

Integration of Probability Collectives for Collision Avoidance in AGENTFLY

David Šišlák[†], Pavel Jisl[†], Přemysl Volf[†], Michal Pěchouček[†], David Nicholson[‡],
David Woodhouse[‡] and Niranjan Suri[§]

[†]Agent Technology Center, FEE, Czech Technical University in Prague, Czech Republic

[‡]Advanced Technology Centre, BAE Systems, United Kingdom

[§]Florida Institute for Human & Machine Cognition, Florida, USA

contact author: sislakd@fel.cvut.cz

ABSTRACT

Rising deployment of Unmanned Aerial Assets in complex operations requires innovative automatic coordination algorithms, especially for decentralized collision avoidance. The paper investigates a stochastic Probability Collectives (PC) optimizer for the collision avoidance problem defined as an optimization task where efficiency criteria, collision penalties and airplanes' missions are integrated in an objective function. The paper provides two different implementation approaches of the proposed method – complex multi-agent deployment distributes the PC optimization process among airplanes and the Process Integrated Mechanism approach replaces communication with optimizer migration. Both implementations have been developed and tested on an airspace multi-agent framework AGENTFLY.

Categories and Subject Descriptors

I.2.11 [Computing Methodologies]: Artificial Intelligence—*Multi-agent systems*; I.2.9 [Computing Methodologies]: Artificial Intelligence—*Autonomous vehicles*

General Terms

Algorithms, Experimentation

Keywords

Autonomous aircraft, Probability Collectives, Collision avoidance, Decentralized algorithms

1. INTRODUCTION

Application of automatic coordination algorithms in the field of *collision avoidance* (CA) in the air traffic domain is very important nowadays. Several unmanned air vehicles (UAVs) operate cooperatively fulfilling their mission goals in the shared air space [1] and thus airplanes are required to implement automatic See & Avoid capability [9]. A distributed See & Avoid capability allows airplanes to implement the *free flight* concept [14] – an approach of autonomous routing

Cite as: Integration of Probability Collectives for Collision Avoidance in AGENTFLY, David Šišlák, Pavel Jisl, Přemysl Volf, Michal Pěchouček, David Nicholson, David Woodhouse, Niranjan Suri, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 69 – 76

Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org), All rights reserved.

of the aircraft based on a local collision avoidance mechanism. The free flight concept is also studied within manned air traffic control in the *Next Generation Air Transportation Systems* (NGATS) for coordination within the enroute part [19] while traditional *Air Traffic Management* (ATM) is still responsible for operations in terminal parts. Such approach to the collision avoidance problem removes the permanent communication connection between airplanes and the air space control center during enroute parts. The industrial potential of automatic collision avoidance is great as it should be applied to all multiple UAVs deployments [22]. For example in the forest fire monitoring application [7], UAVs monitor large forest fire in areas inaccessible by ground vehicles. The paper addresses the field of the *co-operative collision avoidance* problem where airplanes are equipped with bi-directional communication devices allowing communication within a limited range.

The automated decentralized conflict resolution in a free flight domain is widely addressed by the research community [16]. One approach is based on a hybrid system combining discrete events and individual dynamics modeled by differential equations [20]. A computational geometry approach based on a Delaunay diagram is reported in [8]. Next, potential and vortex fields can also be used for generating heading changes for multi-airplane conflicts [15]. Others use solutions based on agent to agent negotiation using various protocols [30].

In the paper, an optimization approach to the collision avoidance problem has been adopted – airplanes search such a series of actions that would allow them to avoid collision effectively. Efficiency criterion, collision penalty and airplanes' goals are incorporated into a shared objective function. The optimal control is a set of actions which minimizes the function (see Section 3). The *Probability Collectives* (PC) framework [17, 31] is used as an optimization solver. The PC is a stochastic optimizer using probabilistic operators optimizing over a variable space. A brief introduction to the PC algorithm is provided in Section 2. Other existing stochastic approaches such as Genetic Algorithms [13] and Particle Swarm Optimization [23] operate on the design variables rather than on their probability distributions.

Using the PC algorithm makes the designed control *robust*, *adaptive* and *distributable*. The PC algorithm minimizes the objective over all possible instances of noise and disturbance. There is no initial assumption about construction of an objective function which can be easily modified

(also during the flight) without updating the optimization framework. The major benefit of the PC optimizer is that the whole optimization process can be distributed among several agents controlling airplanes – several parts can be performed simultaneously. The PC algorithm has already been successfully deployed for the flight vehicle control [5] – a large number of small, simple, trailing-edge devices controlling vehicle dynamics.

The next contribution of the paper is in comparison of two implementation approaches to the described problem: *multi-agent* and *Process Integrated Mechanism* (PIM). In the first approach (Section 4) each optimized variable from PC is mapped to one agent controlling one airplane. This approach can fully profit from a parallelized execution of the PC optimization, but on the other hand it requires a complex negotiation protocol. The PIM approach (Section 5) replaces negotiation with migration in the respective airplane group and the whole PC algorithm is performed only by the PIM process. However, in our approach the PIM process is required to migrate during the optimization procedure because airplane model restrictions are not shared among them.

