The velocity assignment problem for conflict resolution with multiple aerial vehicles sharing airspace

D. Alejo*, J. M. Díaz-Báñez**, J. A. Cobano*, P. Pérez-Lantero*** and A. Ollero*$\tilde{*}$
*Robotics, Vision and Control Group
Engineering School, University of Seville
41092 Seville, Spain
**Applied Mathematics II Department
Engineering School, University of Seville
41092 Seville, Spain
***Escuela de Ingeniería Civil en Informática, Universidad de Valparaíso
2340000 Valparaíso, Chile
$\tilde{*}$Center for Advanced Aerospace Technologies (CATEC)
Aeropolis, Aerospace Technological Park of Andalusia
41309 La Rinconada-Seville, Spain
[dalejo,jacobano,aollero]@cartuja.us.es,dbanez@us.es,pablo.perez@uv.cl

***Abstract*—Efficient conflict resolution methods for multiple aerial vehicles sharing airspace are presented. The problem of assigning a velocity profile to each aerial vehicle in real time, such that the separation between them is greater than a given safety distance, is considered and the total deviation from the initial planned trajectory is minimized. The proposed methods involve the use of appropriate airspace discretization. In the paper it is demonstrated that this aerial vehicle velocity assignment problem is NP-hard. Then, the paper presents three different collision detection and resolution methods based on speed planning. The paper also presents simulations and studies for several scenarios.**

I. Introduction

Automation technologies will play an important role in the different applications of Unmanned Aerial Vehicles (UAVs) and Air Traffic Management (ATM) systems that should satisfy high air traffic demand. Moreover, the integration of UAVs in non-segregated aerial spaces is being also considered for the future ATM [1]. Systems with multiple UAVs present significant advantages in different applications by increasing efficiency, performance, reconfigurability, and robustness [2][3]. In these scenarios, the UAVs should maintain as much as possible the planned trajectories but also maintaining minimum separation between them.

UAV conflict detection and resolution (CDR) methods have been studied extensively. An overview of papers on deconfliction can be found in [4], where the CDR methods are characterized depending on the following factors: dimensions of the state information, technique for dynamic state propagation, conflict detection threshold, conflict resolution technique, maneuvering dimensions, and management of multiple UAV conflicts. There are methods for optimal coordinated maneuvers but these require significant computational effort [5]. Another CDR method is based on a mixed-integer linear program (MILP) to optimize the total flight time by modifying velocity or heading [6]. However, this method only allows one speed change for each aircraft. There are also several methods in which conflicts are solved by changing only the speed [7]. Some of these methods are devoted only to pairwise conflicts and involve computational studies to forecast future traffic levels [8]. The geometric approach presented in [9] is also for pairwise non-

cooperative aircraft collision avoidance. In [10] the uncertainties encountered when solving pairwise conflicts are considered. The method described in [11] is based on MILP to solve pairwise conflicts by changing speeds for a large number of aircrafts.

The method proposed in [12] considers the problem in three dimensions using mixed-integer nonlinear programming to solve the conflicts with only velocity changes by minimizing the total flight time. However, this approach requires a high computation time.

The method in [13] is based on the use of genetic algorithms to solve conflicts by considering the problem of optimal path planning between given waypoints. The main drawback of genetic algorithms is that the computation time is not predictable and the convergence to a solution is not ensured in a finite time interval. The optimal control method in [14] uses a kinematic model of the aircraft flying in a horizontal plane with constant velocity and solves conflicts for two or three aircraft by changing the heading. Another approach [15] involves the application of a hybrid system model that generates safe maneuvers. On the other hand, the stochastic method described in [16] is based on the Monte Carlo approach and the computation time is again high.

In this paper, a 3D conflict resolution problem for multiple UAVs sharing airspace is studied, where a state propagation method is needed to predict trajectories. There are numerous techniques to predict positions in conflict detection [17][18][19][20][21]. In this work it is assumed that the initial trajectories of each UAV are known (given by a sequence of waypoints) and the detection method is based on a grid model. Therefore, each trajectory is defined by the cells through which the corresponding UAV passes. The methods proposed in this paper are based on speed planning to solve conflicts. The velocity profiles for all the UAVs involved in a conflict are computed and the problem is solved by maintaining the planned 3D space trajectory. This approach has the advantage that the probability of creating new conflicts with other UAV in the airspace is low.

A dynamic environment is also considered. In this environment the conflicts between two or more UAVs are solved in real time, at low computational cost, once they are detected. Three methods are presented here. In the first one, the problem is reduced to a scheduling problem and then a greedy approach is considered in order to find an optimal solution. In the second method, a discrete velocity allocation (DVA) problem considering pairs of velocities is implemented to solve the conflicts. Finally, the third proposed method is based on the technique presented in [22], in which suboptimal solutions are found. This method presents several advantages with respect to [22]: it solves conflicts with more than two UAVs and ensures minimum separation between UAVs. In this paper different kind of scenarios are simulated to point out the characteristics of the proposed methods.

This paper is organized in seven sections. Section II presents the problem formulation. Proof that the Velocity Assignment Problem (VAP) is NP-hard is provided in Section III. The proposed conflict resolution methods are described in Section IV. Finally, section V shows the simulations carried out and Section VI details the conclusions.

II. PROBLEM FORMULATION

The problem considered in this paper concerns conflict detection and resolution between UAVs in a common airspace. The detection algorithm is based on the discretization of the airspace divided into cubic cells, also called the grid model (see Figure 1). Other possible way to encode the statement of the problem are [23][24]. The discretization is chosen in this paper because the detection algorithm is simpler and faster. Moreover, a trajectory can be parameterized by the number of cells that the UAV passes through with entrance and departure time. The size of the cells is a parameter and the safety distance is given by a number of cells. The resolution algorithm is based on changing the velocity profile of the UAVs involved in the potential collision. Note that velocity refers to speed in this problem because changes of velocity direction are not considered.

The time for which each UAV stays in a cell depends on its model. It is assumed that each UAV knows the trajectories of other UAVs, i.e., the list of cells that other UAVs will fly across (see Figure 2). Thus, in Figure 1 the cells through which UAV1 passes are: 7, 4, 5, 2, 3. In the case of UAV2 they are: 9, 6, 5, 4, 1. Both UAVs fly in the same flight level.

Method described in [22] presents a disadvantage. Collision is defined when two UAVs lay in the

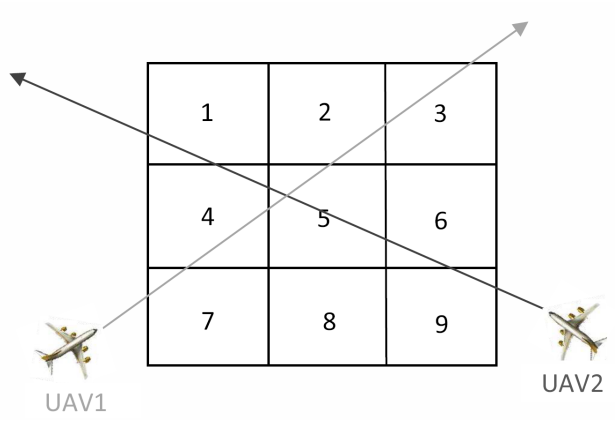

Figure 1. UAV trajectories in a discretized airspace divided into cells.

