Automated Conflict Resolution Utilizing Probability Collectives Optimizer

David Šišlák, Přemysl Volf, Michal Pěchouček, and Niranjan Suri

***Abstract*—Rising manned air traffic and deployment of unmanned aerial vehicles in complex operations requires integration of innovative and autonomous conflict detection and resolution methods. In this paper, the task of conflict detection and resolution is defined as an optimization problem searching for a heading control for cooperating airplanes using communication. For the optimization task, an objective function integrates both collision penalties and efficiency criteria considering airplanes' objectives (waypoints). The probability collectives optimizer is used as a solver for the specified optimization task. This paper provides two different implementation approaches to the presented optimization-based collision avoidance: 1) a parallel computation using multiagent deployment among participating airplanes and 2) semicentralized computation using the process-integrated-mechanism architecture. Both implementations of the proposed algorithm were implemented and evaluated in a multiagent airspace test bed AGENTFLY. The quality of the solution is compared with a negotiation-based cooperative collision avoidance method—an iterative peer-to-peer algorithm.**

***Index Terms*—Air traffic, collision avoidance, conflict resolution, distributed control, multiagent systems, optimization.**

I. Introduction

THE currently worldwide-used air traffic management (ATM) system [1] is based on human controllers responsible for defined airspace sectors, and it is reaching its limits. However, the number of flights is increasing rapidly. Boeing in [2] predicts that the number of cargo flights will triple within next 20 years. The current centralized ATM reacts slowly to changing weather conditions and minor local delays imply large regional congestion. The U.S. Federal Aviation Administration (FAA) estimates [3] that the weather and the national aviation system caused 606 500 delays (513 420 h of delays) in 2008, by which fuel is wasted unnecessarily and atmospheric pollution increases [4]. The most straightforward way to better utilization of airspace is the removal of predefined airways and the *free flight* concept [5], [6] adoption. The free flight concept introduces an idea, where airplanes take care about their separation by themselves instead of a centralized ground air-traffic control. The free flight concept applied to the enroute part of the flight is studied in Next Generation Air Transportation Systems (NGATS) [7]. In NGATS, airplanes can optimize their flight corridors according to their priorities in enroute parts of their flights (midparts), but they are under control of existing ATM mechanisms in terminal parts. The use of the free flight concept will reduce ATM controllers' work load (they will provide other services on top of automated collision avoidance) and minimize failures, which can occur in the centralized ATM [8].

Besides the relation of collision-avoidance techniques to the civilian air-traffic management, autonomous conflict resolution mechanisms are very important to unmanned aerial vehicles (UAVs) cooperatively fulfilling their mission goals in the shared air space [9]. In such a case, UAVs are required to implement automatic see-and-avoid capability [10]. There exist many multiUAV deployment use cases [11], e.g., in an application for forest fire monitoring [12], UAVs monitor a large forest fire in areas inaccessible to ground vehicles. This article addresses the field of the *cooperative collision avoidance* problem, where airplanes are equipped with bidirectional communication devices allowing communication within a limited range.

Automated conflict resolution methods supporting the free flight concept are widely addressed by the research community (see Section II). In this paper, an optimization-based approach to the collision avoidance problem has been adopted—airplanes search for a series of actions that would allow them to avoid a collision effectively. It is supposed that the airplanes can communicate and cooperate together during the optimization. Efficiency criteria, collision penalties, and airplanes' goals are incorporated into a shared objective function. The optimal control is a set of actions, which minimize the objective function (see Section III). In comparison with the approach where the optimization function is defined by an efficiency criterion only and where collision penalties are constraints, this approach is able to provide solutions for situations where there is not enough space to separate airplanes. The collision penalty has to be much higher than efficiency part of the objective function. A high penalty for the collision part of the objective function cause that the algorithm prefers results guaranteeing separation distance among airplanes. In the other case when there is not enough

Manuscript received May 13, 2009; revised September 11, 2009 and June 1, 2010; accepted October 10, 2010. Date of publication December 3, 2010; date of current version April 19, 2011. The Process Integrated Mechanism concept is supported in part by the U.S. Air Force Office of Scientific Research under Grant FA9550-08-1-0218, in part by the Office of Naval Research Coordinated Operations program and the U.S. Army Research Laboratory's TEAM Performance program, and by the Italian MiUR in the frame of the PRIN project "MEnSA - Agent oriented methodologies: engineering of interactions and relations with the infrastructures." The AGENTFLY is supported by the Air Force Office of Scientific Research, Air Force Material Command, United States Air Force, under Grant FA8655-06-1-3073 and by Czech Ministry of Education under Grant 6840770038. This paper was recommended by Associate Editor T. Busch.

D. Šišlák, P. Volf, and M. Pěchouček are with the Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University, Prague 121 35, Czech Republic (e-mail: sislakd@fel.cvut.cz; volf@labe.felk.cvut.cz; pechouc@labe.felk.cvut.cz).

N. Suri is with the Florida Institute for Human and Machine Cognition, Pensacola, FL 32502 USA (e-mail: nsuri@ihmc.us).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCC.2010.2089448

space to separate all airplanes, it minimizes separation violations. The *probability collectives* (PC) framework [13]–[15] is used as an optimization solver. The PC is a stochastic optimizer using probabilistic operators optimizing over a variable space. The PC approach, in contrast with other existing stochastic approaches, such as genetic algorithms [16] and particle swarm optimization [17], operates on probability distributions of variables rather than the variables themselves. Since the collectives approach operates directly on probability distributions, it also offers a direct approach to incorporating uncertainty, which is typically represented through probabilities.

The use of the PC algorithm creates a *distributable* control. There is no initial assumption about the construction of an objective function, which can be, therefore, easily modified (also during the flight) without updating the optimization framework. A major benefit of the PC optimizer is that the whole optimization process can be distributed among several agents controlling airplanes—several parts can be executed simultaneously. The PC algorithm has been already successfully deployed for the flight vehicle control [18]—a large number of small, simple, trailing-edge devices controlling vehicle dynamics.

Besides an application of PC to conflict resolution, another aspect of this paper is the comparison of two implementation approaches to the described PC deployment for the collision avoidance problem: 1) *parallelized optimization* and 2) *semicentralized optimization*. In the first approach, the PC optimization process is done cooperatively by a group of agents. Multiagent approach has been successfully used in many industrial applications before [19] and [20]. Each of the optimized variables from PC is mapped to one agent controlling one airplane. This approach can profit from a parallelized execution of the PC optimization, but it requires a complex negotiation protocol (see Section V). The second approach requires the collection of optimization inputs, selection of a host where the optimization will be performed, and distribution of the solution to all involved airplanes. To implement these tasks in the most straightforward way, the process-integrated mechanism (PIM) has been adopted (see Section VI). The programmer does not need to care about any synchronization or communication issues. All is done automatically using a mobile coordinating process (CP) that provides a shared memory and resource access. However, this second approach cannot utilize the parallelization potential of the stochastic optimization. Both approaches have been implemented in the AGENTFLY system—a scalable, agent-based technology for free-flight simulation [21]. The default airspace simulation based on complex flight plans [22] has been replaced by airplane models reacting to a given control action (e.g., change of heading) resulting from the optimization process specified earlier.

The rest of the paper is organized as follows. Section II provides a summary of existing confliction resolution methods supporting the free flight concept. Section III defines collision avoidance as an optimization task. A brief introduction to the PC algorithm is provided in Section IV. The parallelized multiagent implementation approach is presented in Section V. Section VI describes the PIM implementation approach. The iterative peer-to-peer collision avoidance method, with which the proposed one is compared, is briefly presented in Section VII. Section VIII documents experimental validation comparing both PC-based implementation approaches with the iterative peer-to-peer method. Finally, Section IX concludes the article and discusses the practicality of the implementation.

