Decentralized Algorithms for Collision Avoidance in Airspace

David Šišlák, Jiří Samek and Michal Pěchouček
Agent Technology Group, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague
Technicka 2, Prague, 166 27, Czech Republic
sislakd@fel.cvut.cz, {samek|pechouc}@labe.felk.cvut.cz

ABSTRACT

The paper proposes decentralized deconfliction algorithms deployed on multiple autonomous aerial vehicles in free-flight operations. The paper provides two separate algorithms for collision avoidance - one based on the iterative peer-to-peer negotiation solving a singular collision and second based on multi-party negotiation about a cluster of collisions. The presented decentralized algorithms allow the vehicles operating in the same area to utilize the given airspace more efficiently. The algorithms have been developed and tested on a multi-agent prototype and the properties of both algorithms are discussed on a set of large scale experiments.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial intelligence—*Multi-agent systems*; I.6 [**Computing Methodologies**]: Simulation and modeling

General Terms

Algorithms, Performance

Keywords

agent-based collision avoidance, autonomous aircrafts, distributed algorithm

1. INTRODUCTION

The operation of a group of autonomous airplanes fulfilling a given objective of their mission requires the See & Avoid capability [1]. Such capability guarantees that each airplane is able to monitor local environment using the onboard radar or information from a headquarter to detect possible collision situation and to take actions to avoid that collision. Such approach is known as the *free flight* concept [8] – each airplane plans its own best flight corridor but still respects separation from the others. There are two categories of collision avoidance algorithms: cooperative and noncooperative. The paper addresses the area of cooperative collision avoidance algorithms in the three-dimensional space restricted by the limited communication range and by no-flight zones which represent some strategic places.

The research community in the field of cooperative collision avoidance aims on many different approaches and comparing using different metrics [3]. One possibility for the distributed solution is the use of principled negotiation [10]. Others use the approaches based on the game theory [2] and the fully distributed solution based on agent to agent negotiation using various protocols [11].

We use the agent based approach to implement the cooperative collision avoidance by a local collision avoidance implementation. The detailed specification of the airspace domain is stated in the Section 2. The restriction to use only decentralized algorithm is given by the fact that airplanes in the domain can communicate only with planes located nearby. Well known techniques for collision avoidance based on the potential fields are not suitable for the domain (3D, no-flight zones, the need smooth flight plan also in its first derivation, allowed speed changes) due to their complexity and because of the complicated dynamic mission airplane specification.

The main contribution of the paper is in the novel multi-party collision avoidance (MPCA) algorithm (Section 6) for the described airspace domain. The algorithm removes the iterations from the iterative peer-to-peer (IPPCA) algorithm [9] (new formalized description is in the Section 5) for multi-collisions among several airplanes. The use of searching for the best combination of evasion applications used in the proposed algorithm leads to better solutions (in means of used criterion) and thus for better utilization of given airspace than in the IPPCA. The 'See' ability of the algorithms is the same, see the Section 3. Both algorithms use the same set of possible evasion manoeuvres (Section 4). The properties of the proposed MPCA are studied in experimental way by a comparison to the existing IPPCA (Section 7). The IPPCA algorithm was selected for the comparison because it provides the best result from all implemented ones in the domain multi-agent simulator.

2. DOMAIN DESCRIPTION

Let us define the airspace collision avoidance problem in the free-flight domain (FFD). In FFD the autonomous aircrafts, all members of $\mathcal{A}$, operate in a shared three dimensional *airspace* $Air \subseteq R^3$ that is limited by the ground surface and airspace boundaries. The airspace that can be occupied by an individual aircraft A_i is made smaller by *no-flight zones* $\mathcal{Z}_i = \{Z_1, Z_2 \ldots\}$, where the A_i cannot fly (thus $Air_i = Air - \sum_{Z_k \in \mathcal{Z}_i} Z_k$). The no-fly zones represent strategic places (e.g. nuclear power plants), nonstandard weather conditions or dangerous enemy territories.

The behavior of the airplane $A_i \in \mathcal{A}$ is given by the specific *flight plan* $fp_i = (sp, e_1, e_2 \ldots e_n)$ defined as a start

Cite as: Decentralized Algorithms for Collision Avoidance in Airspace, Šišlák, D. - Samek, J. - Pěchouček, M., *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril,Portugal,pp. 543-550.

situation and ordered sequence of flight elements. The start situation includes the position, initial direction, start time and initial velocity $sp = \langle \overline{x}_0, \overline{dir}_0, t_0, v_0 \rangle$. Each of the flight elements can be of the following type: (i) straight element $e_{straight} = \langle l, a \rangle$ specified by its length and acceleration, (ii) horizontal or vertical turn elements $e_{turn} = \langle r, d \rangle$ given by the turn radius (sign define turn direction) and angle, and (iii) spiral element $e_{spiral} = \langle r, d, c \rangle$ similar to horizontal turn element extended by a climbing rate. The elements are constrained by the airplane type that often specifies minimal and maximal flying velocity, minimal and maximal acceleration, minimal turn radius and max climbing/descending rate. Each element e_{k+1} specifies the flight path part relatively from the end of the previous element e_k or from sp for the first element e_0. Thus the path given by fp_i is continuous and must be smooth. First derivation in all coordinates and time must be continuous too. Let us introduce $\mathcal{FP} = \{fp_i\}_{A_i \in \mathcal{A}}$ as a set of all actual flight plans of the aircrafts in the airspace.

The behavior of an aircraft is not random, while it is specified by a *mission*. All points on the path of fp_i must be in Air_i and must be constrained by the airplane mission $M_i = \langle wp_1, wp_2 \ldots wp_n \rangle$, a sequence of *way-points* $wp_k = \langle \overline{x}, t_1, t_2 \rangle$. The way-point in Air_i specifies the time interval by specifying t_1 and t_2 when the aircraft is allowed/requested to fly through. Let us introduce the function $\mathbf{p}(fp_i, t)$ that returns the position of the individual airplane A_i at the given time t. The way-point wp_k is *completed* by the A_i's flight plan fp_i if $\exists t$ so that $\mathbf{p}(fp_i, t) = \overline{x}^{wp_k}$ and $t_1^{wp_k} \le t \le t_2^{wp_k}$. In the other words, the path defined by fp_i must go through each way-point at least once in the specified order of M_i.

Definition 1. The **planning problem** in FFD from the perspective of an individual aircraft A_i with respect to the mission M_i is the process of finding such a flight plan fp_i so that $\forall wp_k \in M_i$ are completed.