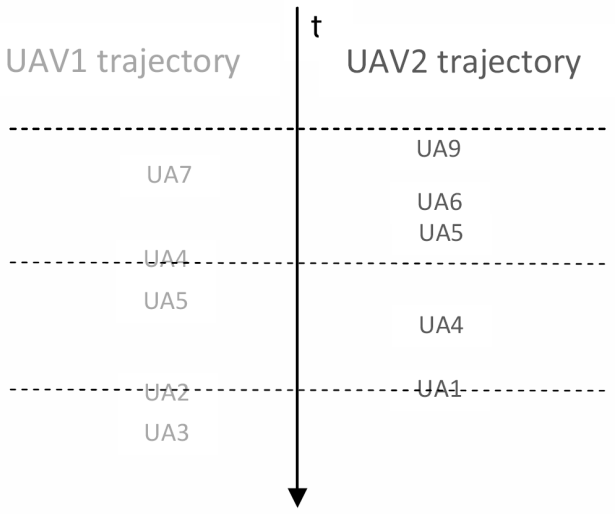

Figure 2. The UAV trajectory is described as a sequence of visited cells.

same cell and at the same time. However, with this definition two UAVs that are not in the same cell can be closer than other two ones that are actually in the same cell (see Figure 3). Therefore the algorithm based on the simple space discretization in cells does not ensure minimum separation between UAVs. A natural idea to cope with this disadvantage is simply to change the conflict definition.

In this paper we consider that two UAVs maintain the minimum separation if they are separated vertically and horizontally by a safety distance. If there is an UAV in a cell C, there is a conflict when there is another UAV in a neighboring cell to C. The following definitions are considered in this paper:

- *Neighboring cells to C*: is the set of cells whose distance is less than the safety distance considered (see Figure 4).

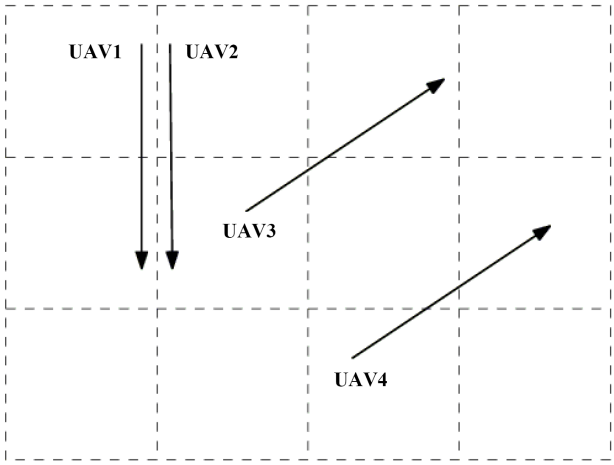

Figure 3. Disadvantage of the grid model: UAV3 and UAV4 are in conflict while UAV1 and UAV2 are not.

- *Conflict*: a cell C crossed by an UAV is in conflict if there is another one which crosses a cell in the neighborhood of C.
- *Conflict Zone* (CZ): set of consecutive cells of two or more UAVs that are in conflict (see Figure 5).
- *Collision*: there are two UAVs crossing a CZ at once.

One of the first approaches of the collision avoidance problem in robotics was proposed in [25], where once all paths have been planned by each aerial vehicle, a velocity profile that avoids collisions in all paths is found by means of the proposed Velocity Planning Problem (VPP) methods assigning velocities.

The VAP addressed in this paper can be defined mathematically as follows: Let $U = \{U_1, ..., U_n\}$ be a set of UAVs represented by points in a three-dimesional space moving with constant initial velocities onto straight lines. Each UAV has a constrained interval of available velocities and when a collision is detected, a velocity is assigned to each of the involved UAV. The initial velocities are modified under the constraints, such that the collision is avoided and the total deviation from the initial velocities is minimized.

In this model, each UAV U_i has an initial trajectory identified by its initial velocity v_i. These velocities are computed to optimally perform a given mission. The method finds new velocities such that a given criterion is optimized. The objective function or total deviation can be modeled as:

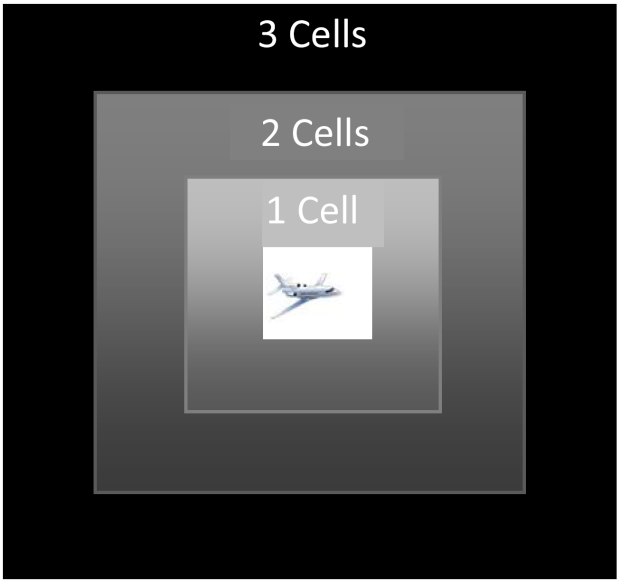

Figure 4. Examples of neighboring cell definition with different safety distances.

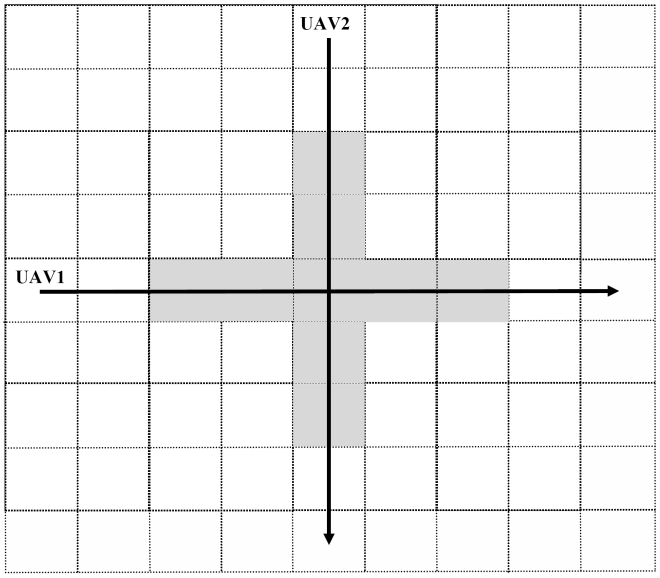

Figure 5. Conflict zone (gray) in a scenario with two UAVs. The safety distance is given by two cells.

$$J = \sum_{i=1}^{n} \sum_{j=1}^{C_i} \left(t_{ij} - t'_{ij}\right)^2 \qquad (1)$$

where n is the number of UAVs of the system, C_i is the number of cells crossed by the U_i, t_{ij} and t'_{ij} are the stay times of the U_i when crossing the cell number j in the solution trajectory and the original trajectory respectively. Basically, the objective is to minimize this total deviation with respect to the initial stay time in each cell.

The VAP is NP-hard; that is to say, a polynomial time algorithm is not possible or, unless, P=NP [26]. Although it is generally assumed that this problem is NP-hard, we do not know where the proof is published. This is proved in Section III. Therefore, we propose some strategies based on velocity planning for the collision avoidance problem of UAVs sharing airspace. Three methods are proposed to avoid collisions (see Section IV): 1) the Greedy method (G), 2) the use of a constrained version of the VAP problem in which pairs of velocities are allocated, and 3) a heuristic approach for velocity planning (VP) based on [22] with non trivial modifications made in order to improve the possible applications.

