UAV Path Planning Using Evolutionary Algorithms

Ioannis K. Nikolos, Eleftherios S. Zografos, and Athina N. Brintaki

Department of Production Engineering and Management,
Technical University of Crete, University Campus,
Kounoupidiana, GR-73100, Chania, Greece
jnikolo@dpem.tuc.gr

Abstract. Evolutionary Algorithms have been used as a viable candidate to solve path planning problems effectively and provide feasible solutions within a short time. In this work a Radial Basis Functions Artificial Neural Network (RBF-ANN) assisted Differential Evolution (DE) algorithm is used to design an off-line path planner for Unmanned Aerial Vehicles (UAVs) coordinated navigation in known static maritime environments. A number of UAVs are launched from different known initial locations and the issue is to produce 2-D trajectories, with a smooth velocity distribution along each trajectory, aiming at reaching a predetermined target location, while ensuring collision avoidance and satisfying specific route and coordination constraints and objectives. B-Spline curves are used, in order to model both the 2-D trajectories and the velocity distribution along each flight path.

1 Introduction

1.1 Basic Definitions

The term *unmanned aerial vehicle* or UAV, which replaced in the early 1990s the term *remotely piloted vehicle* (RPV), refers to a powered aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non lethal payload [1]. UAVs are currently evolving from being remotely piloted vehicles to autonomous robots, although ultimate autonomy is still an open question.

The development of autonomous robots is one of the major goals in Robotics [2]. Such robots will be capable of converting high-level specification of tasks, defined by humans, to low-level action algorithms, which will be executed in order to accomplish the predefined tasks. We may define as *plan* this sequence of actions to be taken, although it may be much more complicated than that. Motion planning (or trajectory planning) is one category of such problems. Besides the great variety of planning problems and models found in Robotics, some basic terms are common throughout the entire subject.

The *state space* includes all possible situations that might arise during the planning procedure. In the case of an UAV each state could represent its position in physical space, along with its velocity. The state space could be either discrete or continuous; motion planning is planning in continuous state spaces. Although its definition is an important component of the planning problem formulation, in most cases is implicitly represented, due to its large size [3].

Planning problems also involve the *time* dimension. Time may be explicitly or implicitly modeled and may be either discrete or continuous, depending on the planning problem under consideration. However, for most planning problems, time is implicitly modeled by simply specifying a path through a continuous space [3].

Each state in the state space changes through a sequence of specific actions, included in the plan. The connection between actions and state changes should be specified through the use of proper functions or differential equations. Usually, these actions are selected in a way to "move" the object from an initial state to a target or goal state.

A planning algorithm may produce various different plans, which should be compared and valued using specific criteria. These criteria are generally connected to the following major concerns, which arise during a plan generation procedure: *feasibility* and *optimality*. The first concern asks for the production of a plan to safely "move" the object to its target state, without taking into account the quality of the produced plan. The second concern asks for the production of optimal, yet feasible, paths, with optimality defined in various ways according to the problem under consideration [3]. Even in simple problems searching for optimality is not a trivial task and in most cases results in excessive computation time, not always available in real-world applications. Therefore, in most cases we search for suboptimal or just feasible solutions.

Motion planning usually refers to motions of a robot (or a collection of robots) in the two-dimensional or three-dimensional physical space that contains stationary or moving obstacles. A motion plan determines the appropriate motions to move the robot from the initial to the target state, without colliding into obstacles. As the state space in motion planning is continuous, it is uncountably infinite. Therefore, the representation of the state space should be implicit. Furthermore, a transformation is often used between the real world where the robots are moving and the space in which the planning takes place. This state space is called the *configuration space* (C-space) and motion planning can be defined as a search for a continuous path in this high-dimensional configuration space that ensures collision avoidance with implicitly defined obstacles. However, the use of configuration space is not always adopted and the problem is formulated in the physical space; especially in cases with constantly varying environment (as in most of UAV applications) the use of configuration space results in excessive computation time, which is not available in real-time in-flight applications. *Path planning* is the generation of a space path between an initial location and the desired destination that has an optimal or near-optimal performance under specific constraints [4]. A detailed description of motion and path planning theory and classic methodologies can be found in [2] and in [3].

1.2 Cooperative Robotics

The term *collective behavior* denotes any behavior of agents in a system of more than a single agent. *Cooperative behavior* is a subclass of collective behavior which is characterized by cooperation [5]. Research in cooperative Robotics has gained increased interest since the late 1980's, as systems of multiple robots engaged in cooperative behavior show specific benefits compared to a single robot [5]:

- Tasks may be inherently too complex, or even impossible, for a single robot to accomplish, or the performance is enhanced if using multiple agents, since a single robot, despite its capabilities and characteristics, is spatially limited.
- Building or using a system of simpler robots may be easier, cheaper, more flexible and more fault-tolerant than using a single more complicated robot.

In [5] cooperative behavior is defined as follows: Given some tasks specified by a designer, a multiple robot system displays cooperative behavior if, due to some underlying mechanism, i.e. the "mechanism of cooperation", there is an increase in the total utility of the system.

Geometric problems arise when dealing with cooperative moving robots, as they are made to move and interact with each other inside the physical 2D or 3D space. Such geometric problems include multiple-robot path planning, moving to and maintaining formation, and pattern generation [5].

According to Fujimura [6], path planning can be either *centralized* or *distributed*. In the first case a universal path planner makes all decisions. In the second case each agent plans and adjusts its path. Furthermore, Arai and Ota [7] allow for hybrid systems that are combinations of *on-line*, *off-line*, *centralized*, or *decentralized* path planners. According to Latombe [2], centralized planning takes into account all robots, while decoupled planning corresponds to independent computation of each robot's path. Methods originally used for single robots can be also applied to centralized planning. For decoupled planning two approaches were proposed: a) prioritized planning, where one robot at a time is considered, according to a global priority, and b) path coordination, where the configuration space-time resource is appropriately scheduled to plan the paths.