Airplane can alter its own current fp_i anytime, but only the future part can be changed. The processes of (re)planning and collision avoidance are carried out in the same time as the process of mission accomplishment. Thus, the airplane is allowed to change its flight plan in some future time t to be able to apply new changed fp'_i, see the Figure 1.

Around each airplane there is defined a *safety zone* – a spherical space with a given radius rsz_i for each A_i. It defines surrounding airspace where no other airplane is allowed to appear so that effects of turbulence caused by other airplane and inaccuracy of flight plan execution (there are allowed small deviations from the flight path) can be avoided. Let us introduce the function

$$\mathbf{col}(fp_i, fp_j, t) = \begin{cases} 1 & \text{if } |\mathbf{p}(fp_i,t), \mathbf{p}(fp_j,t)| \le \max(rsz_i, rsz_j) \\ 0 & \text{otherwise} \end{cases}$$

specifying the fact that two flight plans fp_i and fp_j have a *collision* in time t.

Definition 2. Two aircrafts A_i and A_j (with their flight plans fp_i and fp_j respectively) are **colliding** (denoted as $A_i \otimes A_j$) if and only if $\exists t : \mathbf{col}(fp_i, fp_j, t) = 1$.

Clearly, $A_i \otimes A_j \equiv A_j \otimes A_i$. The set $\mathcal{A}$ is dynamic as there can be new planes created or removed (e.g. after landing) during the process of mission execution. In the FFD it is guaranteed that newly created plane A_i has no collision on its flight plan fp_i with any other existing fp_j in next δ from its start. The δ value is specified for each aircraft and guarantees that there is enough air space to avoid future collision which appears just after the plane creation. After δ the flight plan of the newly created aircraft can collide.

Let us discuss how the multiple collisions can influence each other. We introduce $\mathcal{C}_{all} \subset \mathcal{A} \times \mathcal{A}$ as a set of all colliding aircrafts. We are working with the *multi-collision set* of collisions $\mathcal{C} \subset \mathcal{C}_{all}$ that includes all related collisions. In $\mathcal{C}$ there is at least one pair of colliding airplanes $A_i \otimes A_j$ and in the same time there is no such collision $A_k \otimes A_l \in \mathcal{C}$ so that neither A_i nor A_j does not have a collision with either A_k or A_l and there is no other collision in $\mathcal{C}$ that is linked with both $A_i \otimes A_j$ and $A_k \otimes A_l$ by a finite number of collisions. Let us view $\mathcal{C}$ as undirected graph. Let us assume that each collision from the set $\mathcal{C}$ has one vertex in a graph, an edge between any two vertices exists if and only if there is at least one A_i involved in both collisions represented by vertices. The $\mathcal{C}$ is the multi-collision set if and only if its graph representation is connected (for every pair of distinct vertices in the graph there exists a path linking them both). Note that the concept of multi-collision set includes also the collision of two airplanes only. $\mathcal{A}_\mathcal{C} \subseteq \mathcal{A}$ is the set of all aircrafts which are implied in at least one collision in $\mathcal{C}$.

Let us work with the encounter [6] as a subject of collision avoidance problem. For a given multi-collision set $\mathcal{C}$ an *encounter* en_k is tuple $\langle t, \{fp_i\}_{A_i \in \mathcal{A}_\mathcal{C}} \rangle$ such that $t \ge \texttt{now}$ is a time point in the future from which the flight plans of the colliding airplanes can be changed. Clearly, the *collision avoidance problem* (CA) in FFD can be defined as the process of finding such $\mathcal{FP}$ for which $\mathcal{C}_{all} = \emptyset$. In this paper we are solving CA by solving *local collision avoidance problems* (LCA) applied on top of $\mathcal{FP}$.

Definition 3. **Local collision avoidance problem** (LCA) (*replanning*) with respect to an encounter $en_k = \langle t, \{fp_i\}_{A_i \in \mathcal{A}_\mathcal{C}} \rangle$ [6] and $\mathcal{FP}$ is the process of finding such a solution $\{fp'_i\}_{A_i \in \mathcal{A}' \subseteq \mathcal{A}}$ founded in time $t' < t$ so that the encounter en_k is eliminated.

The current time and time t from encounter gives the maximal interval in which LCA algorithm can search for the solution. The selection of right time t in encounter is the part of the algorithm and can take into account its own properties. Two CA algorithms applied to the same situation can be compared using their final flight plan *utility values* after accomplishment of all aircrafts' missions. The utility function value $\mathbf{u}_i(fp_i)$ used for comparison [5] includes aircraft's intention to proposed solution of the conflicts – e.g. to be as short as possible, to use minimum of fuel, to fulfill time constraints of the way-points, etc. The $\mathbf{u}_i(fp_i) \in \langle 0, 1 \rangle$ is evaluated as weighted sum of its components. Each airplane can have different components, but each airplane must use the same in both compared CA algorithms. E.g. for the social welfare criterion we can say that one CA algorithm is better if $\sum_{A_i \in \mathcal{A}} \mathbf{u}_i(fp_i)$ is higher where fp_i represents final flight plan of airplane A_i after applying CA algorithm.

The airplane can host one or more *agents* and provide them communication infrastructure via its on-board communication transceivers with the limited range of communication c_i. So, the agents at A_i can negotiate with agents at A_j in time t only if $| \mathbf{p}(fp_i, t), \mathbf{p}(fp_j, t) | \le \min(c_i, c_j)$. The agents on airplane A_i have full information about its flight status and can call functions for altering fp_i. Using this transceiver airplane agents are aware of existence of other airplane if they can negotiate together. There is no central element in the domain so the agent knows only information which can be obtained from its hosting airplane or by negotiation with other agents. Even though the range c_i is relatively large, not all aircrafts can communicate together. The domain allows also airplanes which are not capable to communicate with others due to several reasons: broken transceiver, or they don't want to communicate. But in this paper we are focused only on the case where all airplanes $\mathcal{A}$ are able to communicate together if distance condition is satisfied. Thus, agent controlled airplanes can cooperate to do collision avoidance.