It is worth noting that it is possible to find a solution that avoids the initial collision but generates a new collision with other UAVs. Therefore, when solving collisions we distinguish three types of UAVs: the UAVs that are directly involved in the detected potential collision, the UAVs whose trajectories collide with the possible solution trajectories of the directly involved UAVs and whose velocities can be changed, and the non-cooperative UAVs that would maintain their initial velocities.

III. NP HARDNESS PROOF

The NP-hardness is showed by considering a reduction from one of the so-called one-machine scheduling problems (see [27] for a comprehensive survey). The problem that is reduced in this case is the "Sequencing with Release Times and Deadlines" (SRD), which is strongly NP-complete [27], [26] and defined as follows: Given N jobs - each associated with a release time, a deadline, and a processing time - decide whether there is a non-preemptive schedule of these N jobs on a single machine such that all jobs meet their deadlines. In other words, the SRD asks for a sequence of execution of the N jobs so that no execution of any job is interrupted to execute any other job, no two jobs are executed at a same time, and every job starts not before its release time and finishes not after its deadline. By exploiting the similarity between jobs which are to be non-preemptively scheduled on a single machine, and UVAs that must pass one after another through a single cell of the discretized space, we prove the following theorem:

Theorem 3.1: The VAP is NP-hard.

Proof: Let I_{SRD} be an instance of the SRD consisting of n jobs $j_1, j_2, \ldots, j_n$. Each job j_i has release time $t_{min}(i)$, processing time $D(i)$, and deadline $t_{max}(i)$. We reduce I_{SRD} to the following instance VAP_{SRD} of the VAP:

- All aerial vehicles $UAV_1, UAV_2, \ldots, UAV_n$ pass at the same time through a same cell C of the discretized space.
- Once a vehicle enters C, it increases its speed to the maximum possible value in order to lie inside C the minimum amount of time, giving more chances to other UAVs of entering C.
- The speed range of each vehicle allows UAV_i enter C at least at time $t_{min}(i)$ and at most at time $t_{max}(i) - D(i)$, where $D(i)$ is the amount of time in which UAV_i is inside C moving at its maximum speed.

Let $t(i)$ denote the time in which UAV_i enters C. If instance VAP_{SRD} has any feasible solution, then we can obtain a sequence $UAV_{\pi_1}, UAV_{\pi_2}, \ldots, UAV_{\pi_n}$ of the UAVs so that they enter C in this order. Furthermore, for all $1 \leq i \leq n$, UAV_{π_i} satisfies both $t_{min}(\pi_i) \leq t(\pi_i)$ and $t(\pi_i) + D(\pi_i) \leq t_{max}(\pi_i)$. Then, jobs $j_1, j_2, \ldots, j_n$ can be scheduled as $j_{\pi_1}, j_{\pi_2}, \ldots, j_{\pi_n}$, where job j_{π_i} starts at time $t(\pi_i)$, giving a solution to instance I_{SRD}. Proving that instance I_{SRD} has a solution implies that instance VAP_{SRD} has a feasible solution is similar. Therefore, there exists a solution to I_{SRD} if and only if VAP_{SRD} has a feasible solution. Then, deciding if the VAP has a feasible solution is NP-complete and the result follows. ■

In subsequent research, the use of heuristics for scheduling problems can be explored. For example, a related problem is the following one: Given N tasks with release times, durations and deadlines, find a non-preemptive schedule such that all tasks meet their deadlines and the number of machines used to process all tasks is minimum. This is known as the "Scheduling with Release Times and Deadlines on a Minimum Number of Machines" and it is NP-hard [27]. In the same paper some approximation algorithms and heuristics were also proposed. It should be noted that if one is able to modify not only the speed but also the altitude of the UAVs in the cell of the space through which they pass at the same time, the cell could be considered a multiprocessor. Each processor of the cell will correspond to a different altitude, in charge of scheduling the entrance and departure times of the UAVs.

IV. PROPOSED METHODS

In this section we present three CDR methods for UAVs, all based on speed planning. When the trajectories are computed, our methods check whether there are some UAVs whose trajectories could be in conflict. When such a case is identified, a further computation decides whether there would be a collision. A potential collision is solved by changing the stay times of each UAV in each cell; that is, by assigning a velocity profile to each UAV.

The proposed methods are centralized. Scalability is one of the most important advantages of decentralized methods have, while centralized methods could present a disadvantage for being prone to failures in the central system. The aim in this paper is to search for near-optimal solutions. For this reason, the planned trajectories of all UAVs must be known. However, scalability can be achieved in centralized methods by applying it not to the whole system but to a subset composed only by the UAVs involved in a conflict (see section II).

A. Greedy method

The Greedy method (G) works as an on-line algorithm in which the decision concerning some UAVs in conflict, whose speed must be modified, does not use any information related to future conflicts. The problem can be reduced to an on-line scheduling problem [28].

Let $U = \{U_1, U_2, ..., U_n\}$ be the UAVs and assume that their initial speeds are the possible maximum ones, respectively. It is also assumed that the cells of the space are enumerated. We further consider that each UAV U_i flies from time $t = 0$ to time $t = t_f$ passing through some cells of the space. Then, for each UAV we partition its flight time-line $[0, t_f)$ into intervals, where each interval corresponds to the period of time in which U_i passes through a cell. In other words, each time-line of any UAV is a consecutive sequence of labeled intervals, meaning the sequences of cells UAV U_i passes through (see Figure 6(a)).

For the sake of simplicity, the Greedy method is presented for conflict cells. However, it can be generalized to conflict zones.

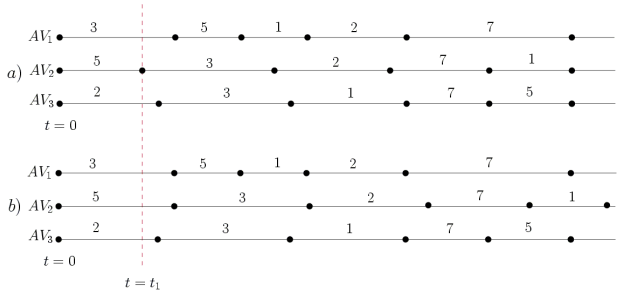

Figure 6. Greedy algorithm. UAV_1 passes through cells 3, 5, 1, 2 y 7. When $t = t_1$ (a) the velocity of UAV_2 is decreased delaying its stay on cell 5 for avoiding the collision with UAV_1 in cell 3 (b).

Let $U_i(t)$ denote the label of the cell in which UAV U_i is at instant of time t. The Greedy method considers the arrangement of labels $U_i(t)$ and makes a sweep from $t = 0$ to t_f while changing the speed of the UAVs accordingly in order to maintain the invariant that, at any time t, no pair of distinct UAVs U_i and U_j, such $U_i(t) = U_j(t)$ exists. In order to achieve this goal the following rule is applied: If at instant t some UAV U_i enters to the cell labeled k and there exists an UAV U_j such that $U_j(t) = k$, then the speed of U_i is decreased to avoid the collision with U_j on that cell. Figure 6 shows an example where the outcome of situation (a) in which UAV_2 tries to enter in cell 3 in which UAV_1 is passing through, is situation (b) where UAV_2 enters cell 3 at the same time as UAV_1 leaves.

