Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios

Eva Besada-Portas, Luis de la Torre, Jesús M. de la Cruz, *Member, IEEE*, and Bonifacio de Andrés-Toro

***Abstract*—This paper presents a path planner for multiple unmanned aerial vehicles (UAVs) based on evolutionary algorithms (EAs) for realistic scenarios. The paths returned by the algorithm fulfill and optimize multiple criteria that 1) are calculated based on the properties of real UAVs, terrains, radars, and missiles and 2) are structured in different levels of priority according to the selected mission. The paths of all the UAVs are obtained with the multiple coordinated agents coevolution EA (MCACEA), which is a general framework that uses an EA per agent (i.e., UAV) that share their optimal solutions to coordinate the evolutions of the EAs populations using cooperation objectives. This planner works offline and online by means of recalculating parts of the original path to avoid unexpected risks while the UAV is flying. Its search space and computation time have been reduced using some special operators in the EAs. The successful results of the paths obtained in multiple scenarios, which are statistically analyzed in the paper, and tested against a simulator that incorporates complex models of the UAVs, radars, and missiles, make us believe that this planner could be used for real-flight missions.**

***Index Terms*—Aerial robotics, multiobjective evolutionary algorithms (EAs), path planning for multiple mobile robot systems.**

I. Introduction

UNMANNED aerial vehicles (UAVs) are aircrafts without onboard pilots that can be remotely controlled or can fly autonomously based on preprogrammed flight plans [1]. The autonomy level achieved by the UAVs depends on the methodology used to control the vehicle and to generate its routes. Although both tasks are equally important (and become especially critical when the UAVs must operate in urban terrains [2]), this paper focuses only on the second: the development of a planner capable of solving the route-generation problem by obtaining a near-optimal path in every possible situation that UAVs can face [3].

The advances in microcontroller design, optimization techniques, and control theory increment the number of fields, both civil and military, where UAVs are currently being used: surveillance, reconnaissance, geophysical survey, environmental and meteorological monitoring, aerial photography, search-and-rescue tasks, etc. Although all these missions seem different, the optimality of a route for any of them can be defined by different optimization planning criteria (such as minimal flying time and/or path length) and fulfillment of some mission constraints (such as flying at a given altitude or visiting some points). Besides, the physical characteristics of the UAVs and the environment also restrict the feasibility of any route and should be considered by a realistic planner.

On military missions, UAVs work in dangerous environments, and therefore, it is vital to always keep their routes apart from any type of threat and restricted zone. Therefore, the best routes are those that minimize the risk of destruction of the UAVs, optimize some other planning criteria, and fulfill all the constraints that are imposed by the proposed mission, the physical characteristics of the UAVs, and the environment. The original routes that are obtained offline by the planner are not always valid in dynamic environments where the position of all the threats is not known beforehand. Therefore, the planner must be able to also work online in order to propose a new path during the UAVs mission when a pop-up (i.e., unknown threat) appears.

Finding the optimal solution to the route-planning problem is nondeterministic-polynomial-time complete (NP-complete) [4], and therefore, the time required to solve it increases very quickly as the size of the problem grows. The problem has been tackled with different heuristics, such as mixed-integer linear programming [5]–[8] and A* [9]–[12], and with nonlinear programming [13]. The planners based on those techniques work with a simplified version of the original problem, where the addition of new constraints or objective criteria is a difficult task. They consider only point-mass dynamics, discretize the solution space, and, in some cases [5]–[8], linearize the models. Although the majority solve the single-UAV case, Richards and How [7] and Raghunathan *et al.* [13] tackle the multi-UAV one.

Evolutionary algorithms (EAs) are versatile optimizers that have already been used to solve different UAV path-planning problems: Mittal and Deb [14], Nikolos *et al.* [15], [18], Hasircioglu *et al.* [16], Pehlivanoglu *et al.* [17], and Nikolos

Manuscript received December 1, 2009; revised March 22, 2010; accepted April 13, 2010. Date of publication May 24, 2010; date of current version August 10, 2010. This paper was recommended for publication by Associate Editor G. Antonelli and Editor J.-P. Laumond upon evaluation of the reviewers' comments. This work was supported by the Community of Madrid under Project "COSICOLOGI" S-0505/DPI-0391, by the Spanish Ministry of Education and Science under Project DPI2006-15661-C02-01 and Project DPI2009-14552-C02-01, and by the European Aeronautic Defense and Space Company (Construcciones Aeronauticas Sociedad Anonima) under Project 353/2005. The work of E. Besada-Portas was supported by the Spanish Postdoctoral Grant EX-2007-0915 associated with the Prince of Asturias Endowed Chair of the University of New Mexico, University of New Mexico, Albuquerque, NM. This paper was presented in part at the Genetic Evolutionary Computation Conference, Atlanta, GA, 2008, and in part at the 8th International FLINS Conference, Madrid, Spain, 2008.

E. Besada-Portas, J. M. de la Cruz, and B. de Andrés-Toro are with the Departamento de Computadores y Automática, Universidad Complutense de Madrid, Ciudad Universitaria, Madrid 28040, Spain (e-mail: evabes@dacya.ucm.es; jmcruz@dacya.ucm.es; deandres@dacya.ucm.es).

L. de la Torre is with the Department of Computer Sciences and Automatic Control, Spanish University for Distance Education, Madrid 28080, Spain (e-mail: ldelatorre@bec.uned.es).

Digital Object Identifier 10.1109/TRO.2010.2048610

and Tsourvelouds [19] optimized the paths of UAVs flying over a terrain, and Zheng *et al.* [20] and Zhang *et al.* [21] searched for optimal UAV paths in military missions. All formulated the problem as finding the trajectory that minimizes and fulfills a set of optimization indexes and constraints, and while Mittal and Deb [14], Nikolos *et al.* [15], Hasircioglu *et al.* [16], Pehlivanoglu *et al.* [17], and Zhang *et al.* [21] solved the single-UAV problem, Nikolos *et al.* [18], Nikolos and Tsourvelouds [19], and Zheng *et al.* [20] solved the multi-UAV one.

This paper presents our evolutionary planner to solve a multi-UAV route-planning and cooperation problem. It extends [14]–[21] and our previous works, which are presented in [22] and [23], as follows.

1) *Using complex models and new optimization criteria and constraints for the military problem*. To minimize the time response of the planner, an extremely simple model of the UAVs, terrain, and threats is used in [14]–[21], and the performance of the planner is tested against the same model instead of a realistic complex simulator. However, in [22], [23], and in this paper, the routes are evaluated according to the properties of a realistic aircraft for the UAVs, real maps of the world for the terrain, and the characteristics of the radars and missiles of the complex simulator used to test the results of our planner. Besides the UAVs position that is considered in [14]–[21], we also use their velocity to calculate the minimum turning radius, the fuel consumption, and the risk of the trajectories. Not only do we check that the UAVs do not collide against the terrain, as in [14]–[21], but we also use maps of any part of the world to represent it, as in [17] and [20], and to inhibit the radar detection's capability when UAVs fly hidden behind mountains. We calculate the probability to detect and/or to destroy an UAV based on the range of detection, the tracking probability, and the fire range of the three types of radars and missiles considered in our simulator. Our planner also supports the definition of prohibited zones (i.e., no flying zones, NFZs), which UAVs have to avoid due to mission restrictions. Finally, this is the first of our papers that introduces the mathematical models used in all our work.
2) *Implementing a new codification for the trajectories*. Although 3-D trajectories can be defined pointwise as the linear segments determined by a list of 3-D absolute Cartesian points (x, y, z), as in [20] and [21], smoother curves, which are easier to follow by the UAVs, can be obtained with other representations [24]. Therefore, the list of 3-D points determines the B-spline curve followed by each UAV in [14]–[16], [18], and [19], a Bezier curve in [17], and a cubic spline (which lets us include in the trajectory all the fixed points that each UAV has to visit, and not only one, as in [14]) in our study, i.e., in [22] and [23]. Besides, the EAs in [14]–[17] codify the 3-D points with absolute Cartesian coordinates, while the EAs in [18], [19], [22], and [23] use relative polar coordinates for (x, y). Using relative polar coordinates significantly reduces the search space; however, the mutation and crossover steps produce global changes in the trajectory instead of the local ones of the absolute Cartesian codification. Our new codification takes advantage of all the approaches: We use a list of 3-D absolute Cartesian points, whose (x, y) values are generated using the relative polar coordinates, to define cubic-spline trajectories.
3) *Using a general multiobjective evaluation method based on goals and priorities*. The evaluation method that combines the different objectives and constraints used in [14]–[21] does not provide an easy support to modify the priorities of the objectives and their goals for different missions, and only in [14] is the concept of Pareto optimality [25] used. The planners of [22] and [23], and this paper use the generic multiobjective Pareto-evaluation function with goals and priorities presented in [26], which easily permits to change the levels of priority of the different objectives as well as to include new objectives and constraints. This way, our planners can be used for different types of missions very easily.
4) *Developing a general evolutionary framework to optimize the behavior of multiple cooperating agents*. The multiple path-generation problem is solved using, for each UAV, an EA that optimizes its path based on the objectives and constraints associated with that UAV and some cooperation objectives related with the others. As the evaluation of the cooperation objectives requires some knowledge of the optimal solutions of all the UAVs, all the EAs need to run simultaneously and send some information of the optimal solution they have obtained so far to the remaining EAs, i.e., coordination between the solutions (which, in the particular case of this work, means coordination between UAVs) is obtained by introducing coordination objectives and sharing information between the EAs. The result of this approach is a multiple coordinated agents coevolution EA (MCACEA), which is a general evolutionary framework to optimize some characteristics (i.e., the path in our problem) of multiple cooperating agents. MCACEA can obtain the path of a single UAV (as in [14]–[17], and [21]) or the paths of several cooperating UAVs that have to be flying simultaneously without colliding (as in [18]–[20]). Although the core idea of MCACEA is shortly described in [22], this paper presents it in detail and includes new and more challenging problems that illustrate its performance better.
5) *Speeding up the computation by combining some problem-specific features with standard efficient EA operators, whose parameters are tuned after a statistical performance analysis*. Following the ideas in [16], [17], and [20], our EAs also include some special features, such as the option of forcing the UAVs to fly at constant altitude and the initialization of the algorithm with previous solutions. The substitution method based on hypercubes [27], which is used in [22] and [23], is replaced by the more efficient one that is included in the second version of the Nondominated Sorting Genetic Algorithm (NSGA-II) [28]. Besides, some of the EA parameters are selected using the statistical performance analysis based on dominance ranking that is presented in [29]. The lack of a significative

statistical difference between the results of the optimizations carried out with and without the local-search operator that is used periodically over the best solutions in our previous work has made us eliminate it as well. Therefore, the improved EAs that constitute our current multi-UAV planner are computationally quicker and more standard.

Finally, we want to highlight that our EA planner works offline and online. The offline paths are obtained by the multiple EAs working inside the multiple-agent framework before the mission starts based on the original information that the system has about the environment. During the simulations, if an unexpected threat suddenly pops up, the EAs of the affected UAVs start working online to find a new path that avoids the threat.