3. LOCAL COOPERATIVE DETECTION

The *see* capability [1] in the domain is implemented by the negotiation and flight plan comparison. Due to the limited communication range each airplane A_i is aware only of the planes located within this range $A_j \in \widetilde{\mathcal{A}_i}$. $\widetilde{\mathcal{A}_i}$ denotes the set of airplanes $A_j \in \mathcal{A}$ of which A_i is aware. Both described algorithms solve encounters locally where they can be detected. It is not necessary to identify collision $A_i \bigotimes A_j$ for whole fp_i and fp_j. The airplane can share only *part of its current flight plan* $\widehat{fp}_i$ from current time t_c for interval t_{share} where $\mathbf{p}(\widehat{fp}_i, t) = \mathbf{p}(fp_i, t)$ for $\forall t : t_c \leq t \leq t_c + \delta_t$. δ_t is selected by the airplane not to expose all its future plan including its mission objectives, but to give enough part to identify possible collision. Such local sharing of the flight plans also reduces necessary communication flow. The flight plan sharing is implemented by the subscribe-advertise protocol [12]. Every time when the airplane is aware of new other airplane A_j it subscribes for its $\widehat{fp}_j$. The plane A_j, by accepting subscription request from A_i, will provide regular updates of its $\widehat{fp}_j$ such there will be enough part of future part of the flight plan from current time. If A_j changes its fp_j for any reason – change of its mission objectives or as a result of other collision avoidance – it provides new fresh $\widehat{fp}_j$ to the subscriber as soon as possible.

Airplane A_i who received $\widehat{fp}_j$ from all its neighbors $\widetilde{\mathcal{A}_i}$ *performs check* if $\exists t$ where $\mathbf{col}(fp_i, \widehat{fp}_j, t) = 1$ upon every received update. If such t exists A_i tries to identify first and last collision point $t_1^{A_i \otimes A_j}$ and $t_2^{A_i \otimes A_j}$ (Section 4). Airplanes are also able to detect multi-collision group $\mathcal{A}_\mathcal{C}$ by exchanging information about collisions. A_i prepares its local view of an encounter $en_k^{A_i} = \langle t, \{fp_i\} \bigcup \{\widehat{fp}_j\}_{A_j \in \mathcal{A}_\mathcal{C} \setminus A_i} \rangle$. Selection of $t_1^\mathcal{C} > t > t_c$ depends on the used algorithm or the combination of algorithms and is chosen somewhere between current time t_c and time of the earliest collision $t_1^\mathcal{C}$ for given multi-collision $\mathcal{C}$. $t - t_c$ defines timeout which is then given for invoked collision avoidance algorithm. If the result for $\mathcal{C}$ is not provided within specified timeout, the algorithm is interrupted and next algorithm is invoked for the same $\mathcal{C}$ and new encounter. Note that using local cooperative detection the encounter contains full fp_i and only parts of $\widehat{fp}_j$, but it is enough to do distributed local collision avoidance. The LCAP algorithm still provides solution containing full flight plans fp_j for $\forall A_j : A_j \in \mathcal{A}_\mathcal{C}$ because all flight plans are still provided by its final implementor A_j.

4. EVASION MANOEUVRES

Both CAP algorithms described in this paper are based on the application of evasion manoeuvre to the place of the collision $A_i \otimes A_j$ – identification of the first and last collision time of the first collision between their fps. $t_1^{A_i \otimes A_j}$ is the *first collision time* between fp_i and fp_j if there is $\mathbf{col}(fp_i, fp_j, t_1) = 1$ and $\forall t : t < t_1$ is $\mathbf{col}(fp_i, fp_j, t) = 0$. The *last collision time* $t_2^{A_i \otimes A_j}$ is defined by $\forall t : t_1 \leq t \leq t_2$ is $\mathbf{col}(fp_i, fp_j, t) = 1$ and for $t = \lim_{\delta \to 0_+} (t_2 + \delta)$ is $\mathbf{col}(fp_i, fp_j, t) = 0$. From the fact that $\max(rsz_i, rsz_j) > 0$ we are certain that $t_2 > t_1$ holds. Note that the times $t_1^{A_i \otimes A_j}$ and $t_2^{A_i \otimes A_j}$ are the same from both perspective A_i and A_j, but the $\mathbf{p}(fp_i, t_1) \neq \mathbf{p}(fp_j, t_2)$.

There are defined seven *evasion manoeuvres* $\mathcal{EM}_i = \{\mathbf{em}_L, \mathbf{em}_R, \mathbf{em}_U, \mathbf{em}_D, \mathbf{em}_F, \mathbf{em}_S, \mathbf{em}_0\}$: left, right, climb up, descend down, fly faster, fly slower and leave plan as it is. The set $\mathcal{EM}_i$ can contain different manoeuvres for each airplane $A_i \in \mathcal{A}$, but there must be included $\mathbf{em}_0$ for all planes. The evasion manoeuvre can be seen as a function applied to the flight plan with time specification and returning new changed flight plan. The simplest one is defined as $fp_i = \mathbf{em}_0(fp_i, p, t, t_1, t_2)$. It returns exactly the same fp_i and is used for the simplification of cases where airplane A_i can also do nothing in CA algorithm. All other $\mathbf{em}$s return changed fp'_i respecting constraints given for flight plan and changes elements only from specified time t – time known from definition of *encounter* en_k. All manoeuvres are applicable only if $t < t_1 < t_2$ and have also specified application strength parametrization p for the evasion manoeuvre.

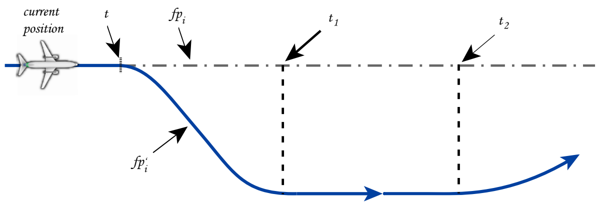

Figure 1: Application of right evasion manoeuvre – $fp'_i = \mathbf{em}_R(fp_i, p, t, t_1, t_2)$

The application of right evasion manoeuver to the fp_i is shown in the Figure 1. It changes fp_i from time t that the new fp'_i passes through the points specified by moved $\mathbf{p}(fp_i, t_1)$ and $\mathbf{p}(fp_i, t_2)$ to the right side. The size of the shift is given by the parameter p – for larger p the evasion manoeuvre makes larger evasion. The $\mathbf{em}_L, \mathbf{em}_U$ and $\mathbf{em}_D$ are defined similarly to $\mathbf{em}_R$ only with different shift direction of the points to the appropriate side. The velocity changing manoeuvres $\mathbf{em}_F$ and $\mathbf{em}_S$ change the flight plan by time t'_1 that $\mathbf{p}(fp'_i, t'_1) = \mathbf{p}(fp_i, t_1)$. For fly faster manoeuvre is $t'_1 < t_1$ and for fly slower $t'_1 > t_1$. The application of $\mathbf{em}_F$ manoeuvre is restricted by the maximal flying velocity of the plane. The $\mathbf{em}_S$ manoeuvre is not restricted, because there can be inserted holding orbit if the minimal flying velocity is reached. Evasion manoeuvres can be combined together by their sequential application.