Suppose that at instant t in the sweep, UAV U_i is in cell k, and UAV U_j and UAV U_p enter cell k. It must be decided whether U_j or U_p is more retarded. This case is called *decision situation* and the decision taken might cause new conflicts for future instants of time, or possibly lead to a point in which there is no way of avoid collisions between the UAVs.

The Greedy method described above has been implemented as a first or preprocessing step, working as follows. If no decision situation is found then an optimal solution for the VAP problem is obtained. Otherwise, the Greedy method is stopped because it cannot guarantee to obtain an optimal solution. At this point any approximation method (e.g. any of those proposed in following sections) can be applied.

B. The discrete allocation problem

In this section we consider the discrete velocity allocation problem (DVA), that is, the VAP subject to a discrete set of velocities. A solution for this problem gives an approximation to the general continuous VAP. Given a partition of the interval of velocities $v_1, v_2, \ldots, v_m$ assign a v_i to each UAV such that there is no collision between the UAVs when they move constantly with the allocated speed. Firstly, we deal with an easier problem, called the Two Velocity Assignment Problem (2-VAP) in which two fixed velocities v_0 and v_1 are only considered. Let $UAV_1, UAV_2, \ldots, UAV_n$ be the n UAVs.

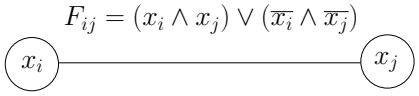

Figure 7. F_{ij} means that UAV_i and UAV_j do not collide if they have the same velocity, either v_0 or v_1.

Theorem 4.1: The 2-VAP can be solved in $O(n^2)$ time in the worst case.

Proof: Consider n logic variables $x_1, x_2, \ldots, x_n$, where $x_i = 0$ if UAV_i is assigned velocity v_0, and $x_i = 1$ otherwise. For each pair UAV_i, UAV_j of UAVs we can test that assignments of velocities do not collide (i.e. assignment of 0's and 1's to x_i and x_j). Then, we can obtain a logic formula $F_{i,j}$, with variables x_i and x_j, such that all its positive interpretations imply an assignment of velocities to both UAV_i and UAV_j, so that they never collide. Each $F_{i,j}$ can be rewritten in Conjunctive Normal Form with two variables per clause (i.e. x_i and x_j) (see Figure 7).

Therefore, our problem is reduced to assigning 0's and 1's to $x_1, x_2, \ldots, x_n$ such that $F = \bigwedge_{i=1}^{n} F_{i,j} = 1$, where F is also in Conjunctive Normal Form with two variables per clause. This is an instance of the 2-SAT problem which can be solved in polynomial time as follows [29]: Let $G_F = \langle V, E \rangle$ be a directed graph, where $V = \{x_1, \bar{x_1}, x_2, \bar{x_2}, \ldots, x_n, \bar{x_n}\}$ and $(\alpha, \beta) \in E$ if and only if there exists in F a clause logically equivalent to $\bar{\alpha} \vee \beta$ (see Figure 8).

It holds that F is unsatisfiable if and only if for some x_i there exist in G_F a path from x_i to $\bar{x_i}$ and a path from $\bar{x_i}$ to x_i. Thus, polynomial time algorithms on path searching in graphs can be applied. A refined algorithm running in $O(n + n')$ time is possible [29], where n' is the number of edges of G_F. Since n' is $O(n^2)$ in the worst case, the result follows. ■

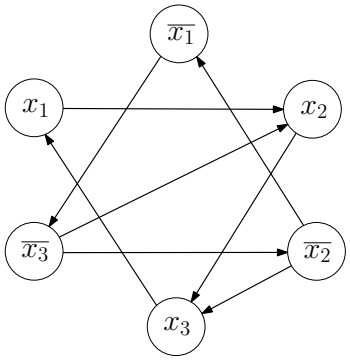

Figure 8. The graph G_F corresponding to $F = (\bar{x_1} \vee x_2) \wedge (\bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_3}) \wedge (x_2 \vee x_3)$.

It is worth noting, using arguments similar to the above ones, that whenever three or more velocities are used instead of two, the problem can be reduced to an instance of the k-SAT problem ($k \geq 3$), which is NP-complete [26].

Using the above algorithm to solve the 2-VAP, we can give an approximation algorithm to the discrete version of the VAP. We solve the 2-VAP for each pair of velocities and, if a pair of velocities avoiding collision exists, it can be identified as an approximation to the optimal trajectory. Then, we select the best pair of velocities found, that is, that minimizing the deviation from the initial trajectory. By applying this approach, we assign two velocities to the UAVs, without collisions, from a finite set of velocities. Since $O(m^2)$ pairs of velocities are tested, the time complexity of this approximation is $O(m^2(n^2 + t))$ in the worst case, where t is the time spent in computing the total deviation from the initial trajectories.

This algorithm is very efficient in dense airspace although the solution is an approximation to the optimal deviation. Therefore, the use of this algorithm involves considering a trade-off between computation time and flight plan deviation.

C. Heuristic velocity planning with optimization phase

This method can also be applied in the case where the Greedy method does not solve the avoiding collision problem. The method, that will be called VP, is based on the algorithm presented in [22]. The method has two steps:

- *Search tree step*, which finds a solution if it exists by exploring all possible arrival orders to each conflict zone, and
- *Optimization step*, which minimizes a cost function.

In this method each UAV trajectory is decomposed into zones that consist of groups of cells (see Section II). In contrast, the method proposed in [22] considers each cell separately.

1) Search tree step: This step asks if the collision avoidance problem can be solved by performing changes in the velocity profiles of the UAVs. The goal of the search tree algorithm is to obtain arrival orders to each CZ that provide a valid solution. In this step, we assume that all the vehicles travel at their maximum velocity. Then it is only possible to decrease their velocities to avoid a collision.

Let us consider a scenario with two UAVs and only one CZ described in Figure 5. The building of a tree is described in Algorithm 1. Each node of a tree represents a visited cell for the UAV. Let us consider two nodes, N_i and N_{i+1}, that are related to neighbor cells, $C(i)$ and $C(i+1)$. We assume that N_i is the parent of N_{i+1}. The corresponding edge between them has assigned a weight w. This weight is calculated according to the following formula and considering maximum speed of each UAV (see step 7):

$$w(N_i, N_{i+1}) = t_{in}(C(i+1)) - t_{in}(C(i)) \qquad (2)$$

where $t_{in}(C(j))$ represents the entrance time to the cell $C(j)$.

Let us also define the arrival order to a CZ as the order in which the UAVs pass through it. This order is determined by the estimated arrival time to the CZ of each UAV (see step 2) and influences the building of each tree in order to decide if the building stops or continuous by comparing the arrival time to the CZ with the time of the UAVs preceding. Therefore, for n vehicles, a CZ has $n!$ different arrival orders. All the possible arrival orders are explored until a solution is found. In the scenario showed in Figure 5, there are two arrival orders and the first one to be tested is UAV2-UAV1 because UAV2 arrives before.