II. Related Conflict Resolution Techniques

There exist many conflict resolution methods, which differ in several aspects, such as the type of control actions used for collision avoidance and centralized or decentralized approach. In this section, selected relevant techniques are discussed. Pappas *et al.* [23] proposed decentralized conflict resolution based on a hybrid system including both discrete events and individual dynamics modeled by differential equations. Projected conflicts are resolved in two phases: 1) A game-theory approach is used by each airplane to search for speed changes guaranteeing the necessary separation regardless of the opponents' actions, and 2) if it fails, coordinated constant speed heading changes are used to avoid the conflict. In contrast to the presented algorithm, their approach is not suitable for airplanes, which want to cooperate together and optimize a shared objective function. Their original algorithm operates only with speed changes, but there is a subsequent extension of the game-theoretic approach for both heading and speed changes [24].

Krozel *et al.* [25] described a distributed algorithm providing heading changes. However, it resolves future collisions in a pairwise manner. The colliding airplane is passed in front or behind the conflicting airplane using two different strategies: 1) The myopic strategy prefers the smallest heading changes, and 2) the look-ahead strategy furthermore ensures that the selected maneuver does not create a conflict earlier than the original one. However, Krozel's algorithm is suitable for self-interested airplanes, which cannot optimize the solution considering global criteria. In contrast, the presented algorithm is able to integrate both 1) airplanes' self-oriented criteria as well as 2) the global criterion. On the other hand, it assumes that airplanes trust each other and would like to cooperate together in the solution, which is the case for many UAV operations. Hill *et al.* [6], [26] used an approach based on the satisficing game theory with dual social utility: selectability and rejectability. Unlike conventional game-theory models maximizing self-interest metrics, they proposed a satisficing extension, where airplanes take preferences of others into consideration. By integrating other preferences into the airplane's decision, this approach implements a kind of cooperation in the final solution. However, it is very hard to take the desired optimization ensuring the best solution for the selected metrics and integrate it into the decision model of the airplanes.

Christodoulou and Kodaxakis [27] formalized the 3-D air traffic collision problem as a mixed-integer nonlinear programming problem optimizing the desired function. However, it is very hard to solve the given nonlinear programming problem for airplanes with more degrees of freedom. Therefore, the article analyzes the solution with maneuvers changing velocity only. Tumer and Agogino [28] applied multiagent algorithms for traffic flow management on predefined airways. In contrast

to many other approaches, Tumer and Agogino associated each agent with a fix (a specific location in 2-D space), which reduces the number of agents for high-traffic areas with thousands of airplanes. Agents are responsible for setting the required separation among the airplanes going through that fix by assigning speed-up or slow-down actions to reduce congestion.

III. Conflict Resolution Optimization Task

The conflict resolution for group of airplanes can be defined as an optimization task formulated as finding the control inputs producing trajectories that minimize a common objective function. The objective function penalizes deviations from goal-oriented controls (flying to their desired waypoints) and collision occurrences (based on the required separation distance). The optimization is subject to constraints, which ensure that the control inputs are within the flight envelope limitations (e.g., the maximum angular velocity of heading changes). Collision occurrences are included as an optimization criterion minimizing separation violations, which are highly penalized in comparison with deviations from goal-oriented controls. In comparison with integration of a collision criterion as a hard constraint, such approach provides airplane controls also in very dense situations, where airplanes cannot be fully separated. For the simplicity of description, conflict resolution actions (control inputs) have been limited to only horizontal control—heading changes. However, the presented approach can be extended and actions can include also vertical and speed control, if necessary.

To reduce the complexity of the optimization task, it is supposed that the objective function considers only a limited time interval into the future for computation of collision occurrences (called look-ahead interval) and searches for the control input that is applied from the given time and can be changed to the optimal control steering airplane toward its next waypoint after some time. Such an approach requires that the optimization task is solved periodically to ensure that the moving look-ahead interval is always placed in the future—a receding horizon optimal control. The interval of this period is denoted Δt. If an airplane changes its next waypoint and, thus, its mission, the optimization is invoked immediately to reflect the changes.

For simplicity, it is supposed that all airplanes can communicate with each other, all airplanes always participate in the optimization task and the optimization provides only heading controls[1] from a discrete set of headings. Alternatively, several independent optimizations can be running, covering airplanes within a communication range, as there are no mutual influences between distant airplanes due to the look-ahead limitation in the construction of the objective function. However, the paper addresses the optimization providing the heading control only; the optimization task can be extended to also provide the altitude and speed control.

The next part of this section formally defines the optimization task with discrete variables specified as

$$\arg\min_{\overline{x}\in\mathcal{X}} \mathrm{G}(\overline{x}) \tag{1}$$

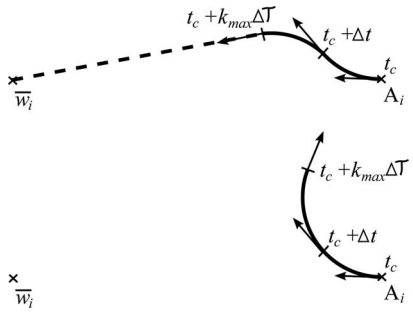

Fig. 1. Application of control actions. (Top) $x_i = \langle \omega_i, 1\rangle$ and (bottom) $x_i = \langle \omega_i, 0\rangle$.

where G is the *objective function*, $\overline{x} \in \mathcal{X}$ is the control vector, also called *joint move* $\overline{x} = \langle x_1, x_2, \ldots, x_N\rangle$, where x_i is the control input for the *i*th aircraft, and it is supposed that there are N airplanes $\mathcal{A} = \langle A_1, A_2, \ldots, A_N\rangle$. The control action x_i for every A_i is defined as a tuple $\langle \omega_i, g_i\rangle$, where ω_i is the heading angular velocity (positive for the right turn and negative for the left turn) and $g_i \in \{0, 1\}$ specifies whether the action is applied to the entire considered future trajectory ($g_i = 0$) or just to the next interval Δt and then proceeding to the airplane's next waypoint ($g_i = 1$). Fig. 1 presents examples of two control actions with the same heading angular velocity, but one with $g_i = 1$ and the second with $g_i = 0$. In Fig. 1, t_c denotes A_i's current position and orientation and $k_{\max}\Delta\tau$ stands for a considered look-ahead interval (which is described later). Cruise speeds v_i of all A_i are constant. For each A_i, the next waypoint is denoted as $\overline{w}_i$. $\overline{w}_i$ is accomplished by A_i, if its current position is within the specified tolerance around $\overline{w}_i$. In such a case, $\overline{w}_i$ is set to the next mission waypoint of A_i. The separation distance R_i specifies that there should not be any other airplane closer than R_i to A_i.

The heading control ω_i is limited by the flight dynamics not to be larger than the maximum angular velocity $\omega_i^{\max}$. The optimizer searches for the value of each x_i within the final number of values in its definition set $\mathcal{X}_i$. $\mathcal{X}_i$ contains m_i actions with ω_i values evenly selected from an interval $\langle -\omega_i^{\max}, \omega_i^{\max}\rangle$ with $g_i = 0$, m_i actions with ω_i values selected the same way, but with $g_i = 1$ and the single action x_i^{opt}. x_i^{opt} is an optimal control action navigating A_i directly toward $\overline{w}_i$, which does not consider other airplanes but respects A_i flight dynamics constraints—the maximum turn rate and flight smoothness. m_i is an odd integer greater than 2, which ensures that the straight flight maneuver is included. Using such a construction, $\mathcal{X}_i$ contains $2m_i + 1$ control actions.[2]

The objective function is constructed using the sampling interval $\Delta\tau$, which is common to all airplanes. The function $\mathbf{f}_i(x_i, k)$ returns the future position of A_i after k intervals $\Delta\tau$ when A_i applies the action x_i considering its current position and A_i flight dynamics constraints. The objective function $\mathrm{G}(\overline{x})$ is defined as

$$\mathrm{G}(\overline{x}) = \mathrm{G}^{\mathrm{col}}(\overline{x}) + \alpha \mathrm{G}^{\mathrm{dev}}(\overline{x})\ . \tag{2}$$

[1]Flight speed is constant and altitude changes are given by the altitude of the next waypoint.