The rest of this paper is organized as follows. Section II describes the problem by means of the mathematical models used to define the missions. Sections III–VI, which are, respectively, related with points 2)–5) of this section, explain the different parts of our EA planner. Section VII presents the results of our planner in different scenarios, and Section VIII contains the final discussion and conclusions. Finally, the Appendix contains the most-used acronyms.

II. Problem Description

The military missions of the optimization problem that are solved by our EA planner are defined by a set of optimization planning criteria and constraints, which include the minimization of the risk of destruction of the UAVs as well as the restrictions imposed by the UAVs' dynamics and the environment.

Currently, our problem considers 11 objectives. The first ten measure the fitness of the trajectory of each UAV independently. Six are constraints, while the other four are values that should be minimized in order to optimize the solution. The last, i.e., the eleventh one, is a cooperation constraint that checks the paths feasibility when all the UAVs are flying simultaneously.

Their values are calculated over the UAV trajectories, which are cubic-spline curves obtained from the list of 3-D Cartesian waypoints used to codify the solutions of each EA; for more details, see Section III. To be able to evaluate their values in the computer, the continuous-spline curves are discretized in N points, whose values in the absolute Cartesian-coordinate system are (x_i, y_i, z_i). The evaluation process uses the discretized curves, and the properties of the elements that appear in the optimization scenarios: the terrain, prohibited flying zones, radars, missiles, and UAVs. The characteristics of our radars and missiles have been estimated after many simulations over our final test bench. The properties of our UAVs are obtained with a computationally efficient simplified version of their complete nonlinear dynamic model. Therefore, they are only valid in standard flight conditions, such as UAV velocity close to 250 m/s.

Their 11 mathematical models are presented next, with the variables in the International System of Measurements.

1) *Minimum turning radius:* The UAV maneuverability is constrained by its minimum turning radius $R_i^{\min}$, which depends on the maximum load factor $n_i^{\max}$, altitude z_i, and velocity v. For each point of the trajectory, $R_i^{\min}$ and $n_i^{\max}$ are obtained as follows:

$$R_i^{\min} = \frac{v^2}{g\sqrt{(n_i^{\max})^2 - 1}} \tag{1}$$

$$n_i^{\max} = 5.3809 \times 10^{-9} z_i^2 - 4.4291 \times 10^{-4} z_i + 6.1000 \tag{2}$$

where g is the gravity. All the trajectory points, whose turning radius R_i is smaller than the minimum permitted one, are simultaneously penalized by the use of (3), whose minimum value (i.e., zero) ensures the fulfillment of the constraint. R_i is calculated as the radius of the circumference that is defined by the points $i-1$, i, and $i+1$

$$\sum_{i=1}^{N} c_i^1 \text{ with } c_i^1 = \begin{cases} 1, & R_i \le R_i^{\min} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

2) *Limited UAV slope:* The UAV maneuverability is also constrained by its maximum climbing slope α_i and its minimum gliding slope β_i, which, in our case, depend on its altitude z_i. These slope limits are calculated with

$$\alpha_i = -1.5377 \times 10^{-10} z_i^2 - 2.6997 \times 10^{-5} z_i + 0.4211 \tag{4}$$

$$\beta_i = 2.5063 \times 10^{-9} z_i^2 - 6.3014 \times 10^{-6} z_i - 0.3257. \tag{5}$$

The UAV slope at the ith point (i.e., S_i) is obtained with

$$S_i = \frac{z_{i+1} - z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}. \tag{6}$$

Similarly to the previous constraint, all the trajectory points, whose slope is out of the permitted range, are penalized using the following expression, whose minimum value (i.e., zero) is the optimal for this constraint:

$$\sum_{i=1}^{N} c_i^2 \text{ with } c_i^2 = \begin{cases} 0, & \beta_i < S_i < \alpha_i \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

3) *Fuel:* UAVs carry a limited quantity of fuel and they have to reach their destination before consuming all of it.

The fuel consumption for each trajectory point is calculated based on the expressions of the following three basic cases.

a) The fuel consumption when the UAV is flying horizontally and straight (i.e., F_i^{CH}), which is dependent on z_i

$$F_i^{\mathrm{CH}} = 9.553 \times 10^{-8} z_i^2 - 2.4524 \times 10^{-3} z_i + 29.5. \tag{8}$$

b) The fuel consumption when the UAV is flying at maximum slope (i.e., F_i^{MS}), which is also dependent on z_i

$$F_i^{\mathrm{MS}} = 1.6679 \times 10^{-11} z_i^3 - 2.4832 \times 10^{-7} z_i^2 - 4.259 \times 10^{-3} z_i + 87.881. \tag{9}$$

c) The fuel consumption when the UAV is flying with maximum bank angle (i.e., F_i^{MT}), which is dependent on $n_i^{\max}$

$$F_i^{\mathrm{MT}} = -3.0435 \times 10^{-1} \left(n_i^{\max}\right)^2 + 16.552 \times n_i^{\max} + 0.3565. \quad (10)$$

The real cases are considered as a combination of the previous cases, where only the main contributions that appear in each type of maneuver are taken into account.

a) When the UAV is ascending, the real consumption (i.e., FC_i^A) depends basically on the fuel consumption at constant height (i.e., F_i^{CH}) and at maximum slope (i.e., F_i^{MS}), as well as on the discrepancy between the actual and maximum slopes (i.e., S_i and α_i)

$$\mathrm{FC}_i^A = F_i^{\mathrm{CH}} + \frac{S_i}{\alpha_i}\left(F_i^{\mathrm{MS}} - F_i^{\mathrm{CH}}\right). \quad (11)$$

b) When the UAV is turning but not ascending, the fuel consumption (i.e., FC_i^T) depends on the fuel consumption at constant height (i.e., F_i^{CH}) and at maximum turning angle (i.e., F_i^{MT}), as well as on the ratio between the actual and maximum load factors (i.e., n_i and $n_i^{\max}$)

$$\mathrm{FC}_i^T = F_i^{\mathrm{CH}} + \frac{n_i}{n_i^{\max}}\left(F_i^{\mathrm{MT}} - F_i^{\mathrm{CH}}\right) \quad (12)$$

where n_i can be obtained from (1), using R_i and n_i instead of $R_i^{\min}$ and $n_i^{\max}$.

c) Finally, when the UAV flies straight and descending, the fuel consumption (i.e., $\mathrm{FC}_i^{\mathrm{CH}}$) is approximated by the one for horizontal and straight-flight conditions F_i^{CH}. This assumption is conservative since in this situation, the consumption is lower.

Considering that between two points of the trajectory, the fuel consumption is constant, the total fuel consumption is calculated as the summation of the product of the fuel needed at point *i* (i.e., $\mathrm{FC}_i^{\mathrm{case}(i)}$) and the time needed to go from i to $i+1$ (Δt_i), which is estimated as follows:

$$\Delta t_i = \frac{\sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2 + (z_{i+1}-z_i)^2}}{v}. \quad (13)$$

The total fuel consumption value is limited by the fuel capacity of the UAV (i.e., Fuel). The following expression only penalizes the constraint when the consumption exceeds the capacity. Its minimum value (i.e., zero) is obtained when the fuel consumption is less than the fuel capacity. A positive value implies that the consumption is higher than the capacity. Therefore, the following expression calculates the extra fuel that is needed to follow the trajectory:

$$\max\left(\sum_{i=1}^{N} \mathrm{FC}_i^{\mathrm{case}(i)} \Delta t_i - \mathrm{Fuel}, 0\right)$$

$$\text{with case}(i) = \begin{cases} A, & \text{when UAV ascends} \\ T, & \text{when UAV turns but does not ascend} \\ \mathrm{CH}, & \text{otherwise.} \end{cases} \quad (14)$$

4) *Terrain:* A feasible path cannot go through the terrain and has to avoid collisions with mountains. To ensure this behavior, the algorithm penalizes the solutions that have at least one point of the spline trajectory inside the terrain. If $\mathrm{map}(x_i, y_i)$ is the function that returns the altitude of the terrain at any point (x_i, y_i), the following expression, which calculates the number of points that are inside the terrain, is used to penalize this constraint:

$$\sum_{i=1}^{N} c_i^4 \text{ with } c_i^4 = \begin{cases} 1, & z_i \le \mathrm{map}(x_i, y_i) \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

5) *Map limits:* To penalize the points of the trajectory that are not inside the map limits, (16), shown below, is used, where x_l^m and x_u^m are the map lower and upper limits for the "x" coordinate, and y_l^m and y_u^m are the equivalent for the "y" coordinate. Its minimum value (i.e., zero) ensures the fulfillment of the following map constraint:

$$\sum_{i=1}^{N} c_i^5$$

$$\text{with } c_i^5 = \begin{cases} 0, & \mathrm{InMap}(x_i, y_i) \\ 1, & \text{otherwise} \end{cases}$$

$$\mathrm{InMap}(x_i, y_i) = \left(x_l^m \le x_i \le x_u^m\right) \wedge \left(y_l^m \le y_i \le y_u^m\right). \quad (16)$$

6) *Flight-prohibited zones (i.e., NFZs):* The user can define certain zones where UAVs must not enter because they are considered high-risk zones, unknown zones, etc.

They are defined as M rectangular regions constrained by their external limits ($x_l^{\mathrm{NFZ},j}$, $x_u^{\mathrm{NFZ},j}$, $y_l^{\mathrm{NFZ},j}$ and $y_u^{\mathrm{NFZ},j}$, with $j = 1{:}M$). The N points of the trajectory have to avoid them. To penalize this objective, for each of the trajectory points that are inside a NFZ, we accumulate the distance to the closest NFZ edge, with the purpose of distinguishing between two solutions that have the same number of points inside the NFZs, but at different distances of their frontiers. The following expression imposes the selected penalization and the constraint is fulfilled with its minimal value (i.e., zero):

$$\sum_{i=1}^{N}\sum_{j=1}^{M} d_i^j$$

$$\text{with } d_i^j = \begin{cases} \min\limits_{k,a}(d_i^{j,k,a}), & \text{if InNFZ}\,(i,j) \\ 0, & \text{otherwise} \end{cases}$$

$$d_i^{j,k,a} = \left|a_k^{\mathrm{NFZ},j} < a_i\right|, \quad \text{with } k = l, u \wedge a = x, y$$

$$\mathrm{InNFZ}(i,j) = \left(x_l^{\mathrm{NFZ},j} \le x_i \le x_u^{\mathrm{NFZ},j}\right) \wedge \left(y_l^{\mathrm{NFZ},j} \le y_i \le y_u^{\mathrm{NFZ},j}\right). \quad (17)$$

7) *Minimum path length:* For military applications, shorter paths are better than longer ones (if all the other objectives are equal) because they require less time of flight and

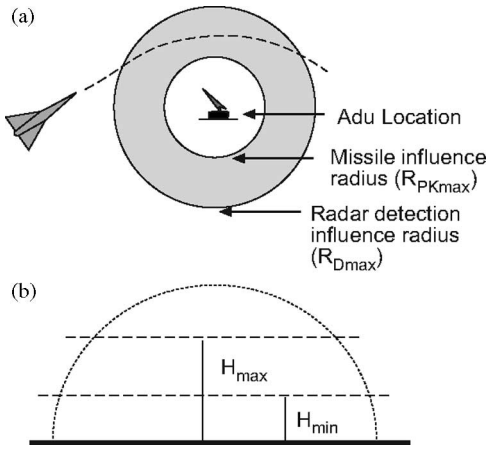

Fig. 1. ADU model. (a) Eagle eye view. (b) Cross-section view.

usually have a lower chance of finding an unknown threat in a dynamic environment.