Cooperation of UAVs has gained recently an increased interest due to the potential use of such systems for fire fighting applications, military missions, search and rescue scenarios or exploration of unknown environments (space-oriented applications). In order to establish a reliable and efficient framework for the cooperation of a number of UAVs several problems have to be encountered:

- UAV task assignment problem: a number of UAVs is required to perform a number of tasks, with predefined order, on a number of targets. The requirements for a feasible and efficient solution include taking into account: task precedence and coordination, timing constraints, and flyable trajectories [8]. The task re-assignment problem should be also considered, in order to take into account possible failure of a UAV to accomplish its task. The task assignment problem is a well-known optimization problem; it is *NP*-hard and, consequently, heuristic techniques are often used.
- UAV path planning problem: a path planning algorithm should provide feasible, flyable and near optimal trajectories that connect starting with target points. The requirement for feasible trajectories dictates collision avoidance between the cooperating UAVs as well as between the vehicles and the ground. The requirement of flyable trajectories usually dictates a lower bound on the turn radius and speed of the UAVs [8]. Additionally, an upper bound for the speed of each UAV may be required. The path optimality can be defined in various ways, according to the mission assigned. However, a typical requirement is to minimize the total length of the paths.
- Data exchange between cooperating UAVs and data fusion: exchange of information between cooperating UAVs is expected to enhance the effectiveness of the team. However, in real world applications communication imperfections and constraints are expected, which will cause coordination problems to the team [9]. Decentralized implementations of the decision and control algorithms may reduce the sensitivity to communication problems [10].
- Cooperative sensing of the targets: the problem is defined as how to co-operate the UAV sensors in terms of their locations to achieve optimal estimation of the state of each target [11] (a target localization problem).
- Cooperative sensing of the environment: the problem is defined as how to cooperate the UAV sensors in order to achieve better awareness of the environment (popup threats, changing weather conditions, moving obstacles etc.). In this category we may include the coordinated search of a geographic region [12].

1.3 Path Planning for Single and Multiple UAVs

Compared to the path-planning problem in other applications, path planning for UAVs has some of the following characteristics, according to the mission [13, 14, 15]:

- Stealth, in order to minimize the probability of detection by hostile radar, by flying along a route which keeps away from possible threats and/or has a lower altitude to avoid radar detection.
- Physical feasibility, which refers to the physical (or technology) limitations from the use of UAVs, such as limited range, minimum turning angle, minimum and maximum speed etc.
- Performance of mission, which imposes special requirements, including maximum turning angle, maximum climbing/diving angle, minimum and/ or maximum flying altitude and specific approaching angle to the target point.
- Cooperation between UAVs in order to maximize the possibility of mission accomplishment.
- Real-time implementation, which asks for computationally efficient algorithms.

The characteristics above imply special issues that have to be considered for an efficient modeling of the (single or multiple) UAV path planning problems.

Path modeling:

The simpler way to model an UAV path is by using straight-line segments that connect a number of way points, either in 2D or 3D space [12, 15]. This approach takes into account the fact that in typical UAV missions the shortest paths tend to resemble straight lines that connect way points with starting and target points and the vertices of obstacle polygons. Although way points can be efficiently used for navigating a flying vehicle, straight-line segments connecting the corresponding way points cannot efficiently represent the real path that will be followed by the vehicle. As a result, these simplified paths cannot be used for an accurate simulation of the movement of the UAV in an optimization procedure, unless a large number of way points is adopted. In that case the number of design variables in the optimization procedure explodes, along with the computation time. The problem becomes even more difficult in the case of cooperating flying vehicles.

In [16], paths from the initial vehicle location to the target location are derived from a graph search of a Voronoi diagram that is constructed from the known threat locations. The resulting paths consist of line segments. These paths are subsequently smoothed around each way point, in order to provide feasible trajectories within the dynamic constraints of the vehicle. A great advantage of the Voronoi diagram approach is that it reduces the path planning problem from an infinite dimensional search, to a finite-dimensional graph search. This important abstraction makes the path planning problem feasible in near-real time, even for a large number of way points [16].

Vandapel et al. [17] used a network of free space bubbles to model the path of small scale UAVs, in order to solve the path planning problem of autonomous unmanned aerial navigation below the forest canopy. Using a priori aerial data scans of forest environments, they compute a network of free space bubbles, which form safe paths within the forest environment. Their approach is tailored to the problem of small scale UAVs and can be decomposed into two steps: 1) the scene made of 3-D points is segmented into three classes (ground, vegetation and tree trunk-branches). 2) A path planning algorithm explores the segmented environment and computes connected obstacle-free areas, which will subsequently form a network of tunnels intersecting at some locations.

An alternative approach is to model the UAV dynamics using the Dubins car formulation [18]. The UAV is assumed to fly with constant altitude, constant flight speed and to have continuous time kinematics [19]. This approach cannot efficiently model real world scenarios, which may include 3D terrain avoidance or following of stealthy routes. However, this approach seems to be sufficient enough for task assignment purposes to cooperating UAVs flying at safe altitudes [19, 8, 20].

B-Spline curves have been used for trajectory representation in 2-D environments (simulated annealing based path line optimization, combined with fuzzy logic controller for path tracking) [21], and in 3-D environments (Evolutionary Algorithm based path line optimization for a UAV over rough terrain) [22, 23]. B-Spline curves need a few variables (the coordinates of their control points) in order to define complicated 2D or 3D curved paths, providing at least first order derivative continuity. Each control point has a very local effect on the curve's shape and small perturbations in its position produce changes in the curve only in the neighborhood of the repositioned control point.