Both approaches have been implemented in the AGENT-FLY system – a scalable, agent-based technology for free-flight simulation, planning and collision avoidance [25]. The default airspace simulation based on complex flight plans [27] has been replaced by airplane models reacting to a given control action (e.g. change heading) resulting from the optimization process specified above. Experiments comparing properties of both implementation approaches and studying their integration complexity as well are provided in Section 7.

2. PROBABILITY COLLECTIVES

Details about the Probability Collectives (PC) theory applicable to an optimization problem with discrete variables are provided in this section. The unconstrained optimization problem is defined as

$$\arg \min_{\bar{x} \in \mathcal{X}} G(\bar{x}), \quad (1)$$

where G is the *objective function* and $\bar{x} = \langle x_1, x_2, \dots, x_N \rangle$ is the variable vector, also called *joint move* $\bar{x} \in \mathcal{X}$. The PC theory can be viewed as an extension of the conventional game theory. There exists 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) [12]. Each player i chooses 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 coupling among players occurs indirectly – their probabilities are updated using the received reward based on the objective function $G(\bar{x})$ combining all variables. The probability distribution of the *joint-moves* $q(\bar{x})$ is

$$q(\bar{x}) = \prod_{i \in \mathcal{I}} q_i(x_i). \quad (2)$$

Bounded rational players [11] balance their choice of the best move with the need to explore other possible moves. 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 [31, 32]. This equilibrium corresponds to at least a local minimum of the original objective function G . The expected world utility of all players with a common world utility G under given players' distributions $q_i(x_i)$ is

$$E_q(G(\bar{x})) = \sum_{\bar{x} \in \mathcal{X}} [G(\bar{x}) \prod_{i \in \mathcal{I}} q_i(x_i)]. \quad (3)$$

In the Nash equilibrium, every player adopts a mixed strategy that minimizes its expected utility with respect to the mixed strategies of the others. 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* [24]¹ of the distribution $q(\bar{x})$

$$S(q) = - \sum_{\bar{x} \in \mathcal{X}} [q(\bar{x}) \ln[q(\bar{x})]]. \quad (4)$$

Using the *maximum entropy principle* (Maxent) [18], each player searches for the probability distribution q that minimizes the expected utility

$$\arg \min_q E_q(G(\bar{x})), \quad (5)$$

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

$$\mathcal{L}(q, T) \equiv E_q(G(\bar{x})) + T[s - S(q)], \quad (6)$$

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

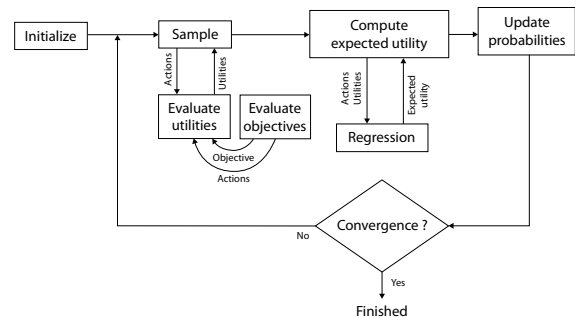


Figure 1: Iterative procedure lowering $E_q(G(\bar{x}))$ [4].

The algorithm lowering $E_q(G(\bar{x}))$ is an iterative procedure of the following steps (see Figure 1, [4]):

1. Initialize the value of the Lagrange parameter T .
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).

¹The distribution with minimal information is the one that does not distinguish at all between the various x and the most informative distribution is the one that specifies a single possible x .

4. The \bar{x} selected according to the final q is the solution of Equation 1.

The sequence lowering T is the *annealing schedule*, in reference to the Simulated Annealing [2]. For a given T , each player i optimizes

$$\begin{aligned} \mathcal{L}_i(q_i, T) &= \sum_{x_i \in \mathcal{X}_i} [q_i(x_i)E[G|x_i]] - TS(q_i) \\ &= \sum_{x_i \in \mathcal{X}_i} [q_i^2(x_i) \prod_{j \in \mathcal{I}, j \neq i} [q_j(x_j)G(x_i, x_j)]] \\ &\quad - T \sum_{x_i \in \mathcal{X}_i} [q_i(x_i) \ln[q_i(x_i)]] , \end{aligned} \quad (7)$$

The function \mathcal{L}_i is convex, has a single minimum in the interior, the temperature T controls the trade-off between exploration and exploitation [32]. The first term $\sum_{x_i \in \mathcal{X}_i} [q_i(x_i)E[G|x_i]]$ in Equation 7 is minimized by a perfectly *rational* player while the second term $TS(q_i)$ is minimized by a perfectly *irrational* player (by a perfectly uniform mixed strategy q_i). In the limit, $T \rightarrow 0$, the set of q that simultaneously minimizes the Lagrangian is the same as the set of q minimizing the objective function G .

3. PROBABILITY COLLECTIVES FOR COLLISION AVOIDANCE

The problem of detection and removal of collisions in a group of airplanes is included in the *optimal control* problem. The optimal control problem is formulated as finding the control inputs (heading changes) which minimize a common objective function that penalizes deviations from target oriented controls (fly to their desired way-points) and collision occurrence (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). It is supposed that all airplanes can communicate together and always participate in the running PC optimization² which is invoked regularly after a given interval Δt .