[2]For the used optimizer, it is required that actions in the set $\mathcal{X}_i$ are ordered by ω_i values (it does not matter whether ascending or descending).

It consists of two parts, which are summed together using a balancing factor α. $\mathrm{G}^{\mathrm{col}}$ penalizes separation violation among all airplanes $\mathcal{A}$ using their future positions and the required separation distances. It is computed as

$$\mathrm{G}^{\mathrm{col}}(\overline{x}) = \sum_{A_i \in \mathcal{A}} \sum_{A_j \neq A_i\, A_j \in \mathcal{A},} \mathrm{G}_i^{\mathrm{col}}(x_i, x_j)$$

$$\mathrm{G}_i^{\mathrm{col}}(x_i, x_j) = \sum_{k=1}^{k_{\max}} \beta^{(k-1)} [\max(R_i - \|\mathbf{f}_i(x_i,k), \mathbf{f}_j(x_j,k)\|, 0)]^2 \quad (3)$$

where the factor $\beta \in (0,1)$ is used for balancing the penalty between earlier and later squared collision penalties expressed as $\max(R_i - \|\mathbf{f}_i(x_i,k), \mathbf{f}_j(x_j,k)\|, 0)$ (smaller β more strongly penalizes earlier collisions), $k_{\max}$ defines the look-ahead horizon and $\|\overline{v}_1, \overline{v}_2\|$ stands for the Euclidean distance between $\overline{v}_1$ and $\overline{v}_2$. The second part of the objective function $\mathrm{G}^{\mathrm{dev}}$ penalizes the deviation from airplanes' optimal trajectories to their next waypoints

$$\mathrm{G}^{\mathrm{dev}}(\overline{x}) = \sum_{A_i \in \mathcal{A}} \mathrm{G}_i^{\mathrm{dev}}(x_i)$$

$$\mathrm{G}_i^{\mathrm{dev}}(x_i) = |\mathbf{f}_i(x_i, k_{\max}), \mathbf{f}_i(x_i^{\mathrm{opt}}, k_{\max})|^2 \,. \quad (4)$$

The deviation is expressed as the squared Euclidean distance between the position at the end of the look-ahead horizon $k_{\max}$ after applying the evaluated action and the optimal control action.

In this approach, the collision penalties are integrated into an optimization function instead of the solution constraints. Such construction of the objective function allows to find a solution also when there is not enough space to separate all airplanes. In such case, optimization provides a solution minimizing the violations among them, as they are integrated as a sum of squared violations. The balancing factor α should be selected so that the values of $\mathrm{G}^{\mathrm{col}}(\overline{x})$ are much higher than the values of $\mathrm{G}^{\mathrm{dev}}(\overline{x})$, if there exists any violation.

IV. Probability Collectives Optimizer

In this section, we describe the details about the PC theory applicable to the optimization problem with discrete variables. The PC theory can be viewed as an extension to the conventional game theory. Let's have a game with N players $i \in \mathcal{I}$. A mixed strategy of the player i is a probability distribution $q_i(x_i)$ over player's possible pure strategies (a definition set of x_i) [29]. Each player i chooses its strategy (a value of the variable x_i) independently by sampling $q_i(x_i)$. There is no direct communication between players in the game. Players learn to cooperate through repeated plays. All the coupling among players occurs indirectly—their probability distributions are updated using the received reward based on the objective function $G(\overline{x})$ combining all variables. The probability distribution of the *joint move* $q(\overline{x})$ is

$$q(\overline{x}) = \prod_{i \in \mathcal{I}} q_i(x_i) \,. \quad (5)$$

Bounded rational players [30] balance their choice of the best move with the need to explore other possible moves. The information theory shows that the equilibrium of a game played by bounded rational players is the optimum of a Lagrangian of the probability distribution of the agents' joint moves [15], [31]. This equilibrium corresponds to at least a local minimum of the original objective function G. The expected world utility of all the players with a common world utility G under given players' distributions $q_i(x_i)$ is

$$\mathrm{E}_q(\mathrm{G}(\overline{x})) = \sum_{\overline{x} \in \mathcal{X}} \mathrm{G}(\overline{x}) q(\overline{x}) = \sum_{\overline{x} \in \mathcal{X}} \left[\mathrm{G}(\overline{x}) \prod_{i \in \mathcal{I}} q_i(x_i) \right]. \quad (6)$$

In the Nash equilibrium, every player adopts a mixed strategy that maximizes its expected utility with respect to the mixed strategies of others.[3] The Nash equilibrium assumption requiring full rationality (every player can calculate strategies of the others) is replaced by the information available to the players. This amount of information is the negative of the *Shannon entropy* [32] (the distribution with minimal information is the one that makes no distinction between the various x at all and the most informative distribution is the one that specifies a single possible x) of the distribution $q(\overline{x})$

$$S(q) = -\sum_{\overline{x} \in \mathcal{X}} [q(\overline{x}) \ln[q(\overline{x})]] \,. \quad (7)$$

Using the *maximum entropy principle* (Maxent) [33], each player searches for the probability distribution q that minimizes the expected utility

$$\arg\min_q \; \mathrm{E}_q(\mathrm{G}(\overline{x})) \quad (8)$$

subject to $\sum_{x_i \in \mathcal{X}_i} q_i(x_i) = 1$ and $q_i(x_i) \geq 0$ for each $i \in \mathcal{I}$. From the *gradient-based optimization,* we have to find the critical point of the *Maxent Lagrangian*

$$\mathcal{L}(q, T) \equiv \mathrm{E}_q(\mathrm{G}(\overline{x})) - TS(q) \quad (9)$$

where T is the Lagrange parameter (which is also referred to as the *temperature*). We need to find q and T such that $\partial\mathcal{L}/\partial q = \partial\mathcal{L}/\partial T = 0$.

The algorithm lowering $\mathrm{E}_q(\mathrm{G}(\overline{x}))$ is an iterative procedure with the following steps (see Fig. 2 and [34]):

1) Initialize the value of the Lagrange parameter T.

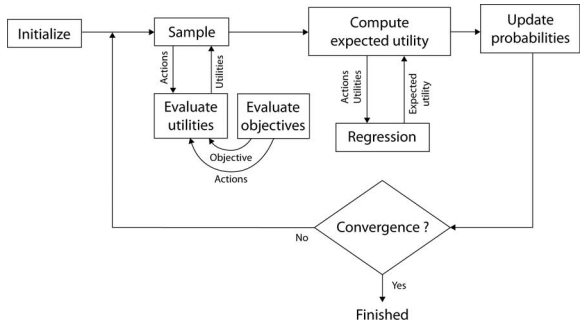

Fig. 2. Iterative procedure lowering $\mathrm{E}_q(\mathrm{G}(\overline{x}))$ [34].

[3] In this paper, maximization of utility is replaced with minimization of cost to be consistent with defined objective function.

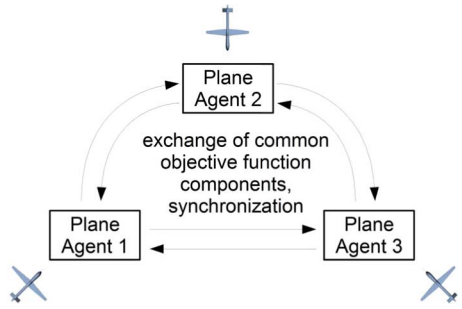

Fig. 3. Multiagent implementation of PC optimization.