Cooperation Scenarios:

Path planning algorithms were initially developed for the solution of the problem of a single UAV. The increasing interest for missions involving cooperating UAVs resulted in the development of algorithms that take into account the special characteristics and constraints of such missions. The related works present various scenarios, formulations and approaches connected to cooperating UAV path planning problems. Some of the most representative scenarios are presented below.

Beard et al. [16] considered the scenario where a group of UAVs is required to transition through a number of known target locations, with a number of threats in the region of interest. Some threats are known a priori, some others "pop up" or become known only when a UAV flies near them. It is desirable to have multiple UAVs arrive on the boundary of each target's radar detection region simultaneously. Collision avoidance is ensured by supposing that individual UAVs fly at different pre-assigned altitudes. In this work the problem is decomposed in several sub-problems: a) The assignment problem of a number of UAVs to a number of targets in a way that each target has multiple UAVs assigned to it, with a high preference to specific targets. b) The determination for each team of UAVs assigned to a target of an estimated time over target that ensures simultaneous intercept and is feasible for all UAVs in the team. c) The determination of a path (specified via waypoints) that can be completed within the specified time over target, taking into account minimum and maximum velocity constraints. d) The transformation of the initial path into a feasible UAV trajectory. e) The development of controllers for each UAV to track their computed trajectory.

A simpler scenario is presented in [15], where the problem under consideration is to generate routes for cooperating UAVs in real time, which take into account the exposure of UAVs to the threats and enable the vehicles to arrive at their goal location simultaneously. Some of the threats are known a priori, some of them “pop up” or become known only when a UAV approaches to it. For each UAV are imposed minimum and maximum velocity constraints. The cooperation related constraints are: a) the simultaneous arrival of all UAVs at goal locations, and b) the collision avoidance between UAVs.

In [20] the motion-planning problem for a limited resource of mobile sensor agents (MSAs) is investigated, in an environment with a number of targets larger than the available MSAs. The MSAs are assumed to move much faster than the targets. In order to keep the targets in surveillance the members of the MSA team have to fly back and forth to update the targets’ status. This *NP*-hard problem is essentially a combination of the problems of sensor resource management and robot motion planning. The problem is formulated as an optimization problem whose objective is to minimize the average time duration between two consecutive observations of each target.

In [12] the objective is to provide a coordinated plan for searching a geographic region, represented by a grid of cells, using a team of searchers. Each cell is characterized by its elevation and a cost parameter that corresponds to the danger of visiting it. Each vehicle carries a sensor, characterized by scan radius, angle and direction. The mission objective is the target coverage, i.e. the percentage of the region that must be scanned during the mission. Scans can be performed only from safe cells (the cell and all 8 neighbors should have been previously scanned). Additionally the path that connects scanning points should traverse through already scanned cells (a soft constraint). For safety reasons scanning paths are not allowed to be very close to each other.

Solution methodologies:

Path planning problems are actually multi-objective multi-constraint optimization problems, in most cases very complex and computationally demanding [24]. The problem complexity increases when multiple UAVs should be used. Various approaches have been reported for UAVs coordinated route planning, such as Voronoi diagrams [16], mixed integer linear programming [25, 26] and dynamic programming [27] formulations.

In [25, 26] mixed-integer linear programming (MILP) is used to solve tightly-coupled task assignment problems with timing constraints. The advantage to this approach is that it yields the optimal solution for the given problem. The primary disadvantage is the high computational time required.

In [16] the motion-planning problem was decomposed into a waypoint path planner and a dynamic trajectory generator. The path-planning problem was solved via a Voronoi diagram and Eppstein's k-best paths algorithm. The trajectory generator problem was solved via a real-time nonlinear filter that explicitly accounts for the dynamic constraints of the vehicle and modifies the initial path. This decomposition of the motion-planning problem has the advantage of decomposing a non-polynomial optimization problem into two sub-problems that can be computed in near-real time, with the disadvantage of providing a suboptimal solution [16].

Computational intelligence methods, such as Neural Networks [28], Fuzzy Logic [29] and Evolutionary Algorithms (EAs) [15, 23] (or, in some cases, a combination of them) have been successfully used in the development of algorithms that produce trajectories for guiding mobile robots in known, unknown or partially known environments.

During the past few years, it has been shown by many researchers that EAs are a viable candidate to solve path planning problems effectively and provide feasible solutions within a short time without demanding excessive computer power. The reasons behind choosing EAs as an optimization tool for the path-planning problem are their high robustness compared to other existing directed search methods, their ease of implementation in problems with a relatively high number of constraints, and their high adaptability to the special characteristics of the problem under consideration [23].

Traditionally, EAs have been used for the solution of the path-finding problem in ground based or sea surface navigation [30]. Commonly, the generated trajectory composed of straight line segments, connecting successive way points, that guided a mobile robot or a vehicle along a 2-D path on the earth's or sea's surface. The design variables used represented the coordinates of the way points, where the vehicle changes its direction. Other approaches took into account the time dimension by using design variables that also described the vehicle steady speed as it traversed a part of its path. When the vehicle's operational environment was partially known or dynamic, a feasible and safe trajectory was planned off-line by the EA, and the algorithm was used on-line whenever unexpected obstacles were sensed [31, 32]. EAs have been also used for solving the path-finding problem in a 3-D environment for underwater vehicles, assuming that the path is a sequence of cells in a 3-D grid [33, 34].

