java 64-bit 1.6.0_18 environment.

Results

In four measured scenarios there are significant differences between scenarios with middle waypoints of the flight plans and without middle waypoints. The difference between scenarios with collision avoidance module and without the module is insignificant. This is due to limited amount of losses of separation, mainly caused by the 120 seconds long time shifts.

The Figure 4 presents clustering of the airplanes based on the total flight duration in 15 minute clusters, i.e. airplanes with total flight duration between 0 and 15 minutes are clustered together.

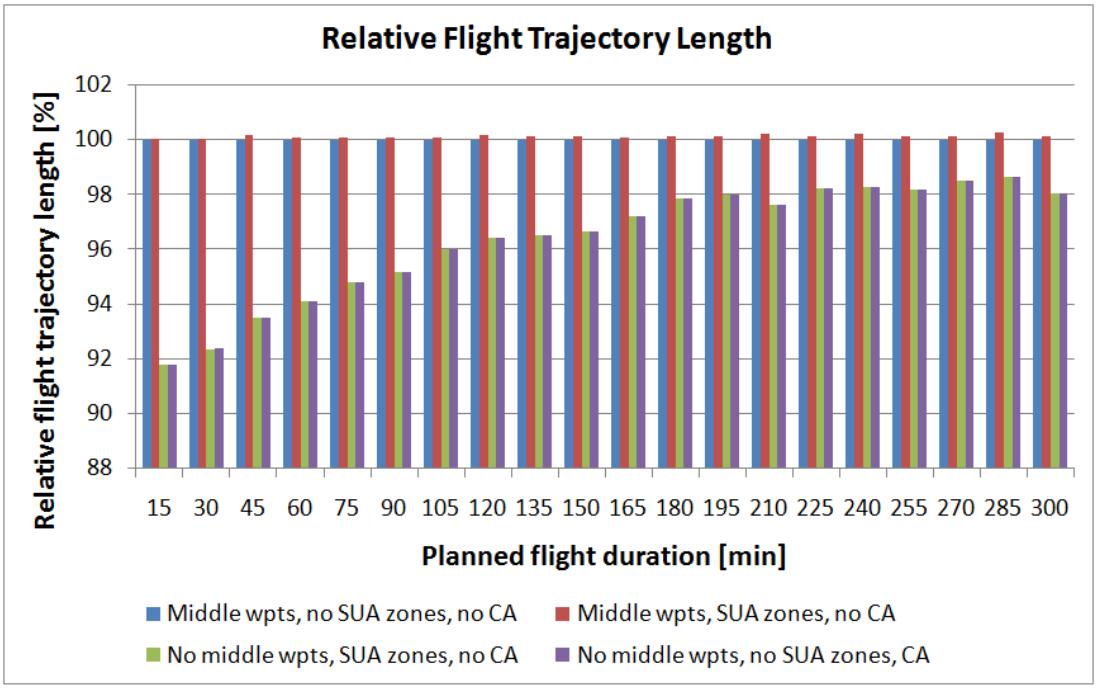

Figure 4. Relative Flight Trajectory Length

The chart in Figure 5 presents the relative fuel consumption of the airplanes. The scenario with middle waypoints, no SUA zones and no collision avoidance is regarded as 100% and the other values are relative to this base. The settings with no middle waypoints (using the free-flight concept) significantly outperform the traditional approaches and save up to 8% of fuel. In these settings the airplanes are using the trajectory planner to find the trajectory between the start waypoint and the final waypoint (keeping the required distance while the CA module is on).