5. ITERATIVE PEER-TO-PEER CA

This section introduces *iterative peer-to-peer CA* [9] used as a provider of comparison result for multi-party CA. The algorithm solves an encounter $en_k = \langle t, \{fp_i\}_{A_i \in \mathcal{A}_\mathcal{C}} \rangle$ by the selection of most important collision airplane pairs $\mathcal{I} = \{A_1 \bigotimes A_2, A_3 \bigotimes A_4 \ldots\}$ where each airplane from $\mathcal{A_C}$ can be included only once in $\mathcal{I}$. Identification of the set $\mathcal{I}$ is done in the distributed manner. Each $A_i \in \mathcal{A_C}$ selects its opponent A_j from local view of the encounter $en_k^{A_i} = \langle t, \{fp_i\} \bigcup \{\widehat{fp}_j\}_{A_j \in \mathcal{A_C} \setminus A_i} \rangle$ (see Section 3) using

$$\underset{A_j \in \mathcal{A_C}}{\arg\min}\ \underset{t_1}{\arg\min}\ \mathrm{col}(fp_i, \widehat{fp}_j, t_1) = 1 \ . \tag{1}$$

Each $A_i \in \mathcal{I}$ starts *pair negotiation on removing collision* (PNRC) with its colliding opponent A_j. If there is a conflict – A_i selects A_j which is already negotiating with other airplane A_k – A_j will stop its current negotiation if and only if $t_1^{A_i \otimes A_j} < t_1^{A_j \otimes A_k}$. Note that for the same times the first selected opponent by A_j will stay. The encounter in which A_j is included can be changed during its PNRC. A_j stops current PNRC if the solved collision no more exists within new encounter or there is identified more important opponent to A_j.

Within PNRC these airplane pairs $A_i \bigotimes A_j \in \mathcal{I}$ prepare a set of possible changed flight plans with their utility value

$$\mathcal{F}_i = \{\langle fp'_{i1}, \mathbf{u}_i(fp'_{i1})\rangle, \langle fp'_{i2}, \mathbf{u}_i(fp'_{i2})\rangle \ldots\} \ . \tag{2}$$

First, the flight plans $\langle fp'_i, \mathbf{u}_i(fp'_i)\rangle \in \mathcal{F}_i$ are given by the application of all $\mathbf{em} \in \mathcal{EM}_i$ to the place of collision $A_i \otimes A_j$ using the lowest parametrization p first, see the Section 4. Note, that the set $\mathcal{EM}_i$ can contain different manoeuvres for each airplane and there is always included *leave plan as it is* manoeuvre $\mathbf{em}_0$ for all airplanes. The changed flight plan fp'_i of $\mathbf{em}$ application is included in $\mathcal{F}_i$ if and only if

$$\forall A_j \in \widetilde{\mathcal{A}_i} \ \text{ is } \ (\underset{t}{\arg\min}\ \mathrm{col}(fp'_i, \widehat{fp}_j, t) = 1) > t_1^{A_i \otimes A_j} \ . \tag{3}$$

Then both planes A_i and A_j exchange their $\widehat{\mathcal{F}}$ – local future parts of proposed changes with their utility values (counted from the whole flight plans) for next δ interval, see the Section 3. A_i prepares the negotiation set

$$\begin{aligned} \mathcal{S}_i^{A_i \otimes A_j} = \ & \{\langle fp'_{i1}, \mathbf{u}_i(fp'_{i1}), \widehat{fp}'_{j1}, \mathbf{u}_j(fp'_{j1})\rangle, \\ & \langle fp'_{i1}, \mathbf{u}_i(fp'_{i1}), \widehat{fp}'_{j2}, \mathbf{u}_j(fp'_{j2})\rangle \ldots\} \end{aligned} \tag{4}$$

as a cartesian product of $\mathcal{F}_i$ and $\widehat{\mathcal{F}}_j$. Each tuple $\langle fp'_{ik}, \mathbf{u}_i(fp'_{ik}), \widehat{fp}'_{jl}, \mathbf{u}_j(fp'_{jl})\rangle \in \mathcal{S}_i^{A_i \otimes A_j}$ is included if and only if

$$(\underset{t}{\arg\min}\ \mathrm{col}(fp'_{ik}, \widehat{fp}'_{jl}, t) = 1) > t_1^{A_i \otimes A_j} \ . \tag{5}$$

If the negotiation set $\mathcal{S}^{A_i \otimes A_j}$ is empty, A_i adds to the $\mathcal{F}_i$ flight plans as a result of application of evasion manoeuvres $\mathbf{em} \in \mathcal{EM}_i$ using the next larger parametrization p. This is done by both A_i and A_j and new $\mathcal{S}_i^{A_i \otimes A_j}$ is generated. The whole process is repeated until the negotiation set holds at least one element.

A_j does the same from its perspective and its view of negotiation set $\mathcal{S}_j^{A_i \otimes A_j}$ holds equivalent elements as $\mathcal{S}_i^{A_i \otimes A_j}$, but has full flight plan for fp'_j and local future parts for fp'_i. Both airplanes A_i resp. A_j propose the solution

$$\begin{aligned} & \underset{\langle fp'_{ik}, \mathbf{u}_i(fp'_{ik}), \widehat{fp}'_{jl}, \mathbf{u}_j(fp'_{jl})\rangle \in \mathcal{F}_i}{\arg\max} \mathbf{u}_i(fp'_{ik}) + \mathbf{u}_j(fp'_{jl}) \ \text{ resp.} \\ & \underset{\langle fp'_{jl}, \mathbf{u}_j(fp'_{jl}), \widehat{fp}'_{ik}, \mathbf{u}_i(fp'_{ik})\rangle \in \mathcal{F}_j}{\arg\max} \mathbf{u}_i(fp'_{ik}) + \mathbf{u}_j(fp'_{jl}) \ . \end{aligned} \tag{6}$$

If there exists more candidates with the same value of selection criterion, the proposed solution is selected randomly from these candidates. To agree with one of two different randomly proposed solutions A_i and A_j can use protocol based on commitment scheme known from cryptography [4].

After distributed application of all solutions for $A_i \otimes A_j \in \mathcal{I}$ the last encounter is partially modified and new one is detected by the *local cooperative detection* (Section 3) and described peer-to-peer CA is started again. The restrictions 3 and 5 ensure that the possible application of the solution selected from $\mathcal{S}^{A_i \otimes A_j}$ cannot cause new earlier collision with any airplane's current flight plan of which they are aware than the solved one. This assertion with combination of the creation of $\mathcal{I}$ guarantees the termination of algorithm.