Instead of minimizing the actual path length, we minimize its value normalized by the minimum flight distance $l_{\min}$, because both values are equivalent and the values of the second represent ratios that are considered admissible. The path length ratio (PLR) is calculated with the following expression:

$$\text{PLR} = \frac{\sum_{i=1}^{N-1} \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2 + (z_{i+1}-z_i)^2}}{l_{\min}} \tag{18}$$

where $l_{\min}$ is calculated as the distance between the points selected by the user to define the mission: the start, intermediate, and ending points of each UAV.

8) *Minimum probability of kill:* Trajectories that accumulate lower probability of destruction are safer than those with higher accumulated probability values.

The probability-of-kill (PKill) function depends on the model used for the Air Defense Units (ADUs) that are groups of radars and missiles. For each ith point of the trajectory, the jth ADU only has a certain possibility to destroy the UAV if the UAV is inside the region defined by the ADU's maximum-risk distance (i.e., R^j_{PKmax}) and minimum- and maximum-risk altitudes (i.e., $H^j_{\min}$ and $H^j_{\max}$). The shape of that region is presented in Fig. 1, which also shows the maximum detection distance (R^j_{Dmax}) that will be defined in the ninth objective. The three types of ADUs used in our scenarios have different R^j_{PKmax}, $H^j_{\min}$, and $H^j_{\max}$. A tabular representation of the probability of destruction inside that region, which is dependent on the orientation (θ^j_i) and altitude (H^j_i) of the UAV relative to the ADU, is obtained after many simulations against our test bench. Interpolating their values with the function $\text{TabPK}(\theta^j_i, H^j_i)$, we obtain the PKill of the ith point of the trajectory for the jth ADU. The PKill accumulated along the trajectory due to the different A active ADUs of the problem is calculated with the following expression, whose value has to be minimized:

$$\text{PKill} = 1 - \prod_{i=1}^{N}\prod_{j=1}^{A}(1 - \text{PK}^j_i \Delta t_i)$$
$$\text{with } \text{PK}^j_i = \begin{cases} \text{TabPK}(\theta^j_i, H^j_i), & \text{if InPK}(i,j) \\ 0, & \text{otherwise} \end{cases}$$
$$\text{InPK}(i,j) = (R_{ij} \le R^j_{\text{PKmax}}) \wedge (H^j_{\min} \le H^j_i \le H^j_{\max}). \tag{19}$$

9) *Minimum probability of radar detection:* When the radars do not detect the UAVs, not only the ADUs cannot destroy them (and, therefore, trajectories that accumulate lower probabilities of detection are safer), but also, the flight of the UAV and its mission are kept secret.

The probability of radar detection (PRD) depends on the model used for the ADU's radars. For each ith point of the trajectory and jth radar, the detection-probability model considers the radar cross section RCS_{ij} of the UAV, the distance R_{ij} between the UAV and the radar, the existence of a line of sight $\text{LoS}(i,j)$ between the UAV and the radar, and some properties of the radar.

The RCS_{ij} depends on the orientation of the UAV with respect to the ADU and can be calculated by the following expression [30] that considers the UAV as an ellipsoid with semiaxis "*a*," "*b*," and "*c*":

$$\text{RCS}_{ij} = \frac{\pi a^2 b^2 c^2}{\sqrt{(a\alpha_z\beta_\phi)^2 + (b\alpha_z\alpha_\phi)^2 + (c\beta_z)^2}}$$
$$\text{with } \alpha_z = \sin(\text{az}^e_{ij}) \quad \text{and} \quad \beta_z = \cos(\text{az}^e_{ij})$$
$$\alpha_\phi = \sin(\phi^e_{ij}) \quad \text{and} \quad \beta_\phi = \cos(\phi^e_{ij}) \tag{20}$$

where az^e_{ij} is the angle between the velocity of the UAV and the segment that joins the UAV and radar positions, and ϕ^e_{ij} is given by

$$\phi^e_{ij} = \phi_{ij} - \arctan\left(\frac{\tan(\text{el}_{ij})}{\sin(\text{az}_{ij})}\right) \tag{21}$$

with ϕ_{ij} the roll, el_{ij} the elevation, and az_{ij} the azimuth between the UAV at position *i* and the *j*th ADU.

The LoS between two points (i,j) of the space is calculated with the function $\text{LoS}(i,j)$ that considers the terrain elevation at the line that joins the two considered points. Although this function significantly increments the computation time of the algorithm, it lets it find solutions where the terrain inhibits the radar effects.

The PRD accumulated along the trajectory due to the radars of the different A active ADUs is calculated with (22), whose value should be minimized. R^j_{Dmax} is the maximum detection distance of the *j*th ADU, and ζ^j_1 and

ζ_2^j are parameters that are dependent on the model of its radar. All the ADUs of the scenarios have the same $R^j_{\mathrm{D}\max}$ (i.e., 50 km)

$$\mathrm{PRD} = 1 - \prod_{i=1}^{N}\prod_{j=1}^{A}\left(1 - \mathrm{DP}_i^j \Delta t_i\right)$$

$$\text{with } \mathrm{DP}_i^j = \begin{cases} 0, & \text{if } R_{ij} > R^j_{\mathrm{D}\max} \vee \neg \mathrm{LoS}(i,j) \\ \dfrac{1}{1+\zeta_2^j\left(R_{ij}^4/\mathrm{RCS}_{ij}\right)^{\zeta_1^j}}, & \text{otherwise.} \end{cases} \tag{22}$$

10) *Minimum flight altitude:* UAVs flying at low altitude can benefit from the terrain-mask effect that will help them to avoid unknown radars. Although this increases the fuel consumption (which is optimal at cruise altitude), the combination of objectives 3) and 10) makes the planner search low-altitude trajectories whose fuel needs are within the maximum permitted.

To achieve this objective, we calculate the accumulated difference between z_i and the altitude of the terrain at that point (i.e., map(x_i, y_i)) and divide it by the number of points in the trajectory, i.e., N. Therefore, the algorithm minimizes the elevation of the UAV over the terrain and not the elevation of the UAV over the sea level. The following expression calculates this objective value:

$$\frac{\sum_{i=1}^{N} c_i^{10}}{N}$$

$$\text{with } c_i^{10} = \begin{cases} 0, & \text{if } z_i < \mathrm{map}\left(x_i, y_i\right) \\ z_i - \mathrm{map}(x_i, y_i), & \text{otherwise.} \end{cases} \tag{23}$$

11) *Avoiding UAVs collisions:* When generating paths for several UAVs, it is important to check whether two UAVs are getting too close while following their respective paths. For that, the planner has to test if two paths coincide in space and time.

With this purpose, given the trajectory of the u UAV and the v trajectory for the remaining UAVs, every spline-curve point of the first trajectory is compared with every point of the others. If the distance d_{ij}^{uv} between the ith point of the trajectory of the uth UAV and the jth point of the trajectory of the vth UAV is smaller than a minimum critical value $d^{\min}$, it is checked if the arrival times to those points (i.e., t_i^u and t_j^v) are too close to constitute a risk, which happens if their differences are lower than the minimum permitted time $t^{\min}$. The parameters $d^{\min}$ and $t^{\min}$, which are chosen by the user, specify the requested safety level. Each dangerous pair of points is penalized by (24), whose minimum value ensures that the trajectory of the u UAV does not collide against any of the other v UAVs trajectories. The indices in N^k, t_l^k, x_l^k, y_l^k, and z_l^k are related with the the kth UAV and l trajectory point

$$\sum_{\forall v \neq u}\sum_{i=1}^{N^u}\sum_{j=1}^{N^v} c_{ij}^{11}$$

$$\text{with } c_{ij}^{11} = \begin{cases} 1, & \text{if } d_{ij}^{uv} < d_{\min} \wedge |t_i^u - t_j^v| < t_{\min} \\ 0, & \text{otherwise} \end{cases}$$

$$d_{ij}^{uv} = \sqrt{(x_i^u - x_j^v)^2 + (y_i^u - y_j^v)^2 + (z_i^u - z_j^v)^2}. \tag{24}$$

III. Solution Codification For The Evolutionary Algorithm Planner

EAs work with populations of possible solutions that, in this problem, codify the possible 3-D trajectories of each UAV.

Our codification is a list of 3-D points that are used to define the cubic-spline curve [24] that each UAV has to follow. This list, whose points are floating-point coded, contains some fixed points (i.e., start, end, and intermediate) that the UAVs are forced to visit and some undetermined points (i.e., waypoints) that the EA planner calculates to find the feasible and optimal cubic-spline trajectory for each UAV. The cubic-spline curve, which is only used in our works, automatically ensures that the trajectory visits all the points in the list.

The offline planner sets the number of waypoints that exists between each pair of fixed points, which is based on their distance, and calculates their values. When the UAVs are flying, the online planner adds some new waypoints to those lists whose paths are affected by the pop-ups and recalculates the values of all the waypoints placed after the UAVs current positions.

Each waypoint in the 3-D space is defined by three coordinates, whose values can have different meaning depending on the selected codification. For example, in [14]–[17], [20], and [21], these coordinates are the absolute Cartesian "x," "y," and "z," while in [18] and [19], and ours, i.e., [22] and [23], they belong to a relative polar-coordinate system, with absolute "z" (i.e., the altitude) and polar coordinates "r" (i.e., radius) and "θ" (i.e., angle) relative to the previous point in the list. Fig. 2 shows both codifications for the same cubic-spline curve.

The absolute Cartesian codification generates a huge search space in real scenarios since any point of the space can be reached from the previous point of the list. The relative polar-coordinate system dramatically reduces the search space, because it forces the EA to have a predefined order of points. However, any change of the values of the waypoints in the relative polar codification modifies the whole shape of the part of the spline curve that comes after it. In the absolute Cartesian codification, the changes only modify the spline curve locally.

