Agent-Based Cooperative Decentralized Airplane-Collision Avoidance

David Šišlák, Přemysl Volf, and Michal Pěchouček, *Member, IEEE*

***Abstract*—The efficiency of the current centralized air-traffic management is limited. A next-generation air transportation system should allow airplanes (manned and unmanned) to change their flight paths during the flight without approval from a centralized en route control. Such a scheme requires decentralized peer-to-peer conflict detection and collision-avoidance processes. In this paper, two cooperative (negotiation-based) conflict-resolution algorithms are presented: *iterative peer-to-peer* and *multiparty* algorithms. They are based on high-level flight-plan variations using evasion maneuvers. The algorithms work with a different level of coordination autonomy, respect realistic assumptions of imprecise flight execution (integrating required navigation performance), and work in real time, where the planning and plan-execution phases interleave. Both algorithms provide a resolution in a 4-D domain (3-D space and time). The proposed algorithms are evaluated experimentally, and their quality is studied in comparison with a state-of-the-art agent-based method—the *satisficing game theory* algorithm.**

***Index Terms*—Agent-based collision avoidance (CA), air-traffic control, conflict resolution, cooperation, distributed coordination, optimization.**

I. Introduction

TODAY, air-traffic management (ATM) uses a centralized air-traffic control, which is responsible for safe flight separation. Each airplane is required to provide the traffic control with its suggested flight plans (FPs) covering the entire flight trajectory before takeoff. Traffic control authorizes the flight before takeoff. During the flight, the airplane is in permanent contact with the respective traffic-control centers in the appropriate air sector [1]. The responsibility for airplanes' separation lies on the traffic controllers who do not optimize airplanes' trajectories with respect to their individual priorities but only provide safety separation for them. By this, unnecessary fuel is wasted, and atmospheric pollution increases [2]. In 2005, 322 272 h of delays were estimated within the U.S. airspace with a total cost estimated to exceed $3 billion [3]. Although the growth of the air-transport demand has been reduced by the global financial crisis in the last two years, the demand for air transportation will be increasing in the few next decades. In [4], Boeing anticipates that cargo will triple over the next 20 years. Thus, the current system cannot guarantee that it will work efficiently and provide safe control in the future, particularly when the number of flights is increasing.

In a large research program studying Next-Generation Air Transportation Systems [5], the *free-flight* concept [6] for the en route part of the flight is addressed—the airplane can fly freely according to its own priorities, and only its operations in the terminal parts are coordinated with ground-based ATM. The free-flight concept relaxes the requirements for permanent contact with the air-traffic controllers, and the separation is achieved by negotiation among airplanes. The free-flight approach requires decentralized algorithms providing collision detection and avoidance for the flight. The same mechanism is addressed in the Unmanned Aerial System domain where each airplane is required to implement an automatic *See-and-Avoid* capability [7] to provide safe operation from the collision-avoidance (CA) perspective. The most challenging task is a mixed operation of both manned (either civilian or military) and unmanned airplanes in the same airspace.

This paper presents a decentralized CA approach compatible with the free-flight concept considering the following realistic assumptions for ATM. Airplanes cannot stop flying while the algorithms detect potential conflicts on the airplanes' trajectories and searching for conflict resolution. Thus, the CA algorithms have to consider that the airplanes are flying while collisions are being resolved. Conflict resolution can utilize all degrees of freedom of airplane control—the heading, altitude, and cruise speed of the airplane can be altered. All these control actions can be combined together but still need to respect the physical constraints of the airplane as given by its flight dynamics. A conflict-resolution maneuver has to respect the airspace that is restricted by the existing no-flight zones (also known as special-use airspaces (SUAs) [1]). The proposed conflict-resolution maneuver is executed imprecisely by the airplane due to imprecise position sensors, very complex control systems, and external factors such as weather conditions. Airplanes are equipped with communication transceivers with limited transmitting power. Communication links among airplanes can be established, but only within the adequate range.

The approach presented in this paper provides agent-based cooperative decentralized airplane CA—each agent provides control for one airplane. Conflict detection is based on the concept where located airplanes nearby share their limited

Manuscript received July 1, 2008; revised March 12, 2010 and June 21, 2010; accepted June 28, 2010. Date of publication July 26, 2010; date of current version March 3, 2011. This work was supported in part by the Air Force Office of Scientific Research, Air Force Material Command, U.S. Air Force, under Grant FA8655-06-1-3073, and in part by the Czech Ministry of Education under Grant 6840770038. 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 authors 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 Associate Editor for this paper was S. C. Wong.

The authors are with the Agent Technology Center, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, 166 27 Prague, Czech Republic (e-mail: sislakd@fel.cvut.cz).

Digital Object Identifier 10.1109/TITS.2010.2057246

future intentions in the form of partial future FPs providing 3-D descriptions of their positions in time. Such a concept guarantees the privacy of the airplanes' future plans. The key difference of this from all other CA approaches is the extensible component-based approach to airplane control consisting of a flight executor, CA, and flight-trajectory-planning components. This paper presents two conflict-resolution algorithms: 1) *iterative peer-to-peer CA* (IPPCA) and 2) *multiparty CA* (MPCA). Both algorithms provide high-level variation of flight trajectories using conflict-resolution waypoints (WPs) and delegate trajectory planning to the planning component. IPPCA solves colliding trajectories of several airplanes iteratively by pairwise negotiation (PN), where the most important collision is removed in each iteration. MPCA searches for the best combination of several evasion maneuvers among multiple airplanes. Both conflict-resolution algorithms were introduced in our previous conference paper [8]. In this paper, the algorithms are presented in relation to the component-based airplane-control approach considering realistic assumptions for ATM. The properties of the algorithms are experimentally evaluated within a new set of experiments and compared against the satisficing game theory algorithm (SGTCA) [9].

The rest of this paper is organized as follows. Section II summarizes and discusses the work related to this paper. Section III presents a component-based control approach to the CA and introduces a basic notation and interactions among components. Section IV introduces conflict detection based on sharing of local flight intentions among airplanes. Section V presents the common parts of conflict-resolution algorithms, which are then used in both presented algorithms. Section VI describes the iterative peer-to-peer conflict-resolution algorithm, and Section VII provides a description of the multiparty conflict-resolution algorithm. Finally, Section VIII documents all experiments and results.

II. Related Work

Collision detection and avoidance methods supporting the free-flight concept are widely addressed by the research community. The presented approaches differ in several aspects, such as their centralized and decentralized nature, different levels of coordination, or the set of control actions used for CA. Several approaches [10]–[14] formulate conflict resolution as a game. An evader tries to avoid a collision against all possible pursuer's maneuvers. Such game-theoretic approach is very useful for noncooperative cases where airplanes cannot communicate together. For our case where airplanes can communicate together, these methods become inefficient because they cannot be used for mid- and long-term conflict resolution where possible pursuer's maneuvers cover large volumes of airspace.

Field approaches to CA [15]–[18] are very close to a reactive-control mechanism. Based on various kinds of information, these algorithms prepare a field or components that are used for field construction. A field-based CA algorithm generates an airplane's control actions depending on its current state with respect to the field. Such approaches require computationally intensive preparation of the field when the input for the computation is updated. Similar to game approaches, these methods are not suitable for a long-term airplane operation optimization.

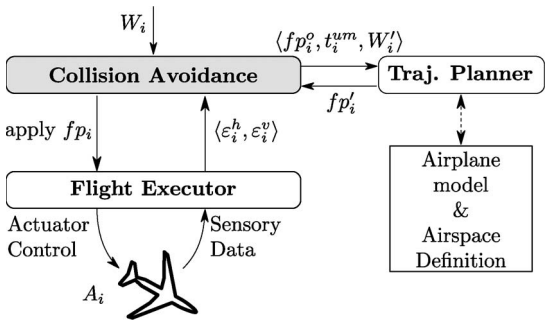

Fig. 1. Airplane-control approach: Components and their interactions. The CA component is addressed in this paper.

Optimization approaches [19]–[22] define conflict resolution as an optimization problem that searches for airplanes' controls. With an increasing number of airplanes, the problem becomes computationally unsolvable, and a limited time horizon is employed. It is fairly complicated to integrate airspace limitations (SUAs, excluded areas, and ground surface separation) into these methods. In contrast to the presented approaches, all airplanes have to provide their preferences and constraints as input to the optimizer.