The described iterative peer-to-peer algorithm is also suitable for the application of any other solution selection from the negotiation set than described maximal sum of utilities in criterion 6. For example there can be used classical *monotonic concession protocol* (MCP) [13] – the simple protocol for automated agent to agent negotiations in cooperative domain. Use of such protocol guarantees that both participating agents want to maximize their expected utility. In this case both agents leave only *pareto optimal* deals in the negotiation set $\mathcal{S}$ and then the agents can use *Zeuthen strategy* [13] for finding acceptable deal for both agents.

6. MULTI-PARTY CA

This section describes the collision avoidance algorithm based on the creation of groups of airplanes which together solve a collision or collisions. In a more dense airspace, this approach enables better utilization of the airspace. As a motivation for this approach, we can imagine a situation where two airplanes have a collision $A_i \otimes A_j$, but it is difficult for them to avoid the collision as other airplanes are in the airspace near to them. The situation can be so difficult that they can have only two options, dramatically deviate from their courses, or deviate only slightly but make their flight plans colliding with another airplanes' flight plans. However, they can create a group with the other airplanes and solve the collision $A_i \otimes A_j$ together with them. Basically, we can say that the two colliding airplanes will ask other airplanes to make space for their evasion maneuvers.

The basic idea behind the presented multiparty algorithm is to search the state space of possible applications of sequences of evasion maneuvers on the flight plans of airplanes. The goal of the searching is to solve a multi-collision with respect to given criterion evaluating the fitness of solution. In our experiments we use the sum of flight plan utilities for decimalization of the social welfare. There is no restriction how many evasion maneuvers an airplane can apply to its flight plan. This means that the state space is infinite. The multiparty collision avoidance is motivated by A* algorithm [7]. The A* algorithm finds the optimal solution in a finite time if there is a solution. Our situation is more difficult, each airplane has its local information about other airplanes. This information can change during searching. This is the reason why we can not use pure A* algorithm, we can not specify searching space in the beginning of the searching. When the new plane appears in the communication range, its flight plan can collide with some airplane in multiparty group. Then the A* algorithm should be restarted because of the state space change. Our algorithm just updates states in the open list and continues with the search. This approach removes the lost of the progress of the search by restart. There are no cycles in the state space so the list for storing of already expanded states can be omitted. We will use only open list $\mathcal{O}$ for storing of states generated by the expansion of other states and not yet expanded.

For a given encounter $en_k = \langle t, \{fp_i\}_{A_i \in \mathcal{A_C}} \rangle$ a *multi-party group* $\mathcal{G} \supseteq \mathcal{A_C}$ is a set of airplanes whose flight plans are involved in a solution searching process. The goal of the group is to find a solution for the encounter. Note that solution provided by multi-party algorithm has to contain flight plans for airplanes A_C, but usually it will contain additional flight plans for airplanes located nearby. A *state* s is a set $\{s_i\}_{A_i \in \mathcal{G}}$, where $s_i = \langle \widehat{fp}_i, \mathbf{u}_i(fp_i) \rangle$ is a tuple containing the local flight plan $\widehat{fp}^s$ of airplane A_i for state s and its utility value computed by airplane A_i from full flight plan fp_i^s. For the *initial state* s_0 the $\mathcal{G}$ is $\mathcal{A_C}$ and thus the s_0 contains current local flight plans derived from en_k with their utility values. The *child state* s' of the state s is defined as a set of changed local flight plans with their utilities by application of evasion united with the set of unchanged flight plans from the predecessor, $s' = \{\langle \widehat{fp}_i^{s'}, \mathbf{u}_i(fp_i^{s'}) \rangle\}_{A_i \in \mathcal{I}} \cup \{\langle \widehat{fp}_i^{s}, \mathbf{u}_i(fp_i^s) \rangle\}_{A_i \in \mathcal{G} \setminus \mathcal{I}}$. The set $\mathcal{I} \subseteq \mathcal{G}$ holds exactly two airplanes which apply the evasions to remove selected collision as described later. The *final state* s_f is a state which is the solution of an encounter, basically a set of non-colliding flight plans.

For each state s there is defined the *evaluation function* $\mathbf{f}(s) = \mathbf{g}(s) + \mathbf{h}(s)$. The $\mathbf{g}(s)$ is the cost of the application of all evasion maneuvers applied to the s_0 to get to the state s. It is clear that $\mathbf{g}(s_0) = 0$. The $\mathbf{g}(s)$ is defined recursively. For the child state s' of s the

$$\mathbf{g}(s') = \mathbf{g}(s) + \sum_{A_i \in \mathcal{I}} \mathbf{u}_i(fp_i^s) - \sum_{A_i \in \mathcal{I}} \mathbf{u}_i(fp_i^{s'}). \qquad (7)$$

In the other words, the $\mathbf{g}(s') - \mathbf{g}(s)$ is the cost of application of evasion maneuvers with goal to remove one single collision of two flight plans for $A_i \in \mathcal{I}$. The $\mathbf{h}(s)$ is the heuristics function estimating the cost to remove all remaining collisions among flight plans in the state s.

At the beginning the multiparty group contains only the airplanes which create it, $\mathcal{G} = \mathcal{A}_C$. The searching algorithm of the group proceeds in a cycle until it finds the solution. The state with the lowest value of evaluation function is selected $s^* = \arg\min_{s \in \mathcal{O}} f(s)$. All flight plans from s^* are checked for collision with local flight plans of airplanes from the set $\mathcal{A} \setminus \mathcal{G}$. Any colliding airplane not already included in the set $\mathcal{G}$ is asked to join the group. If the airplane A_y joins the group, then its actual local flight plan is added to all states in $\mathcal{O}$ and to state s^*. Precisely, all states $s \in \mathcal{O}$ are replaced by a new states $s \cup \langle \widehat{fp}_y, \mathbf{u}_y(fp_y) \rangle$ (similarly for s^*). Note that the cost of the state $\mathbf{g}(s)$ does not change by addition of new flight plan, there is no application of evasion maneuvers. If the state s is the final state, algorithm finishes and the planes in the multiparty group change their actual flight plans to the flight plans $\{fp_i^{s_f}\}_{A_i \in \mathcal{G}}$ which correspond to local flight plans in the chosen final state s_f. From the description of the algorithm below, it is clear, that each airplane A_x knows for each generated local flight plan $\widehat{fp}_i^{s_f}$ its full version $fp_i^{s_f}$.