In [23] an EA based framework was utilized to design an off-line/on-line path planner for UAVs autonomous navigation. The path planner calculates a curved path line, represented using B-Spline curves in a 3-D rough terrain environment; the coordinates of B-Spline control points serve as design variables. The off-line planner produces a single B-Spline curve that connects the starting and target points with a predefined initial direction. The on-line planner gradually produces a smooth 3-D trajectory aiming at reaching a predetermined target in an unknown environment; the produced trajectory consists of smaller B-Spline curves smoothly connected with each other. For both off-line and on-line planners, the problem is formulated as an optimization one; each objective function is formed as the weighted sum of different terms, which take into account the various objectives and constraints of the corresponding problem. Constraints are formulated using penalty functions.

Changwen Zheng et al. [15] proposed a route planner for UAVs, based on evolutionary computation, which can be used to plan routes for either single or multiple vehicles. The flight route consists of straight-line segments, connecting the way points from the starting to the goal points. A real coded chromosome representation is used; for each way point its physical coordinates are used as design variables, along with a state variable, which provides information on the feasibility of the corresponding way point and the feasibility of the route segment connecting the point to the next one. The cost function of flight route penalizes the length of the route, penalizes flight routes at high altitudes and routes that come dangerously close to known ground threat sites. The imposed constraints on route segments are relevant to: minimum route leg length, maximum route distance, minimum flying height, maximum turning angle, maximum climbing/diving angle, simultaneous arrival at target location and no collision between vehicles. The route planning problem is formulated as the problem of minimization of the cost function under the aforementioned constraints.

In [8] a multi-task assignment problem for cooperating UAVs is formulated as a combinatorial optimization problem. A Genetic Algorithm is utilized for assigning the multiple agents to perform multiple tasks on multiple targets. The algorithm allows efficiently solving this *NP*-hard problem and, additionally, allows taking into account requirements such as task precedence and coordination, timing constraints, and flyable trajectories. The performance metric for the optimization problem is defined as the cumulative distance traveled by the vehicles to perform all of the required tasks; the objective is to minimize this metric subject to the above requirements. Integer encoding is used for the chromosomes, which are composed of two rows; the first row presents the assignment of a vehicle to perform a task on the target appearing on the second row. The algorithm was compared to a stochastic random search and a deterministic branch and bound search methods and found to provide near optimal solutions considerably faster than the other methods.

1.4 Outline of the Current Work

The following scenario was considered in this work: Assuming a number of UAVs at different known initial locations, the issue is to produce 2-D trajectories, with a desirable velocity distribution along each trajectory, reaching a common target under specific coordination and route constraints. The constraints and objectives refer to: minimum path lengths, collision avoidance between the flying vehicles and the ground, predefined minimum and maximum UAV velocity magnitudes during their flights, predefined safety distance between UAVs, near simultaneous arrival to the target and target approach from different directions.

This work is an extension of a previous one [35], which used Differential Evolution (DE) in order to find optimal paths of coordinated UAVs, with the paths being modeled with straight line segments. The main drawback of that approach was the need of a large number of segments for complicated paths, resulting in a large number of design variables and, consequently, generations to converge. In this work the Differential Evolution (DE) algorithm is combined with a Radial Basis Functions Network (RBFN), which serves as a surrogate approximation, in order to reduce the number of exact evaluations of candidate solutions. The candidate paths are modeled in the physical space and evaluated with respect to the physical (working) space. B-Spline curves are used for path line modeling, and complicated paths can be produced with a small number of control variables.

The rest of the chapter is organized as follows: section 2 contains B-Spline and Evolutionary Algorithms fundamentals; the solid terrain formulation, used for experimental simulations, is also presented. An off-line path planner for a single UAV will be briefly discussed in section 3, in order to introduce the concept of UAV path planning using Evolutionary Algorithms. Section 4 deals with the concept of coordinated UAV path planning using Evolutionary Algorithms. The problem formulation is described, including assumptions, objectives, constraints, objective function definition and path modeling. Section 5 presents the optimization procedure using a combination of Differential Evolution and a Radial Basis Functions Artificial Neural Network, which is used as a surrogate model in order to enhance the converge rate of Differential Evolution algorithm. Simulations results are presented in section 6, followed by discussion and conclusions in section 7.

2 B-Spline and Evolutionary Algorithms Fundamentals

2.1 B-Spline Curves

Straight-line segments cannot represent a flying objects path line, as it is usually the case with mobile robots, sea and undersea vessels. B-Splines are adopted to define the UAV desired path, providing at least first order derivative continuity. B-Spline curves are well fitted in the evolutionary procedure; they need a few variables (the coordinates of their control points) in order to define complicated curved paths. Each control point has a very local effect on the curve's shape and small perturbations in its position produce changes in the curve only in the neighborhood of the repositioned control point.

B-Spline curves are parametric curves, with their construction based on blending functions [36, 37]. Their parametric construction provides the ability to produce non-monotonic curves. If the number of control points of the corresponding curve is $n+1$, with coordinates $(x_0, y_0, z_0), \ldots, (x_n, y_n, z_n)$, the coordinates of the B-Spline curve may be written as

$$x(u) = \sum_{i=0}^{n} x_i \cdot N_{i,p}(u), \tag{1}$$

$$y(u) = \sum_{i=0}^{n} y_i \cdot N_{i,p}(u), \tag{2}$$

$$z(u) = \sum_{i=0}^{n} z_i \cdot N_{i,p}(u), \tag{3}$$

where u is the free parameter of the curve, $N_{i,p}(u)$ are the blending functions of the curve and p is its degree, which is associated with curve's smoothness ($p+1$ being its order). Higher values of p correspond to smoother curves.