For simplicity of description, it is supposed that there are N airplanes $\mathcal{A} = \langle A_1, A_2, \dots, A_N \rangle$ which are flying at the same flight level and this level is not changed during the flight and 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 and $g_i \in \{0, 1\}$ specifies whether an action is taken considering the entire future trajectory ($g_i = 0$) or just for the next interval Δt and then proceeding to the airplane's target ($g_i = 1$). Cruise speeds v_i of all A_i are fixed. For each A_i , the next target way-point \bar{w}_i is specified. The function $\mathbf{pos}(A_i)$ returns the current position \bar{p}_{A_i} . Whenever the Euclidean distance $|\bar{p}_{A_i}, \bar{w}_i|$ is below the specified tolerance, \bar{w}_i is accomplished by the airplane A_i . In this case, \bar{w}_i is set to the next mission way-point of A_i if there is any. If its final mission way-point has been already reached, the A_i is removed from the set of existing airplanes \mathcal{A} . The separation distance R_i specifies that there should not be any other airplane closer than R_i to A_i .

For each PC optimization, the definition set \mathcal{X}_i for x_i contains $2m_i + 1$ control actions³. There are m_i actions with

²Alternatively, there can be running several separate independent PC optimization covering airplanes within communication range, as there are no mutual influences between distant airplanes due to the look ahead limitation in the construction of objective function.

³The PC optimization objective function is based only on

ω_i values evenly selected from an interval $\langle -\omega_i^{max}, \omega_i^{max} \rangle$ with $g_i = 0$ (repeated application of ω_i), m_i actions with ω_i values selected the same way but with $g_i = 1$ (a single application of ω_i and then proceeding to \bar{w}_i) and the single action x_i^{opt} . The ω_i^{max} bounds the maximum available angular velocity, m_i is an odd integer greater than 2 (this ensures that the straight flight manoeuvre is included) and x_i^{opt} is an optimal control action with respect to the specified Δt navigating A_i directly to \bar{w}_i . Actions in the set \mathcal{X}_i are ordered by ω_i values (it does not matter whether ascending or descending).

The $\Delta\tau$ is a general parameter for all \mathcal{A} defining the future trajectory sampling interval. 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 $\mathbf{pos}(A_i)$ and A_i constraints. The common objective function $G(\bar{x})$ used for evaluation of joint samples in the PC optimization is

$$G(\bar{x}) = G^{col}(\bar{x}) + \alpha G^{dev}(\bar{x}) . \quad (8)$$

It consists of two parts which are summed together using a balancing factor α . The G^{col} penalizes separation violation among all \mathcal{A} using planned future positions and requires separation distances. It is computed as

$$\begin{aligned} G^{col}(\bar{x}) &= \sum_{A_i \in \mathcal{A}} \sum_{\substack{A_j \in \mathcal{A} \\ A_j \neq A_i}} G_i^{col}(x_i, x_j) , \\ G_i^{col}(x_i, x_j) &= \sum_{k=1}^{k_{max}} \beta^{(k-1)} [\max(R_i - |\mathbf{f}_i(x_i, k), \mathbf{f}_j(x_j, k)|, 0)]^2 , \end{aligned} \quad (9)$$

where the factor $\beta \in (0, 1)$ is used for balancing the penalty between earlier and later collisions (smaller β penalizes more earlier separation violations) and the k_{max} defines the look-ahead horizon which is reflected in the objective function. The second part of the objective function G^{dev} penalizes deviation from airplanes' optimal trajectories to their next way-points,

$$\begin{aligned} G^{dev}(\bar{x}) &= \sum_{A_i \in \mathcal{A}} G_i^{dev}(x_i) , \\ G_i^{dev}(x_i) &= [\mathbf{f}_i(x_i, k_{max}) - \mathbf{f}_i(x_i^{opt}, k_{max})]^2 . \end{aligned} \quad (10)$$

The deviation is expressed as the 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 the case when other airplanes are not considered.

4. MULTI-AGENT APPROACH

The Probability Collectives approach is usable for distributed stochastic optimization using Multi-Agent Systems [3]. Collectives in the PC algorithm can be taken as groups of self-interested, learning agents that act together to minimize the objective function (Equation 1), see Figure 2. Each variable is maintained by one agent. Thus, each agent searches for the optimal action for one airplane (Section 3). In this section, $A_i \in \mathcal{A}$ denotes the agent providing control to 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

the current control input ω_i and do not consider any future control changes after Δt . Thus, such extension provides wider chance to select given action which is anyway updated in next optimization after Δt .

$G(\bar{x})$, Equation 8, and the convergence test requires cooperation of all agents. Sampling and updating of all variables in the iterative procedure of the PC algorithm are performed independently.