In the other case - when s^* is not the final state, the pair of airplanes A_i, A_j with the earliest collision in the state s^* is selected by

$$\arg\min_{A_i, A_j \in \mathcal{G}, A_i \neq A_j} t_1^{fp_i^{s^*} \otimes fp_j^{s^*}} \qquad (8)$$

and these airplanes A_i, A_j generate combinations $S^{fp_i^{s^*} \otimes fp_j^{s^*}}$ of flight plans to remove their (earliest) collision. The airplanes prepare sets F_i resp. F_j of possible changed flight plans with their utility value

$$\mathcal{F}_i = \{\langle fp'_{i1}, \mathbf{u}_i(fp'_{i1}) \rangle, \langle fp'_{i2}, \mathbf{u}_i(fp'_{i2}) \rangle \ldots\} \qquad (9)$$

and their local versions $\widehat{F}_i$ resp. $\widehat{F}_j$ for next δ interval, see the Section 3,

$$\widehat{\mathcal{F}}_i = \{\langle \widehat{fp}'_{i1}, \mathbf{u}_i(fp'_{i1}) \rangle, \langle \widehat{fp}'_{i2}, \mathbf{u}_i(fp'_{i2}) \rangle \ldots\} \ . \qquad (10)$$

The flight plans $\langle fp'_i, \mathbf{u}_i(fp'_i) \rangle \in \mathcal{F}_i$ are given by the application of all $\mathbf{em} \in \mathcal{EM}_i$ to the place of collision $fp_i^{s^*} \otimes fp_j^{s^*}$ using the lowest parametrization p, see the Section 4. Note, that the set $\mathcal{EM}_i$ can contain different manoeuvres for each airplane and there is always included *leave plan as it is* manoeuvre $\mathbf{em}_0$ for all airplanes. $\mathcal{S}^{fp_i^{s^*} \otimes fp_j^{s^*}}$ is then a subset of combinations of flight plans from $\widehat{\mathcal{F}}_i$ and $\widehat{\mathcal{F}}_j$ which have no collision or the first collision point of the collision is not earlier that the old collision point, precisely

$$\begin{aligned} \mathcal{S}^{fp_i^{s^*} \otimes fp_j^{s^*}} = \; & \{\{\langle \widehat{fp}'_{ik}, \mathbf{u}_i(fp'_{ik}) \rangle \in \widehat{\mathcal{F}}_i, \\ & \langle \widehat{fp}'_{jl}, \mathbf{u}_j(fp'_{jl}) \rangle \in \widehat{\mathcal{F}}_j\} : \\ & t_1^{\widehat{fp}'_{ik} \otimes \widehat{fp}'_{jl}} > t_1^{\widehat{fp}_i^{s^*} \otimes \widehat{fp}_j^{s^*}} \ \} \end{aligned} \qquad (11)$$

The set of new states $\mathcal{N}$ is created using

$$\begin{aligned} \mathcal{N} = \; & \{s' = old \cup new | \\ & old = s^* \setminus \{s_i^*, s_j^*\}, new \in \mathcal{S}^{fp_i^{s^*} \otimes fp_j^{s^*}}, \\ & \forall A_x \in \{A_i, A_j\} \ \forall A_y \in \mathcal{G} \setminus \{A_i, A_j\} : \\ & t_1^{\widehat{fp}_x^{new} \otimes \widehat{fp}_y^{old}} > t_1^{\widehat{fp}_i^{s^*} \otimes \widehat{fp}_j^{s^*}} \ \} \ . \end{aligned} \qquad (12)$$

New states $\mathcal{N}$ are added to $\mathcal{O}$ and the searching continues with expansion of the state with the smallest value according to the evaluation function.

By default the algorithm uses the zero heuristics. With general utility function for flight plan, it is possible to have evasion maneuvers that do not change the utility of flight plane and can be used for collision avoidance, so only zero heuristic is admissible [7]. For example assume we have a scenario where two planes have a 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 and in this case the utility value is the same is in the initial state.

The algorithm with zero heuristics finds the final state with the lowest value of evaluation function which corresponds to the lowest utility lost for the sum of utilities of flight plans. However, in the worst case the expanded state space can grow exponentially with the number of collisions in a multiparty group. We can also use different not admissible heuristic. Such as "collision" heuristic which would combine the main utility criterion with giving more preferences to the states with less collisions. The search process with such heuristics is faster, but its result can be far from the best possible utility.

6.1 Interaction of Multi-party groups

When the airplane is asked to join the multiparty group $\mathcal{G}_1$, it can be already participating in another multiparty group $\mathcal{G}_2$. In this case, the airplane checks if $\mathcal{G}_1$ is more important than $\mathcal{G}_2$ and if so, the airplane terminates its interaction with group $\mathcal{G}_2$ and joins the group $\mathcal{G}_1$. When an airplane terminates the interaction with the group, the group is dissolved. The relation of importance of groups is defined according to the earliest collision in their encounter. When the group $\mathcal{G}$ is searching for solution of the encounter $\langle t, \{fp_i\}_{A_i \in \mathcal{A}_C} \rangle$ then the time t_G of the soonest collision is

$$t_{\mathcal{G}} = \min_{A_i, A_j \in \mathcal{G}, A_i \neq A_j} t_1^{A_i \otimes A_j} \tag{13}$$

The group $\mathcal{G}_1$ *is more important* than $\mathcal{G}_2$ – $\mathcal{G}_1 \succ \mathcal{G}_2$ if and only if $t_{\mathcal{G}_1} < t_{\mathcal{G}_2}$. To make $\succ$ relation total, it must be defined also for situations when $t_{\mathcal{G}_1} = t_{\mathcal{G}_2}$, also it is not important if $t_{\mathcal{G}_1} < t_{\mathcal{G}_2}$ or $t_{\mathcal{G}_1} > t_{\mathcal{G}_2}$. It can be chosen for example randomly, or deterministically - with help of the lexicographic ordering of the string representation of the groups.

7. EXPERIMENTAL EVALUATION

The experimental evaluation and comparison of novel MPCA algorithm has been carried out within the system AGENTFLY. The AGENTFLY [9] implements the airspace domain specified in the Section 2 and provides the precise flight modeling of a large number of aircrafts. Each unmanned aerial vehicle is controlled by an autonomous agent which provides a flight plan updates for an execution, can use communication infrastructure and can subscribe for receiving local perceptions (existence and position from onboard simulated radar and transceiver equipment). The planning problem specified in the definition 1 is supported by the flight plan wrapping object with a highly optimized searching algorithm using a dynamic size of state space element building blocks. The changes to the flight plan are defined by the insertion and the alteration of the special control dummy way-points.