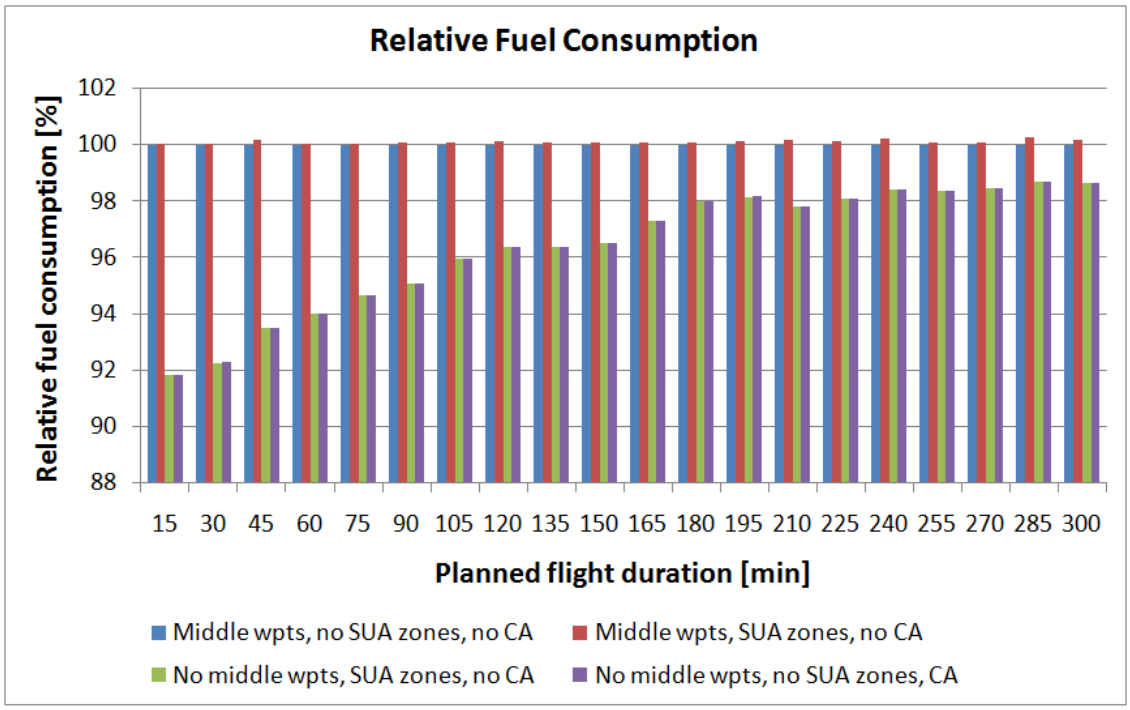

Figure 5. Relative Fuel Consumption

Figure 6 presents the relative flight trajectory length summed for all airplanes from each cluster. The chart is highly correlated with the chart at Figure 3.

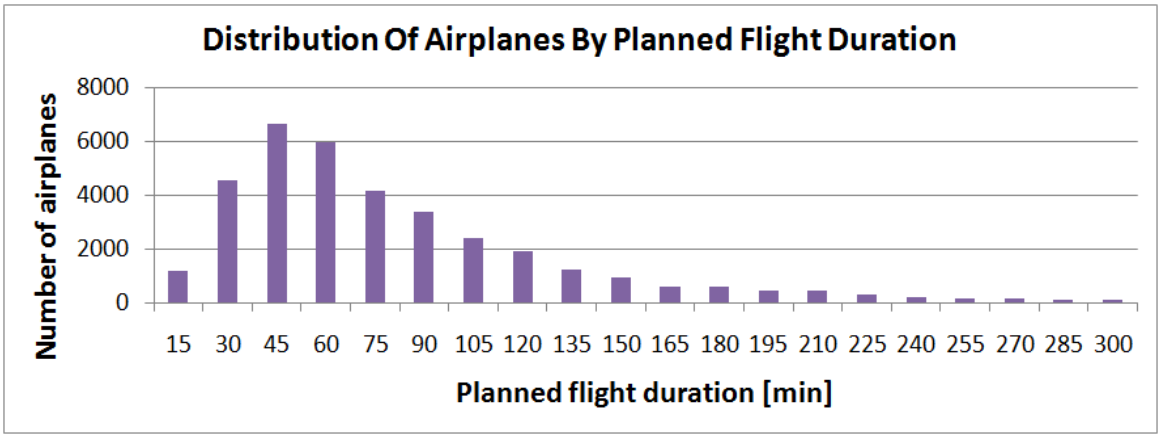

Figure 6. Distribution of Airplanes by Planned Flight Duration

Conclusion

In the paper we present the multi-agent approach to the air traffic simulation and to the flight control of UAVs. We describe the architecture of the system and justify using of the Aglobe multi-agent platform.

In the second part of the paper we present the collision detection and resolution (collision avoidance) methods for air traffic domain. The three presented methods use the decentralized iterative approach. The methods were empirically evaluated and together with the planner are the core components of the AgentFly system.

The experimental part of the work presents the real scenario in which the air traffic in the U.S. NAS is simulated based on the real data. The high – fidelity simulation works with the data set from August 30, 2007, special use airspaces and real wind conditions. In the experiment we show the difference in trajectory length and fuel consumption for the current system of the air traffic control and for the free – flight concept in the en-route phase of the flight.

The experiment also proved that AgentFly is suitable for computational intensive high – fidelity large scale simulations. Thanks to the distributed simulation framework it is possible to use AgentFly on computer grids or in clouds of computers and push the simulation limits even further. In our opinion AgentFly is one of the tools suitable for simulations of the NextGen concept.

References

[1] David Sislak and Martin Rehak and Michal Pechoucek, 2005, A-globe: Multi-Agent Platform with Advanced Simulation and Visualization Support. In *Web Intelligence*. IEEE Computer Society,. ISBN 0-7695-2415-X.

[2] Bellifemine, F. and Rimassa, G. and Poggi, A., 1999, JADE - A FIPA-compliant Agent Framework, Proceedings of 4th International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, London

[3] David Sislak and Michal Pechoucek and Premysl Volf and Dusan Pavlicek and Jiri Samek and Vladimir Marik and Paul Losiewicz, 2008, AGENTFLY: Towards Multi-Agent Technology in Free Flight Air Traffic Control. Defense Industry Applications of Autonomous Agents and Multi-Agent Systems. Birkhauser Verlag.