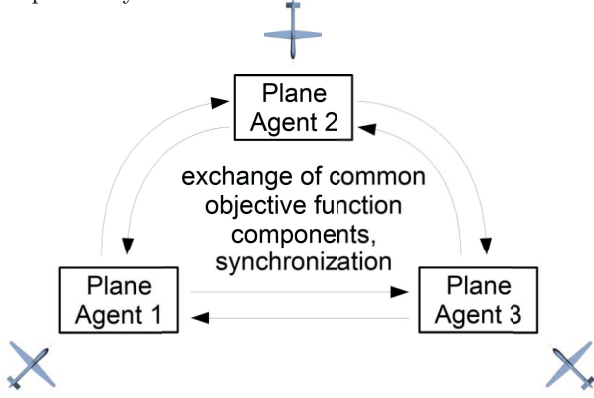


Figure 2: Multi-agent implementation of Probabilistic Collectives.

The A_i is configured using several airplane-oriented parameters: the maximum available angular velocity ω_i^{max} , the number of discrete steers m_i and the separation distance R_i . Moreover, the A_i manages its given mission and defines the next way-point 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 look-ahead step interval $\Delta\tau$, the balancing factor α , the collision penalty time factor β and the annealing schedule parameters.

Algorithm 1 presents a distributed implementation of the PC optimization procedure executed by each agent. The same algorithm is invoked regularly with Δt period based on the current time for all airplanes. It is required that all airplanes have their time synchronized with an error much smaller than Δt . First, each agent performs an initial setup of the optimal action x_i^{opt} , definition set \mathcal{X}_i , probability distribution q_i as uniform discrete distribution over \mathcal{X}_i and temperature T (lines 1–4).

The iterative optimization loop lowering $E_q(G(\bar{x}))$ from Figure 1 is implemented at lines 5–26. Agents prepare sample blocks s_i (N_{SB} actions selected from \mathcal{X}_i using Monte-Carlo sampling) and prediction points $pred_i$ (for each action in s_i agents apply function $f_i(x_i, k)$ where $k = 1..k_{max}$), lines 6 and 7. Then agents exchange their $pred_i$, lines 8 and 9. Computation of the common objective function $G(\bar{x})$ (equations 8, 9 and 10) for each joint action \bar{x} in sample block s_i is distributed among all agents. Each agent A_i computes G_i of the rewritten objective function

$$G(\bar{x}) = \sum_{A_i \in \mathcal{A}} G_i(\bar{x}),$$

$$G_i(\bar{x}) = \alpha G_i^{dev}(x_i) + \sum_{\substack{A_j \in \mathcal{A} \\ A_j \neq A_i}} G_i^{col}(x_i, x_j). \quad (11)$$

The deviation part of the objective function $G_i^{dev}(x_i)$ is prepared at line 10. Each agent waits for other sample block predictions from $\mathcal{A} \setminus A_i$ agents and adds the collision part of the objective function for each processed $pred_j$, lines 11–15. After processing all predictions each agent has the value $G_i(\bar{x})$ for each sample block action and it sends these values to all other agents, lines 16 and 17. Then, the agent

Input: \mathcal{A}

Output: ω_i

```

{1}  $x_i^{opt} \leftarrow$  Optimal action( $\bar{w}_i$ );
{2}  $\mathcal{X}_i \leftarrow$  Get  $x_i$  definition set( $\omega_i^{max}, m_i, x_i^{opt}$ );
{3}  $q_i \leftarrow$  Get initial distribution( $\mathcal{X}_i$ );
{4}  $T \leftarrow$  Initialize temperature;
{5} while true do
{6}    $s_i[N_{SB}] \leftarrow$  Sampling( $\mathcal{X}_i, q_i, N_{SB}$ );
{7}    $pred_i[N_{SB} \times k_{max}] \leftarrow$  Predictions( $s_i, k_{max}$ );
{8}    $wait\_set \leftarrow \mathcal{A} \setminus A_i$ ;
{9}   Send( $pred_i, wait\_set$ );
{10}   $G_i[N_{SB}] \leftarrow$  Own dev cost( $pred_i, x_i^{opt}, \alpha$ );
{11}  while  $wait\_set \neq \emptyset$  do
{12}     $A_j, pred_j \leftarrow$  Fetch other predictions;
{13}     $G_i \leftarrow$  Add col cost( $G_i, R_i, \beta, pred_i, pred_j$ );
{14}     $wait\_set \leftarrow wait\_set \setminus A_j$ ;
{15}  end
{16}   $wait\_set \leftarrow \mathcal{A} \setminus A_i$ ;
{17}  Send( $G_i, wait\_set$ );
{18}  while  $wait\_set \neq \emptyset$  do
{19}     $A_j, G_j \leftarrow$  Fetch other costs;
{20}     $G_i \leftarrow$  Add costs( $G_i, G_j$ );
{21}     $wait\_set \leftarrow wait\_set \setminus A_j$ ;
{22}  end
{23}   $q_i \leftarrow$  Update distribution( $\mathcal{X}_i, q_i, s_i, G_i, T$ );
{24}   $T \leftarrow$  Update temperature( $T$ );
{25}  if Converged( $G_i$ ) then break;
{26} end
{27} return Sample final control( $\mathcal{X}_i, q_i$ );

```

Algorithm 1: Agent PC optimization pseudocode

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 .