Both tested algorithms are implemented as modules of the *multi-layer collision avoidance* (MLCA) architecture [9] in AGENTFLY. They are configured to use the optimization criterion based on the maximization of the sum of utilities while utilities have only the component for the length of the flight plan. The MLCA provides implementation of exchange of local future flight plans $\widehat{fp}$, identification of encounters in local perspective and combines several methods by use of specified timeout which comes from an encounter. The AGENTFLY system has implemented two cooperative collision avoidance methods: *rule-based* (RBCA) and *iterative peer-to-peer* (IPPCA) collision avoidance (implemented as plug-ins). The IPPCA was chosen as a solution provider to which new algorithm is compared. In the Section 5 we've provided its description. The implementation is based on the concept of active and passive participants of PNRC process. The active one is responsible for making set $\mathcal{S}^{A_i \otimes A_j}$ and selection of the solution for $A_i \otimes A_j$. On the other hand, the passive participant provides its $\widehat{\mathcal{F}}$ and accepts the selected solution. The identification who is active and who is passive is based on natural ordering of the unique airplanes' ids.

We've implemented MPCA algorithm (Section 6) as a new plug-in in MLCA architecture of the system. It is implemented mainly as a *multiparty coordinator* agent who cooperates with participating airplane agents. One coordinator agent manages one multiparty group and resides on one of the airplanes contained in initial multi-collision which caused creation of the encounter and its corresponding multiparty group. The concept of the coordinator is used for simplification of the synchronization issues in the communication among airplanes. A multiparty coordinator is responsible for the state space expansion and searches for an optimal solution of multi-collision. During the expansion state it requests for the changed flight plans the airplanes in the group needed for the expansion of the state space. After finding the final state, it informs airplanes in the group that they may implement one of the generated flight plans.

7.1 Random Experiment

In the random experiment we performed a set of repetitive tests while collecting several characteristic properties for the comparison of both methods. The sequence of 900 runs (configuration with 5, 10 ... 90 airplanes each 50 times repetitively) for each method was carried out in the limited airspace area of 31 x 31 units. The mission M_i holds exactly two way-points randomly generated on the two opposite airspace borders, thus each airplane needs to fly across the square. All way-points have the same altitude during the whole experiment and each new airplane's way-points are generated on the adjacent borders of the square in clock-wise direction. Such generating scheme guarantees high number of collisions in the middle of the airspace. The safety zone size $rsz = 0.25$ units and the communication range $c = 10$ units are the same for all airplanes. The airplane flying speed can vary between 0.075 and 0.125 units per second. Totaly 85,500 airplanes were simulated during more than 230 hours of flight time. The collision-free solution was found in every simulation run – there is no collision between any two final airplanes' flight plans.

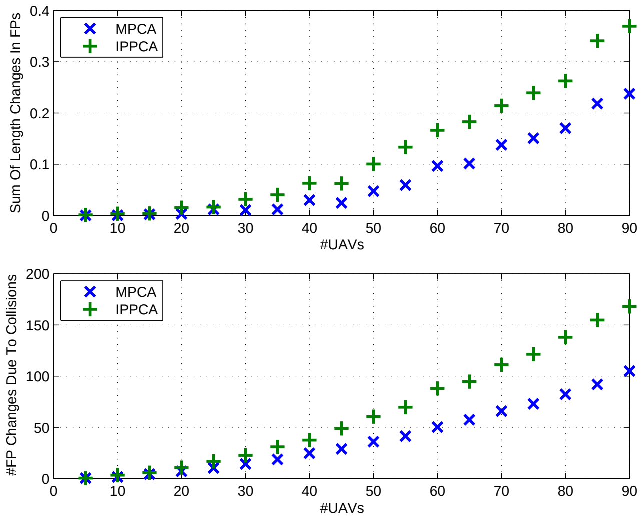

Figure 2: Top: The sum of differences between final collision-free paths and shortest regardless collisions. Bottom: The number of applied changes by all planes in particular experiment run. Presents average values from 50 repeated experiments.

The top chart in the Figure 2 presents the comparison of the average sum of all differences between the final collision-free flight plans and the shortest path from start to end way-point in given run. The MPCA solves the collisions using only 50 to 80 percentage of average flight plan length generated by the IPPCA in the same scenario. The value varies due to the fact that the same numbers and types of collisions during each randomized experiment are not guaranteed. The bottom plot displays the average sum of numbers of changes of flight plans applied by all airplanes in the given configuration. More changes occur in the experiment using the IPPCA due to its iterative nature (Section 5). The difference in the number of applied changes in a flight plan correlates with the difference depicted in the previous chart. This shows that there must exist multi-party groups $\mathcal{G}$ with more than two airplanes that participate in the MPCA.

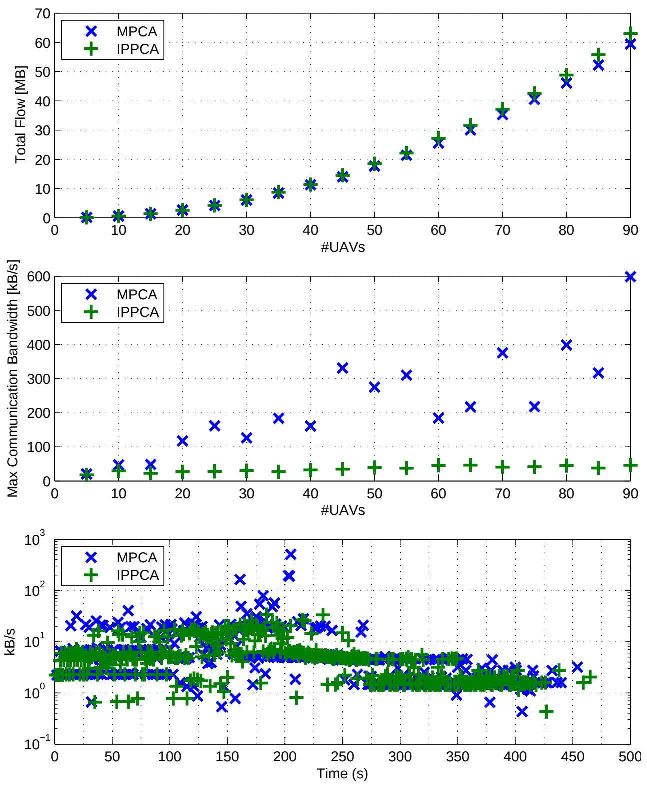

Figure 3: The communication flow analysis: total communication flow (top), maximum network bandwidth (middle) and network flow distribution over time in one specific run for 90 UAVs (bottom).