Rong *et al.* [23] described a cooperative agent-based solution based on constraint-satisfaction problems. Conflicting airplanes negotiate pairwise until a mutually acceptable solution is found. The proposed algorithm can fail, and no solution is provided. In such situation, the centralized ground controller forces its own resolution. This approach cannot be used for the free-flight concept where there is no such high-level coordinator.

Wollkind *et al.* [24] proposed a solution for a two-airplane conflict using the monotonic concession protocol (MCP) [25]. This approach has been used as a motivation for the presented iterative peer-to-peer algorithm where it is extended for multi-airplane collisions, and a modification of the flight trajectory is formulated as a high-level evasion maneuver that utilizes the flight-trajectory planner.

Archibald *et al.* [9] used an agent-based approach based on the satisficing game theory [26] with dual social utility: selectability (characterizing effectiveness) and rejectability (characterizing resource sharing). They proposed a satisficing extension where airplanes take into consideration the preferences of others as well. This decentralized agent-based solution to CA is used in the experiments evaluating the presented approach.

Mao *et al.* [27], [28] addressed conflict resolution for multiple airplane flows intersecting at a fixed point. They proved that the resolution scheme based on a minimal heading change upon entering the airspace provides a stable solution to this conflict. The presented scenario featuring intersecting airplane flows is very interesting and is used in this paper for evaluation of the presented algorithms.

III. Component-Based Control Approach

Each airplane $A_i \in \mathcal{A}$, where $\mathcal{A}$ is a set of all airplanes operating in the airspace, contains several components providing airplane control (see Fig. 1). There is no requirement for any

centralized controller in this approach to provide safe flight in its *en route* part. There are two important data structures that play a crucial role in the component approach: the *WP sequence* and the FP.

The WP sequence $W_i = \langle w_0, w_1, \ldots, w_m \rangle$ is an ordered list of *WPs*. The order of WPs defines the order in which these WPs have to be fulfilled by A_i. Each WP specifies a geographical position, altitude, and optional constraints. The following constraints can be defined for each WP: 1) the time interval when A_i is requested to fly through it; 2) the desired velocity of flight at the WP; and 3) the orientation under which A_i has to fly through the WP. The position of a WP is not limited to the position of navigation aids [1] only but can also be defined freely utilizing all available navigation equipment for *Area Navigation* (RNAV) [1].

An FP $\mathrm{fp}_i = \langle \langle t_0, \overline{c_i}(t_0) \rangle, e_0, e_1, \ldots \rangle$ is a sequence of FP elements e_k following an initial configuration $\langle t_0, \bar{c}_i(t_0) \rangle$, defining an initialization time t_0 and an initial A_i's state $\bar{c}_i(t_0)$ (position, orientation, and velocity). fp_i provides a precise description of the movement of A_i's pivot.[1] Each e_k uniquely describes the geometrical shape of the trajectory part including the cruise-speed control. Any fp_i is valid with respect to the specified W_i if and only if the path given by fp_i is smooth (continuous in its first derivative) in all coordinates, the path respects airplane-flight model constraints, all positions on the path are within the A_i's operating airspace (a limited area with parts excluded), and the WPs in W_i are fulfilled (satisfied) in the given order. A WP is fulfilled at time t with fp_i if and only if all its mandatory (position and altitude) and all its specified optional constraints (time, velocity, and/or direction) are matched by fp_i at time t, and all previous WPs are fulfilled before t.

The quality of fp_i is evaluated by the cost function $\mathbf{cost}_i(\mathrm{fp}_i)$, where a lower cost value corresponds to a better FP. The construction of $\mathbf{cost}_i(\mathrm{fp}_i)$ integrates A_i's priorities for its operation. Typically, $\mathbf{cost}_i(\mathrm{fp}_i)$ is constructed as a multicriterion cost function expressed as a weighted sum of selected components: to be as short as possible, to use minimum fuel, to have minimum curvature, and to have as few altitude changes as possible.

In addition to WP and FP structures, the *unchangeable part marker* t_i^{um} and *flight execution performance* $\langle \varepsilon_i^h, \varepsilon_i^v \rangle$ are used. t_i^{um} uniquely identifies the position in FP from which the FP can be altered. t_i^{um} is the crucial element that makes the control approach usable for real-time airplane control. The components perform their internal actions while considering that A_i is still flying. fp_i, which is modified by the CA algorithm, can be executed only when the state defined by the previous FP is the same as the state defined in fp_i for the time of fp_i's application. $\langle \varepsilon_i^h, \varepsilon_i^v \rangle$ defines the horizontal and vertical flight-trajectory-tracking accuracies. The actual airplane's position should be within these boundaries around the position defined by fp_i. Tracking inaccuracy is caused by errors of an airplane's onboard sensory equipment and actuators and by external factors such as weather. The concept of execution accuracy level is defined in the *required navigation performance* (RNP) standard [29].

[1]The A_i's pivot is positioned in the center of gravity of A_i.

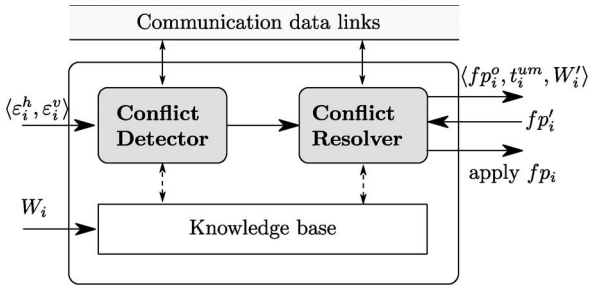

Fig. 2. CA concept.

A. Flight Executor

The flight executor implements autopilot functions to track the requested flight trajectory fp_i as precisely as possible. It is known as the *flight-management system* (FMS) [1]. Based on the required flight trajectory in fp_i and sensory data (position, altitude, speed, and orientation), it provides control for the A_i's actuators (ailerons, elevators, rudder, thrust power, and optional secondary actuators). The FMS is aware of sensory and tracking errors and, thus, is able to express its execution performance in $\langle \varepsilon_i^h, \varepsilon_i^v \rangle$. The FMS can report updated performance data if the previous value is no longer valid, e.g., A_i enters an area with high turbulence. The design of FMS is tightly coupled with the construction of the airframe and is not in the scope of this paper.

B. Trajectory Planner

The trajectory-planner component performs replanning of the flight trajectory. The planner works directly with the airplane's model, its constraints, and airspace definition. The replanning task $\langle \mathrm{fp}_i^o, t_i^{um}, W_i' \rangle$ contains fp_i^o, which should be modified from the specified unchangeable part marker t_i^{um} with respect to a WP sequence W_i' that determines where the resulting FP should fly through. Based on the replanning task, the planner searches and returns the modified fp_i' that has the minimum $\mathbf{cost}_i(\mathrm{fp}_i')$ and is valid with respect to W_i'. The valid FP was introduced earlier. The part of fp_i' up to t_i^{um} is the same as in the original fp_i^o. If it is not able to finish the replanning task (i.e., it cannot find a valid fp_i' with respect to W_i'), the planner reports a planning failure. The description of such a planner component is not provided in this paper. Details about the planner can be found in [30].

C. CA

The CA component provides decentralized collision detection and resolution for a free-flight operation (see Fig. 2). A finite time horizon δ_t for which each airplane detects and solves collisions is specified. δ_t is defined for all airplanes, and all airplanes use the same value. Depending on the selection of δ_t, each A_i solves conflict in a short-term, midterm, or long-term horizon. δ_t specifies the length (timewise) of future intents shared by the airplanes.

The CA components hosted on different airplanes can communicate together via reliable wireless-communication data links. The transmission power of the communication equipment is set so that the communication range is equal to the distance that A_i can fly at the maximum cruise speed over the δ_t time

interval. The onboard communication equipment allows establishing bidirectional connection between two airplanes when the airplanes are mutually within each other's communication range. Each bidirectional connection preserves the ordering of messages—messages are delivered in the same order as they are sent. The communication equipment also provides notification when a new airplane appears within the range or an existing one disappears. There is one external input W_i specifying the mission of A_i. To change the FP, CA instantiates a replanning task and invokes the trajectory planner. WPs in W_i have to be provided in each replanning task in an unchanged order.