Our current codification takes advantage of both approaches. We maintain a cubic-spline representation to force the visit of all the points of the list while the waypoints are codified in absolute Cartesian coordinates to ensure only local changes in the crossover and mutation steps. Also, to dramatically reduce the search space, the initial values of the absolute Cartesian coordinates are obtained from a random process that creates the radios, angles, and altitudes of the waypoints in relative polar representation, i.e., the waypoints of a trajectory are created

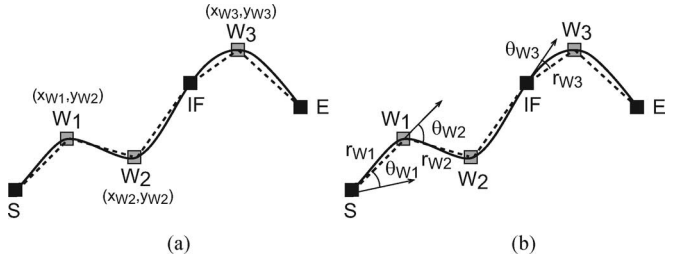

Fig. 2. Absolute Cartesian and relative polar codifications for the same (solid line) cubic-spline curve defined by the fixed points S, IF, and E and the waypoints W_1, W_2, and W_3. Although only the "x" and "y" values are presented in the curves, the codification also includes the absolute "z" value. (a) (Absolute Cartesian coordinates) Each waypoint is represented with its $(x_{W_i}, y_{W_i}, z_{W_i})$ values. (b) (Relative polar coordinates) Each waypoint is represented with its $(r_{W_i}, \theta_{W_i}, z_{W_i})$ values. r_{W_i} is the distance from the previous list point to the ith waypoint. θ_{W_i} is the angle change with respect to the angle defined by the segment that joins the two previous list points (i.e., dashed lines and vectors). The r_{W_i} and θ_{W_i} values to any fixed point are fixed by that point and the previous waypoint.

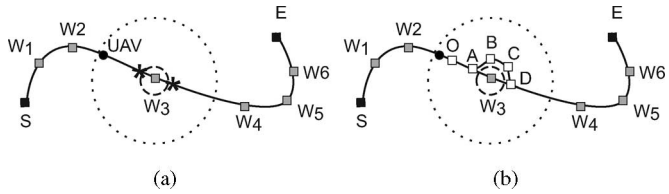

Fig. 3. Waypoint list online initialization process. (a) (Final optimal offline waypoints) ($W_{\#}$), and start (S) and end (E) points that define the optimal trajectory obtained by the offline planner, the areas defined by the (dotted circle) maximum-radar-detection distance and by the (dashed circle) maximum-risk distance (i.e., R^j_{PKmax}) of a pop-up ADU, (black filled circle) the UAV position when the online planner starts working, and the (asterisks) intersection of the discretized spline trajectory with the region defined by the maximum-risk distance (i.e., R^j_{PKmax}). (b) (Initial online waypoints) Original fixed point (O) of the online trajectory and a U-shaped four waypoint sublist (A, B, C, and D) created randomly by the online planner to avoid the pop-up R^j_{PKmax} area. A new online 3-D list is defined by (O, A, B, C, D, W_4, W_5, W_6, and E).

from scratch by the offline planner within the polar relative representation and translated into its absolute Cartesian representation, the absolute Cartesian waypoints are manipulated by the EA operators, and the discretized spline, which is defined by the fixed points and waypoints, is only used to determine the goodness of each trajectory. The initialization in the polar relative representation is carried out using uniform random distributions for the floating-point-coded (r, θ, z) values. The "z" limits constitute the minimal and maximal altitude that the UAV can reach, the "r" limits represent the original permitted separation between two consecutive waypoints, and the "θ" limits are related with the UAV maneuverability. The absolute Cartesian values that they are translated into are codified as real floating-point-coded numbers. Finally, the objective values are obtained with the expressions given in Section II, using the N (x_i, y_i, z_i) points in which the cubic spline is discretized. $N = p * (N_{\mathrm{IF}} + N_W + 1) + 1$, where N_{IF} is the number of intermediate fixed points, N_W the number of waypoints, and p a user-selected parameter (which is 40 in this paper).

The process of initializing the lists of waypoints of the online planner, which we will explain using the example presented in Fig. 3, is different. When a flying UAV following the offline optimal trajectory identifies a new pop-up, the online planner starts working. First, the planner finds the points of the list (W_3, W_4, W_5, W_6, E) of the offline trajectory situated after the UAV position (black filled circle). It also finds the segment (limited by asterisks) of the UAV-discretized spline trajectory that falls inside the new pop-up maximum-risk-distance region (dashed circle) and identifies the waypoints (i.e., W_3) it bypasses. Then, it creates four waypoint sublists (such as the one defined by A, B, C, and D), whose values, which are generated in the polar relative representation, define, when translated into the absolute Cartesian one, a random U-shape around the pop-up R^j_{PKmax} area. Next, it creates the new lists combining in the appropriate order an origin fixed point (i.e., O) close to the UAV position, the U-shaped waypoint sublists, and the set of offline waypoints located after the UAV current location that is not bypassed by the intersected segments. This way of proceeding gives the online planner a high chance of creating lists (O, A, B, C, D, W_4, W_5, W_6, and E) that usually avoid the pop-ups to start with. Finally, their not-fixed-point values are optimized to ensure that the new 3-D lists fulfill all the optimization criteria (for instance, avoiding that the UAV collides against others).

IV. Evaluation Function

Our problem has the 11 objectives presented in Section II. Therefore, the planner needs a function that combines their values to obtain the fitness of each solution of the population.

For this purpose, we use the generic-evaluation multiobjective method based on goals, priorities, and Pareto sets proposed in [26], where the objectives are placed in levels of priority and where limits (goals) can be imposed on each of them. With this method, which is already used successfully in other complex multicriteria constrained problems [31], [32], the EA compares all pairs of solutions in the population, decides if one Pareto dominates the other, and ranks them with the technique proposed by Goldberg [33].

To be able to set the priority levels of the 11 objective values, we divide them in two groups: (I) the ones related with each UAV (i.e., single-UAV objectives) and (II) the cooperation objectives (multi-UAV objectives). Furthermore, both groups are divided in two different types: a) constraints that the path of the UAV has to satisfy completely due to its physical restrictions, the terrain, or mission considerations and b) optimization indexes that must be minimized according to the selected mission. The rows *Name* and *Eq.* of Table I show this classification and the equations that calculate them.

Based on this classification, the objective values used to measure when the constraints are satisfied are placed as high-priority objectives whose values must be zero. The objective indexes to be minimized are located at lower priority levels organized according to the goal of the mission. With this organization, solutions that fulfill the constraints are better than the ones that do not, and the latter are basically organized according to how far the constraints have been optimized. The rows of Table I *Level*, *Min,* and *Max*, show a typical election of priority levels and limits for the objectives. The limits establish the accepted

TABLE I
USUAL EVALUATION FUNCTION PRIORITIES AND LIMITS

I. Individual Objectives						
I.a) Constraint Conditions						
Name	Turning Radius	Limited slope	Fuel (kg)	Terrain	Map Limits	NFZs (m)
Eq.	(3)	(7)	(14)	(15)	(16)	(17)
Level	1st	1st	1st	1st	1st	1st
Min	0	0	0	0	0	0
Max	0	0	0	0	0	0
I.b) Optimization Indexes						
Name	PLR	PKill		PRD	Flight Altitude (m)	
Eq.	(18)	(19)		(22)	(23)	
Level	2nd	2nd		3rd	3rd	
Min	1	0		0	50	
Max	1.2	0		0.5	1000	
II. Coordination Objectives						
II.a) Constraint Conditions						
Name	Avoiding UAVs Collision					
Eq.	(24)					
Level	1st					
Min	0					
Max	0					
II.b) Optimization Indexes						

interval of values for each objective and are tuned by the user considering the mission goals.

Furthermore, as we are using a general method to evaluate the fitness of the function given the constraints and objective indexes, we can easily add or remove constraints or optimization indexes as needed. For instance, we could consider other coordinating constraints, such as the timing restrictions proposed in [34] and [35], which are the simultaneous arrivals of two or more aircrafts to a selected point, and tight/loose sequencing (especially important if the mission considers on-air UAV refueling). The autonomous aerial-refueling problem could also be formulated here in terms of the four parameters presented in [36]: maximum waiting time, refueling time, return-to-field priority, and refueling sequence number. Besides, a cooperation-optimization index (none is currently considered in our problem), such as the minimization of the distance between two different UAVs in order to carry out a tracking mission, could easily be included.

Although the missions presented in [14]–[21] are also defined as multiobjective problems, their evaluation functions are significantly less flexible than ours.

Finally, it is important to highlight that the time needed to evaluate the solutions is especially problematic when the online planner starts running due to the presence of a sudden pop-up. In order to speed up the algorithm, the online planner uses a simpler model of the problem and less optimization indexes. This difference is justified because the critical objective of the online planner is to avoid the threat. Therefore, minimizing the PRD or altitude is no longer important. To sum up, both planners use the same evaluation function, the seven constraints, and different objective values. The differences in this respect are the following:

1) The offline planner uses the four optimization indexes that we want to minimize. Table I shows the configuration used in the offline planner in all the scenarios presented in Section VII, except in scenario B1.
2) The online planner uses only two optimization indexes: the PKill and the PLR. PKill is more important than the PLR when replanning due to the appearance of an unexpected ADU and, therefore, in the online planner, PLR priority level is reduced (i.e., set to 3).

V. MULTIPLE UNMANNED-AERIAL-VEHICLES OPTIMIZATION FRAMEWORK

The previous sections show how to codify the solutions and evaluate them. However, they do not explain how the planner solves the single and multiple-UAV trajectory problem. This section and the following contain that information.

Calculating the paths of several UAVs that are flying simultaneously is a complex task due to the space constraints that one path imposes on the others and the cooperative objectives of the set of UAVs that the planner has to optimize. The problem has already been tackled by nonlinear-programming and mixed-integer linear-programming methods, and the results of both techniques have been compared in [37].

Our approach, i.e., MCACEA, uses multiple EAs (one per each UAV) that evolve their own populations to find the best path for its associated UAV according to their individual and cooperation constraints and objective indexes. Each EA is a single-UAV planner that runs in parallel and that exchanges some information with the others during its evaluation step. This information is needed to let each EA measure the coordination objectives of the paths encoded in its own population for its UAV, taking into account the possible optimal paths of the remaining UAVs. Therefore, each single planner has to receive information related to the best solutions of the remaining single planners before evaluating the cooperative objectives of each possible solution of its own population.

As the cooperation objective values depend on the best solutions of the other populations and the optimality of a solution depends both on the individual and cooperation objectives, it is not really possible to select and send the best solution of each planner to the others. However, we can divide the evaluation step inside each EA in three parts: In the first part, the EA identifies the best solution considering only the individual objective values and sends it to the others; in the second part, the cooperation objective values of all the paths are calculated taking into account the received information; and in the third part, the EA calculates the fitness of the solutions considering all the individual and cooperation objective values. However, there is another problem that also needs to be solved. Although each UAV can only follow a unique optimal path, each EA maintains a Pareto set of optimal paths and selects the unique optimal path at the end, when the last population has already been obtained (see Section VI). Therefore, to be able to determine a unique optimal path according with the individual objectives in each generation, the step in charge of selecting the final optimal path is included in the evaluation step of each EA.

TABLE II
EVALUATION STEP IN THE COOPERATING PLANNERS

1. Evaluating the individual objectives of each path, which are presented as the ten first objectives from section II.
2. Calculating the fitness of each path with the evaluation function of section IV with only the individual objectives.
3. Finding the best solution of the population, as explained in section VI-G.
4. Sending (and receiving) the best solution to (of) the other single UAV planners.
5. Calculating the cooperation objectives taking into account the received information from the other EAs.
6. Calculating the fitness of each path with the evaluation function of section IV and the individual and cooperation objectives, which have been obtained in steps 1 and 5.