2) Minimize the Lagrangian $\mathcal{L}(q,T)$ with respect to q at specified T (sample the system, update the probabilities).
3) Reduce the value of T and repeat from Step 2 until q converges (q does not vary more than the specified threshold for a couple of iterations).
4) The $\overline{x}$ selected according to the final q is the solution of (1).

The sequence lowering T is the *annealing schedule*, in reference to the simulated annealing [35]. For a given T, each player i optimizes

$$\mathcal{L}_i(q_i,T) = \sum_{x_i\in\mathcal{X}_i}[q_i(x_i)\mathrm{E}[\mathrm{G}|x_i]] - T\sum_{x_i\in\mathcal{X}_i}[q_i(x_i)\ln[q_i(x_i)]]\,. \tag{10}$$

The function $\mathcal{L}_i$ is convex, has a single minimum in the interior, the temperature T controls the tradeoff between exploration and exploitation [31]. The first term $\sum_{x_i\in\mathcal{X}_i}[q_i(x_i)\mathrm{E}[\mathrm{G}|x_i]]$ in (10) is minimized by a perfectly *rational* player, while the second term $-T\sum_{x_i\in\mathcal{X}_i}[q_i(x_i)\ln[q_i(x_i)]]$ is minimized by a perfectly *irrational* player (by a perfectly uniform mixed strategy q_i). In the limit $T\to 0$, the set of q that simultaneously minimizes the Lagrangian is the same as the set of q minimizing the objective function G.

V. Parallelized Approach

The PC optimization (see Section IV) can be fully distributed and implemented in a parallel way as a multiagent system. Collectives in the PC algorithm can be viewed as groups of self-interested, learning agents that act together to minimize the objective function (1) (see Fig. 3). Each variable is maintained by one agent. Thus, each agent searches for an optimal action for a single airplane (see Section III). In this section, $A_i\in\mathcal{A}$ denotes the agent providing control to the airplane A_i. Each A_i keeps the current probability distribution q_i for its action variable x_i. Computation of the expected utility value [value of the common objective function $\mathrm{G}(\overline{x})$, (2)], and the convergence test requires cooperation of all agents. Sampling and updating of all variables in the iterative procedure of the PC algorithm can be performed independently in a parallel way.

Each A_i is configured using several airplane-oriented parameters: the maximum available angular velocity $\omega_i^{\max}$, the number of discrete steers m_i, and the separation distance R_i.

Input: $\mathcal{A}$
Output: ω_i

```
{1}  x_i^opt ← Optimal action(w̄_i);
{2}  X_i ← Get x_i definition set(ω_i^max, m_i, x_i^opt);
{3}  q_i ← Get initial distribution(X_i);
{4}  T ← Initialize temperature;
{5}  while true do
{6}      s_i[N_SB] ← Sampling(X_i, q_i, N_SB);
{7}      pred_i[N_SB × k_max] ← Predictions(s_i, k_max);
{8}      wait_set ← A\A_i;
{9}      Send(pred_i, wait_set);
{10}     G_i[N_SB] ← Own dev cost(pred_i, x_i^opt, α);
{11}     while wait_set ≠ ∅ do
{12}         A_j, pred_j ← Fetch other predictions;
{13}         G_i ← Add col cost(G_i, R_i, β, pred_i, pred_j);
{14}         wait_set ← wait_set\A_j;
{15}     end
{16}     wait_set ← A\A_i;
{17}     Send(G_i, wait_set);
{18}     while wait_set ≠ ∅ do
{19}         A_j, G_j ← Fetch other costs;
{20}         G_i ← Add costs(G_i, G_j);
{21}         wait_set ← wait_set\A_j;
{22}     end
{23}     q_i ← Update distribution(X_i, q_i, s_i, G_i, T);
{24}     T ← Update temperature(T);
{25}     if Converged(G_i) then break;
{26} end
{27} return Sample final control(X_i, q_i);
```

Algorithm 1. Agent PC optimization pseudocode.

Moreover, the A_i manages its given mission and defines its next waypoint w_i. All A_i use common configuration parameters: the size of the sample block in each iteration N_{SB}, the look-ahead horizon $k_{\max}$, the sampling interval $\Delta\tau$, the balancing factor α, the collision penalty time factor β, and annealing schedule parameters.

Algorithm 1 presents a distributed implementation of the PC optimization procedure executed by each agent. First, each agent performs an initial setup of the optimal action x_i^{opt}, the definition set $\mathcal{X}_i$, the probability distribution q_i as an uniform discrete distribution over $\mathcal{X}_i$, and the temperature T according to the selected annealing schedule initial value (lines 1–4).

The iterative optimization loop lowering $\mathrm{E}_q(\mathrm{G}(\overline{x}))$ from Fig. 2 is implemented at lines 5–26. Agents prepare sample blocks s_i (N_{SB} actions selected from $\mathcal{X}_i$ using Monte Carlo sampling [36]) and prediction points $pred_i$ (for each action in s_i agents apply function $\mathbf{f}_i(x_i,k)$, where $k=1,\ldots,k_{\max}$) (lines 6 and 7). Then, agents exchange their $pred_i$ (lines 8 and 9). The computation of the common objective function $\mathrm{G}(\overline{x})$ (2)–(4) for each joint action $\overline{x}$ in the sample block s_i is distributed among all agents. Each agent A_i computes G_i of the rewritten objective function

$$\mathrm{G}(\overline{x}) = \sum_{A_i\in\mathcal{A}}\mathrm{G}_i(\overline{x})$$
$$\mathrm{G}_i(\overline{x}) = \alpha\mathrm{G}_i^{\mathrm{dev}}(x_i) + \sum_{A_j\neq A_i\,A_j\in\mathcal{A}}\mathrm{G}_i^{\mathrm{col}}(x_i,x_j)\,. \tag{11}$$

The deviation part of the objective function $\mathrm{G}_i^{\mathrm{dev}}(x_i)$ is prepared at line 10. Each agent waits for other sample block predictions from all agents from the set $\mathcal{A}\backslash A_i$ and adds the collision part of

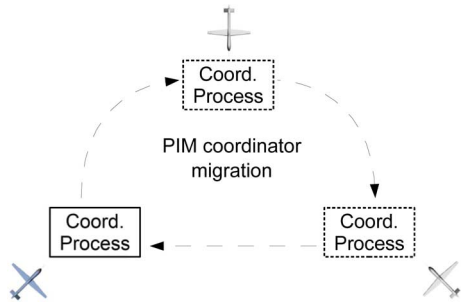

Fig. 4. PIM approach to PC optimization.

the objective function for each processed $pred_j$ (lines 11–15). After processing all predictions, each agent has the value $\mathrm{G}_i(\overline{x})$ for each sample block action and it sends these values to all others agents (lines 16 and 17). Then, the agent waits for all other parts and sums the values into G_i (lines 18–22). At this point, all agents have the same values of the objective function evaluation in their G_i.

The update of the agent's probability distribution minimizing Lagrangian $\mathcal{L}_i(q_i, T)$ with the current temperature T is done at line 23. Then, the temperature T is decreased according to the common annealing schedule (line 24). The convergence test of the iterative optimization procedure is done simultaneously by all agents (line 25). It is not necessary to communicate during this phase, as all agents have the same $\mathrm{G}(\overline{x})$ value. Finally, the agent selects the final control according to its stabilized probability distribution q_i (line 28).

VI. Semicentralized Approach

The semicentralized PC optimization approach requires collecting of optimization inputs, selecting the host where the optimization will be performed, and distributing the solution to all involved airplanes. The concept of the PIM has been adopted to implement the semicentralized approach. First, the PIM model is briefly presented. Then, the implementation of the automated conflict resolution based on the PIM approach is described.

A. Process-Integrated Mechanism

PIM [37] is an architecture that benefits from a single controlling authority while avoiding structural difficulties that have traditionally led to its rejection in many complex settings. The core idea of PIM is to retain the perspective of the single controlling authority, but abandon the notion that it must have a fixed location within the system. Instead, the computational state of the CP is moved among the component parts of the PIM.