Update of the agent probability distribution minimizing Lagrangian $\mathcal{L}_i(q_i, T)$ with the current 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 $G(\bar{x})$ values. Finally, the agent selects the final control according to its stabilized probability distribution q_i , line 27.

5. PROCESS INTEGRATED MECHANISM APPROACH

The Process Integrated Mechanism (PIM) [10] is a novel architecture which enjoys the advantages of having a single controlling authority while avoiding the 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 coordinating process is rapidly moved among the component parts of the PIM.

The PIM model consists of a single *Coordinating Process* (CP) and a set of *Components* each capable of running the CP. The CP cycles among the components at a speed that is sufficient to meet the overall coordination needs of the PIM (e.g. so the planes can coordinate their reaction to

new events in a timely manner), see Figure 3. The time that CP runs on a component is called its *Residency Time*. Each component maintains the code for CP, so the controlling process can move from component to component by passing only a small run-time state using the mechanism of *strong mobility* [28]. The underlying PIM runtime 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 coordinating process operating with a unified global view where, in fact, data remains distributed across the components.

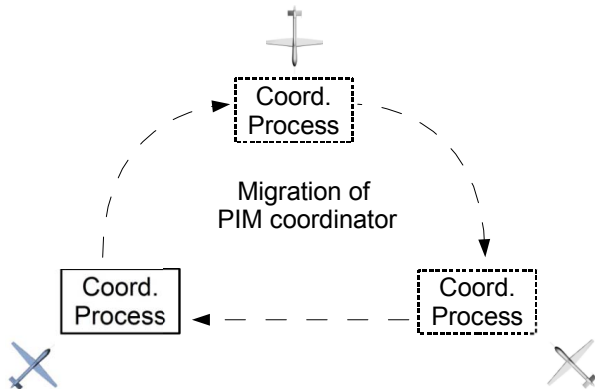


Figure 3: Process Integrated Mechanism approach to Probability Collectives.

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 at the same time as though each was the only process running on that machine. The programmer could construct the program with no concern for the details of the process switching that is actually happening on the processor. To the programmer it is as though 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 the 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 appear to be a single entity (e.g. a PIM). The programmer needs not be concerned with the details of the process moving among the processors. In the time sharing model, it is important that each process is run sufficiently frequently to preserve the illusion that it is constantly running on the machine. Likewise, with the PIM model, it is important that the CP runs on each component sufficiently frequently to react appropriately to any changing circumstances in the environment.

There are two implementations of JVM supporting PIM concept natively. The first implementation is based on the Aroma VM [29], which provides the necessary primitives to asynchronously capture and move the execution state of threads running inside the VM. Aroma allows the capture and migration of the CP between any two Java bytecode instructions, thereby providing fine-grained and accu-

rate control over the residency time of the CP at each node. However, Aroma does not provide a Just-in-time Compiler, thereby providing less performance than other VMs. The second prototype implementation is based on the Mobile Jikes RVM [6, 21], which is a modified version of the Jikes Research VM developed by IBM. The state capture mechanism provided by the Mobile Jikes RVM is not as fine-grained as Aroma. However, the Jikes RVM provides a Just-in-time Compiler, thereby providing performance that is close to the commercial Java VMs.

6. PIM AS MOBILE AGENT

Evaluation experiments were performed using the AGENTFLY system which requires at least Java 1.6. Thus we do not modify AGENTFLY to run on Aroma or Jike RVM. For these reasons we decided to implement the PIM control concept as a *mobile agent* (MA). The AGENTFLY system is built on top of AGLOBE multi-agent platform [26]. The migration in AGLOBE uses a *weak mobility* – a mobile agent moves from one entity to another with some data in the network upon its own request. It is based on messaging. When MA intends to migrate, the state of the agent is saved and is serialized using the JVM methods. The serialized state of the agent is then migrated to the next entity where the agent is restarted. The agent is migrated with its knowledge, stored in internal structures of the agent. The code for the agent can be already present on the target computer, thus during migration AGLOBE verifies its version. The detailed description of migration of agents can be found in [26]. In contrast, the original PIM approach exploits *strong mobility* (or *thread migration*), that allows a thread to transparently transfer its execution to another JVM, along with its computational state.

Traditionally, the PIM’s CP runs on each component for a specified *residency time*. This time specifies how long the CP resides on each component and its amount of computation it can do. There is a trade-off between the reactivity of PIM and this amount of computation. A longer residency time reduces the total fraction of time lost to transmission delays, thereby increasing the computational efficiency of the PIM at the cost of increasing the latency of the CP as it moves among components, thereby decreasing the coordination and reactivity of the PIM. Conversely, a shorter residency time enhances the system’s ability to coordinate overall responses to new and unexpected events since the overall cycle time of the CP will be shorter. But as we reduce the residency time, we increase the ratio of the overhead associated with moving the CP and thus decrease the computation available to the PIM for problem solving.

In the MA implementation we do not prefer to migrate the agent in specified time slices. The MA is migrated when the computation of each step of the PC algorithm is finished. The MA is notified when it has migrated to another host and requests the data it needs for computation (e.g. samples or future plans), then the computation is started. When the computation is finished on the host, MA migrates to another host.