Table II shows the complete evaluation step of the individual cooperating planners, including references to the sections where each function is explained in detail. When the planner calculates the path of a single UAV, it only uses the first two steps of this new evaluation process.

Note that in our evaluation process, the trajectory that is considered the best in accordance with the individual objectives is not necessary the best after evaluating the coordination objectives. In other words, the EAs may sometimes evaluate the coordination objectives against the "wrong" trajectories. This could make the EAs have some initial difficulties to converge to collision-free trajectories, although the results presented in Section VII show that this problem is usually solved after a few generations. Another possibility, which is considered in [20], is to send the best completely evaluated trajectory of the previous generation to the other EAs instead of our current best-only individual objective evaluated one. Nevertheless, not only does the approach in [20] not solve the problem, but, we believe, it worsens it because it introduces a bias toward outdated completely evaluated trajectories, while our approach does it toward currently good individual objective evaluated ones. Finally, Nikolos *et al.* [18], and Nikolos and Tsourvelouds [19] optimize the path of multiple UAVs without considering interdependent objectives.

To reduce the bandwidth of the distributed planner, the information exchanged by the EAs are the 3-D point lists instead of the pointwise discretized cubic splines. The time used to exchange this information among the UAVs' onboard computers depends on the UAVs position, environment, and/or communication technology. The communication delays affect every EA generation, and therefore, they are especially problematic for the online planner since it needs to obtain the alternative routes before the pop-ups destroy the UAVs. To minimize their effects, when the UAVs are flying, the replanning process is done sequentially: Only the EA of an UAV is run until it finishes to optimize its own trajectory considering the trajectories of the other UAVs constant because their associated EAs are not working. Therefore, only the final 3-D list of the u UAV is broadcasted to the others when the u EA has finished after T_r^u s. Which EAs need to be started is determined whenever a new pop-up is identified, and their initialization order is calculated based on the time, T_i^u, needed by each u UAV to intersect the R_{PKmax} regions of the identified pop-ups that affect the remaining part of its current cubic spline. All the UAVs survive, when the started EAs are able to find *feasible trajectories* with PKill $=$ 0 on time, i.e, when $T_i^v > \sum_{k\in\mathrm{previous(order},v)} T_r^k$. Therefore, it is extremely important to minimize T_r^u. To achieve it, we use the U-shaped inclusion of waypoints in the 3-D lists described in Section III and the stop criterion presented in Section VI-F. Section VII shows an illustrative example, where we analyze real T_r^u and T_i^u times.

Finally, MCACEA is not only valid for the UAV problem, but can also be applied to other optimization problems with cooperating agents. Although this way of proceeding may look similar to the habitual parallelization of EAs (which is used in [38] and [39] to solve UAV problems), in this case, instead of distributing the solutions of the whole problem between different EAs that share their solutions periodically, we are dividing the problem into smaller problems that are solved simultaneously by each EA taking into account the solutions of the part of the problems that the other EAs are obtaining.

VI. EVOLUTIONARY OPERATORS AND OTHER RELEVANT CHARACTERISTICS OF THE EVOLUTIONARY ALGORITHM

The basic characteristics of each EA, which are given by the type of operators and functions implemented inside it, are presented in this section. The selection, recombination, and mutation methods differ from the ones used in our previous work: the first two to make it closer to NSGA-II and the last to take into account the new codification. Besides, the previously used local-search procedure has been disabled.

A. Selection

The binary tournament method [40] is used to select N_s pairs of elements from the current population of N_p solutions.

B. Crossover

A uniform distribution with crossover probability P_c is used to decide if each pair of selected solutions (lists with N_W waypoints) is mated to create two new solutions or left without change. The crossing point C_p of each pair of mating solutions is selected from the interval $[1, N_W]$, and the first new solution is formed with the $1 : C_p$ waypoints of the first parent and the $C_p + 1 : N_W$ of the second, while the second new solution has the $1 : C_p$ waypoints of the second parent and the $C_p + 1 : N_W$ of the first.

C. Mutation

We increment the values of each of the $3 * N_W$ elements of each solution (list with N_W 3-D $(x_{W_i}, y_{W_i}, z_{W_i})$ waypoints) with the values obtained from a zero-mean normal distribution with a small C_{ms} covariance. Besides, some of the elements, which are selected according to an uniform distribution with probability P_m, are incremented with the values of another zero-mean normal distribution with bigger C_{mb} covariance. This two-level incremental mutation lets the EA explore any region of the

space and escape from the confined region associated with the relative polar-representation initialization.

D. Immigration

To increment gene's diversity, a fixed number of new solutions (i.e., immigrants) is created using the random relative polar-representation initialization.

E. Recombination

The NSGA-II recombination method [28] is used to create the new population selecting a total of N_p solutions from the current population, the pairs of solutions that have been selected and undergone the crossover and mutation processes, and the immigrants. The population size N_p is maintained by means of preserving all the Pareto sets that fit into N_p, with a good Pareto-front distribution of the elements of the last fitted set.

F. Stop Criterium

Offline EAs stop when the generation limit is reached. Online EAs stop after 1) the routes are feasible (i.e., constraints fulfilled), safe (PKill $\neq 0$), and short enough, or 2) the generation limit is reached.

G. Selection of the Final Solution

We also need a criterion to select at the end, among the solutions belonging to the first Pareto set, the trajectory that the UAV will follow. We implement a method based on the priorities of the evaluation function and some additional rules.

For example, for the offline planner and priority levels presented in Table I, we consider following two cases.

1) There is a feasible solution with PKill $= 0$. Therefore, there is only one element in the first Pareto set because from the two objectives in the secondary priority level PKill has reached the minimum and PLR determines which solution is the best.
2) For all the feasible solutions, PKill > 0. We first obtain the relative PKill_i of each *i*th solution in the first Pareto set as $\text{PKrel}_i = \text{PKill}_i / \min_i(\text{PKill}_i)$. Then, we discard the solutions with $\text{PKrel}_i \geq V$ (for example, with $V = 1.05$), which makes the nondiscarded *j*th solutions have a PKill_j that is not significantly higher than the minimum. Finally, we select the solution with $\min_j(\text{PLR}_j)$. Therefore, the final solution has a PKill that is a bit higher than the minimum, as well as a reasonable PLR.

H. Reuse of Solutions

When an offline EA returns the final solution, the planner copies the waypoints that codify it to be able to use them for future optimizations. This lets us reduce the computational time when we want to find a trajectory over a scenario slightly different from another one that the planner has already solved. We can also include the results of any algorithm that finds a suboptimal solution of the problem in the first generation of the EA, as long as it is codified as described in Section III.

VII. Results

Our planner is part of a complex system that tests the performance of UAV planners in hostile environments. This system lets us 1) set the known and pop-up ADUs positions and the UAVs fixed points, 2) obtain their routes with a planner (including ours), and 3) test them against a simulator that uses real-world maps and complex UAV and ADU models.

We have successfully tested our planner in multiple scenarios, and in this paper, we present some examples that reflect its more important characteristics. Fig. 4 shows the scenarios (identifiable by a capital letter followed by the number of UAVs they contain) and the final trajectories returned by the planner. The big dashed gray circles mark each ADU's maximum distance of detection (i.e., R^j_{Dmax}), while the small red solid circles represent their maximum-risk distance (i.e., R^j_{PKmax}). Although each circle, respectively, represents the areas that can increment PRD and PKill, they do not have to do it necessarily because those values also depend on other factors (see Section II). The label ADU_i identifies the *i*th known ADU, and POP_i the *i*th pop-up ADU. Unlabeled ADUs are always known. The brown rectangles show the NFZs. Labels S, E, and IF identify the start, end, and intermediate fixed points of the UAVs. The offline planner trajectories are the solid lines that are not black or red. Sometimes they are labeled close to its S point as UAV_i. The thick black and red lines in Fig. 4(i) show the alternative routes calculated, during a simulation where pop-ups are identified, by the online planner. Note that some paths cover tens of kilometers, while others hundreds, and therefore, the examples show the good scalability of our planner. Besides, although we only show the isometric view of E2 in Fig. 4(h), the planner is always searching for 3-D routes.

We also present how some tunable parameters of the planner affect the quality of the trajectories in the different scenarios. To compare the results obtained by the planner for two different sets of parameter values, each planner is run N_r times with the same set of values, because the EAs stochastic foundation makes them obtain different trajectories in each execution. The comparison is carried out using the statistical front-dominance-ranking procedure (SFDRP) presented in [29], which measures if the results of algorithms A and B are statistically significantly different applying the nonparametric Mann–Whitney rank tests [41] to the number of times that each of the N_r final best Pareto fronts obtained by algorithm A is dominated by each of the N_r final best Pareto fronts obtained by algorithm B, and *vice versa*. We pick SFDRP because it can be directly applied to the single-UAV problem using the generic multiobjective evaluation method based on goals, priorities, and Pareto sets (see Section IV) when calculating the number of times the Pareto fronts are dominated. To use it in the multi-UAV problem, we have to consider the existence of a different best Pareto front for each UAV. To overcome this difficulty, the Pareto fronts used by SFDRP are created from the Cartesian product of the UAVs Pareto fronts. Therefore, the multi-UAV statistical tests are carried out over the space of all the possible trajectories for all the UAVs in the scenario. Finally, it is important to highlight that although our EAs incorporate a final step that selects a single best trajectory