The PIM model consists of a single CP and a set of *components*, each capable of running CP. CP cycles among the components as required in order to execute the CP algorithm (see Fig. 4). The time in which CP runs on a component is called the *residency time*. Each component maintains the code for CP; therefore, the controlling process can move from component to component by passing only a run-time state using the mechanism of *strong mobility* [38]. The underlying PIM run-time system manages the actual movement of the CP across the components, and presents the programmer with a virtual machine in which there is a single CP operating with a unified global view where, in fact, data remains distributed across the components. The programmer need not be concerned with the details of the process moving among the processors.

The PIM model can be viewed as the inverse of *time sharing*. Time-sharing models revolutionized computing because they allowed multiple processes to run on the same computer processing unit (CPU) at the same time as though each was the only process running on that machine. In such a model, the programmer could construct the program with no concern for the details of the process switching that is actually happening on that CPU. To the programmer, it is as if their program has the entire processor, even though in reality it is only running in bursts as it is switched in and out. The PIM model, on the other hand, provides the reverse. To the programmer, it still appears that there is one program controlling all the components, but CP is actually cycling from component to component. Even further, it is as though the memory and data available on each processor is also always available, as in a distributed memory system. In other words, the set of components appears to be a single entity.

There are two implementations of Java virtual machine (JVM) supporting PIM concept natively. The first implementation is based on the Aroma VM [39], which provides the necessary primitives to asynchronously capture and move the execution state of threads running inside VM. Aroma allows the capture and migration of CP between any two Java byte-code instructions, thereby providing fine-grained and accurate control over the residency time of CP at each node. However, Aroma does not provide a *just-in-time* (JIT) compiler, thereby Aroma provides less performance than other JVMs. The second implementation is based on the Mobile Jikes RVM [40], [41], which is a modified version of the Jikes Research VM developed by IBM [42]. The Mobile Jikes RVM provides also JIT, thereby a performance which is close to the commercial JVMs.

However, both these JVMs do not support the latest Java specification, which is required for running the multiagent airspace evaluator AGENTFLY [21]. Thus, the native requirement for *strong migration* has been replaced by *weak migration* available in all existing JVMs. Using weak migration, a mobile CP moves from one entity to another upon its own request. When CP code requires access to currently unavailable memory by calling a respective command, it triggers the migration. Its state is serialized and migrated to the node having the required data. Finally, CP is restarted on that node. AGENTFLY system is built on top of AGLOBE multiagent platform [43] and the implemented CP utilizes its agent weak migration support.

B. Conflict Resolution CP

The semicentralized implementation of the described conflict resolution is straightforward and does not require any modifications of the algorithm described in Sections III and IV. The pseudocode of the coordination process is stated in Algorithm 2.

First, CP reads the initial configuration specifying a set of involved airplanes $\mathcal{A}$, PC configuration parameters c_{PC}, and

```
{1} ⟨A, c_PC, id⟩ ← Get initial configuration();
{2} while true do
{3} |  S ← Get airplanes' states(id, A);
{4} |  Ω ← PC optimization(c_PC, S);
{5} |  Apply new control(id, Ω);
{6} |  id ← id + 1;
{7} end
```

Algorithm 2. PIM coordination process pseudo-code implementing centralized PC optimization for CA.

an optimization ID id (line 2). The c_{PC} includes the following parameters for PC optimization: N_{SB}, $k_{\max}$, α, β, and annealing schedule parameters. Then, CP repeats infinitely lines 2–2. CP provides periodical control of involved airplanes with period Δt. Each control input is uniquely identified by the optimization ID id. CP does not finish implicitly by default, but it can be removed along with the component (airplane) where it is actually running. For example, if an airplane finishes its mission (its last waypoint is accomplished). All the components (airplanes $\mathcal{A}$) monitor that CP is visiting them at least once every Δt. If any component detects that CP is not visiting it, it contacts all others and invoke a recovery mechanism that creates and starts a new CP. The same mechanism is used for the creation of the first CP. All airplanes are the only components that provide PIM underlying architecture for hosting CP.

The invocation of the method `Get airplanes' states` blocks until the new control is required for a specified id and also causes migration of CP among all the airplanes $\mathcal{A}$ to collect information about their current state and parameters (line 2). Once CP has the necessary information about all the airplanes, it performs local `PC optimization` (see Section III) (line 2). Finally, CP calls the method `Apply new control` that causes migration among all the airplanes $\mathcal{A}$ and sets a new control action ω_i, which is included for each airplane in the set Ω (line 2). After that, CP increments id (line 2).

VII. Iterative Peer-to-Peer Collision Avoidance

This section briefly introduces the *iterative peer-to-peer collision avoidance* (IPPCA) [22] that is used as a comparator. Similar to the presented approach, IPPCA provides collision detection and resolution for cooperating airplanes, which can communicate together. IPPCA is based on high-level flight plan variations using evasion maneuvers. The *flight plan* (FP) is a geometrical description including time information of airplane's flight trajectory consisting of a sequence of basic elements. In IPPCA, a valid FP fulfilling all restrictions on airplane's flight dynamics (like a bounded angular velocity as in our case) is produced by the flight path planner, which constructs the optimal FP (with respect to the selected criterion) using a given sequence of waypoints and considering their speed and time constraints.

The conflict detection part in IPPCA is implemented in the following way. Airplanes use the subscribe-advertise protocol for sharing their local intentions. Each airplane provides a limited future part of its current FP (local FP) to others. This future part is restricted to cover only a time horizon where the collision avoidance should be applied. Each airplane provides an updated local FP each time when its current flight plan is modified or the old local FP does not cover the required future time horizon. Each time an airplane receives an updated local FP from another airplane, it performs a *collision inspection*. During the collision inspection, the received local FP and airplane's current flight plan is searched for a collision—a situation where both airplanes are closer than the required separation distance. If such a collision exists, the conflict resolution part is started.

The conflict resolution in IPPCA is based on the *pair negotiation on removing collision* (PNRC) process. During PNRC, an airplane still receives and inspects local FPs from other airplanes searching for a collision. If a new collision is detected, it is checked which one has higher priority (earlier collision). If the currently running PNRC is started for a collision with a lower priority, it is interrupted and the new PNRC for the collision with a higher priority is invoked.

The PNRC process searches for modified flight plans of both participating airplanes removing the identified collision. First, PNRC prepares sets of their FP alternatives initially including the airplanes' current FPs and initializes the parameter defining the strength of deviation applied by an evasion maneuver. PNRC works in the loop until the solution is found. In each loop, both participating airplanes extend their sets of FP alternatives with new modified FPs generated by the flight path planner as a result of evasion maneuvers using the current deviation parameter. In this paper, IPPCA works with only two evasion maneuvers: *turn left* and *turn right*. The turn left maneuver deviates the original flight trajectory to the left starting from the point of the identified collision. The deviation parameter defines how much the trajectory is deviated from the original one. The turn right maneuver does the same but to the right side. Only FPs, which do not collide prior to the collision being solved with known FPs of other airplanes are included in the sets of alternatives. Then, both these sets are combined together and only pairwise FP combinations, which do not cause the same or earlier mutual collisions are inserted in the candidate set.

If the candidate set is empty, the loop is repeated with an increased deviation parameter. In the next round, new larger deviations are included and more combinations are tested. If there is at least one combination of FPs in the candidate set, PNRC is finished and the PNRC solution is selected as the combination with the minimal cost according to the selected criterion. For example, the combination that has the minimal fuel consumption for both airplanes together. If there are several alternatives with the same minimal cost, the solution is chosen randomly from them. Finally, both involved airplanes apply modified FPs and dispatch updated local FPs to all their subscribers. A new collision inspection is then performed and new PNRCs are invoked for remaining collisions. Thus, IPPCA solves a complex collision situation of several airplanes (like the configuration used for evaluation) by a sequence of FPs' modifications given by pairwise negotiations. The detailed description of IPPCA can be found in [22].