The main advantage of the PIM is simplified development of coordination processes. In PIM implementation of PC, we run only one instance of CP in the group of entities and we don’t need to solve synchronization issues. In fact, there is one implementation of PC algorithm (Section 3) distributed among all airplanes. Using the PIM concept, the paralleliza-

tion benefits of the coordination algorithm cannot be used.

7. EVALUATION

The *Probability Collectives* algorithm for collision avoidance was tested with the distributed multi-agent approach (Section 4) and semi-centralized Process Integrated Mechanism (Section 6). Both approaches use the same PC algorithm with differences in the coordination part of the algorithm. In the multi-agent implementation, distributed message-based negotiation is utilized and in the PIM implementation the centralized migrating agents is exploited.

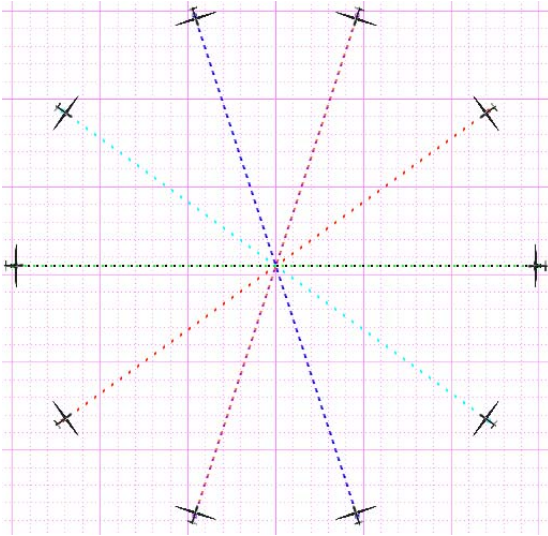


Figure 4: The super-conflict experimental setup with 10 airplanes. The diameter of the circle where the start and goal way-points are located is 13 km.

We performed a set of repetitive tests for comparison of both methods. A sequence of 20 repetitive runs in a super-conflict configuration (see Figure 4) with a varying number of airplanes (from 2 to 10) was carried out for each method. The parameters were set as follows: $N_{SB} = 20$, $\alpha = 1$, $\beta = 0.995$, $\Delta t = 5 s$, $\Delta \tau = 1 s$, $k_{max} = 125$, $v_i = 35 m/s$, $\omega_i^{max} = 4^\circ/s$, $m_i = 5$ and $R_i = 500 m$. Airplane parameters v_i , ω_i^{max} , m_i and R_i are the same for all airplanes in the configuration.

In Figure 5, the time spent in all optimizations for given configuration is presented. The presented time sums the duration required for all PC optimizations which are started regularly each 5 seconds during the flight simulation. The PIM approach consumes much more time than the multi-agent approach. This is caused by the fact that only one coordinator is running in the PIM architecture. The measured time also includes the time required for the coordinator's migration. The number of required migration loops increases with the number of airplanes in the configuration and the size of coordinator knowledge base increases as well. All other airplanes waste their time when they are not hosting the coordinator process. The time decrease for configurations with 5 and 6 airplanes is caused by a lower number of PC algorithm loops before it converges to the solution. The same decrease was observed also in the multi-agent implementation.

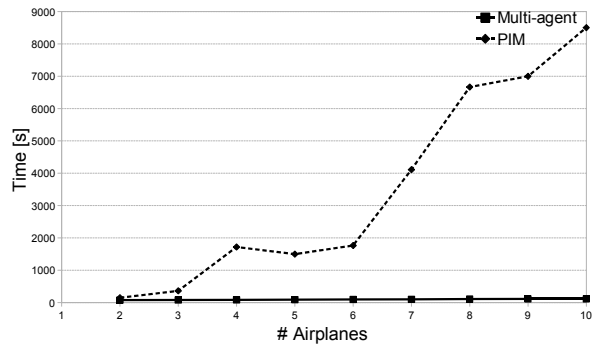


Figure 5: The time spent for all optimizations during whole flight (averaged from 20 repetitive runs).

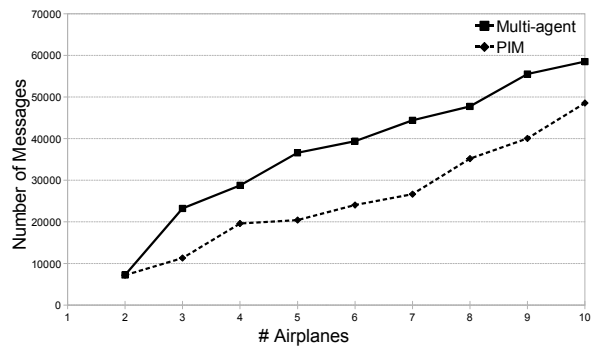


Figure 6: The overall number of messages exchanged among all agents in the given configuration (averaged from 20 repetitive runs).

In Figure 6, the overall number of messages required for coordination behavior of all airplanes is shown. The number of messages required for the PIM approach is lower than in the distributed implementation, because each message in PIM means one jump of the coordinating process. As the multi-agent approach is using the complex negotiation protocol, this approach needs more messages for PC optimization.