IV. Conflict Detection

For purposes of conflict detection, a horizontal separation h and a vertical separation v are defined. These values are predefined and are used by all airplanes. Typically, the horizontal separation is 5 nmi, and the vertical separation is 2000 ft above flight level (FL) 290 and 1000 ft below FL 290, as defined in [1]. It is possible to have different separation limits based on particular airplane types, but for the sake of simplicity, this is not considered in this paper. Conflict detection is implemented by sharing limited future flight trajectories among airplanes. The *local part of an FP* (LFP) fp_i is denoted as $\widehat{\mathrm{fp}}_i$. $\widehat{\mathrm{fp}}_i$ defines the planned positions of A_i within the time interval given by the current time t_{now} and $t_{\mathrm{now}} + \delta_t$, and these positions are the same as in the original fp_i. LFP sharing is implemented by the subscribe–advertise protocol [31].

When the conflict detector (CD) of A_i receives a notification that a communication link with A_j has been established, A_i subscribes for $\widehat{\mathrm{fp}}_j$ and $\langle \varepsilon_j^h, \varepsilon_j^v \rangle$. By accepting a subscription request from A_i, A_j provides regular updates of its $\widehat{\mathrm{fp}}_j$. If A_j changes its fp_j for any reason (e.g., due to a change of W_j or as a result of its conflict resolution), A_j provides updated $\widehat{\mathrm{fp}}_j$ to A_i as soon as possible. A_j provides $\langle \varepsilon_j^h, \varepsilon_j^v \rangle$ upon subscription and when the A_j's flight executor detects new flight-execution performance $\langle \varepsilon_j^h, \varepsilon_j^v \rangle$. A_i stores the last known $\widehat{\mathrm{fp}}_j$ and $\langle \varepsilon_j^h, \varepsilon_j^v \rangle$ for each airplane in its knowledge base.

When A_i receives $\widehat{\mathrm{fp}}_j$ or $\langle \varepsilon_j^h, \varepsilon_j^v \rangle$ from any airplane located within its communication range, or its own fp_i or $\langle \varepsilon_i^h, \varepsilon_i^v \rangle$ is updated, A_i performs a collision inspection. Airplanes A_i and A_j have colliding flight trajectories, which are denoted as $A_i \otimes A_j$, if and only if there exists a point in time where the 3-D positions defined by fp_i and $\widehat{\mathrm{fp}}_j$ are horizontally closer than $r + \varepsilon_i^h + \varepsilon_j^h$ and vertically closer than $h + \varepsilon_i^v + \varepsilon_j^v$ at the same time (see Fig. 3). For $A_i \otimes A_j$, the collision start time $t_1^{A_i \otimes A_j}$ and the collision end time $t_2^{A_i \otimes A_j}$ of the first collision between fp_i and $\widehat{\mathrm{fp}}_j$ are identified.

The information about colliding flight trajectories is stored in A_i's knowledge base, and the conflict resolver (CR) is notified. Collision inspection may conclude that the trajectories do not collide. In such a case, the collision flag is cleared in the knowledge base, and the CR is notified as well. The same collision identified by A_i is also identified by the respective airplane A_j because it receives $\widehat{\mathrm{fp}}_i$ and performs the collision inspection between fp_j and $\widehat{\mathrm{fp}}_i$ and, thus, identifies the same collision.

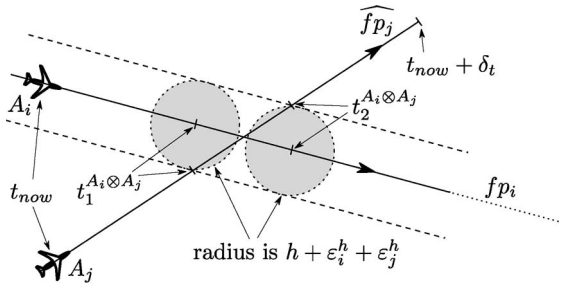

Fig. 3. Collision detection from A_i's perspective. Both A_i and A_j are flying at the same altitude.

V. Conflict Resolution

The CR component integrates conflict-resolution algorithms. Within each airplane, a variety of different conflict-resolution algorithms can be integrated using *multilayer CA framework* [32]. For the sake of simplicity of description, only one algorithm is present in each CR, and all airplanes have the same algorithm. Two different conflict-resolution algorithms are presented in this paper: IPPCA and MPCA algorithms. Both algorithms internally use cost values for the selection of the best flight trajectories resolving a conflict. Variation of flight trajectories as well as expression of their cost value is performed by the airplanes whose trajectories are conflicting. A_i generates a modification of fp_i considering its mission WPs W_i and all flight-model constraints and its operational airspace. However, each time, it shares only a limited part of its future intent $\widehat{\mathrm{fp}}_i$ with others. A_i uses the same cost function $\mathbf{cost}_i(\mathrm{fp}_i)$ for trajectory planning and the evaluation of the flight variant for CA. The cost value is computed from the whole fp_i and is shared along with $\widehat{\mathrm{fp}}_i$ from which it cannot be computed as there is only a limited part of the flight trajectory.

Conflict-resolution algorithms work with the unchangeable time marker t_i^{um}. t_i^{um} is set by the multilayer CA framework. For the purpose of this paper, there is a preconfigured time interval τ for each algorithm, which is the same for all airplanes.[2] t_i^{um} is set as $t_{\mathrm{now}} + \tau$ once the conflict-resolution algorithm is started. The lower limit for τ is defined as the maximum duration required for the conflict-resolution algorithm between its start and the time when a new FP for application is provided. If the value of τ is selected below this limit, it can happen that the provided FP is not applicable because the airplane already follows the trajectory specified in the previously valid FP, which is no longer present in the new one. The upper limit for τ is given by δ_t. If $\tau > \delta_t$, no airplane is able to solve any collision because the time of an identified collision $t_1^{A_i \otimes A_j} \leq t_{\mathrm{now}} + \delta_t$ (δ_t is equal to the length of $\widehat{\mathrm{fp}}_j$), and the flight trajectory can be altered after $t_i^{um} = t_{\mathrm{now}} + \tau$. These two equations cannot be satisfied if $\tau > \delta_t$.

Both presented algorithms use a set $\mathcal{M}$ of *evasion maneuvers* to generate modified flight trajectories. There are seven basic evasion maneuvers defined $\mathcal{M} = \{\mathbf{m}_L, \mathbf{m}_R, \mathbf{m}_U, \mathbf{m}_D, \mathbf{m}_F, \mathbf{m}_S, \mathbf{m}_0\}$: turn-left, turn-right, climb-up, descend-down, fly-faster, fly-slower, and no-change

[2]For all experiments in this paper, $\tau = 2$ s.

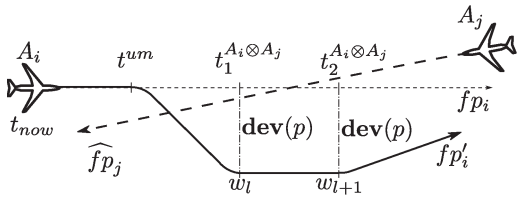

Fig. 4. A_i applies the turn-right evasion maneuver at the place of the first identified collision $A_i \otimes A_j$ — $\mathrm{fp}_i' = \mathbf{m}_R(\mathrm{fp}_i, W_i, p, t^{um}, t_1^{A_i \otimes A_j}, t_2^{A_i \otimes A_j})$.

evasion maneuver.[3] An evasion maneuver is a function $\mathbf{m}_k(\mathrm{fp}_i, W_i, p, t_{um}, t_1, t_2)$, where fp_i is supposed to be modified, W_i is the WP sequence defining the mission for the airplane, $p \in \{1, 2, 3, \ldots\}$ is the evasion-maneuver parameterization, t_{um} defines the position in fp_i from which it can be modified, and times t_1 and t_2 define the position where the evasion maneuver should be applied. The function internally uses WPs that are inserted at appropriate positions in W_i and constructs a replanning task for the trajectory planner. The result of replanning is then returned as a result of the evasion-maneuver function. The evasion-maneuver function can fail if the maneuver cannot be applied. All maneuvers can be applied only if $t_i^{um} < t_1 < t_2$. The no-change maneuver is defined as $\mathbf{m}_0(\mathrm{fp}_i, p, t_i^{um}, t_1, t_2) = \mathrm{fp}_i$; it returns the unmodified input fp_i and is used for simplification of cases where A_i can also leave its FP unmodified as a result of conflict resolution.