VIII. Evaluation

Both *parallelized PC* (see Section V) and *semicentralized PC* (see Section VI) approaches have been compared against the *iterative peer-to-peer collision avoidance* (IPPCA)

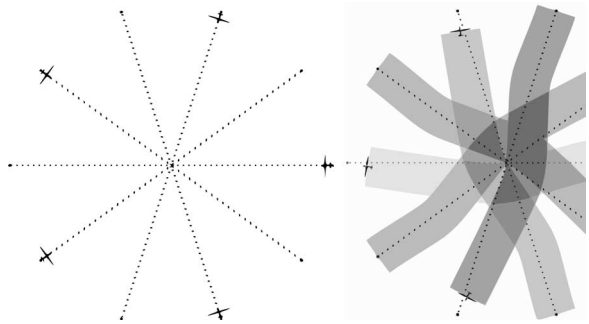

Fig. 5. Superconflict experimental setup with five airplanes (left). Final trajectories after optimization-based conflict resolution—the width of the gray area behind each airplane correspond to the required separation distance (right).

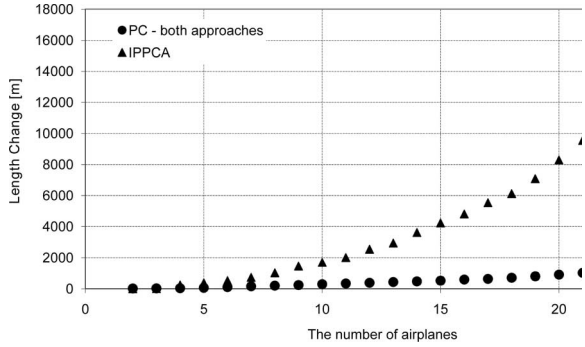

Fig. 6. Average length addition to the original flight trajectory (diameter of the circle 13 km) of each airplane to provide required separation.

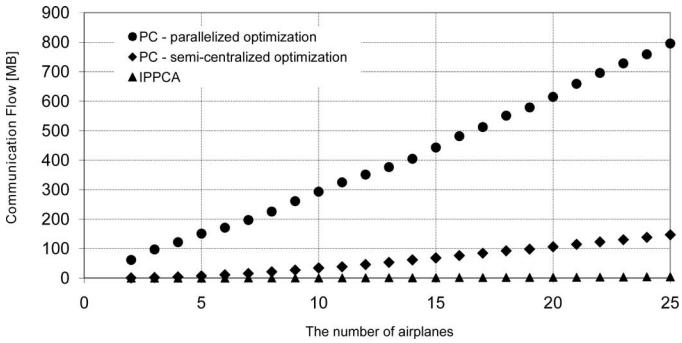

Fig. 7. Average communication flow among all airplanes in given configuration to safely fly across the circle.

(see Section VII) in AGENTFLY [21]—a multiagent airspace evaluator. The algorithms have been tested in a superconflict configuration with a varying number of airplanes (from 2 to 25) (see the left side of Fig. 5). In the superconflict setup, all airplanes are initially located on a horizontal circle with diameter 13 km and their missions contain one waypoint for each airplane located on the opposite side of the circle at the same altitude. Airplanes' initial positions are well-proportioned on the circle—the spacing between any two neighboring airplanes is the same. All airplanes are started at the same time and they have the same constant flight speed $v_i = 35$ m/s. Such a setup causes all airplanes to collide in the circle's center, if no conflict resolution is applied. All airplanes have the same restriction on the flight dynamics restricting the maximum angular velocity to $\omega_i^{\max} = 4^\circ/\text{s}$. The collision resolution is required to provide separation at least $R_i = 500$ m at any time during the flight.

Although the optimization task is well defined for an automated conflict resolution, the PC optimizer is based on a stochastic procedure and thus the conflict resolution result can be different in each run. On the other hand, IPPCA solves collisions in an iterative manner in pairs. The negotiation-based resolution of a conflict for any one pair is deterministic, but if there are several results with the same cost, the one that is actually applied is chosen randomly. The order of pairwise negotiations in collisions of more than two airplanes is given by an asynchronous nature of the collision detection process. Thus, IPPCA can also provide a different result in each run. Each superconflict setup with the given number of airplanes has been measured in 50 repetitive runs. All results present average values from the same configurations. The parameters for the PC optimization were set as follows: the sample block size $N_{\text{SB}} = 180$, the balancing factor $\alpha = 1$, the collision penalty time factor $\beta = 0.995$, the optimization period $\Delta t = 10$ s, the sampling interval $\Delta\tau = 1$ s, the look-ahead horizon $k_{\max} = 125$, the number of discrete actions $m_i = 7$. IPPCA was configured to use only three evasion maneuvers: *straight*, *turn left*, and *turn right*. To provide a correct comparison of both conflict resolution methods, the restriction of flight dynamics is the same. IPPCA includes flight dynamics restriction in the path planner, where the minimal horizontal turn radius is used. For experiments, the minimal turn radius restriction corresponds to the value uniquely derived from v_i and $\omega_i^{\max}$. Among all runs, no experiment was observed that violated the separation radius. An example result of the optimization-based conflict resolution for five airplanes in the superconflict setup is shown on the right side of Fig. 5.

The first chart (see Fig. 6) compares the quality of the solution provided by optimization-based and IPPCA negotiation-based methods. In both algorithms, the solution is optimized for lengths of trajectories (deviations from optimal solutions in the optimization-based version) in the optimization criterion. Average values for *semicentralized* and *parallelized* integration approaches are almost the same. Using the same restriction on flight dynamics, the trajectory lengthening for PC-based methods is up to 12% and for IPPCA is up to 118% of the original length for each airplane in the configuration with 25 airplanes. For configurations with two and three airplanes, PC provides a solution with a similar lengthening as IPPCA and for an increasing number of airplanes, the lengthening for PC is up to ten times smaller than for IPPCA.

The chart in Fig. 7 presents the overall communication flow, which was exchanged among all airplanes during the whole flight. For a semicentralized PC, messages used for the migration of CP are included. The CP (see Algorithm 2) is required to visit each airplane twice during each optimization: 1) once CP collects current airplanes' states and 2) when CP applies control actions to all airplanes. A very small amount of the communication flow is used for the migration protocol itself and the major part contains an internal state of CP during the migration (values of its variables and an execution stack). For parallelized PC (see Algorithm 1), the communication flow is given

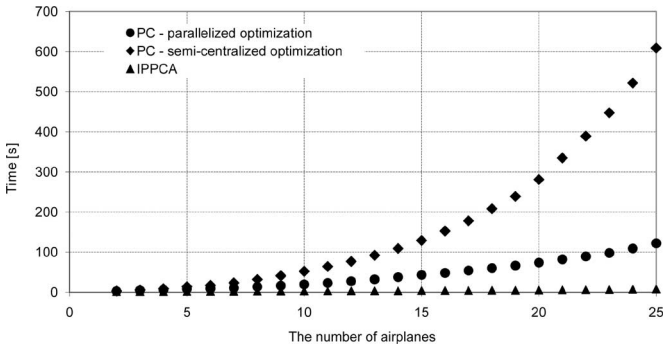

Fig. 8. Averagetime spent for finding a solution for the given setup.

by messages exchanging components of the common objective function and synchronizing the PC optimization process. For IPPCA, the communication flow includes messages necessary for exchanging partial flight plans, proposed flight plan modifications, and messages required by the peer-to-peer negotiation protocol. There are also included messages, which are used for regular updates of flight plans. The lowest communication flow is generated by IPPCA. It requires about 5 MB in total for 25 airplanes. But experiments showed that the flow amount increases quadratically in the number of airplanes in the collision configuration. On the other hand, both PC-based methods require more communication: up to 140 MB for semicentralized PC and up to 800 MB for parallelized PC. The parallelized PC requires about six times higher communication flow than the semicentralized PC for the same collision situation. However, the flow increases linearly in the number of airplanes for both PC. The linear dependence is given by using a fixed sample block size in the PC optimization. The semicentralized PC linearly increases only the number of required coordination process migrations. The parallelized PC uses multicast messages for exchanging predictions and partial costs, and thus, the increasing number of airplanes causes only a linear increase of dispatched messages (each airplane produces such messages). In both cases, the overall number of optimization iterations is limited due to the annealing schedule, which speeds up the convergence of finding an extreme of the objective function. However, in the case when multicast messaging is not usable, the dependency will be also quadratic for parallelized PC.