The blending functions are defined recursively in terms of a *knot* vector $U = \{u_0, \ldots, u_m\}$, which is a non-decreasing sequence of real numbers, with the most common form being the *uniform non-periodic* one, defined as

$$u_i = \begin{cases} 0 & if \quad i < p+1 \\ i-p & if \quad p+1 \leq i \leq n \\ n-p+1 & if \quad n < i. \end{cases} \tag{4}$$

The blending functions $N_{i,p}$ are computed, using the knot values defined above, as

$$N_{i,0}(u) = \begin{cases} 1 & if \quad u_i \leq u < u_{i+1} \\ 0 & \quad otherwise, \end{cases} \tag{5}$$

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u). \tag{6}$$

If the denominator of either of the fractions is zero, that fraction is defined to have zero value. Parameter u varies between 0 and $(n-p+1)$ with a constant step, providing the discrete points of the B-Spline curve. The sum of the values of the blending functions for any value of u is always 1.

The use of B-Spline curves for the determination of a flight path provides the advantage of describing complicated non-monotonic 3-dimensional curves with controlled smoothness with a small number of design parameters, i.e. the coordinates of the control points. Another valuable characteristic of the adopted B-Spline curves is that the curve is tangential to the control polygon at the starting and ending points. This characteristic can be used in order to define the starting or ending direction of the curve, by inserting an extra fixed point after the starting one, or before the ending control point. Figure 1 shows a quadratic 2-dimensional B-Spline curve ($p=2$) with its control points and the corresponding control polygon.

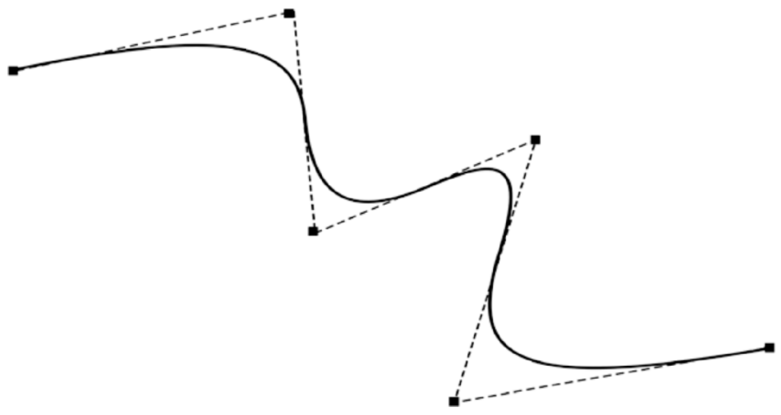

Fig. 1. A quadratic ($p = 2$) 2-dimensional B-Spline curve, produced using a uniform non-periodic knot vector, and its control polygon

2.2 Fundamentals of Evolutionary Algorithms (EAs)

EAs are a class of search methods with remarkable balance between exploitation of the best solutions and exploration of the search space. They combine elements of directed and stochastic search and, therefore, are more robust than directed search methods. Additionally, they may be easily tailored to the specific application of interest, taking into account the special characteristics of the problem under consideration [38, 39, 30].

The natural selection process is simulated in EAs, using a number (*population*) of individuals (candidate solutions to the problem) to evolve through certain procedures. Each individual is represented through *chromosome* - a string of numbers (bit strings, integers or floating point numbers), in a similar way with chromosomes in nature; it contains the design variables of the optimization problem. Each individual's quality is represented by a *fitness function* tailored to the problem under consideration.

Classic Genetic Algorithms (GAs) use binary coding for the representation of the *genotype*. However, floating point coding moves EAs closer to the problem space, allowing the operators to be more problem specific; this provides a better physical representation of the space constraints. Additionally, directed search techniques gain physical meaning and they are easily applicable.

In general, EA starts by generating, randomly, the initial chromosome population with their *genes* (the design variables in the case of floating point coding) taking values inside the desired constrained space of each design variable. The lower and higher constraints of each gene may be chosen in a way that specific undesirable solutions may be avoided. Although the shortening of the search space reduces the computation time, it may also lead to sub-optimal solutions, due to the lower variability between the potential solutions.

After the evaluation of each individual's fitness function, operators are applied to the population, simulating the according natural processes. Applied operators include various forms of *recombination*, *mutation* and *selection*, which are used in order to provide the next generation chromosomes. The first classic operator applied to the selected chromosomes is the *one-point crossover* scheme. Two randomly selected chromosomes are divided in the same (random) position, while the first part of the first one is connected to the second part of the second one and vice-versa. The crossover operator is used to provide information exchange between different potential solutions to the problem.

The second classic operator applied to the selected chromosomes is the uniform mutation scheme. This asexual operator alters a randomly selected gene of a chromosome. The new gene takes its random value from the constrained space, determined in the beginning of the process. The mutation operator is used in order to introduce some extra variability into the population.

The resulting intermediate population is evaluated and a fitness function is assigned to each member of the population. Using a selection procedure (different for each type of EA) the best individuals of the intermediate population (or the best individuals of the intermediate and the previous population) will form the next generation. The process of a new generation evaluation and creation is successively repeated, providing individuals with high values of fitness function.

2.3 The Solid Boundary Representation

In the simulation results that will be presented the terrain where UAVs fly is represented by a meshed 3-D surface, produced using mathematical functions of the form

$$\begin{aligned} z\left(x,y\right) = {} & \sin\left(y+a\right) + b\cdot\sin\left(x\right) + c\cdot\cos\left(d\cdot\sqrt{x^2+y^2}\right) \\ & + e\cdot\cos\left(y\right) + f\cdot\sin\left(f\cdot\sqrt{x^2+y^2}\right) + g\cdot\cos\left(y\right), \end{aligned} \tag{7}$$

where a, b, c, d, e, f, g are constants experimentally defined, in order to produce either a surface with mountains and valleys (as shown in Fig. 2) or a maritime environment with islands close to each other (as shown in Fig. 6).