We've studied also the communication aspects of both algorithms, see Figure 3. Both algorithms have almost the same amount of transmitted bytes in total during the specific configuration, but there is huge difference in the flow distribution. The middle chart presents the maximal communication bandwidth depending on the number of airplanes in current configuration. This chart presents the overall maximum from all 50 repeated runs. It is apparent that the MPCA requires several times wider communication bandwidth than the IPPCA, especially for the configurations with more airplanes. The higher number for larger setup is given by the larger multi-party groups appearing in the collision area (the center of restricted airspace). The MPCA requires more communication during the state expansion phase within the coordination group but on the other hand it requires lower number of detected encounters. This leads to almost the same sum of total transmitted bytes among all aircrafts. The characteristic distribution of communication flow over time is shown in the bottom plot. The peeks around time 200 seconds are caused by the higher occurrence of multi-collisions (airplane needs around 400 seconds to fly to the opposite side). The computation requirements of both algorithms for the entire configurations have been also analyzed. The computation requirements are measured as a sum of time necessary to solve all collisions in one run. It has been measured on the same identical computer for both algorithms. The requirements are almost the same for both methods, but the distribution is different similarly as the network flow.

7.2 Specific Scenarios

The differences between both algorithms have been tested on two selected worse-case based scenario setups. In the first setup there are flying 13 airplanes located in the geometrical vertically oriented plane. Their position in the plane is shown in the Figure 4 left. Initially all of them fly in the same direction and at the same flight speed. There is another airplane which flies in the opposite direction and has a head collision with the airplane located in the middle of the first group. The final results comparing both collision avoidance methods are depicted on the right side in the Figure 4. When using the IPPCA only one plane avoids the group of airplanes and therefore no other planes participate in the solution. On the other hand, while using the MPCA the middle airplane in the group performs a combination of several flight plan changing manoeuvres and creates a small hole in the middle of the group flying in geometrical plane to let the opposite airplane fly through. Then the airplane goes back to its original central position within the group. The difference is given by the fact that the multi-party solution is found by the search for the solution with best criterion through the large space of possible combinations. In the IPPCA two negotiating airplanes must find solution of their collision and next negotiation cannot take already applied changes back. The MPCA provides a solution that is only 0.213 units longer than the initial flight plan while the IPPCA gives a solution that is 2.843 units longer. These values were calculated as an average from 20 consecutive experiments.

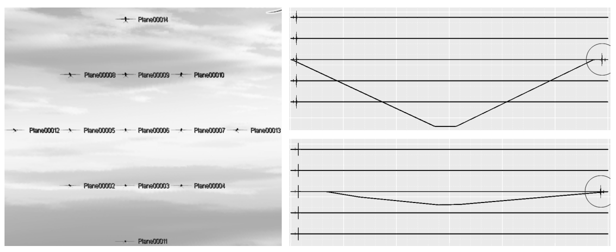

Figure 4: Left: the setup of the first test scenario. Right: the result after IPPCA (top) and MPCA (bottom) negotiations.

The second selected worse-case based scenario setup places ten airplanes equally to the horizontal circle. All of them start at the same flight altitude and want to fly to the opposite side of the circle through its center. Therefore in the center of the circle there is a multi-collision of all airplanes where each one has collision with all others. Both algorithms find a solution for the setup – final flight plans are not colliding together. The IPPCA produces the final flight plans 1.963 units longer then the initial ones and the MPCA provides solution only 1.441 units longer.

8. CONCLUSION

The novel multi-party approach addressing the problem of decentralized collision avoidance among agent-controlled autonomous airplanes participating in the free-flight operation is presented in the paper. The *multi-party* algorithm for the specified airspace domain is described and experimentally compared to the *iterative peer-to-peer* method. The algorithm utilizes the benefits of a centralized solution search and optimization but still respects the decentralized manner in the air domain with restricted communication range. Several airplanes having mutual collision are grouped together and perform state space generation and exploration for finding the best solution for the given criterion function. The multi-party group is dynamically changed if it is necessary or two groups can be merged into one.

The multi-party algorithm has been tested on two selected worse-case based scenario setups where it has been proved that it has the potential to provide the solution with better value of given criterion function than the peer-to-peer method. An extensive set of experiments has been carried out where the properties of both methods have been studied. In random experiment it was validated that the multi-party algorithm never provides worse solution than the iterative peer-to-peer in any configuration. For the setups with mutual collisions among several airplanes the multi-party solution is much better for the given criterion. The algorithms have been compared in their communication properties. The necessary communication total flow is the same for both methods. But the distribution of the communication flow over time is different. The proposed multi-party algorithm requires wider communication bandwidth among airplanes during the state space expansion phase. In our next work we would like to address the influence of the coordination group size to the state space expansion and to the time necessary to find a solution. We are thinking of implementation of a limit for group size and of definition of a membership function which will be used for the identification which airplane should be replaced by another one for the fully occupied groups. Such restriction should provide solution also for the scenarios with hundreds of airplanes where this unlimited version will not work due to the fast expansion of the searched state space and long searching time. In this case the current multi-party version will be timeouted because it will not provide the solution until the time specified in the encounter.

9. ACKNOWLEDGEMENT

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3073. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon[1]. Also supported by Czech Ministry of Education grant 6840770038.

10. REFERENCES

[1] DOD. Unmanned aircraft systems roadmap 2005-2030, 2005.

[2] 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.

[3] J. Krozel, M. Peters, K. D.Bilimoria, C. Lee, and J. S. 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.

[4] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 4(2):151–158, 1991.

[5] S. Parsons and M. Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

[6] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter.* The MIT Press, Cambridge, Massachusetts, 1994.

[7] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Series in Artificial Intelligence, Englewood Cliffs, New Jersey, 1995.

[8] R. Schulz, D. Shaner, and Y. Zhao. Free-flight concept. In *Proceedings of the AiAA Guidance, Navigation and Control Conference*, pages 999–903, New Orelans, LA, 1997.

[9] D. Šišlák, P. Volf, A. Komenda, J. Samek, and M. Pěchouček. Agent-based multi-layer collision avoidance to unmanned aerial vehicles. In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS '07): Modeling, Evolution and Engineering*, 2007.

[10] 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.

[11] 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.

[12] M. Wooldridge, editor. *An Introduction to MultiAgent Systems.* John Wiley and Sons Ltd, 2002.

[13] 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.

[1]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.