First, the tree is built for UAV1, then for UAV2 and so on when there are more UAVs. Whenever a CZ is reached, the tree only calculates its following weight if all UAVs that precede the UAV which is building its tree, given by arrival order, have already calculated the weights of the edges related to that CZ (steps 8 and 9). Figure 9 represents the

Algorithm 1 Search Tree Algorithm

1: **repeat**
2: Get the arrival order to all conflict zones to be checked.
3: Start the trees of all UAVs.
4: **while** there are some trees not completed and there are not any **unavoidable collision**. **do**
5: **for** each tree **do**
6: **if** the tree is not completed **then**
7: Calculate the minimum weights of the next edges up to the next conflict node.
8: **if** the end was not reached **and** all previous vehicles have calculated the weights of their conflict edges **then**
9: Calculate the weight of the conflict egde.
10: **if** a collision has been detected **then**
11: Go back and create new branches that solve the collision. Backtrack also related trees.
12: **if** the beginning of the tree is reached **then**
13: An **unavoidable collision** has been detected
14: **end if**
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **end while**
20: **until** A solution is found or all arrival orders have been unsuccessfully checked.
21: **return** The arrival order of the solution, or an error if not found.

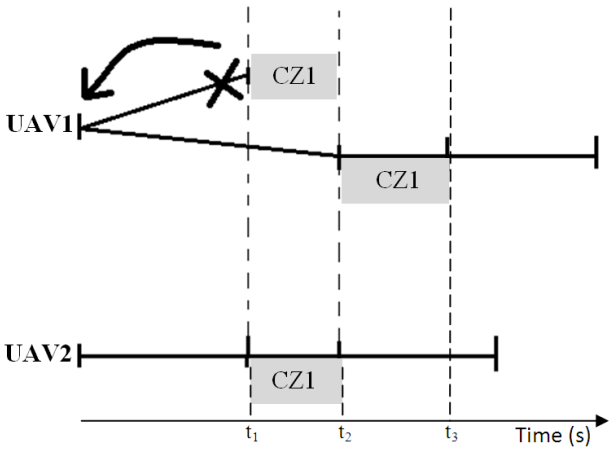

Figure 9. Trees generated in the example scenario. UAV2 tree only has one branch because is the first UAV that passes through CZ1. UAV1 has two branches because a collision has been detected in CZ1 so a backtracking process starts.

complete trees in the proposed example. Note that the horizontal length of each edge is proportional to its weight. In this case, when UAV1 comes in CZ1, instant t_1, it does not continue building its tree because the UAV before it, UAV2, has not calculated its weights of the edge related to that CZ, so the UAV1 tree is stopped and the UAV2 tree is started and completed. Otherwise, the algorithm will continue with the tree of the UAV1.

Whenever a weight related to the CZ is calculated, the algorithm checks if the arrival time to that conflict leads or not to a potential collision with regard to the UAVs preceding. In this case, when the weight associated with the CZ of UAV1 tree is calculated in the upper branch, a potential collision is detected with UAV2. If a potential collision is detected, then the algorithm creates a new branch in the tree, assigning greater weights (that is, greater differences of the entrance time, so speed decreases) in as fewer edges as necessary in order to avoid that potential collision (step 11). The lower UAV1 branch represented in Figure 9 is generated and the potential collision is avoided because UAV1 comes in CZ later, instant t_2, that is, when UAV2 is leaving CZ. If the collision cannot be avoided considering the speed constrains of the UAV, the algorithm fails with the proposed arrival order and another arrival order has to be checked (steps 12 and 13).

The complexity of the algorithm grows as the number of CZs increases. Let us assume that there are m CZs with n UAVs, where n_i of them are involved in the ith conflict. So, a total of $n_1!n_2!, \ldots, n_m!$ different orders should be checked.

The first arrival order is determined by the estimated arrival time to a CZ of each UAV. Then, each tree is built considering the arrival order to detect potential collision. If no solution is found with the first arrival order, a new order should be checked. The algorithm permutes the arrival order to the CZ from the cost function J. The CZ with highest cost is chosen to define the arrival order. This cost is defined as:

$$J_i = \mu_i - \sigma_i \tag{3}$$

where μ_i and σ_i are the mean and standard deviation of the set of estimated arrival times of each UAV to the CZ. The mean of the arrival times to a CZ is considered because a change in earlier CZs can affect the following CZs. The standard deviation is included in order to take into account the differences between the arrival times of the different UAVs to a CZ. In order to find a solution, it is advisable to change the arrival order to a conflict where all the estimated arrival times of the vehicles involved in it are similar.

The backtracking process takes place when a collision is detected and a new tree should be built for one or more UAVs. In this case, the backtracking process can become more complex than the previous case considering one CZ. Note that the backtracking process is only done for the UAVs which arrive to the corresponding CZ later.

2) QP-problem: When the search tree algorithm finds a solution, we have a valid *arrival order* for all the CZs in the collision avoidance problem. At this point, an improvement on the objective function can be done by solving a QP-problem. The QP-problem minimizes a quadratic cost function with linear constraints. Let us consider a cost function in order to obtain the most similar trajectory to the initial one. The considered cost function is:

$$J_{QP} = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \left(\frac{t_{ik} - t_{ik}^{ref}}{t_{ik}^{ref}} \right)^2 \tag{4}$$

where t_{ik} is the stay time in the kth CZ visited by the UAV_i and t_{ik}^{ref} is the stay time in the initial trajectory. n represents the number of UAVs of the system and m_i the number of CZs crossed by the UAV U_i.

It should be noted that a new denominator term has been introduced with respect to (1). This is necessary in this method because the optimization is computed CZ by CZ instead of cell by cell. The stay time in each CZ can be very different, so without this term very large variations in the stay time in small CZs can be produced, leading to saturations in the velocity control signal.

The constraints of the model are:

$$t_{ik} - a_{ik} v_{ik} - b_{ik} \leq 0 \tag{5}$$

$$c_{ik} + d_{ik} - t_{ik} \leq 0 \tag{6}$$

The maximum and minimum stay time in each cell depends on the initial velocity at those cells, v_{ik}. This dependence is in fact non linear, but we have to linearize it in order to formulate the QP-problem. This is achived by interpolation with a_{ik}, b_{ik}, c_{ik} and d_{ik} as the interpolation coefficients. Moreover, the following constraints regarding v_{ik} are considered:

$$v_{ik} - v_{max} \leq 0 \tag{7}$$

$$v_{min} - v_{ik} \leq 0 \tag{8}$$

$$|v_{i,k} - v_{i,k-1}| \leq \frac{a_{i,max} d_{i,k-1}}{v_{ref}} \tag{9}$$

$$\forall i = 1...n, k = 1...m_i$$

The first two constraints are given by the UAV model, and the third one relates the initial velocity in one cell with the other in the previous cell because of the maximum acceleration constraint. In the third equation, $a_{i,max}$ represents the maximum desired acceleration of the UAV_i and $d_{i,k}$ represents the distance traveled by the UAV_i in the kth CZ.

Finally, in order to avoid collisions for each CZ and each UAV_i that has to cross that zone immediately before than an UAV_j, the next constraints should be considered:

$$\sum_{k=1}^{Q} t_{mk} - \sum_{k=1}^{P-1} t_{lk} \leq 0 \tag{10}$$

where P indicates the entry cell in the CZ of the UAV_j and Q indicates the leaving cell of the UAV_i.

The above optimization problem can be solved by means of the QP-solver implemented in the Computational Geometry Algorithms Library (CGAL) [30].

V. Simulations

The three methods have been implemented and several simulations have been carried out. The algorithms have been run in a PC with a 2GHz Dual Core processor and 2 GB of RAM. The operating system used in the simulations was Linux Debian Testing with kernel 2.2.24. The code has been written in the C++ language and compiled with the compiler included in gcc-4.1.2.