The application of a turn-right evasion maneuver to the fp_i is shown in Fig. 4. fp_i is modified beyond t^{um} so that the new fp_i' passes through newly introduced WPs w_l and w_{l+1} specified by shifted positions in fp_i defined for $t_1^{A_i \otimes A_j}$ and $t_2^{A_i \otimes A_j}$ to the right side perpendicular to the direction vectors at those positions. The size of the shift is given by the parameter p—for larger p, the evasion maneuver makes a greater shift $\mathbf{dev}(p)$, and *vice versa*. $\mathbf{m}_L$, $\mathbf{m}_U$, and $\mathbf{m}_D$ are similarly defined to $\mathbf{m}_R$, only with different shift directions for WPs w_l and w_{l+1} to the appropriate side. $\mathbf{dev}(p)$ is defined as $ph/2$ for horizontal maneuvers $(\mathbf{m}_R, \mathbf{m}_L)$ and $pv/2$ for vertical maneuvers $(\mathbf{m}_U, \mathbf{m}_D)$.

The velocity-changing maneuvers $\mathbf{m}_F$ and $\mathbf{m}_S$ use only one WP w_l, which is positioned at the position defined in fp_i for $t_1^{A_i \otimes A_j}$. w_l is defined along with a time constraint—which is *not earlier* for $\mathbf{m}_S$ and *not later* for $\mathbf{m}_F$. The differences between time constraints and $t_1^{A_i \otimes A_j}$ are specified by p—a larger p produces a greater difference, and *vice versa*. The difference is given by $ph/(2v_1)$, where v_1 is the velocity defined by fp_i at $t_1^{A_i \otimes A_j}$. All evasion maneuvers are restricted by a constrained airplane model and an operational airspace that are integrated in the trajectory planner. Sequential application of evasion maneuvers produces flight-trajectory changes that are not covered by the limited set $\mathcal{M}$. The implementation of evasion maneuvers guarantees that $\mathbf{cost}_i(\mathbf{m}_k(\mathrm{fp}_i, p, t^{um}, t_1, t_2))$ is a nondecreasing function of parameter p. The conflict-resolution algorithms are described in the next two sections.

[3]Each airplane can have different set $\mathcal{M}$, but in this paper, the same set is used for all airplanes.

VI. Iterative Peer-to-Peer Collision Avoidance

The IPPCA algorithm removes collisions during PN. Thus, a multiairplane collision is solved completely after several runs of IPPCA (several PNs). If more than three airplanes are involved in a collision, two parallel PNs between different airplanes can run because the algorithm has no centralized component. For collisions with a higher number of airplanes, the quality of the final conflict resolution depends on the order of PNs because each pairwise solution leads to an application of a new FP, which cannot be fully reverted back in any of the next iterations as the airplanes are moving all the time. However, the undefined order of PNs leads to the aforementioned issue only if there exist several identical first-collision times for different pairs of airplanes. For different first-collision times, the collisions are resolved in the order as given by those times.

A_i receives a notification from its CD every time the knowledge about detected collisions is updated. Initially, when A_i is not running PN, A_i selects the most important opponent A_j whose $\widehat{\mathrm{fp}}_j$ produces the earliest first-collision time $t_1^{A_i \otimes A_j}$. It can happen that more LFPs of different airplanes produce collisions at the same time. In such case, A_i selects its most important opponent A_j randomly from the set of these airplanes. Then, A_i sends the request for PN to A_j. A_i can receive such request from A_j, too. Once A_i receives such request from A_j, A_i has already updated the LFP of A_j, and thus, A_i has identified their mutual collision as well because bidirectional communication links preserve the order of messages. In such case, A_i checks whether it is willing to negotiate with the sender of the request. If so, PN is established. If not, it rejects the request. If the request of A_i is rejected by the other side, A_i selects another opponent from the set of airplanes whose LFPs produce the earliest collision time. If there is no other airplane producing a collision with the same earliest time, A_i does nothing and waits for a new notification from its CD or an incoming request for PN. It can happen that a new collision is identified while A_i is running PN, and this collision occurs earlier than the collision time of the collision for which PN is currently running. In such case, the current negotiation is interrupted, the other airplane from the interrupted negotiation is notified, and A_i sends the new negotiation request to the airplane whose LFP produces the most important collision.

```
{1}  p^rnd ← 1;
{2}  F_i ← ∅;
{3}  repeat
{4}      F_i ← F_i ⋃ PrepareEvasions(fp_i, W_i, p^rnd, t_1^{A_i⊗A_j},
         t_2^{A_i⊗A_j}, M);
{5}      SendAlternatives(F_i, A_j);
{6}      F̂_j ←
         WaitForOpponentsAlternatives(A_j);
{7}      S_i^{A_i⊗A_j} ← PrepareNegotiationSet(F_i, F̂_j);
{8}      p^rnd ← p^rnd + 1;
{9}  until S_i^{A_i⊗A_j} ≠ ∅;
{10} fp_i' ← SelectSolution(S_i^{A_i⊗A_j}, A_j);
{11} ApplyFlightPlan(fp_i');
```

Algorithm 1: Pairwise negotiation over $A_i \otimes A_j$ from A_i's perspective.

The pseudocode of the PN is in Algorithm 1. The PN works in *negotiation rounds* (NR) (lines 3–9) until a solution is found. For each NR, a unique parameter p^{rnd} is defined. In NR, A_i prepares a set of trajectory alternatives along with their costs, which are inserted into the set $\mathcal{F}_i$ (line 4). The function `PrepareAlternatives` returns a set of tuples holding a flight trajectory and its cost evaluated by $\mathbf{cost}_i$. Flight trajectories are prepared using the set $\mathcal{M}$ where evasion maneuvers are parameterized by p^{rnd}. Based on the maximum cost value of all flight trajectories generated using p^{rnd}, flight trajectories are also generated with a higher parameterization p whose costs are lower than or equal to that identified maximum cost. This mechanism removes the issue with different cost increases for different evasion maneuvers. Such issue can result to a situation that the applied FP is not the best one capable of removing $A_i \otimes A_j$ because a better FP may exist for another evasion maneuver with a higher parameterization p. `PrepareAlternatives` returns only FPs that do not produce collisions that are earlier than $t_1^{A_i \otimes A_j}$ with known LFPs of airplanes other than A_j.

The updated set $\mathcal{F}_i$ is then sent to A_j, but only LFPs that define flight trajectories for the next δ_t interval are sent along with their cost values (line 5). A_i keeps the whole FPs in $\mathcal{F}_i$ so that the selected one can be applied. Then, A_i waits for alternatives prepared by A_j (line 6). The negotiation set $\mathcal{S}_i^{A_i \otimes A_j}$ is prepared as a Cartesian product of $\mathcal{F}_i$ and $\widehat{\mathcal{F}}_j$ (line 7) and contains only those combinations of FPs that do have mutual collision time that is earlier than or equal to $t_1^{A_i \otimes A_j}$. NRs are repeated with higher p^{rnd} until the negotiation set contains at least one combination.

At this moment, both A_i and A_j have the same negotiation set, but A_j has full FPs for itself and local parts for A_i. Both airplanes propose a solution based on the social welfare criterion [33] where they select the candidate from the set of variants that have the lowest sum of FP costs (line 10). If there exist more candidates with the same value of the selection criterion, each airplane randomly proposes a solution from these candidates. To agree with one of two different randomly proposed solutions, A_i and A_j use a protocol based on a commitment scheme known from cryptography [34]. Finally, the FPs of the selected candidate are applied and distributed to the subscribers via their CDs. The CDs then recheck the applied flight trajectories for collisions with other airplanes, and IPPCA is invoked again.

Infinite pairwise iterations among airplanes are blocked by the criteria integrated in functions `PrepareAlternatives` and `PrepareNegotiationSet`, where only FPs that do not cause collisions at times that are earlier than or equal to the one being solved are included. In [35], a formal analysis of IPPCA running for a multiairplane conflict configuration where airplanes are configured to work only with a limited degree of freedom for conflict resolution is provided. It is proved that pairwise iterations end in a stable solution. For this constrained case, a worst-case estimation of the number of required PNs and the occupied area after the resolution based on the number of airplanes in the multiairplane collision has been derived. For all degrees of freedom (heading, altitude, and cruise speed), the resolution becomes simpler as there are more options for conflict resolution, but we are not able to do a formal analysis yet.