Figure 7 presents the total amount of message flow among all airplanes required for their coordination in the given situation. The PIM approach has much higher message flow in the system. As the PIM coordinator collects data from all entities in the system, the required amount of transmitted data is higher than in the distributed implementation. In each step, the PIM coordinator receives a new set of data from each entity and this information must be stored and transferred to other entities. Thus this causes high transfers of knowledge. This problem is tightly coupled with the overall speed of the system. In each PIM coordination step, this knowledge must be serialized (see implementation details in Section 6) and this slows down the response of the system. On the other hand, the multi-agent approach can utilize multi-casted messages which can further save the communication bandwidth especially in the configuration with more airplanes.

The presented evaluation compares two different approaches for solving Probability Collectives optimization for collision avoidance coordination of airplanes. Based on the experimental evaluation only, the multi-agent approach utilizing

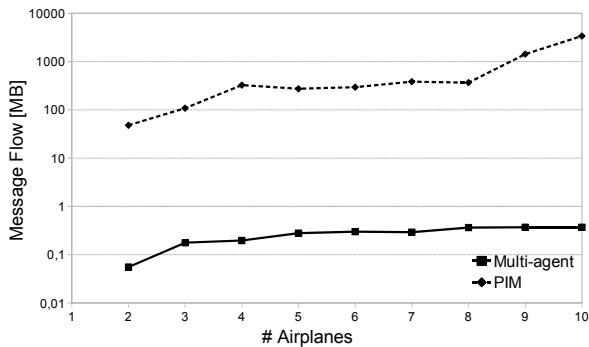


Figure 7: Total communication flow among all airplanes (averaged from 20 repetitive runs).

the parallelization opportunity of the algorithm is much more efficient for this coordination algorithm. The main disadvantage of the PIM approach is the size of knowledge required for coordination. Not all migrating knowledge is necessary at every airplane. Moreover, the loss of parallelization wastes available computational power of all other entities, where the coordinator is not present.

Issues were also identified with the scalability of the PIM approach. In the configuration with 5 and more airplanes, it is almost impossible to compute the optimal solution for collision avoidance in a reasonable time interval (see Figures 7 and 5). The presented values can be slightly affected by the chosen PIM implementation as a mobile agent. In the case where the PIM coordinator will be implemented in one of the PIM frameworks, the values for the total flow in the system will be almost the same, because the mobile agent implementation uses externalization approach minimizing the size of the serialized agent to its minimum. On the other hand, the strong mobility in native PIM frameworks will provide higher performance. This doesn't affect the coordination performance as the computational power is not as constrained as the available communication bandwidth among airplanes.

8. CONCLUSION

The paper addresses the problem of automated decentralized collision avoidance among cooperating airplanes. The conflict resolution task has been defined as an optimization task specified by a common objective function. The required control actions for all airplanes are defined as input variables of the objective function. The *Probability Collectives* (PC) stochastic optimizer is used for optimization of the objective function. Its use makes the optimization robust against noise and introduces a potential for distribution.

The presented collision avoidance approach has been implemented in two different versions: *multi-agent* (MA) and *process integrated mechanism* (PIM). The MA implementation requires transformation of the main PC optimization algorithm that is executed by several agents in parallel. Synchronization parts have to be carefully inserted in the implementation and the common objective function has to be optimally split among all agents so that minimum parts are computed by more agents redundantly. The PIM approach is clearly straight-forward as the PC optimization algorithm is implemented as a semi-centralized part migrating among

airplanes in a loop. Such implementation is very fast and does not require any modification of the algorithm.

During an experimental evaluation of both implementations, it was identified that the price for the complex multi-agent implementation is compensated by the implementation performance. Especially in the configuration with higher number of airplanes, the PIM approach becomes unusable for real-time coordination as the communication flow increases quickly in this specific case. However, advanced techniques can be applied to provide better performance also for larger coordination groups. For example, a large group of airplanes can be divided into several smaller airplane groups, introducing some kind of hierarchy to the coordination. It means that each of these smaller groups is controlled by its own low-level PIM coordinator and group-to-group coordination is done by a high-level PIM process migrating between these groups. This is possible mainly in the coordination tasks where the algorithm can be split into several parts without any impact on the coordination goal.

Acknowledgement

This work formed part of the Argus DARP (Defence and Aerospace Research Partnership) project. This was a collaborative project involving BAE Systems, QinetiQ, Rolls-Royce, the University of Oxford and the University of Southampton, and was funded by the industrial partners together with the UK EPSRC, MoD, and TSB.

The Process Integrated Mechanism concept is supported in part by the U.S. Air Force office of Scientific Research under grant number FA9550-08-1-0218 and in the past by the Office of Naval Research Coordinated Operations program and the U.S. Army Research Laboratory's TEAM Performance program. Further support is 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, USAF, under grant number FA8655-06-1-3073 and by Czech Ministry of Education grant number 6840770038.