A graphical interface has been developed for the visualization of the terrain surface, along with the path lines [23]. The corresponding interface deals with different terrains produced either artificially or based on real geographical data, providing an easy verification of the feasibility and the quality of each solution. The path-planning algorithm considers the boundary surface as a group of quadratic mesh nodes with known coordinates.

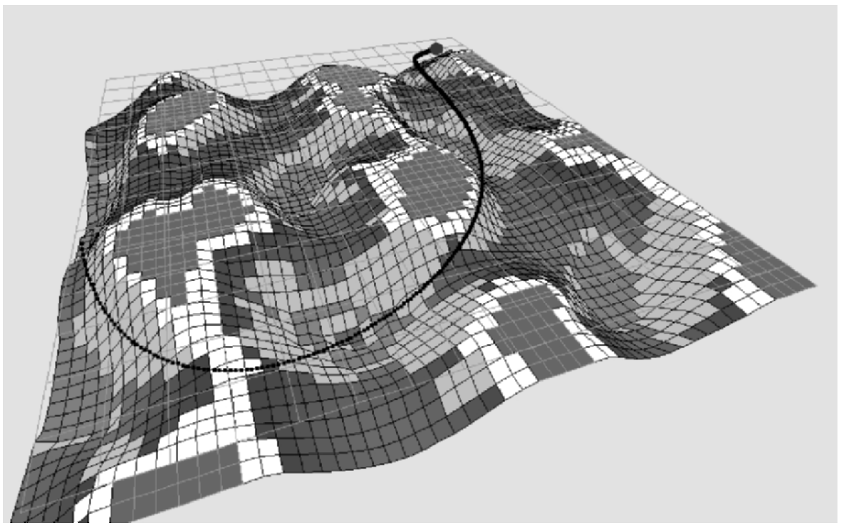

Fig. 2. A typical simulation result of the off-line path planner for a single UAV; the horizontal section of the terrain represents the imposed upper limit to the UAV flight. The starting position is marked with a circle

3 Off-line Path Planner for a Single UAV

The off-line path planner, discussed in detail in [23], will be briefly presented here, in order to introduce the concept of UAV path planning using Evolutionary Algorithms. The off-line planner generates collision free paths in environments with known characteristics and flight restrictions. The derived path line is a single continuous 3-D B-Spline curve, while the solid boundaries are interpreted as 3-D rough surfaces. The starting and ending control points of the B-Spline curve are fixed. A third point close to the starting one is also fixed, determining the initial flight direction. Between the fixed control points, free-to-move control points determine the shape of the curve, taking values in the constrained space. The number of the free-to-move control points is user-defined. Their physical coordinates are the genes of the EA artificial chromosome.

The optimization problem to be solved minimizes a set of four terms, connected to various objectives and constraints; they are associated with the feasibility and the length of the curve, a safety distance from the obstacles and the UAV's flight envelope restrictions. The objective function to be minimized is defined as

$$f = \sum_{i=1}^{4} w_i f_i. \tag{8}$$

Term f_1 penalizes the non-feasible curves that pass through the solid boundary. The penalty value is proportional to the number of discretized curve points located inside the solid boundary; consequently, non-feasible curves with fewer points inside the solid boundary show better fitness than curves with more points inside the solid boundary. Term f_2 is the length of the curve (non-dimensional with the distance between the starting and destination points) used to provide shorter paths.

Term f_3 is designed to provide flight paths with a safety distance from solid boundaries

$$f_3 = \sum_{i=1}^{nline} \sum_{j=1}^{nground} 1/\left(d_{i,j}/d_{safe}\right)^2 , \tag{9}$$

where *nline* is the number of discrete curve points, *nground* is the number of discrete mesh points of the solid boundary, $d_{i,j}$ is the distance between the corresponding nodes and curve points, while d_{safe} is a safety distance from the solid boundary. Term f_4 is designed to provide curves with a prescribed minimum curvature radius [23]. Weights w_i are experimentally determined, using as criterion the almost uniform effect of the last three terms in the objective function. Term $w_1 f_1$ has a dominant role in Eq. 8 providing feasible curves in few generations, since path feasibility is the main concern. The minimization of Eq. 8, through the EA procedure, results in a set of B-Spline control points, which actually represent the desired path.

Initially, the starting and ending path-line points are determined, along with the direction of flight. The limits of the physical space, where the vehicle is allowed to fly (upper and lower limits of their Cartesian coordinates), are also determined, along with the ground surface. The determined initial flight direction is used to compute the third fixed point close to the starting one; its position is along the flight direction and at a pre-fixed distance from the starting point.

The EA randomly produces a number of chromosomes to form the initial population. Each chromosome contains the physical coordinates of the free-to-move B-Spline control points. Using Eqs. 1 to 6, with a constant step of parameter u, a B-Spline curve is calculated for each chromosome of the population in the form of a sequence of discrete points. Subsequently, each B-Spline is evaluated, using the aforementioned criteria, and its objective function is calculated. Using the EA procedure, the population of candidate solutions evolves during the generations; at the last generation the population member with the smallest value of objective function is the solution to the problem and corresponds to the path line with the best characteristics according to the aforementioned criteria.

The simulation runs have been designed in order to search for path lines between “mountains”. For this reason, an upper ceiling for flight height has been enforced, which is represented in the graphical environment by the horizontal section of the terrain. A typical simulation result is demonstrated in Fig. 2.

4 Coordinated UAV Path Planning

This section describes the development and implementation of an off-line path planner for Unmanned Aerial Vehicles (UAVs) coordinated navigation and collision avoidance in known static maritime environments. The problem formulation is described, including assumptions, objectives, constraints, objective function definition and path modeling.