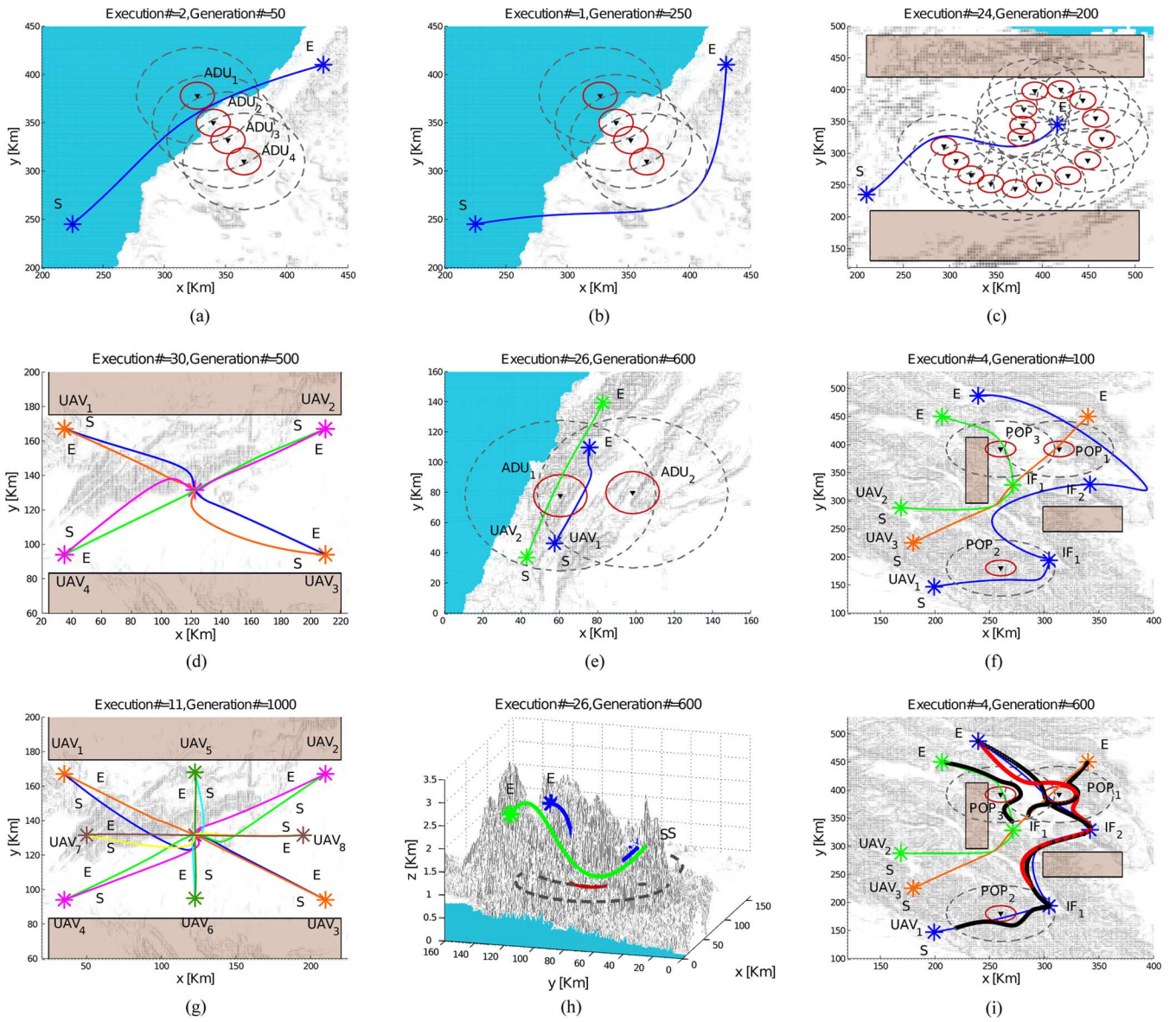

Fig. 4. UAV scenarios and solutions. (a) Scenario A1. (b) Scenario B1 (the priority level of PRD is 2). (c) Scenario C1. (d) Scenario D4. (e) Scenario E2. (f) Scenario F3 (intermediate solutions). (g) Scenario D8. (h) Scenario E2 (isometric view). (i) Scenario F3 (final solution + pop-up alternative route).

for each UAV, SFDRP is carried out over the final best fronts, because the EAs evolve its own population according to the fronts and select the best at the end.

The EA parameters and number of waypoints used to obtain the offline trajectories of Fig. 4 are shown in Tables[1] III–V. The mean execution time (i.e., Time) of the planner in a Pentium IV at 3.0 GHz is also included,[2] although it is highly dependent on the scenario and trajectories that the EAs are evaluating. For instance, evaluating only one point of a trajectory that passes inside an ADU's R^j_{Dmax} region is 100 times slower than when it is not due to the use of the LoS function. Therefore, the LoS function is disabled in the online planner, which prevents the UAVs from using the masking effect in the replanned part of the trajectory, but evicts this costly time consumption.

Finally, our EAs use a conservative model to obtain PKill (see Section II), which ensures that when PKill $= 0$ and there are no pop-up ADUs, the UAV always reaches its final destination. When PKill > 0, the UAV is assuming some risk, and depending on the simulation, it will succeed or fail with a probability close to the value obtained by the EA.

A. *Results in Single Unmanned-Aerial-Vehicle Scenarios*

These scenarios, which are presented in the top row of Fig. 4, show the planner's ability to find no risky paths in static environments with different number of ADUs. In scenario A1 [see Fig. 4(a)], there is a little corridor (of 8 km) between ADU_1 and ADU_2, where PKill $= 0$, which the planner always finds. However, the UAV is detected, because PRD is in the third priority

[1]The meaning of the acronyms in the tables is summarized in the Appendix.

[2]The code has been serialized to be run over a single computer and account only for the total processing time without measuring communication delays.

TABLE III
PLANNER-TUNABLE PARAMETERS UNDER TEST

Parameter	N_p	N_s	P_c	P_m
Nominal	30	12	0.75	0.008
Mininum	30	4	0.5	0.004
Maximum	80	14	1	0.0128
Increment	10	2	0.05	0.0008
Statistically Selected	Any	12 – 14	0.7 – 0.75	Any

level, while PLR is in the second, and therefore, the planner primes shorter paths to nondetectable ones. Scenario B1 [see Fig. 4(b)] minimizes PRD as well, as we increment the priority level of this objective before running the planner. Therefore, the planner now finds paths adjacent to the dashed gray circles. Finally, scenario C1 [see Fig. 4(c)] shows the planner's ability to find trajectories in areas with many ADUs, where the PKill $= 0$ area has a complex shape.

We use SFDRP in these scenarios to see how the values of some tunable parameters of the EAs affect the solutions. Table III shows the parameters under test (first row), which are changed one at a time, their nominal value during all the optimizations except when they are being changed (second row), their minimum value (third row), their maximum (fourth row), the increment used to change them (fifth row), and the range of values that are considered better after analyzing the results of SFDRP (sixth row). For each scenario and set of parameter values, we run the planner $N_r = 30$ times during 800 generations and save the best Pareto fronts obtained by the EA every 50 generations to be able to perform SFDRP at different stages of the algorithm. The results of the test show, with a 5% confidence, that there is no systematic statistical significant difference on the solutions obtained by the algorithm at the same generation, when we change N_p. Therefore, we pick the smaller value. The same happens when we change P_m, and therefore, we maintain the value used in our previous works. However, SFDRP shows that using a $P_c = 0.7$ or $P_c = 0.75$, or a $N_s \geq 12$, lets the EAs obtain fronts less times Pareto-dominated by the fronts obtained with the other P_c or N_s values, and therefore, we pick $P_c = 0.75$ (the EA has more crossing opportunities with bigger values) and $N_s = 12$ (smaller values reduce computational cost).

SFDRP is also used to see how the scenario influence the number of needed generations. In this test, the planner maintains the nominal tunable parameters and SFDRP compares the $N_r = 30$ best fronts obtained at a given X generation with the $N_r = 30$ best fronts obtained at another Y. Therefore, this test [whose results for scenario A1 are presented in Fig. 5(a)] tells us if we can (white) or cannot (gray) expect to obtain an overall better solution when we run the planner once up to generation Y than when we run it once up to generation X. As one expects, Fig. 5(a) shows that incrementing the X-generation number produces better results up to a given point, which is identified as G_S hereafter, after which, no better results should be expected.[3]

[3]For lack of space, we can include neither the SPDRP generation graphics for all scenarios nor the SPDRP-tunable parameter graphics. However, we use them to obtain the G_S of Tables IV and V and the conclusions of Table III.

However, SFDRP cannot quantify the improvements. Therefore, for each execution of the planner, we measure, in some generations, the improvement of the PLR[4] with respect to another generation 50 steps earlier, calculate its mean over the $N_r = 30$ executions, and represent it, for the three scenarios, in Fig. 5(b). These graphics let us identify a generation number, i.e., G_I, after which the planner cannot significantly improve its actual solution. Therefore, G_I is associated with improvements smaller than a threshold inside one execution of the planner, while G_S is related with overall improvements, but does not account for their size. Finally, to analyze the convergency of the planner, we also represent the PLR mean and standard deviation (Std) values of all the solutions that appear in the best Pareto fronts of the $N_r = 30$ optimizations in Fig. 5(c). The amplitude of values of the ordinate axis is maintained; therefore, all PLR Std values are presented in the same scale. Table IV also contains the generation in which the constraints are fulfilled, i.e., G_C; the mean execution time for 50 generations, i.e., Time; the number of waypoints, i.e., N_W; and the number of points of the discretized spline curve, i.e., N.

Finally, we want to point out that we also use SFDRP to decide if our new codification is better than the previous and to disable the local-search procedure used in our previous works. The results show that the new codification is usually better and only punctually worst at generation 50, while the local-search procedure rarely and punctually improves the results.

B. Results in Multiple Unmanned-Aerial-Vehicle Scenarios

The scenarios of the second and third rows of Fig. 4 show the planner's ability to find no risky paths in dynamic environments with multiple UAVs. We use $N_r = 30$ planner executions per scenario, with the nominal values of the tunable parameters of Table III, and perform the same generation tests as in the single-UAV case. The second row of Fig. 5 presents their mean improvement and mean PLR graphics. Table V collects the offline planner G_S, G_I, G_C, and Time numbers, and the N_W and N of each UAV trajectories.

Scenarios D2, D4 [see Fig. 4(d)], D6, and D8 [see Fig. 4(g)] are specially designed to analyze the difficulties of our planner to converge initially to collision-free trajectories. They consist of several pairs of UAVs with opposite starting and final points that are confined in a small region and have to bypass the same intermediate point flying at constant altitude.[5] In all cases, the planner converges to free-collision (and completely feasible) trajectories before generation 50. It also optimizes them considering the restriction that one trajectory imposes on the others: At least one path is straight, and the others bend as little as possible to avoid collisions.

In scenario E2 [see Fig. 4(e) and (h)], we illustrate the advantage of using the terrain inhibition to find free risky paths

[4]We use PLR only because the constraints are fulfilled in the three scenarios before generation 50, and PKill=0 in A1 and B1 before the generation 50 and in C1 before the generation 100.

[5]We impose this restriction because it increments the challenge as the UAVs cannot avoid each other using different altitudes. Each setup is the same as the previous with two extra UAVs. Therefore, D2 and D6, which are not represented for lack of space, are special cases of D4 and D8.