9. REFERENCES

- [1] D. A. R. P. Agency. DARPA Tactical Technology Office Programs. <http://www.darpa.mil/j-ucas>.
- [2] Y. Bar-Yam. *Dynamics of Complex Systems*. Perseus Books, 1997.
- [3] S. Bieniawski and D. H. Wolpert. Adaptive, distributed control of constrained multi-agent systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Washington, DC, USA, 2004. IEEE Computer Society.
- [4] S. R. Bieniawski. Distributed optimization and flight control using collectives. *Dissertation Stanford University*, 2005.
- [5] S. R. Bieniawski, I. Kroo, and D. H. Wolpert. Flight control with distributed effectors. In *2005 AIAA Guidance, Navigation, and Control Conference*, pages AIAA 2005-6074, August.
- [6] G. Cabri, L. Ferrari, L. Leonardi, and R. Quitadamo. Strong agent mobility for aglets based on the IBM JikesRVM. In *SAC '06: Proceedings of the 2006 ACM*

- symposium on Applied computing*, pages 90–95, New York, NY, USA, 2006. ACM.
- [7] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra. Forest fire monitoring with multiple small UAVs. In *Proceedings of the American Control Conference*, pages 3530–3535, 2005.
- [8] Y. J. Chiang, J. T. Klosowski, C. Lee, and J. S. B. Mitchell. Geometric algorithms for conflict detection/resolution in air traffic management. In *Proc. of IEEE Conf. on Decision and Control*, pages 1835–1840, December 1997.
- [9] DOD. Unmanned systems roadmap 2005-2030, 2007.
- [10] 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 *IEEE International Conference on Distributed Human-Machine Systems (DHMS)*. IEEE, 2008.
- [11] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, May 1998.
- [12] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [13] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [14] J. C. Hill, F. R. Johnson, J. K. Archibald, R. L. Frost, and W. C. Stirling. A cooperative multi-agent approach to free flight. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1083–1090, New York, NY, USA, 2005. ACM Press.
- [15] J. Kosecka, C. Tomlin, G. Pappas, and S. Sastry. Generation of conflict resolution manoeuvres for air traffic management. In *Proceedings of Intelligent Robots and Systems*, volume 3, pages 1598–1603, September 1997.
- [16] J. Krozel, M. Peters, K. D. Bilimoria, C. Lee, and J. S. Mitchell. System performance characteristics of centralized and decentralized air traffic separation strategies. In *4th USA/Europe Air Traffic Management R & D Seminar*, Stanta Fe, NM, December 2001.
- [17] C. F. Lee and D. H. Wolpert. Product distribution theory for control of multi-agent systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 522–529, Washington, DC, USA, 2004. IEEE Computer Society.
- [18] D. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.
- [19] National Research Council Panel on Human Factors in Air Traffic Control Automation. *The Future of Air Traffic Control: Human Factors and Automation*. National Academy Press, 1998.
- [20] G. J. Pappas, C. Tomlin, and S. Sastry. Conflict resolution for multi-agent hybrid systems. In *Proc. of IEEE Conf. on Decision and Control*, volume 2, pages 1184–1189, December 1996.
- [21] 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 *PPPJ '08: Proceedings of the 6th international symposium on Principles and practice of programming in Java*, pages 43–51, New York, NY, USA, 2008. ACM.
- [22] Z. Sarris. Survey of UAV applications in civil markets. In *Proceedings of the 9th Mediterranean Conference on Control and Automation*, 2001.
- [23] J. Schutte and A. Gronwold. Optimal sizing design of truss structures using the particle swarm optimization algorithm. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages AIAA 2002–5639, September 2002.
- [24] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, (27):379–423 and 623–656, 1948.
- [25] D. Šišlák, M. Pěchouček, P. Volf, D. Pavlíček, J. Samek, V. Mařík, and P. Losiewicz. AGENTFLY: Towards Multi-Agent Technology in Free Flight Air Traffic Control. In M. Pěchouček, S. Thompson, and H. Voss, editors, *Defense Industry Applications of Autonomous Agents and Multi-Agent Systems*, pages 73–97. Birkhauser Verlag, 2008.
- [26] D. Sislak, M. Rehak, M. Pechoucek, M. Rollo, and D. Pavlicek. AGLOBE: Agent Development Platform with Inaccessibility and Mobility Support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.
- [27] D. Šišlák, J. Samek, and M. Pěchouček. Decentralized algorithms for collision avoidance in airspace. In Padgham, Parkes, Mueller, and Parsons, editors, *Proceedings of 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, New York, NY, USA, 2008. ACM.
- [28] 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. In *Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents (ASA/MA 2000)*.
- [29] 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 *USENIX JVM 2001 Conference Work in Progress Session. Extended version available as a Technical Report*. USENIX, 2001.
- [30] S. Wollkind, J. Valasek, and T. R. Ioerger. Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *Proc. of the American Inst. of Aeronautics and Astronautics Conference on Guidance, Navigation, and Control*, Providence, RI, 2004.
- [31] D. H. Wolpert. Information theory – the bridge connecting bounded rational game theory and statistical physics. In D. Braha, A. A. Minai, and Y. Bar-Yam, editors, *Complex Engineered Systems*, pages 262–290, Berlin, 2006. Springer.
- [32] D. H. Wolpert and S. Bieniawski. Distributed control by lagrangian steepest descent. In *43th IEEE Conference on Decision and Control*, December 2004.