4.1 Constraints and Objectives

The path planner was designed for navigation and collision avoidance of a small team of autonomous UAVs in maritime environments. Known and static environments are considered, characterized by the existence of a number of islands with short distances between them. The flight height is assumed to be almost constant, close to the sea-level, and the path-planning problem is formulated as a 2-D one. Having N UAVs launched from different known initial locations, the issue is to produce N 2-D trajectories, formed by B-Spline curves, with a desirable velocity distribution along each trajectory, aiming at reaching a predetermined target location, while ensuring collision avoidance either with the environmental obstacles or with the UAVs. Additionally the produced flight paths should satisfy specific route and coordination objectives and constraints. Each vehicle is assumed to be a point, while its actual size is taken into account by equivalent obstacle – ground growing.

The general constraint of the problem is the collision avoidance between UAVs and the ground. The route constraints are:

(a) Predefined initial and target coordinates for all UAVs
(b) Predefined initial and final velocity magnitudes for all UAVs, and
(c) Predefined minimum and maximum UAV velocity magnitudes during their flights.

Additionally, a single route objective is imposed: minimum path lengths, for maximizing the effective range of each vehicle. All three route constraints are explicitly taken into account by the optimization algorithm. The route objective is implicitly handled by the algorithm, through the definition of the objective function.

Besides route constraints and objective, coordination-relative constraints and objectives are imposed, which are implicitly handled by the algorithm, through the objective function definition. The coordination objectives used in this work are the following:

(a) Each UAV should arrive at the target, using a different path and a different approach vector, but the time of arrival for all UAVs should be as close as possible.

(b) Approaching the target from different directions. All angles between successive approaching directions should be as equal as possible, in order to assure an almost uniform distribution of UAVs around the target during their approach, for maximizing the probability of mission accomplishment.

The single coordination constraint is defined as keeping a minimum safety distance between UAVs, in order to ensure:

(a) collision avoidance between UAVs, and
(b) a spatial separation between the corresponding flight corridors, which, for some missions, increases the probability of survival.

4.2 Path Modeling Using B-Spline Curves

In this work each path is constructed using a B-Spline curve. Although the resulting curve in the physical space should be a 2-D one, 3-D B-Spline curves are utilized for the construction of each path. The two dimensions are used for the production of the x, y coordinates in the physical space of motion (horizontal plane), while the 3^{rd} dimension corresponds to the velocity c along the path. For this reason, each B-Spline control point is defined by 3 numbers, corresponding to $x_{k,j}, y_{k,j}, c_{k,j}(k = 0, \ldots, n, j = 1, \ldots, N, N$ being the number of UAVs, while $n + 1$ is the number of control points in each B-Spline curve, the same for all curves). In this way a smooth variation of velocity c is defined along the path. The first $(k = 0)$ and last $(k = n)$ control points of the control polygon are the initial and target points of the j^{th} UAV, which are predefined by the user. The corresponding velocities $c_{0,j}, c_{n,j}$ (launch and approaching velocities) are also predefined by the user.

The control polygon of each B-Spline curve is defined by successive straight line segments. For each segment, its length $seg\_length_{k,j}$, and its direction $seg\_angle_{k,j}$ are used as design variables $(k = 1, \ldots, n-1, j = 1, \ldots, N)$. Design variables $seg\_angle_{k,j}$ are defined as the difference between the direction (in deg.) of the current segment and the previous one. For the first segment of each control polygon $seg\_angle_{1,j}$ is measured from x-axis. Additionally, the UAVs' velocities $c_{k,j}$ at each control point are used as design variables, except for the starting and target points (where they are predefined).

Using $seg\_length_{k,j}$ and $seg\_angle_{k,j}$ the coordinates of each B-Spline control point $x_{k,j}$ and $y_{k,j}$ can be easily calculated. The use of $seg\_length_{k,j}$ and $seg\_angle_{k,j}$ as design variables instead of $x_{k,j}$ and $y_{k,j}$ was adopted for two reasons. The first reason is the fact that abrupt turns of each flight path can be easily avoided by explicitly imposing short lower and upper bounds for the $seg\_angle_{k,j}$ design variables. The second reason is that by using the proposed design variables a better convergence rate was achieved compared to the case with the B-Spline control points' coordinates as design variables. The latter observation is a consequence of the shortening of the search space, using the proposed formulation. The lower and upper boundaries of each independent design variable are predefined by the user. Velocity boundaries depend on the flight envelope of each UAV. For the first segment of each control polygon $seg\_angle_{1,j}$ upper and lower boundaries can be selected such as to define an initial flight direction. Additionally, by selecting lower and upper boundaries for the rest of $seg\_angle_{k,j}$ variables close to 0 degrees (for example $-30°$ to $30°$), abrupt turns may be avoided.

4.3 Objective Function Formulation

The optimum flight path calculation for each UAV is formulated as a minimization problem. The objective (cost) function to be minimized is formulated as the weighted sum of five different terms

$$f = \sum_{i=1}^{5} w_i f_i, \tag{10}$$

where w_i are the weights and f_i are the corresponding terms described below.

Term f_1 corresponds to the single route objective of short flight paths and is defined as the sum of the non-dimensional lengths of all N flight paths (B-Spline curves)

$$f_1 = \sum_{j=1}^{N} l_j, \tag{11}$$

where l_j is the non-dimensional length of the j^{th} path, given as

$$l_j = \frac{L_j}{\sqrt{\left(x_{target} - x_{0,j}\right)^2 + \left(y_{target} - y_{0,j}\right)^2}} - 1. \tag{12}$$

In Eq. 12 L_j is the length of the j^{th} path, x_{target}, y_{target} are the coordinates of the target point and $x_{0,j}$, $y_{0,j}$ are the coordinates of the j^{th} starting point. In Eq. 12, for the calculation of the non-dimensional length l_j, the distance between the starting and target points is subtracted, in order to obtain zero f_1 value for straight line paths.