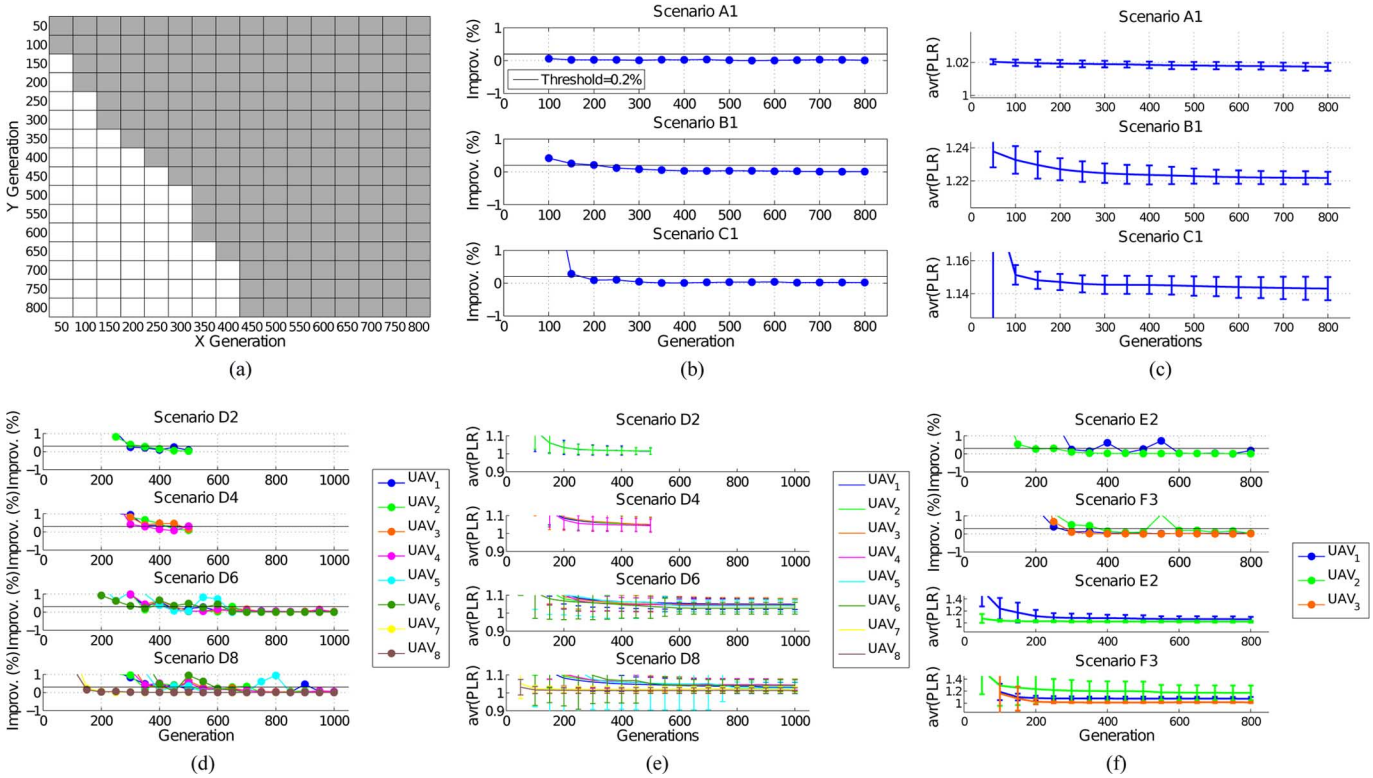

Fig. 5. UAV scenarios graphic results. (a) SFDRP generation graphic for scenario A1. (b) Mean values of the PLR improvements. (c) Mean PLR values of the best pareto solutions. (d) Mean values of the PLR improvements. (e) Mean PLR values of the best pareto solutions. (f) Mean improvements and Mean PLR.

TABLE IV
PLANNER AND SOLUTIONS PROPERTIES FOR THE SINGLE-UAV SCENARIOS

Scenario	Fig.	G_S	G_I	G_C	Time (s)	N_W	N
A1	4(a)	450	50	<50	193.2	2	121
B1	4(b)	650	250	<50	75.3	2	121
C1	4(c)	250	200	<50	209.9	3	161

TABLE V
PLANNER AND SOLUTIONS PROPERTIES FOR THE MULTI-UAV SCENARIOS

Sc.	Fig.	G_S	G_I	G_C	Time (s)	UAV	N_W	N
D2	4(d) 4(g)	400	400	<50	47.7	1,2	4	241
D4		>500	500	<50	308.3	3,4		
D6		>1000	650	<50	458.9	5,6	2	161
D8		>1000	950	<50	596.8	7,8		
E2	4(e) 4(h)	450	600	<50	152.9	1,2	2	121
F3	4(f) 4(i)	550	600	<50	97.9	1	6	361
						2	4	241
						3	4	241

among the mountains. ADU_1 is placed between mountains, and the UAVs find free risky paths inside the R^j_{PKmax} region (red solid circle), where PKill=0 due to the mountains.

In scenario F3 [see Fig. 4(f) and (i)], the UAVs have to visit several IF points, avoid NFZs, and three pop-ups. Fig. 4(f) represents the offline planner results after only 100 generations (as a first try). A visual inspection shows that the UAV_1 trajectory can be improved. Therefore, we restart the planner with the previous solution (see Section VI-H) and run it during 400 extra generations [see Fig. 4(i)]. Therefore, we run the planner during 500 generations, i.e., 100 initially and 400 later. The offline trajectories of UAV_1, UAV_2, and UAV_3, respectively, overfly POP_2 and POP_1, POP_3, and POP_1. The UAVs start flying and UAV_1 detects POP_2 as soon as it enters the R_{Dmax} area, and as its R_{PKill} area pop-up only intersects UAV_1 path, only UAV_1 EA is run to obtain the UAV_1 black trajectory in 11.8 s. Later on, the same happens with UAV_2 and POP_3, and the UAV_2 black trajectory is obtained in 1.8 s. Finally, UAV_3 detects POP_1, whose R_{PKill} area intersects first with UAV_3 trajectory and, later, with UAV_1. Therefore, UAV_3 EA runs for 0.3 s to obtain UAV_3 black trajectory, and once it is finished, UAV_1 EA runs for 2.5 s to obtain UAV_1 red trajectory. We run the online EA several times in this scenario, and observe that $\max(T_r^v) = 24$ s when the online EA run for the limited number of 100 generations. Considering that $R_{\mathrm{Dmax}} - R_{\mathrm{PKill}} = 30\,000$ m and that UAVs are flying at 250m/s, in the worst case, only five UAVs could be entering pop-up detection regions simultaneously and still be able to have their routes recalculated sequentially. However, the mean value of T_r^v is 5 s, and therefore, up to 24 UAVs will not be a problem for our sequential approach.[6]

C. General Remarks

The results of Fig. 5, and the G_S and G_I values of Tables IV and V show that although the number of needed

[6]As $\max(T_r^v)$, R_{Dmax}, and R_{PKill} depend on the scenario, these numbers, which illustrate the range of applicability of our sequential approach in this scenario, are only orientative.

generations depends on the scenario (as one could expect), with the correct number of generations the planner is able to find a solution really close to the optimum in all the cases.[7] Although the correct generation number for a specific scenario cannot be known beforehand, additional generations can be added, restarting the planner with a previous solution.

Besides, G_c shows that the constraints are always fulfilled in a small number of generations. Moreover, the planner quickly overcomes the initial instability free-collision issue ($G_c < 50$ in the multi-UAV problem), even in really crowed scenarios with multiple UAVs. Additionally, the analysis of the scenario with pop-ups shows that the sequential online approach is valid for multiple UAVs entering pop-up detection regions simultaneously, although it can fail when the number of UAVs grows and the online EAs need too much time.

Finally, our compact trajectory representation lets the planner identify appropriate solutions within a few generations with only a few waypoints N_W.

VIII. Conclusion

Our current work solves the problem of finding the paths of a set of cooperating UAVs in military missions. For this purpose, we have created the MCACEA for cooperating agents, which consists of a set of EAs that obtain the optimal behavior of each agent. EAs run in parallel offline and in serial online, and communicate each other information about their optimal solution (i.e., every iteration, offline; at the end, online). Each EA finds the optimal path of its associated UAV, and the optimality of the path for each UAV is measured, considering 1) each UAV mission and 2) the cooperation of the UAVs.

The flexibility of this planner is based on the evaluation method used to calculate the fitness of the solutions, which easily permits adding new optimization indexes and changing their goals and priorities. This characteristic is used to achieve different behaviors of the planner when it searches offline for the initial paths and when it works online to avoid pop-up threats. In both cases, the EAs consider their UAV physical constraints, the terrain, the properties of the known ADUs, the exclusion zones, and the obligatory intermediate points. Speedups can be obtained using a constant flight altitude and/or previously found solutions.

The results of the planner have been statistically analyzed to see its ability to converge to the optimal regions of the space and the influence of the scenarios in a set of parameters. The analysis shows that our planner behaves consistently in different scenarios and that the number of generations highly depends on the problem. It is also able to find feasible and nearly optimal solutions with small population sizes.

Our current work focuses on including the velocities of the UAVs in the problem codification, incrementing the number of cooperation objectives, and making the online planners work in parallel or serial in spite of the communication delays. The first idea will bring important improvements and support new ways to solve collisions problems [42]. It will also increment the survival possibilities of the UAVs because the planner will be able to raise the velocity near the ADUs to stay the shortest time possible inside the dangerous areas. The planner will also be able to slow down an UAV to save fuel or to facilitate its maneuvers. The second idea will consider new ways of coordinating the missions, such as making different UAVs to arrive at the same (or different) point at similar times or using bait UAVs to increment the survival possibilities of others. The third idea will permit testing of the distributed planner in scenarios that include communication delays.

Other UAV-planning tasks, such as target recognition [43], [44], tracking [45], cooperative search [46], and vehicle-routing problem [47], are also being solved with EAs. Although our research is currently focused in the problem of finding the best path that avoids the risk, we are analyzing the possibilities of widening the types of tasks that the EA planner can optimize.

[7]The small improvements and PLR Std of scenario A1 are due to the fact that at generation 50, the planner finds a solution inside the corridor that is really close to the optimum. In scenario B1, the solutions are improved longer, because the planner is trying to fit them around the dashed gray circle. In scenario C1, the big initial improvement and Std are due to the fact that at generation 50, the solution of the planner in 20% of all executions are passing through PKill $\neq 0$ regions. After generation 100, the planner improves the solutions around the optimum. Scenarios D# get harder as the number of UAVs grows, and therefore, the improvements [see Fig. 5(d)] and mean PLR values [see Fig. 5(e)] decrease slowly. In scenario E2, the bigger PLR Std of UAV1 [see Fig. 5(f)] is originated by the high variety of free risky paths that UAV finds among the mountains. Finally, the high PLR Std of UAV2 in scenario F3 is due to the fact that in 10% of the execution, the trajectory after IF_1 surrounds its closer NFZ by its left side.

TABLE VI

Acronym	Description
UAV	Unmanned Air Vehicle
ADU	Air Defense Unit
NFZ	Flight Prohibited Zone, No Flying Zones
PLR	Path Length Ratio
PRD	Probability of Radar Detection
PKill	Probability of being destroyed (Killed)
R_{PKmax}	Missile Influence Radius
R_{Dmax}	Radar Detection Influence Radius
LoS	Line of Sight
N_p	Population size
N_s	Number of pairs of parents
P_c	Crossover probability
P_m	Mutation probability
G_S	Generation number where global improvements, among different runs of the EAs, stop
G_I	Generation number where PLR improvements, within the same run of the EA, stop
G_C	Generation number where constrains are fulfilled
N_W	Number of waypoints of the 3D list
N	Number of discretized points of the spline curve
SFDRP	Statistical Front-Dominance Ranking Procedure

Appendix

Table VI contains the acronyms that were most used in this paper.

Acknowledgment

The authors would like to thank the reviewers for all their useful comments about this work. The EA that has been used to solve the problem uses part of the functionality of the MATLAB Toolbox EVOCOM [48], [49].

REFERENCES

[1] L. R. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicle* (Library of Flight Series). Reston, VA: AIAA, 2004.

[2] T. Samad, J. S. Bay, and D. Godbole, "Network-centric systems for military operations in urban terrain: The role of UAVs," in *Proc. IEEE*, Jan. 2007, vol. 95, no. 1, pp. 92–107.

[3] J. A. Goldman, "Path planning problems and solutions," in *Proc. Nat. Aerosp. Electron. Conf.*, 1994, pp. 105–108.