The chart in Fig. 8 presents the overall time spent on finding a solution for the given setup. For both PC-based optimizations, it includes the time necessary for running all optimizations (regularly invoked every Δt) in the given configuration. For IPPCA, it includes all the time necessary for exchanging partial flight plans, preparing flight modifications using the flight path planner, and negotiations searching for pairs' solutions. Also, time spent on regular collision recheck is included for IPPCA. The smallest amount of time is required by IPPCA—only 10 s for the configuration with 25 airplanes. IPPCA has quadratic dependence of the time on the number of airplanes. Semicentralized PC has cubic dependence on the number of airplanes. It requires 610 s to solve the configuration with 25 airplanes. The time requirements are reduced to quadratic in parallelized PC by the distribution of computation, where only 120 s are spent on solving the largest conflict.

IX. Conclusion

This paper addresses the problem of autonomous conflict resolution for cooperating airplanes based on communication. The conflict resolution task has been defined as an optimization task specified by a common objective function, where the required airplanes' control actions are defined as input variables of the objective function. For the simplicity of description, conflict resolution actions have been limited to only horizontal control—heading changes. However, the presented approach can be extended and actions can include also vertical and speed control, if necessary. The presented concept considers that all airplanes can communicate and cooperate during optimization. However, the concept can be extended to include noncooperative airplanes flying in the same airspace. This can be done by extension of the common objective function. There can be included a part, which will penalize actions of airplanes causing future separation violations with noncooperative airplanes. Such computation should include prediction of movement of those airplanes based on position observations from transponder replies, ADS-B, or radar.

The PC stochastic optimizer has been applied to solve the complex objective function. The presented collision avoidance method has been implemented in two different versions: the *parallelized* and *semicentralized* optimization approach. The parallelized implementation (see Section V) is much more complex than the semicentralized implementation utilizing the PIM model (see Section VI). The parallelized implementation requires a transformation of the main PC optimization algorithm that is executed by several agents in a parallel way. Synchronization parts have to be carefully inserted in the implementation, while the computation of the common objective function has to be optimally split among all agents in order to minimize the number of the same parts computed by multiple agents redundantly and so that only limited information is exchanged among agents. On the other hand, the semicentralized approach utilizing the PIM model is clearly straightforward from the implementation point of view. The PC optimization is implemented in a centralized way and underlying PIM components automatically take care of their transparent migration within the airplane group. Such an implementation is very fast and requires no modification of the algorithm.

Experimental evaluation showed that the presented optimization approach provides up to ten times shorter trajectory lengthening for each airplane than the iterative IPPCA. Considering the evaluation setup and linear dependence of fuel consumption and trajectory length for the specified flight dynamics, it implies almost 46% savings of fuel for the situation with 25 conflicting airplanes. On the other hand, the optimization-based conflict resolution increased the requirements for communication flow and increased the time spent on the optimization. The semicentralized PC requires 28 times higher communication flow than IPPCA. The price for the complex parallel multiagent PC implementation is compensated by the performance of the algorithm—it reduces the dependence from cubic to quadratic, but at the same time, it increases communication flow. Thus, the PC-based optimization conflict resolution is suitable for the cases, where there is higher preference for the quality of the

solution (reduction of fuel consumption) than for the computational resources. If there is limited communication bandwidth, the centralized implementation is better than the presented parallelized version. However, both PC implementation approaches can be combined together, if airplanes are equipped with processors with more execution cores. In such a case, the optimization function in the coordination process can be programmed as a multithreaded optimizer using a similar code as presented for the parallel approach. Moreover, communication among agents should be replaced by thread-to-thread data exchange.

A. Discussion of the Practicality of Implementation

The presented collision avoidance approach can be deployed in both unmanned aerial vehicles and civilian air-traffic domains. The approach requires that airplanes are equipped with bidirectional data communication infrastructure, which provides airplane-to-airplane communication. It is not possible to use this method without communication equipment. The collision avoidance algorithm can be then integrated directly with an existing flight management system or used as an external component. The external integration requires availability of suitable interfaces, where the algorithm can read the current airplane state and apply changes in the current control (e.g., new heading). The airplane state is used as an input for the algorithm. On the other hand, the resulting control provided by the algorithm is filled back to the flight management system. The available computational resources (internal or external) have major influence on the speed of convergence of the optimization process. The presented algorithm does not address the limitations of computational resources. To provide robust conflict resolution, it can be integrated within a complex conflict resolution architecture, such as the *multilayer collision avoidance* framework [44]. Using this multilayer concept, several approaches with different requirements are combined together. The process of collision avoidance is permanently monitored and depending on various symptoms (not enough time until the collision, limited communication bandwidth), optimization can be interrupted and another algorithm capable of solving the conflict under those circumstances is invoked.

ACKNOWLEDGMENT

The work related to the application of PCs into airplane collision avoidance domain formed part of the Argus DARP (Defence and Aerospace Research Partnership) project. This was a collaborative project involving British Aerospace (BAE) Systems, QinetiQ, Rolls-Royce, the University of Oxford, and the University of Southampton and was funded by the industrial partners together with the U.K. Engineering and Physical Sciences Research Council, Ministry of Defence, Technology Strategy Board.

REFERENCES

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

[2] (2008). Current Market Outlook 2008–2027. The Boeing Co., Chicago, IL. [Online]. Available: http://www.boeing.com/commercial/cmo/pdf/Boeing_Current_Market_Out look_2008_to_2027.pdf

[3] (2009). Airline On-Time Statistics and Delay Causes. U.S. Bureau Transp. Statist., Washington, DC. [Online]. Available: http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp

[4] 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.

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

[6] 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 *Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2005, pp. 1083–1090.

[7] National Research Council Panel on Human Factors in Air Traffic Control Automation, *The Future of Air Traffic Control: Human Factors and Automation*, C. D. Wickens, A. S. Mavor, R. Parasuraman, and J. P. McGee, Eds. Washington, DC: Nat. Acad., 1998.

[8] J. Košecká, C. Tomlin, G. Pappas, and S. Sastry, "Generation of conflict resolution maneuvers for air traffic management," in *Proc. Intell. Robots Syst. Conf.*, Sep. 1997, vol. 3, pp. 1598–1603.

[9] (2008). DARPA Tactical Technology Office Programs. Defense Advanced Research Project Agency, Arlington, VA. [Online]. Available: http://www.darpa.mil/j-ucas

[10] *Unmanned Systems Roadmap 2007–2032*. U.S. Dept. Defense, Washington, DC, 2007.

[11] Z. Sarris, "Survey of UAV applications in civil markets," presented at the 9th Meditarranean Conf. Control Autom., Dubrovnik, Croatia, 2001.

[12] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small UAVs," in *Proc. Amer. Control Conf.*, 2005, pp. 3530–3535.

[13] C. F. Lee and D. H. Wolpert, "Product distribution theory for control of multi-agent systems," in *Proc. 3rd Int. Joint Conf. Autonom. Agents Multiagent Syst.*, 2004, pp. 522–529.