IPPCA uses the social welfare criterion minimizing the overall cost for conflict resolution in the function `SelectSolution`. It can be configured to provide solutions for self-interested airplanes where airplanes optimize their own costs instead of the overall cost. In this case, the MCP [25] can be used—the protocol for automated agent-to-agent negotiations in cooperative domains. In this case, airplanes leave only *pareto-optimal* candidates in the negotiation set $\mathcal{S}^{A_i \otimes A_j}$, and airplanes can use the *Zeuthen strategy* [25] to find an acceptable combination for both agents without iterations in the MCP.

VII. Multiparty Collision Avoidance

The MPCA algorithm is based on creation of groups of airplanes with mutually related collisions on their existing flight paths or on considered flight paths detected during the search. In contrast to IPPCA, MPCA does not resolve multiairplane collisions using the cascade effect where the removal of a collision and an update of FPs cause new subsequent collisions. In MPCA, airplanes in the group solve their multiairplane collision within one algorithm run. As shown in experiments, such approach provides better utilization of the airspace because its internal search finds the optimal solution with respect to the selected criterion. The optimization criterion is constructed as the minimization of the overall cost of flight trajectories for all involved airplanes in an MPCA run. The group of airplanes in MPCA is created dynamically during its run. Each airplane can act in two roles within MPCA: 1) a conflict-resolution coordinator (CRC) and 2) a group participant (GP). For one MPCA run, one airplane can act as CRC and GP at the same time, and all others act as GP only. For each MPCA run, the time of the earliest collision for which it was initiated, which is denoted as $t_{\mathcal{G}}$, is defined.

The CD part of CA is the same as for IPPCA. MPCA is notified about changes in the airplane's local knowledge base where LFPs of adjacent airplanes and identified collisions are stored. If A_i is not participating in any MPCA, A_i selects an airplane A_j whose $\widehat{\mathrm{fp}}_j$ generates the earliest collision (if there are several options, a random one is selected). A_i becomes CRC and requests A_j to participate in its MPCA group, which is denoted as $\mathcal{G}_i$. $t_{\mathcal{G}_i}$ and $\|\mathcal{G}_i\|$ are included in this request.[4] After successful acceptance, A_i asks another airplane and so on until all airplanes whose LFPs collide with fp_i are in $\mathcal{G}_i$. A_i can also receive such a participation request at any stage from another airplane A_k. A_i can be in the following states when it receives such a request from A_k.

1) A_i is not participating in any group $\rightarrow$ A_i accepts the request and becomes GP for $\mathcal{G}_k$.
2) A_i is GP in another MPCA group $\rightarrow$ A_i responds with the contact to A_i's group CRC.
3) A_i is CRC, and it has previously sent a request to A_k regarding a collision in their current FPs (both A_i and

[4] $\|\mathcal{G}_i\|$ denotes the size of the MPCA group $\mathcal{G}_i$.

A_k try to be CRC for the same collision) $\rightarrow$ A_i and A_k use the protocol based on the commitment scheme known from cryptography [34] to agree upon who will remain CRC and who will become GP only,

4) A_i is CRC, and there is no collision between A_i and A_k in their current FPs in the knowledge base (A_k tries to involve A_i into A_k's MPCA because there is a collision between $\widehat{\mathrm{fp}}_i$ and LFP of A_k in the state space MPCA run) $\rightarrow$ A_i and A_k compare $t_{\mathcal{G}_i}$ and $t_{\mathcal{G}_k}$, the airplane with a lower value remains CRC, and the other destroys its group and becomes GP. When the considered times are the same, $\|\mathcal{G}_i\|$ and $\|\mathcal{G}_k\|$ are compared (the larger one has priority). If the sizes of the groups are the same, the commitment scheme is used to decide which one will remain CRC.

All airplanes accepting to be GP provide the cost value for their current flight trajectory. A_i, acting as CRC, can receive a response from A_j informing that A_j is participating in another MPCA group where A_l is CRC. In such case, A_i contacts A_l and ask it to release A_j from $\mathcal{G}_l$. $t_{\mathcal{G}_i}$ and $\|\mathcal{G}_i\|$ are sent along with the request. The same resolution scheme as in the fourth option earlier is used to decide who will succeed and remain CRC. Suppose that A_i wins and $\mathcal{G}_l$ is destroyed. A_j then becomes GP of $\mathcal{G}_i$. At this time, A_i has all airplanes whose local flight trajectories collide with fp_i in $\mathcal{G}_i$. A_i also checks whether there is a collision between FPs of the airplanes included in $\mathcal{G}_i$ and those which have yet to be included. If such a collision is found, the respective airplane is asked to join $\mathcal{G}_i$ using the same mechanism, as previously described.

Suppose that A_i is CRC and $\mathcal{G}_i$ contains all airplanes that have mutual collisions in their current local flight trajectories. MPCA does not work as a centralized CA solver. Several conflict-resolution runs with disjoint groups can be active at the same time. Such groups can overlap during the MPCA run where a generated variant of a flight trajectory collides with a nonincluded airplane. MPCA search is motivated by the A* algorithm [36] and works with the $OPEN$ list. The state space is composed of LFPs. Each state contains one LFP for each airplane in $\mathcal{G}_i$ along with its cost value evaluated by the appropriate airplane. Thus, any state contains $\|\mathcal{G}_i\|$ LFPs. The state space is not defined before the search starts. Rather, it is constructed during the search. Initially, $OPEN$ contains only one state where the current LFPs of $\mathcal{G}_i$ and their costs are stored.

The MPCA search loop works as follows. CRC removes the state that has the minimum sum of costs of all flight trajectories from $OPEN$. Then, it checks whether there is a collision among any LFP pair in that state or a collision among any LFP from the same state and any LFP stored in the knowledge base of an airplane not involved in $\mathcal{G}_i$. If no collision is found, MPCA ends, and the respective FPs are applied by all involved airplanes. Although CRC works with LFPs only, the application of the desired flight trajectory is done by the airplane that has the respective FP. If there is a collision and it is detected for LFP of an airplane A_j that is not included in $\mathcal{G}_i$, CRC sends a request for participation to A_j and waits for the result. The addition of a new airplane A_j into the search is done dynamically without restarting the search. Up to this point, there was no collision among A_j's current LFP and any processed LFP variant in all previous states removed from $OPEN$, and the cost of all states should be increased by a constant only. Thus, the order of the removal from $OPEN$ is not affected by this addition. All states in $OPEN$, and the currently removed state are extended with A_j's LFP and A_j's cost value.

Now, CRC performs an expansion of the state so that new states are generated and inserted into $OPEN$. The expansion is done such that for each pair of colliding LFPs in the state, the appropriate airplanes are asked to prepare alternatives for that collision using the set of evasion maneuvers. This *pairwise generation* (PG) among A_j and A_k is similarly driven as PN in IPPCA. During PG, A_j and A_k prepare alternatives using the function `PrepareEvasions` with $p^{rnd} = 1$. Then, sets $\widehat{\mathcal{F}}_j$ and $\widehat{\mathcal{F}}_k$ are sent to CRC. CRC prepares combinations $\mathcal{S}^{A_j \otimes A_k}$ using the function `PrepareNegotiationSet`. Similar to IPPCA, CRC includes only such alternatives that do not produce an earlier collision than $t_1^{A_j \otimes A_k}$ compared with LFP variants of any airplane from $\mathcal{G}_i$ and to known LFPs of any airplane from $\mathcal{A} \setminus \mathcal{G}_i$. If $\mathcal{S}^{A_j \otimes A_k}$ is empty, A_j and A_k are asked to extend alternatives using $p^{rnd} = p^{rnd} + 1$. Based on $\mathcal{S}^{A_j \otimes A_k}$, new states are generated so that LFPs of A_j and A_k are replaced with variants from $\mathcal{S}^{A_j \otimes A_k}$, and LFPs of other airplanes in $\mathcal{G}_i$ are left unchanged. Thus, the same number of states is generated, which is the same as the number of combinations in $\mathcal{S}^{A_j \otimes A_k}$. The described PG producing new states is started for all pairs of colliding LFPs in the currently processed state.