In order to compute the trajectories it was necessary to model the behavior of the aerial vehicles. The simple model for a controlled UAV proposed in [31] was used in the simulations. This model allows

us to reduce the computational time expended in simulations. Nevertheless, in the proposed methods it was also possible to use models of arbitrary complexity.

Three different scenarios (S1, S2 and S3) were considered to perform the following four studies:

1) To describe the changes of speed required to solve conflicts by considering non-cooperative UAVs. S1 shown in Figure 10 is used.
2) To analyze how the computing time of each method depends on the considered scenario (see Figure 12 and Figure 13).
3) To obtain values of (1) introduced in Section II in order to compare the kindness of the solutions obtained by each method. The solution is better when the solution trajectory is closer to the initial trajectory; that is, values of (1) are lower.
4) To check how the computing time depends on the safety distance,i. e., the minimum number of cells between two UAVs.

In S1 the size of the cell is $150m$ and the safety distance is 4 cells. In S2, the size is $150m$ and the safety distance is 3 cells, and in S3 they are $100m$ and 6 cells.

In the first study, the goal is to show how the detected conflicts can be solved by changing the speed of each UAV when there are non-cooperative UAVs. In this simulation, five UAVs fly on a circular scenario sharing airspace (see Figure 10). UAV_1 is non-cooperative so the speed cannot be changed. Therefore, UAV_2, UAV_3, UAV_4 and UAV_5 must change their speed profiles to avoid the detected conflicts in the center of the circle. The speed profiles computed for each cooperative UAV are shown in Figure 11. Firstly, all cooperative UAVs decrease their speed before entering the CZ. When an UAV comes into the CZ it increases its speed in order to exit more rapidly. The entry order in this case is: $UAV1, UAV3, UAV4, UAV2, UAV5$. Finally, each UAV returns to its initial speed. It has been shown how several changes of speed for each UAV can solve the conflicts. Thus, the deviation from the initial trajectory is smaller.

Regarding the second study, the results obtained in the Greedy method in S2 and S3 are shown in Table I for different numbers of UAVs. In all cases, the solution is found. The initial speed has been set to the cruise speed ($53.5m/s$). The computing time, $T(s)$, is low and grows almost linearly with

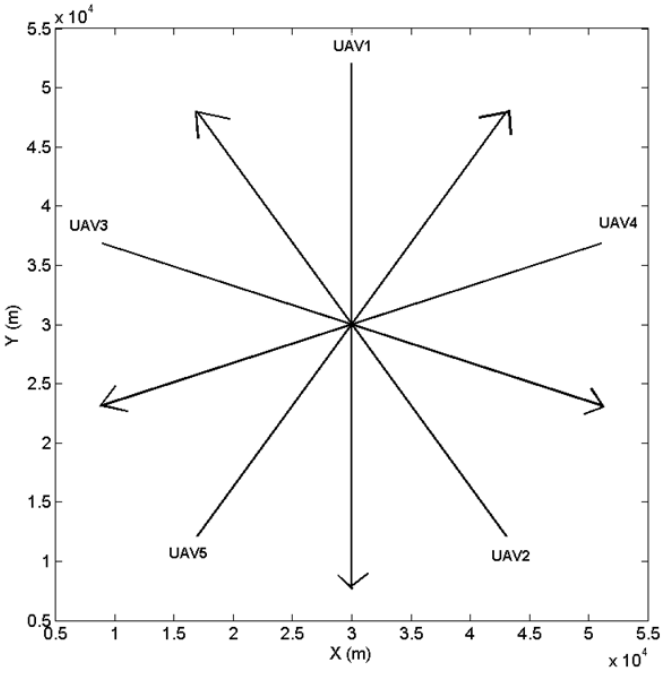

Figure 10. First simulation scenario (S1).

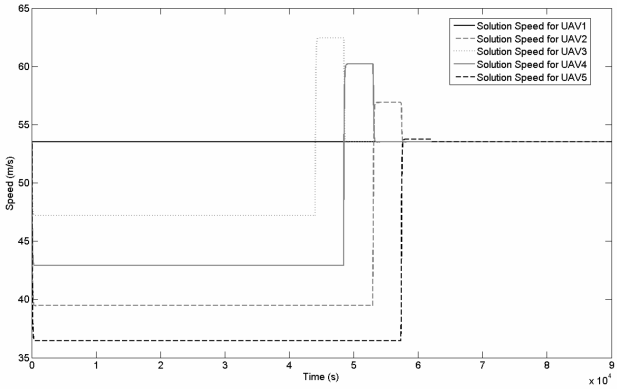

Figure 11. Speeds computed for UAV2, UAV3, UAV4 and UAV5 to avoid the detected conflicts.

the number of UAVs. This method is good when a solution is needed in a short time (from milliseconds to seconds). Finally, if we take into account the comparison criteria J, the values obtained are quite low and in most cases are also the lowest of the three methods.

The results obtained with the DVA method (2-VA) are shown in Table II for $v_1 = 45.0m/s$ and $v_2 = 55.0m/s$. The execution time is the lowest of all three methods that are presented in this paper. Therefore, this method can also be used when a solution needs to be computed in seconds.

An important characteristic of this method is that the computing time and the number of solutions do

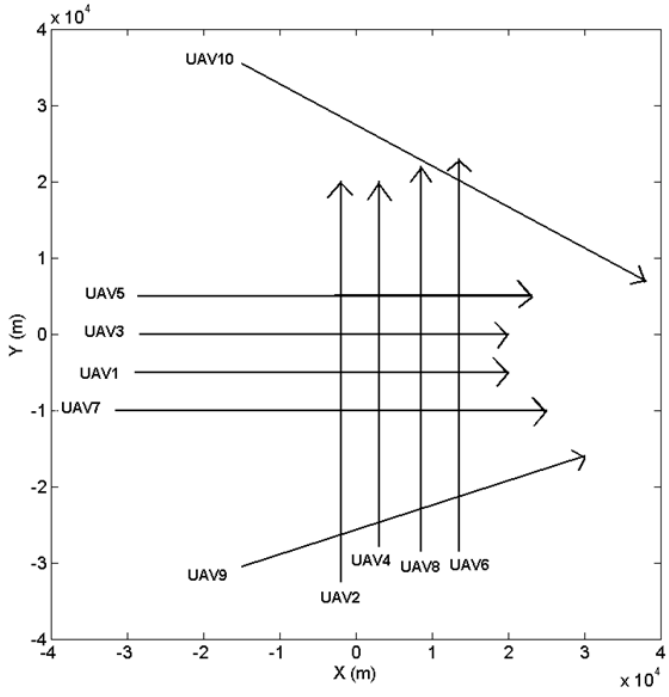

Figure 12. Second simulation scenario (S2).

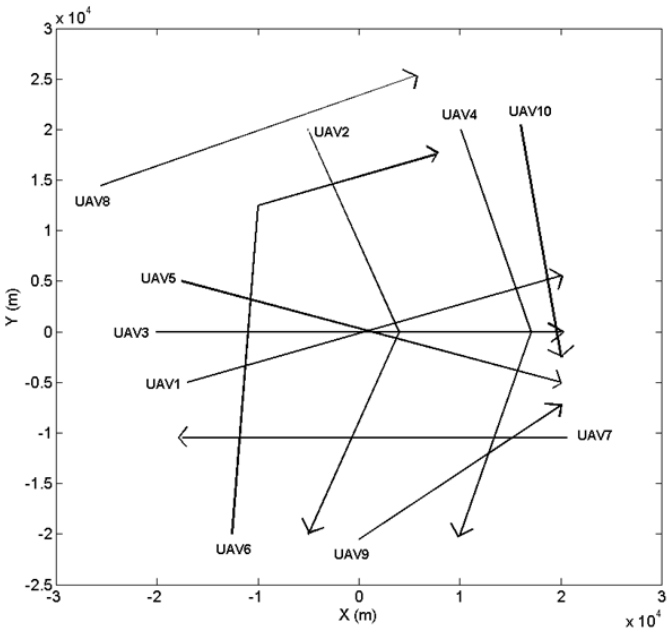

Figure 13. Third simulation scenario (S3).