[14] S. Bieniawski and D. H. Wolpert, "Adaptive, distributed control of constrained multi-agent systems," in *Proc. 3rd Int. Joint Conf. Autonom. Agents Multiagent Syst.*, 2004, pp. 1230–1231.

[15] D. H. Wolpert, "Information theory—The bridge connecting bounded rational game theory and statistical physics," in *Complex Engineered Systems*, D. Braha, A. A. Minai, and Y. Bar-Yam, Eds. Berlin, Germany: Springer-Verlag, 2006, pp. 262–290.

[16] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA: Addison-Wesley/Longman, 1989.

[17] J. Schutte and A. Greonwold, "Optimal sizing design of truss structures using the particle swarm optimization algorithm," in *Proc. 9th AIAA/ISSMO Symp. Multidisciplinary Anal. Optim.*, Sep. 2002, AIAA 2002–5639.

[18] S. R. Bieniawski, I. Kroo, and D. H. Wolpert, "Flight control with distributed effectors," in *Proc. AIAA Guid., Navigat., Control Conf.*, Aug. 2005, pp. 2005–6074.

[19] V. Mařík and J. Lažanský, "Idustrial applications of agent technologies," *Control Eng. Pract.*, vol. 15, no. 11, pp. 1364–1380, Nov. 2007.

[20] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE Intell. Syst.*, vol. 20, no. 1, pp. 27–35, Jan./Feb. 2005.

[21] 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.

[22] D. Šišlák, J. Samek, and M. Pěchouček, "Decentralized algorithms for collision avoidance in airspace," in *Proc. 7th Int. Conf. Autonom. Agents Multi-Agent Syst.*, 2008, pp. 543–550.

[23] G. J. Pappas, C. Tomlin, and S. Sastry, "Conflict resolution in multi-agent hybrid systems," in *Proc. IEEE Conf. Dec. Control*, Dec. 1996, vol. 2, pp. 1184–1189.

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

[25] J. Krozel, M. Peters, K. D. Bilimoria, C. Lee, and J. S. Mitchel, "System performance characteristics of centralized and decentralized air traffic separation strategies," presented at the 4th USA/Eur. Air Traffic Manag. R&D Semin., Santa Fe, NM, Dec. 2001.

[26] 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.

[27] 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.
[28] K. Tumer and A. Agogino, "Improving air traffic management with a learning multiagent system," *IEEE Intell. Syst.*, vol. 24, no. 1, pp. 18–21, Jan./Feb. 2009.
[29] D. Fudenberg and J. Tirole, *Game Theory*. New York: MIT Press, 1991.
[30] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*. New York: MIT Press, May 1998.
[31] D. H. Wolpert and S. Bieniawski, "Distributed control by lagrangian steepest descent," in *Proc. 43rd IEEE Conf. Decis. Control*, Dec. 2004, pp. 1562–1567.
[32] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, no. 27, pp. 379–423 and 623–656, 1948.
[33] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[34] S. R. Bieniawski, "Distributed optimization and flight control using collectives," Ph.D. dissertation, Stanford Univ., Stanford, CA, 2005.
[35] Y. Bar-Yam, *Dynamics of Complex Systems*. New York: Perseus, 1997.
[36] N. Metropolis and S. Ulam, "The monte carlo method," *J. Amer. Stat. Assoc.*, vol. 44, no. 247, pp. 335–341, Sep. 1949.
[37] K. M. Ford, N. Suri, K. Kosnar, P. Jisl, P. Benda, M. Pechoucek, and L. Preucil, "A game-based approach to comparing different coordination mechanisms," in *Proc. IEEE Int. Conf. Distrib. Human-Machine Syst.*, 2008, pp. 20–25.
[38] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, and R. Jeffers, "Strong mobility and fine-grained resource control in NOMADS," presented at the 2nd Int. Symp. Agents Systems Appl. and 4th Int. Symp. Mobile Agents, ETH Zürich, Switzerland, 2000.
[39] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, and R. Saavedra, "State Capture and Resource Control for Java: The Design and Implementation of the Aroma Virtual Machine," in *Proc. USENIX JVM Conf. Work Prog. Session,* 2001, pp. 74–83.
[40] R. Quitadamo, D. Ansaloni, N. Suri, K. M. Ford, J. Allen, and G. Cabri, "The PIM: An innovative robot coordination model based on Java thread migration," in *Proc. 6th Int. Symp. Principles Practice Programm. Java*, 2008, pp. 43–51.
[41] R. Quitadamo, G. Cabri, and L. Leonardi, "Mobile JikesRVM: A framework to support transparent Java thread migration," *Sci. Comput. Programm.*, vol. 70, no. 2–3, pp. 221–240, Feb. 2008.
[42] B. Alpern, S. Augart, S. Blackburn, M. Butrico, A. Cocchi, P. Cheng, J. Dolby, S. Fink, D. Grove, M. Hind, K. McKinley, M. Mergen, J. Moss, T. Ngo, V. Sarkar, and M. Trapp, "The Jikes research virtual machine project: Building an open-source research community," *IBM Syst. J.*, vol. 44, no. 2, pp. 399–417, 2005.
[43] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček, "AGLOBE: Agent development platform with inaccessibility and mobility support," in *Software Agent-Based Applications, Platforms and Development Kits*, R. Unland, M. Klusch, and M. Calisti, Eds. Berlin, Germany: Birkhauser Verlag, 2005, pp. 21–46.
[44] 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.*, 2007, pp. 365–370.

David Šišlák received the Ing. degree in technical cybernetics and the Ph.D. degree in artificial intelligence and biocybernetics, both from Czech Technical University, Prague, Czech Republic.

He is also a Research Scientist at Czech Technical University. He is an author or coauthor of cited publications in proceedings of international conferences and journal papers. His research interests include technical cybernetics and multiagent systems involved in 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 is a recipient of the 2005 IEEE/ Web Intelligence Consortium/ Association for Computing Machinery/ Web Intelligence—Intelligent Agent Technology. Joint Conference Best Demo Award and the 2004 International Cooperative Information Agents Workshop system innovation award for AGLOBE multiagent platform and related simulations.

Přemysl Volf received the Mgr. 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.

He is also a Researcher at Czech Technical University. His current research interests include distributed cooperative algorithms used for collision avoidance in air traffic control and verification of these algorithms using theory and prototypes.

Michal Pěchouček received the Graduate degree in technical cybernetics from the Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic, the M.Sc. degree in information technology: knowledge-based systems, from the University of Edinburgh, Edinburgh, U.K., and the Ph.D. degree in artificial intelligence and biocybernetics from Czech Technical University.

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

Dr. Pěchouček has been a Co-Chair of the Autonomous Agents and Multi-Agent Systems Industry Track, Holonic and Multi-Agent System, Knowledge Systems for Coalition Operations, and Central and Eastern European Conference on Multi-Agent Systems (CEEMAS), and a member of the Program Committee of other relevant conferences and workshops. He is the Chair of the European Workshop on Multi-Agent Systems Advisory Board and a member of CEEMAS Steering Committee.

Niranjan Suri received the B.Sc. and M.Sc. degrees in computer science from the University of West Florida, Pensacola, and the Ph.D. degree in computer science from Lancaster University, Lancaster, U.K.

He is a Research Scientist with the Florida Institute for Human and Machine Cognition, Pensacola. His current research is concerned with the notion of agile computing, which supports the opportunistic discovery and exploitation of resources in highly dynamic networked environments. His other research interests include distributed systems, networking, communication protocols, virtual machines, and software agents. He has been a Principal Investigator with numerous research projects sponsored by the U.S. Army Research Laboratory. the U.S. Air Force Research Laboratory, the Defense Advanced Research Projects Agency, the Office of Naval Research, and the National Science Foundation (NSF). He has authored or coauthored more than 50 papers.

Dr. Suri has been on the Technical Program Committees of several international conferences and has been a Reviewer for NSF as well as several international journals.