The major difference between MPCA and IPPCA is that MPCA tries all possible combinations as well as their ordering while resolving the identified conflict. In IPPCA, once the run is finished, the pairwise resolution is applied, and it can no longer be reverted. By default, MPCA uses the zero heuristics, and the best candidate state is selected based only on the sum of costs from $OPEN$. The zero heuristics is the only one that is admissible for the A* search using a general cost function. For example, the cost function is constructed based on the length of FP. Evasion maneuvers changing the cruise speed do not affect the length of FP and still can resolve collisions.

The number of airplanes participating in one MPCA group is not limited in the current version. The real-time application of the algorithm requires specification of the τ interval, which is used to construct t^{um}. In MPCA, t^{um} is set once CRC is initialized. Then, the same value is used for all replanning tasks. In the worst case, the state space grows exponentially with the number of collisions. CRC monitors the progress, and once the specified amount of τ interval is passed, CRC enables nonadmissible "collision" heuristics, which assigns higher penalization to states with a larger number of unresolved conflicts. The search with collision heuristics is optimized to remove all collisions as fast as possible. If there are several options with the same number of collisions, the state with a lower sum of costs is preferred.

VIII. Experiments

The experimental evaluation and comparison of the presented algorithms has been carried out in AgentFly [37]—a

scalable agent-based prototype for airspace simulation, planning, and CA validation. The experiments use the metrics proposed by Krozel *et al.* in [38]. Various CA algorithms are compared by the sum of properties of all airplanes. The *efficiency* metrics express deviations of airplanes from their nominal trajectories. Any deviation from the nominal FP during the conflict resolution results in additional operational costs. Higher efficiency means that there is a lower additional operational cost, and thus, the algorithm provides FPs that are more efficient. In our case, the cost function is based on the flight-trajectory length. The same criterion is used in the trajectory planner as well as in CA. The efficiency eff is computed as the ratio of the sum of nominal FP lengths to the sum of flight trajectory lengths after the application of all conflict-resolution maneuvers (all airplanes reach their mission goals)

$$eff = \frac{\sum_{A_i \in \mathcal{A}} \mathbf{len}(\mathrm{fp}_i)}{\sum_{A_i \in \mathcal{A}} \mathbf{len}\left(\mathrm{fp}_i^0\right)} 100 \tag{1}$$

where the function $\mathbf{len}(\mathrm{fp}_i)$ is the length of fp_i, and fp_i^0 is the nominal FP (initial FP without any applied conflict-resolution maneuver) of A_i. Efficiency is expressed in percent. The highest efficiency is represented by 100%. In addition to efficiency, *near miss* is studied. A near miss is an observed violation of the required separation among airplanes. No detected near miss is the desired outcome. In the first two experiments, the two presented algorithms have been compared with SGTCA [9], which was introduced in a related work.

A. Perpendicular-Flow Scenario

In the *perpendicular-flow* scenario [27], two linear traffic flows of flights intersecting at right angles are generated—the first flow goes from right to left, and the second flow goes from top to bottom. The airplanes fly at the same altitude, at a constant speed of 500 kn, $h = 5$ nmi, and $\delta_t = 0.2$ h (sharing 100 nmi of FPs). For IPPCA and MPCA, $\varepsilon^h = 0$ (to provide a correct comparison with SGTCA, $\tau = 2$ s, and $\mathcal{M} = \{\mathbf{m}_L, \mathbf{m}_R, \mathbf{m}_0\}$ (only heading changes are allowed to compare them with SGTCA). Every 40 s (preserving 5.5-nmi distance), one new airplane is generated in each flow (see Fig. 5, left). The configuration was measured several times with randomly inserted gaps in the flows. Each gap corresponds to 80 s, which implies that there is one airplane missing in the flow. The scenario parameter μ defines the average number of airplanes generated in each flow before a gap is inserted. Lower μ produces more gaps. The *continuous-flow* value results in no gaps. Each scenario is run for 24 h—for the continuous version, it means that 4320 flights were generated. Each configuration was measured ten times, and the values presented in Table I are average values calculated from repetitive experiments.

Although there is no physical collision among airplanes controlled by SGTCA, there are many near misses detected. Both IPPCA and MPCA completely remove near misses and provide safe separation, although they are configured to use only the maneuvers changing the heading. It is very interesting to note that both algorithms provide a wavelike pattern solution for this scenario as in [27] [see Fig. 5, right (dashed lines show the wavelike pattern)]. IPPCA increases the efficiency in comparison with SGTCA. MPCA is even better than IPPCA. There is a higher increase of efficiency for configurations with fewer gaps.

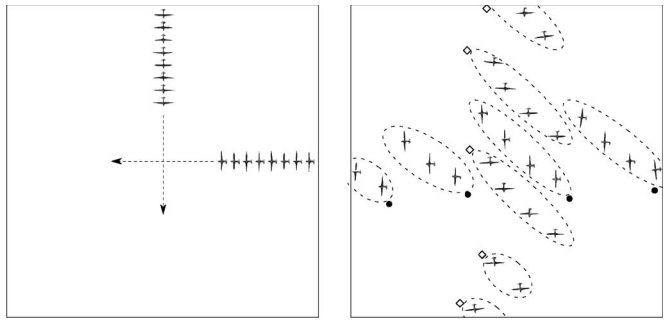

Fig. 5. Perpendicular-flow scenario. (Left) Flights generation and (right) conflict resolution using IPPCA—dashed lines are used to point up direction of the airplanes, •—flying toward left side, and ◇—flying toward the bottom.

B. Superconflict Scenario

The *superconflict* scenario is based on the model used in [20]—airplanes are evenly spaced on a circle with a radius 50 nmi; each airplane's destination WP is the point on the opposite side of the circle so that the nominal FPs go through the center of the circle. Thus, there is a multiairplane collision of all airplanes in the center at the same time. The parameters of the airplanes are the same as in the previous experiment, and due to the comparison with SGTCA, only heading changes are allowed.

Table II presents the values, which are computed as an average from 20 runs for different numbers of airplanes. IPPCA removes all near misses in comparison with SGTCA, but IPPCA results in longer FPs (worse efficiency) than SGTCA. MPCA performs better in this configuration. It also provides better efficiency than SGTCA, while it still provides safe flight separation for all airplanes during the whole flight.

C. Approach Scenario

The convergence of PN in IPPCA was studied in the *approach* scenario (see Fig. 6, left). There are N airplanes generated and positioned at the same altitude and at the same distance from a single contact point. The area around the contact point is restricted by existing SUAs. In this case, the degree of freedom is configured to use the heading and cruise-speed changes $\mathcal{M} = \{\mathbf{m}_L, \mathbf{m}_R, \mathbf{m}_F, \mathbf{m}_S \mathbf{m}_0\}$. The airplanes are configured to have an initial cruise speed of 300 kn, the cruise speed can be changed by $\pm 10\%$ with acceleration of 30 kn/min, $h = 3$ nmi, $\delta_t = 0.2$ h, $\tau = 2$ s, and $\varepsilon^h = 1$ nmi. There is no preprogrammed knowledge of the fact that the airplanes can only use fly-faster and fly-slower maneuvers, and they should use the same cruise speed while they are flying through the gap between SUAs. The resulting situation near the contact point for ten airplanes is shown in Fig. 6, right—the circle around the airplane positioned on the right-hand side shows the $h + 2\varepsilon^h$ range for separation.