Table I
RESULTS OBTAINED FROM THE GREEDY APPROACH CONSIDERING S1 AND S2.

UAVs	Scenario 2		Scenario 3	
	T(s)	J(s^2)	T(s)	J(s^2)
2	0.05	3.91	0.04	2.32
3	0.10	3.92	0.10	8.19
4	0.12	3.92	0.13	8.19
5	0.18	4.64	0.19	22.35
6	0.21	4.64	0.31	22.35
7	0.33	4.64	0.37	22.35
8	0.38	4.65	0.44	22.35
9	0.44	4.65	0.52	22.35
10	0.60	4.65	0.59	22.36

not depend on the scenario considered. This method recalculates the stay times of each UAV in all visited cells for both low and high speeds. This method usually provides more than one possible solution, whenever this happens the one that gives the lowest value of J is chosen.

Table II
RESULTS OBTAINED FROM THE PROBLEM WITH DVA METHOD (2-VA): $v_1 = 45m/s, v_2 = 55m/s$.

UAVs	Scenario 1		Scenario 2	
	$T(s)$	$J(s^2)$	$T(s)$	$J(s^2)$
2	0.01	93.96	0.01	30.44
3	0.04	174.40	0.04	111.50
4	0.08	176.20	0.07	113.10
5	0.13	274.70	0.11	182.10
6	0.20	276.60	0.18	183.90
7	0.28	278.70	0.23	185.40
8	0.36	280.60	0.30	186.70
9	0.49	282.70	0.34	187.50
10	0.69	284.40	0.40	188.40

When the number of UAVs is low, most of the time is spent in simulating the trajectories of each UAV. Only one simulation is done for each UAV, so we can conclude that its complexity is approximately $O(n)$, where n is the number of UAVs in the system. However, the time spent in graph search becomes dominant as the number of UAVs becomes greater, as calculated in Section IV-B.

Table III shows the results obtained when using the DVA in order to check a set composed by more than two velocities (see Section IV-B). In these simulations, a set of five velocities $V = \{45.0, 47.5, 50.0, 53.5, 55.0\}m/s$ is con-

sidered, called the Five Velocity Assignment (5-VA). Considering more velocities, better solutions are obtained, i.e. lower values of J. On the other hand, the computing time becomes greater in a very noticeable way. Moreover, this time grows faster as the number of UAVs increases.

Table III
RESULTS OBTAINED FROM THE PROBLEM WITH DVA METHOD (5-VA): $V = \{45, 47.5, 50, 53.5, 55\} m/s$.

UAVs	Scenario 1		Scenario 2	
	$T(s)$	$J(s^2)$	$T(s)$	$J(s^2)$
2	0.26	1.97	0.12	1.61
3	0.80	3.56	0.57	12.7
4	2.26	14.74	1.69	14.2
5	5.54	28.6	3.10	42.0
6	7.71	30.5	7.47	42.0
7	14.9	32.6	11.9	42.0
8	22.7	34.5	14.0	42.0
9	31.9	36.1	23.9	42.0
10	51.2	37.6	32.4	42.0

The computing times obtained with the heuristic VP method in S2 and S3 are shown in Table IV and Table V, respectively. This method finds good solutions but the computing time is higher than that for the Greedy and DVA methods and it significantly increases with the number of UAVs considered. Despite of this, this method can be efficiently applied in real-time to a system composed of a maximum of 6 UAVs when the solution has to be computed in few seconds.

Note that the computing time required to detect conflicts also increases with the number of UAVs because more CZs are detected. The computing time for a QP-problem increases because new CZs appear, thus adding constraints to the system. Similarly, the number of variables considered in the QP-problem also increases when new UAVs are added to the system. Two variables are needed for each CZ that is crossed by an UAV.

Table IV
COMPUTATIONAL TIME SPENT IN ALL PHASES OF THE ALGORITHM AND CRITERIA RESULTS OF THE SOLUTIONS OBTAINED IN S2 FROM THE HEURISTIC VP METHOD.

UAVs	Conflict(s)	S. T.(s)	QP(s)	$T(s)$	$J(s^2)$
2	0.044	0	0.020	0.064	0.857
3	0.188	0	0.096	0.284	0.859
4	0.716	0.004	0.708	1.428	0.861
5	1.804	0.008	2.608	4.420	0.894
6	4.032	0.024	11.425	15.481	0.896
7	7.456	0.028	29.854	37.332	0.967
8	14.065	0.052	83.717	97.834	0.969
9	23.073	0.06 4	179.455	202.593	1.105
10	38.078	0.104	287.046	325.228	1.109

Table V
COMPUTATIONAL TIME SPENT IN ALL PHASES OF THE ALGORITHM AND CRITERIA RESULTS OF THE SOLUTIONS OBTAINED IN S3 FROM THE HEURISTIC VP METHOD.

UAVs	Conflict(s)	ST(s)	QP(s)	$T(s)$	$J(s^2)$
2	0.032	0	0.016	0.048	5.95
3	0.336	0	0.060	0.396	69.97
4	1.128	0.004	0.472	1.604	69.97
5	2.660	0.012	1.356	4.028	217.30
6	6.200	0.020	14.981	21.201	223.20
7	9.677	0.088	37.190	46.955	223.20
8	13.933	0.088	38.930	52.951	223.20
9	18.481	0.112	46.383	64.976	223.20
10	23.913	21.417	78.061	123.392	223.20

The computing time for each method in S2 and S3, can be compared from the results in Tables I, II, III, IV and V. The strong dependency of the computing time needed by both 5-VA and the VP method with the number of UAVs is very significant. On the other hand, this dependency is less significant for the 2-VA and Greedy methods and the growth of their execution time is almost linear. However, the differences in time required are not noticeable if the number of UAVs in the system does not exceed 6.

The computational cost of the collision avoidance methods, when the safety distance (i.e., the number of cells) changes, is shown in Figure 14. Note that this factor does not have noticeable effect on the computing time in the Greedy and DVA methods. In the VP method, however, the computing time increases as the safety distance is increased. This extra time is spent in the conflict detection phase of the algorithm and arises due to the growth of CZs, which in turn means that the conflict database contains more items and it is more expensive to generate it.

VI. CONCLUSIONS

In this paper, the conflict resolution problem for multiple UAVs in a common airspace is studied. It has been proved that this problem is NP-hard (see Section III). Three different methods have been proposed for collision avoidance of UAVs sharing airspace (see Section IV). These methods are based on different strategies: the greedy approach, a discrete velocity allocation (DVA) problem considering

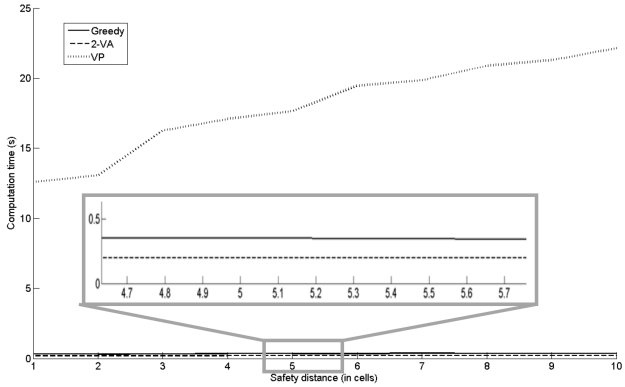

Figure 14. Computing time for each method on varying the number of cells in S2.

pairs of velocities and the velocity planning based on [22] but with some changes made to improve the application conditions.