[4] R. J. Szczerba, "Threat netting for real-time, intelligent route planners," in *Proc. IEEE Symp. Inf., Decis. Control*, 1999, pp. 377–382.

[5] J. Bellingham, "Coordination and control of UAV fleets using mixed-integer linear programming," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, 2002.

[6] Y. Kuwata and J. P. How, "Three dimensional receding horizon control for UAVs," presented at the AIAA Guid., Navigat., Control Conf. Exhib., AIAA, Monterey, CA, 2002.

[7] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed-integer linear programming," in *Proc. Amer. Control Conf.*, 2002, pp. 1936–1941.

[8] J. J. Ruz, O. Arévalo, J. M. de la Cruz, and G. Pajares, "Using MILP for UAVs trajectory optimization under radar detection risk," in *Proc. 11th IEEE Int. Conf. Emerging Technol. Factory Autom.*, 2006, pp. 1–4.

[9] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, and A. Oustaloup, "Consideration of obstacle danger level in path planning using A* and fast-marching optimisation: Comparative study," *Signal Process.*, vol. 83, no. 11, pp. 2387–2396, 2003.

[10] R. J. Szczerba, P. Galkowski, I. Glickstein, and N. Ternullo, "Robust algorithm for real-time route planning," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 3, pp. 869–878, Jul. 2000.

[11] K. Trovato, "A* planning in discrete configuration spaces of autonomous systems," Ph.D. dissertation, Amsterdam Univ., Amsterdam, The Netherlands, 1996.

[12] Y. Qu, Q. Pan, and J. Yan, "Flight path planning of UAV based on heuristically search and genetic algorithms," in *Proc. 31st Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2005, p. 5.

[13] A. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad, "Dynamic optimization strategies for 3d conflict resolution of multiple aircraft," *AIAA J. Guid., Control Dyn.*, vol. 27, no. 4, pp. 586–594, Jul.–Aug. 2004.

[14] S. Mittal and K. Deb, "Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2007, vol. 7, pp. 3195–3202.

[15] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 6, pp. 898–912, Dec. 2003.

[16] I. Hasircioglu, H. R. Topcuoglu, and M. Ermis, "3-d path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, 2008, pp. 1499–1506.

[17] Y. V. Pehlivanoglu, O. Baysal, and A. Hacioglu, "Vibrational genetic algorithm based path planner for autonomous UAV in spatial data based environments," in *Proc. 3rd Int. Conf. Recent Adv. Space Technol.*, 2007, vol. 7, pp. 573–578.

[18] I. K. Nikolos, N. C. Tsourveloudis, and K. P. Valavanis, "Evolutionary algorithm based path planning for multiple UAV cooperation," in *Advances in Unmanned Aerial Vehicles*. Berlin, Germany: Springer-Verlag, Jan. 2007, pp. 309–340.

[19] I. K. Nikolos and N. C. Tsourvelouds, "Path planning for cooperating unmanned vehicles over 3-d terrain," *Inf. Control, Autom. Robot.*, vol. 24, pp. 153–168, Jan. 2009.

[20] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 609–620, Aug. 2005.

[21] R. Zhang, C. Zheng, and P. Yan, "Route planning for unmanned air vehicles with multiple missions using an evolutionary algorithm," in *Proc. IEEE 3rd Int. Conf. Nat. Comput.*, 2007, pp. 1499–1506.

[22] J. M. de la Cruz, E. Besada-Portas, L. de la Torre, B. Andrés-Toro, and J. A. Lopez-Orozco, "Evolutionary path planner for UAVs in realistic environments," in *Proc. Genet. Evol. Comput. Conf.*, 2008, pp. 1447–1484.

[23] J. M. de la Cruz, E. Besada Portas, L. de la Torre, and B. Andrés Toro, "Multiobjective path planner for unmanned air vehicles (UAVs) based on genetic algorithms," presented at the 8th Int. FLINS Conf., Madrid, Spain, 2008.

[24] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*. New York: Academic, 1988.

[25] D. A. V. Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a pareto front," in *Proc. Genet. Program. Conf.*, 1998, pp. 221–228.

[26] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: Unified formulation," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 18, no. 1, pp. 26–37, Jan. 1988.

[27] C. A. Coello-Coello and M. Salazar Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," presented at the Congr. Evol. Comput., Honolulu, HI, 2002.

[28] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Parallel Problem Solving Nat. VI Conf.*, Berlin, Germany: Springer-Verlag, 2000, pp. 849–858.

[29] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastive multiobjective optimizers," Comput. Eng. Netw. Lab., ETH Zurich, Zurich, Switzerland, Tech. Rep. TIK 214, Feb. 2006.

[30] P. T. Kabamba, S. M. Meerkov, and F. H. Zeitz, "Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking," *J. Guid. Control Dyn.*, vol. 29, no. 2, pp. 279–288, Mar./Apr. 2006.

[31] B. Andrés-Toro, E. Besada-Portas, P. Fernandez-Blanco, J. A. Lopez-Orozco, and J. M. de la Cruz, "Multiobjective optimization of dynamic processes by evolutionary algorithms," presented at the 15th Triennial World Congr. IFAC, Barcelona, Spain, 2002.

[32] S. Esteban, B. Andrés-Toro, E. Besada-Portas, J. M. Girón Sierra, and J. M. de la Cruz, "Multiobjective control of flaps and t-foil in high speed ships," in *Proc. 15th Triennial World Congr. IFAC*, 2002, pp. 1–6.

[33] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[34] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions and cooperative-timing missions," *J. Guid. Control Dyn.*, vol. 28, no. 1, pp. 150–161, Jan./Feb. 2005.

[35] R. Beard, T. McLain, M. Goodrich, and E. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.

[36] Z. Jin, T. Shima, and C. J. Schumacher, "Optimal scheduling for refueling multiple autonomous aerial vehicles," *IEEE Trans. Robot. Autom.*, vol. 22, no. 4, pp. 682–693, Aug. 2006.

[37] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, "MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles," in *Proc. Amer. Control Conf.*, Jun. 2006, pp. 5763–5768.

[38] D. Jian and J. Vagners, "Parallel evolutionary algorithms for UAV path planning," in *Proc. AIAA 1st Intell. Syst. Tech. Conf.*, 2004, pp. 1499–1506.

[39] M. A. Darrah, W. M. Niland, B. M. Stolarik, and L. E. Walp, "Increased UAV task assignment performance through parallelized genetic algorithms," Air Force Res. Lab., Dayton, OH, Tech. Rep. AFRL-VA-WP-TP-2006-339, Aug. 2006.

[40] D. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.

[41] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, pp. 50–60, Jan. 1947.

[42] J. J. Rebollo, I. Maza, and A. Ollero, "A two step velocity planning method for real-time collision avoidance of multiple aerial robots in dynamic environments," in *Proc. 17th World Congr., Int. Federation Autom. Control*, Jul. 2008, pp. 1–6.

[43] T. Shima and C. Schumacher, "Assignment of cooperating UAVs to simultaneous tasks using genetic algorithms," in *Proc. AIAA Guid., Navigat., Control Conf. Exhib.*, Aug. 2005, pp. 15–18.

[44] J. Tian, L. Shen, and Y. Zheng, "Genetic algorithm based approach for multi-UAV cooperative reconnaissance mission planning problem," in *Lecture Notes in Computer Science*, vol. 4203. Berlin, Germany: Springer-Verlag, 2006, pp. 101–110.

[45] D. Howden and T. Hendtlass, "Collective intelligence and bush fire spotting," in *Proc. Genet. Evol. Comput. Conf.*, 2008, pp. 41–48.

[46] J. Tian, Y. Zheng, H. Zhu, and L. Shen, "A MPC and genetic algorithm based approach for multiple UAVs cooperative search," in *Lecture Notes in Computer Science*, vol. 3801. Berlin, Germany: Springer-Verlag, 2005, pp. 399–404.

[47] M. Russel and G. Lamont, "A genetic algorithm for unmanned aerial vehicle routing," in *Proc. GECCO*, 2005, pp. 1499–1506.

[48] E. Besada-Portas, J. A. Lopez-Orozco, and B. Andres-Toro, "A versatile toolbox for solving industrial problems with several evolutionary techniques," in *Evolutionary Methods for Design, Optimization and Control*. Chapel Hill, NC: Univ. North Carolina Press, 2002, pp. 325–330.

[49] E. Besada-Portas, J. A. Lopez-Orozco, and B. Andres-Toro, *Evocom Toolbox Ver. 2.0*, Spanish Registro Propiedad Intelectual Comunidad Madrid, Madrid, Spain, 16/2007/5293, 2004.

Eva Besada-Portas was born in Madrid, Spain, on January 16, 1974. She received the M.Sc. degree in physics and the Ph.D. degree in computer engineering from the Complutense University of Madrid (UCM) in 1997 and 2004, respectively.

She was a Teaching Assistant at the UCM, where she has been an Assistant Professor with the Department of Computer Architecture and Automatic Control since 2005. Since 2006, she has been a Postdoctoral Visiting Researcher with the Department of Computer Science, University of New Mexico, Albuquerque. Her current research interests include optimal control, evolutionary algorithms, unmanned aerial vehicles, multisensor fusion systems, and machine-learning techniques.

Luis de la Torre was born in Malaga, Spain, on April 23, 1983. He received the M.Sc. degree in physics from the Complutense University of Madrid (UCM), Madrid, Spain, with specialization in electronic devices and automatic control, in 2008. He is currently working toward the Ph.D. degree in systems engineering and automatic control with Open University of Spain (UNED), Madrid.

He is currently a Research Fellow with the Department of Computer Sciences Automatic and Control, UNED. In 2007, he joined the Department of Computers Architecture and Automatic Control, UCM, as a Research Fellow. His current research interests include evolutionary algorithms, unmanned aerial vehicles, path planning, remote and virtual laboratories, and distance education.

Jesús M. de la Cruz (M'80) received the M.Sc. and Ph.D. degrees in physics from the Physics Science Faculty, Complutense University of Madrid (UCM), Madrid, Spain, in 1979 and 1984, respectively.

From 1985 to 1990, he held teaching appointments at the UCM and at the Open University of Spain (UNED), Madrid. From 1990 to 1992, he was with the Department of Electronics, Cantabria University of Santander, Santander, Spain. In October 1992, he joined the Department of Computer Architecture and Automatic Control, UCM, where he was the Dean from 1997 to 2001 and is currently a Full Professor and the Head of the Automatic Control and Robotics Group. His current research interests include broad aspects of automatic control and its applications, real-time control, optimization, statistical learning, and robotics.

Dr. de la Cruz has been the Chair of the Spanish Chapter of the IEEE Oceanic Engineering Society.

Bonifacio de Andrés-Toro was born in Cadiz, Spain, on April 26, 1940. He received the M.Sc. and Ph.D. degrees in physics from the Complutense University of Madrid (UCM), Madrid, Spain, in 1971 and 1996, respectively.

In 1990, he joined the Department of Computer Architecture and Automatic Systems, UCM, as a Lecturer, where he has been an Associate Professor since 2002. His current research interests include development of evolutionary algorithms to solve complex problems, such as the control of beer-fermentation processes and path planning in unmanned vehicles.