TABLE I
COMPARISON OF IPPCA, MPCA, AND SGTCA IN THE PERPENDICULAR-FLOW SCENARIO

μ	# flights	Near misses			eff [%]		
		IPPCA	MPCA	SGTCA	IPPCA	MPCA	SGTCA
4	3240	0	0	61	99.5	99.6	99.4
8	3780	0	0	378	98.7	98.9	98.1
12	3960	0	0	626	98.0	98.4	97.1
16	4050	0	0	796	97.9	98.3	96.4
20	4104	0	0	1011	97.6	98.2	95.4
cont.	4320	0	0	1847	96.0	96.7	92.3

TABLE II
COMPARISON OF IPPCA, MPCA, AND SGTCA IN THE SUPERCONFLICT SCENARIO

# airplanes	Near misses			eff [%]		
	IPPCA	MPCA	SGTCA	IPPCA	MPCA	SGTCA
12	0	0	0	94.0	98.2	97.9
16	0	0	1	90.5	96.7	95.5
20	0	0	7	88.7	94.1	93.6
24	0	0	8	84.8	87.3	86.9
28	0	0	14	80.3	85.2	84.5

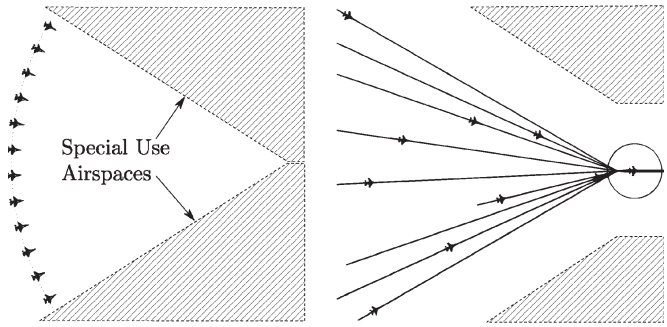

Fig. 6. Approach scenario. (Left) Initial configuration and (right) the detail of the final FPs near the contact position.

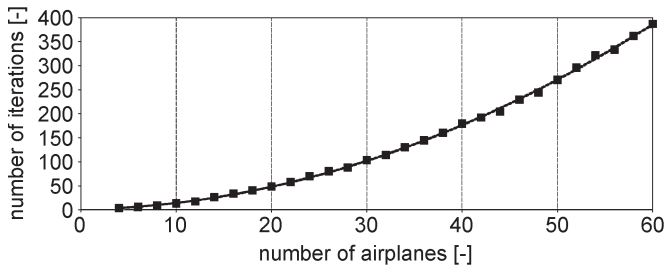

Fig. 7. Number of PNs in IPPCA in the approach scenario.

Fig. 7 shows a chart with the number of PNs in IPPCA for a varying number of airplanes in the approach scenario. Each configuration was measured 20 times, and the values are averaged from their results. In all experiments, there is no near miss detected, and IPPCA provides safe separation in this hard case as well. It is interesting that IPPCA results in the solution where only the changes of the cruise speed are used, and finally, all airplanes are stabilized back to the original flight speed to fly in a chain. The curve in Fig. 7 is polynomial with respect to the number of airplanes, but the mathematical analysis of the simplified version of this scenario provides an exponential worst-case estimation in [35].

D. Sphere Scenario

The *sphere* scenario is a 3-D extension of the superconflict scenario. The airplanes are initially positioned on a spherical surface at three different altitudes. Thus, the airplanes form three circular layers that have the same horizontal center and are positioned above each other. The middle layer corresponds to the superconflict scenario where the airplanes simply fly to the opposite side of the circle. The airplanes in the top layer fly to the opposite position in the bottom layer and *vice versa*. The radii of the top and bottom circles are smaller than the radius of the middle circle so that the nominal FPs have the same length. The top and bottom layers are positioned so that nominal FPs require 80% of their maximum descend and climb rates. The airplanes are started at the same time and with the same cruise speed; thus, their nominal FPs cause a multiairplane collision of all airplanes in the sphere's center. In this scenario, the airplanes can use all degrees of freedom for conflict resolution—heading, vertical, and cruise-speed changes. The airplanes are configured to have an initial cruise speed of 500 kn, the cruise speed can be changed by $\pm 5\%$ with acceleration of 20 kn/min, maximum descend and climb rates of 1500 ft/min, $h = 5$ nmi, $v = 1000$ ft, $\delta_t = 0.2$ h, $\tau = 2$ s, $\varepsilon^h = 2$ nmi, and $\varepsilon^v = 100$ ft. The same climb and descend rates are used, which are set to the common value used for climbing so that the scenario is symmetric.

Again, Table III provides the results showing averages from 20 runs for each number of airplanes in the sphere scenario. There was no near miss detected during any experiment in this scenario. MPCA is more efficient than IPPCA, particularly for a higher number of airplanes. The number of PNs in IPPCA is polynomial with respect to the number of airplanes. We can observe that the addition of a degree of freedom for conflict resolution increases the CA efficiency—for 24 airplanes in the superconflict configuration, there are 84.8% for IPPCA and 87.3% for MPCA, and here, for the same number of airplanes, there are 90.3% for IPPCA and 91.1% for MPCA. For more airplanes, the decrease of efficiency is not as large as with the superconflict scenario.

TABLE III
COMPARISON OF IPPCA AND MPCA IN THE SPHERE SCENARIO

# airplanes	*eff* [%]		# iterations
	IPPCA	MPCA	IPPCA
12	97.4	97.5	33
18	93.8	94.2	49
24	90.3	91.1	72
30	86.8	87.2	98
36	84.2	85.1	146
42	82.0	83.4	194
48	80.8	82.6	227

IX. CONCLUSION

In this paper, two agent-based cooperative decentralized airplane CA algorithms have been presented. The algorithms are integrated in a component-based control approach employing split conflict resolution and trajectory-planning components. The airplanes formulate their flight intention in FPs, providing a detailed intent description, including temporal aspects. For intent description, the concept of RNP [29] is utilized, which was extended with the vertical component. Using RNP, CA handles imprecise flight execution. The conflict detection is based on sharing of local parts of the airplanes' FPs so that they can perform geometrical check to detect a potential collision on their trajectories. The conflict-resolution component utilizes evasion maneuvers that formulate replanning tasks for modification of a flight trajectory, respecting all the constraints of a complex airplane model as well as limits on operational airspace. Conflict-resolution algorithms work in real-time—airplanes are flying while they are planning changes to their current flight trajectories.

Two conflict-resolution algorithms have been presented in this paper: 1) IPPCA) and 2) MPCA algorithms. The algorithms work with a different level of coordination autonomy. IPPCA removes multiairplane collisions by several changes of flight trajectories, which result from PNs, considering the costs of various combinations. The quality of the overall solution provided by IPPCA is affected by the order of PN, which is not uniquely given when multiple collisions occur at the same time. On the other hand, MPCA resolves multiairplane collisions using a single-group negotiation where the algorithm searches through the state space composed of variants of flight trajectories. The MPCA group is established dynamically based on the mutual collisions among airplanes' flight trajectories. Finally, the MPCA group contains all airplanes that have mutually colliding trajectories together.

Both algorithms are evaluated and compared experimentally in the AGENTFLY system [37]. Evaluations were performed in four different scenarios: the 1) perpendicular-flow [27]; 2) superconflict [20]; 3) approach; and 4) sphere scenarios. For IPPCA, it has been shown that the iterative PN ends in a stable solution for all experiments. The number of iterations is polynomial with respect to the number of airplanes. IPPCA and MPCA were compared with an existing multiagent CA algorithm called SGTCA [9] in the first two scenarios. In all experiments, both IPPCA and MPCA provide safe separation control where their efficiency (lower additional operational costs) is better than SGTCA. MPCA provides better efficiency than IPPCA in all cases. In the perpendicular-flow scenario, MPCA dynamically establishes multiparty groups with up to 12 airplanes. In all other scenarios, the MPCA group contains all airplanes because all of them have colliding trajectories in the same place and at the same time. Except for the sphere scenario, both IPPCA and MPCA perform in real time and are able to find a solution and avoid identified conflicting trajectories before airplanes violate the required separation. In the sphere scenario, a $\tau = 10$ s has to be set to ensure real-time performance using a standard desktop computer.

For future work, there are two major issues that are open: 1) the unbounded size of the MPCA group and 2) analysis of the efficiency of both IPPCA and MPCA and the quantification of the impact on air traffic. The current MPCA algorithm has no limits on the number of airplanes participating in the search, but the size of the state space grows exponentially with the number of airplanes. This prompts the employment of the collision heuristics in MPCA that will help it to find a solution in the remaining time, but such a solution is not as efficient as the zero heuristics. MPCA with a limited group size has to consider which airplanes should be included in the group to provide a more efficient result, and airplanes should also be dynamically disjoined from the group.

REFERENCES

[1] M. S. Nolan, *Fundamentals of Air Traffic Control,* 4th ed. Belmont, CA: Thomson Brooks/Cole, 2004.

[2] D. K. Chin and F. Melone, "Using airspace simulation to assess environmental improvements from free flight and CNS/ATM enhancements," in *Proc. Winter Simul. Conf.*, Dec. 1999, pp. 1295–1301.