The proposed methods avoid the conflicts by changing only velocities and maintaining the space trajectories. Furthermore, the conflicts are solved minimizing the time deviations with respect to initial trajectories that are assumed to be optimally computed and then should be maintained as much as possible when avoiding conflicts.

The work described here has led to advances in conflict resolution methods that consider speed changes under dense airspace. The most important advantages with respect to previous works are: low computational time; multiple UAVs are considered in different scenarios with non-cooperative UAV; and the detection of conflicts takes into account several UAVs rather than just a pair of UAVs as, for example in [11]. Furthermore, the third proposed method allows the speed profile to be changed in such a way that each UAV can return to its initial speed after solving the conflict, so more than one speed change is allowed.

The proposed methods can be integrated into a Conflict Detection and Resolution system. The first option would be to compute a solution with the Greedy method because its computing time is lower. This method provides an optimal solution if no decision situation is found. If the Greedy method does not identify a solution, then the DVA method can be used to compute a solution. The main advantage of the Greedy and DVA method is the low requirements to compute the solution in short time. As a final option the VP method can be used to compute a near-optimal solution. This method computes an approximated solution, with a small deviation from the initial trajectory, but the computing time is higher and then its application is constrained by the computational resources required to obtain the solution within the time constraints.

All implemented methods have been validated for a different numbers of UAVs and in different scenarios. Our study suggests that the proposed methods are suitable for application in real time. We plan to validate these methods by experimenting with several UAVs.

Another possible future study would involve new cost functions that consider other objectives of the UAV such as minimum fuel consumption, minimum arrival time, maximum clearance, etc.

VII. Acknowledgement

This work was supported by the European Commission FP7 ICT Programme under project CONET NoE (FP7-INFSO-ICT-224053), project EC-SAFEMOBIL 288082 and project "Geometry optimization applied to problems of classification, communications and aerial robotics" (FEDER-MTC2009-08625). Díaz-Báñez was partially supported by ESF EUROCORES programme EuroGIGA-ComPoSe IP04-MICINN Project EUI-EURC-2011-4306. Pérez-Lantero was partially supported by grant FONDECYT 11110069.

References

[1] "Sesar consortium website:." http://www.eurocontrol.int/sesar/public/subsite_homepage/homepage.html.

[2] I. Maza, F. Caballero, J. Capitan, J. Martinez-de Dios, and A. Ollero, "A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities," *Journal of Field Robotics*, vol. 28, no. 3, pp. 303–328, 2011.

[3] J. A. Cobano, J. R. Martínez-de Dios, R. Conde, J. M. Sánchez-Matamoros, and A. Ollero, "Data retrieving from heterogeneous wireless sensor network nodes using uavs," *Journal of Intelligent and Robotic Systems*, vol. 60, no. 1, pp. 133–151, 2010.

[4] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 179–189, 2000.

[5] J. Hu, M. Prandini, A. Nilim, and S. Sastry, "Optimal coordinated maneuvers for threedimensional aircraft conflict resolution," *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, p. 2002, 2002.

[6] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, pp. 3 –11, mar 2002.

[7] R. Ehrmanntraut, "The potential of speed control," in *23rd Digital Avionics Systems Conference*, vol. 1, pp. 3.E.3–1–7, oct. 2004.

[8] H. Erzberger, "Automated conflict resolution for air traffic control," in *Proceeding International Congress Aeronautical Sciences*, pp. 179–189, 2006.

[9] C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo, "A novel 3d geometric algorithm for aircraft autonomous collision avoidance," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 1580 –1585, dec. 2006.

[10] E.Crück and J. Lygeros, "Subliminal air traffic control: Human friendly control of a multi-agent system," in *American Control Conference*, pp. 462–467, July 2007.

[11] A. Vela, S. Solak, W. Singhose, and J.-P. Clarke, "A mixed integer program for flight-level assignment and speed control for conflict resolution," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 5219 –5226, dec. 2009.

[12] M. A. Christodoulou and S. G. Kodaxakis, "Automatic commercial aircraft-collision avoidance in free flight: the three-dimensional problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 2, pp. 242–249, 2006.

[13] R. Vivona, D. Karr, and D.Roscoe, "Pattern-based genetic algorithm for airborne conflict resolution," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, (Keystone, Colorado), August 2006.

[14] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution for air traffic management systems," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, pp. 221 –231, dec 2000.

[15] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution manoeuvres," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 2, pp. 110 –120, jun 2001.

[16] A. Lecchini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte carlo optimization for conflict resolution in air traffic control," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, pp. 470 –482, dec. 2006.

[17] R. A. Paielli and H. Erzberger, "Conflict probability estimation for free flight," *AIAA JOURNAL OF GUIDANCE CONTROL AND DYNAMICS*, vol. 20, pp. 588–596, 1997.

[18] J. Hu, M. Pr, and S. Sastry, "Aircraft conflict prediction in the presence of a spatially correlated wind field," *IEEE Trans. on Intelligent Transportation Systems*, vol. 6, pp. 326–340, 2005.

[19] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, pp. 199 –220, dec 2000.

[20] R. Irvine, "A geometrical approach to conflict probability estimation," in *Air Traffic Control Quarterly*, vol. 10, pp. 85–113, 2002.

[21] G.Bakker and H. Blom, "Conflict probability and incrossing probability in air traffic management," in *In IEEE Conference on Decision and Control, Las Vegas*, pp. 2421–2426, 2002.

[22] J. J. Rebollo, A. Ollero, and I. Maza, "Collision avoidance among multiple aerial robots and other non-cooperative aircraft based on velocity planning," in *7th Conference on Mobile Robots*, (Albufeira, Portugal), 2007.

[23] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple uavs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Monterey, CA), August 2002.

[24] S. Waslander, G. Inalhan, and C. Tomlin, "Decentralized optimization via nash bargaining," *Theory and Algorithms for Cooperative Systems*, vol. 4, pp. 565–585, 2004.

[25] K. Kant and S. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5(3), 1986.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[27] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, "Sequencing and scheduling: algorithms and complexity," in *Logistics of Production and Inventory* (S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, eds.), vol. 4 of *Handbooks in Operations Research and Management Science*, ch. 9, pp. 445–522, Elsevier, 1993.

[28] J. Sgall, "On-line scheduling - a survey," 1997.

[29] M. F. P. B. Aspvall and R. E. Tarjan., "A linear-time algorithm for testing the truth of certain quantified boolean formulas,"

[30] T. C. Project, *CGAL User and Reference Manual*. CGAL Editorial Board, 3.9 ed., 2011. http //www.cgal.org/Manual/3.9/doc_html/cgal_manual/packages.html.

[31] T. W. McLain, R. W. Beard, and A. Beard, "Coordination variables, coordination functions, and cooperative timing missions," 2003.