[3] *FAA OPSNET Data Jan–Dec 2005*, Federal Aviation Admin., U.S. Dept. Transp., Washington, DC, 2005.

[4] *Current Market Outlook 2008–2027*, Boeing Co., Chicago, IL, 2008.

[5] National Research Council Panel on Human Factors in Air Traffic Control Automation, *The Future of Air Traffic Control: Human Factors and Automation*, Washington, DC: Nat. Acad., 1998.

[6] R. Schulz, D. Shaner, and Y. Zhao, "Free-flight concept," in *Proc. AIAA Guid., Navigat., Control Conf.*, New Orleans, LA, 1997, pp. 889–903.

[7] *Unmanned Systems Roadmap 2007–2032*, 2007.

[8] D. Šišlák, J. Samek, and M. Pěchouček, "Decentralized algorithms for collision avoidance in airspace," in *Proc. 7th Int. Conf. AAMAS*, 2008, pp. 543–550.

[9] J. K. Archibald, J. C. Hill, N. A. Jepsen, W. C. Stirling, and R. L. Frost, "A satisficing approach to aircraft conflict resolution," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 4, pp. 510–521, Jul. 2008.

[10] R. Lachner, "Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1997, vol. 3, pp. 2308–2313.

[11] J. Zhang and S. Sastry, "Aircraft conflict resolution: Lie–Poisson reduction for game on SE(2)," in *Proc. 40th IEEE Conf. Decision Control*, 2001, vol. 2, pp. 1663–1668.

[12] A. Bayen, S. Santhanam, I. Mitchell, and C. Tomlin, "A differential game formulation of alert levels in ETMS data for high-altitude traffic," in *Proc. AIAA Guid., Navigat., Control Conf.*, Austin, TX, Aug. 2003.

[13] C. Tomlin, G. J. Pappas, and S. Sastry, "Noncooperative conflict resolution," in *Proc. IEEE Conf. Decision Control*, San Diego, CA, Dec. 1997, pp. 1816–1821.

[14] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution manoeuvres," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 110–120, Jun. 2001.

[15] K. Zeghal, "A review of different approaches based on force fields for airborne conflict resolution," in *Proc. AIAA Guid., Navigat., Control Conf.*, Boston, MA, Aug. 1998, pp. 818–827.

[16] J. Kosecka, C. Tomlin, G. Pappas, and S. Sastry, "2 1/2 D conflict resolution maneuvers for ATMS," in *Proc. IEEE Conf. Decision Control*, Dec. 1998, vol. 3, pp. 2650–2655.

[17] W. Kelly and M. Eby, "Advances in force field conflict resolution algorithms," in *Proc. AIAA Guid., Navigat., Control Conf.*, Denver, CO, Aug. 2000.

[18] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 199–220, Dec. 2000.
[19] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution for air traffic management systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 221–232, Dec. 2000.
[20] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 3–11, Mar. 2002.
[21] A. Raghunathan, V. Gopal, D. Subramanian, L. Biegler, and T. Samad, "Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft," *J. Guid. Control Dyn.*, vol. 27, no. 4, pp. 586–594, Jul./Aug. 2004.
[22] M. A. Christodoulou and S. G. Kodaxakis, "Automatic commercial aircraft-collision avoidance in free flight: The three-dimensional problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 242–249, Jun. 2006.
[23] J. Rong, S. Geng, J. Valasek, and T. R. Ioerger, "Air traffic control negotiation and resolution using an onboard multi-agent system," in *Proc. Digital Avionics Syst. Conf.*, 2002, vol. 2, pp. 7B2-1–7B2-12.
[24] S. Wollkind, J. Valasek, and T. R. Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *Proc. AIAA Guid., Navigat., Control Conf.*, Providence, RI, 2004.
[25] G. Zlotkin and J. S. Rosenschein, "Negotiation and task sharing among autonomous agents in cooperative domains," in *Proc. 11th Int. Joint Conf. Artif. Intell.*, 1989, pp. 912–917.
[26] J. K. Archibald, J. C. Hill, F. R. Johnson, and W. C. Stirling, "Satisficing negotiations," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 1, pp. 4–18, Jan. 2006.
[27] Z. H. Mao, D. Dugail, E. Feron, and K. Bilimoria, "Stability of intersecting aircraft flows using heading-change maneuvers for conflict avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 4, pp. 357–369, Dec. 2005.
[28] Z. H. Mao, D. Dugail, and E. Feron, "Space partition for conflict resolution of intersecting flows of mobile agents," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 512–527, Sep. 2007.
[29] *Minimum Aviation System Performance Standards: Required Navigation Performance for Area Navigation*, Radio Technical Commision for Aeronautics, Washington, DC, 2000.
[30] D. Šišlák, P. Volf, and M. Pěchouček, "Flight trajectory path planning," in *Proc. 19th ICAPS*, 2009, pp. 76–83.
[31] M. Wooldridge, Ed., *An Introduction to MultiAgent Systems*. Hoboken, NJ: Wiley, 2002.
[32] D. Šišlák, P. Volf, A. Komenda, J. Samek, and M. Pěchouček, "Agent-based multi-layer collision avoidance to unmanned aerial vehicles," in *Proc. Int. Conf. Integr. Knowl. Intensive Multi Agent Syst.*, J. Lawton, Ed., 2007, vol. KSCO 2007, pp. 365–370.
[33] H. Moulin, *Fair Division and Collective Welfare*. Cambridge, MA: MIT Press, 2003.
[34] M. Naor, "Bit commitment using pseudorandomness," *J. Cryptology: J. Int. Assoc. Cryptologic Res.*, vol. 4, no. 2, pp. 151–158, Jan. 1991.
[35] P. Volf, D. Šišlák, M. Pěchouček, and M. Prokopová, "Convergence of peer-to-peer collision avoidance among unmanned aerial vehicles," in *Proc. IEEE/WIC/ACM Int. Conf. IAT*, Nov. 2007, pp. 377–383.
[36] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[37] M. Pěchouček and D. Šišlák, "Agent-based approach to free-flight planning, control, and simulation," *IEEE Intell. Syst.*, vol. 24, no. 1, pp. 14–17, Jan./Feb. 2009.
[38] 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 *Proc. 4th USA/Europe Air Traffic Manage. R & D Semin.*, Santa Fe, NM, Dec. 2001.

David Šišlák received the M.S. degree from Technical Cybernetics, Czech Technical University, Prague, Czech Republic, where he is currently working toward the Ph.D. degree with the Agent Technology Center, Department of Cybernetics.

He is a Research Scientist with the Agent Technology Center. He is an author or coauthor of cited publications in proceedings of international conferences and journals. His research interests are in technical cybernetics and multiagent systems, focusing on decentralized collision-avoidance algorithms in the air-traffic domain, efficient communication, knowledge maintenance in an inaccessible multiagent environment, large-scale multiagent simulations, and agent frameworks.

Mr. Šišlák was the recipient of the 2005 IEEE/WIC/ACM WI-IAT Joint Conference Best Demo Award and the International Cooperative Information Agents workshop system innovation award in 2004 for the AGLOBE multiagent platform and related simulations.

Přemysl Volf received the M.S. degree in software systems from the Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic. He is currently working toward the Ph.D. degree with the Agent Technology Center, Gerstner Laboratory, Department of Cybernetics, Czech Technical University, Prague, Czech Republic.

He is currently a Researcher with the Agent Technology Center. His research is focused on distributed cooperative algorithms used for collision avoidance in air-traffic control and verification of these algorithms using theory and prototypes.

Michal Pěchouček (M'10) received the M.Sc. degree in information technology from the University of Edinburgh, Edinburgh, U.K., in 1996 and the M.Sc. degree in cybernetics and the Ph.D. degree in biocybernetics and artificial intelligence from Czech Technical University, Prague, Czech Republic, in 1995 and 1998, respectively.

He is a Professor of artificial intelligence with the Department of Cybernetics, Czech Technical University, where he is also the Head of the Agent Technology Center, Gerstner Laboratory. He is an author or coauthor of cited publications in proceedings of international conferences and journals.

Dr. Pěchouček is a Chair of the EUMAS advisory board and a member of the CEEMAS steering committee. In addition, he has been a Cochair of the AAMAS Industry Track, HOLOMAS, KSCO, and CEEMAS and is a member of the program committee of other relevant conferences and workshops.