Term f_2 is a penalty term, designed in order to materialize the general constraint of collision avoidance between UAVs and the ground. All N flight paths are checked whether or not pass through each one of the M ground obstacles. Discrete points are taken along each B-Spline path and they are checked whether or not they lie inside an obstacle. If this is true for a discrete point of the path line, a constant penalty is added to term f_2. Consequently, term f_2 is proportional to the number of discrete points that lie inside obstacles. Additionally, for each path line, a high penalty is added in case that even one discrete point of the corresponding path lies inside an obstacle.

Term f_3 was designed in order to take into account the second coordination objective, i.e. the target approach from different directions. For each flight

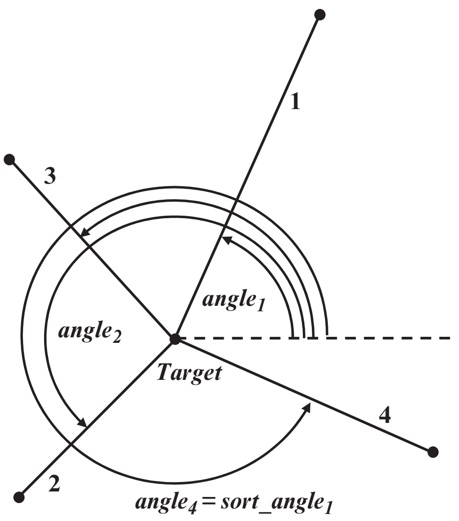

Fig. 3. Definition of azimuth angles, calculated for the last control polygon segment of each flight path

path the opposite to the flight direction azimuth angle of the last B-Spline control polygon segment is calculated as (Fig. 3)

$$angle_j = \begin{cases} \arctan(\Delta y/\Delta x) & if \quad \Delta y \geq 0 \; and \; \Delta x \geq 0 \\ 2\pi - \arctan(\Delta y/\Delta x) & if \quad \Delta y < 0 \; and \; \Delta x \geq 0 \\ \pi + \arctan(\Delta y/\Delta x) & if \quad \Delta x < 0 \end{cases} \tag{13}$$

$$\Delta y = y_{n-1,j} - y_{n,j}, \;\; \Delta x = x_{n-1,j} - x_{n,j}.$$

All calculated azimuth angles $angle_j$, $(j = 1, \ldots, N)$ are sorted in a descending order and stored as variables $sort\_angle_j$. An additional variable $sort\_angle_{N+1}$ is calculated as

$$sort\_angle_{N+1} = sort\_angle_1 - 2\pi. \tag{14}$$

Subsequently, the deference between two successive $sort\_angle_j$ is calculated as

$$\Delta sort\_angle_j = sort\_angle_j - sort\_angle_{j+1}, \;\; j = 1, \ldots, N, \tag{15}$$

where $\Delta sort\_angle_j$ is the angle between two successive flight paths, connected to the target point (Fig. 4). We define opt_angle as

$$opt\_angle = 2\pi/N. \tag{16}$$

Variable opt_angle denotes the optimum angle between successive B-Spline flight paths as UAVs are approaching the target, in order to have uniform distribution of UAVs around the target.

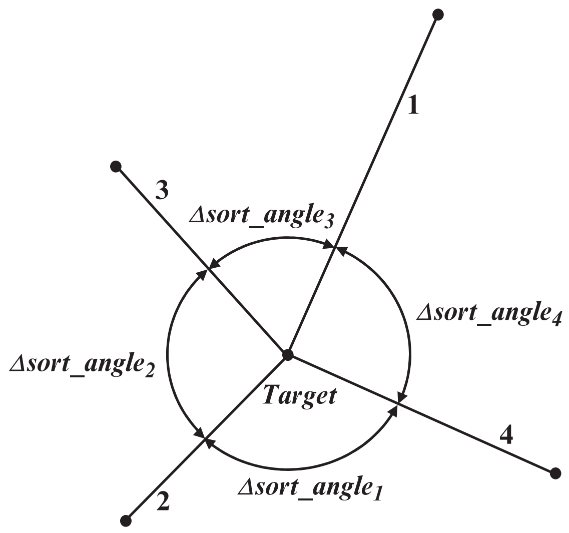

Fig. 4. Definition of $\Delta sort\_angle_j$

Term f_3 is then calculated as:

$$f_3 = \frac{\sum_{j=1}^{N} |opt\_angle - \Delta sort\_angle_j|}{ref\_angle}. \tag{17}$$

In Eq. 17, ref_angle is a small reference angle which is used to provide a non-dimensional form of f_3 and takes a value equal to $\pi/20$.

Term f_4 is relevant to the single coordination constraint (keep a safety distance between UAVs), while term f_5 is relevant to the first coordination objective (arrival at target with minimum time intervals). For their calculation, a flight simulation is needed. Each candidate solution is defined by the corresponding design variables. Then the coordinates of all B-Spline control points are computed, while the coordinates and the velocities at the starting and target points are predefined by the user. Assuming a simultaneous launching of all UAVs at $t = 0$, a simulation of their flights is performed. According to B-Spline theory [36, 37], each curve is constructed in the physical space by giving specific values to the u parameter in the parametric space. Taking a constant increment of u, discrete points are computed along each curve, with the coordinates and velocity provided by the B-Spline function. Having the x, y coordinates and the UAV velocity in each discrete point, the time needed by the UAV to reach the next point can be easily computed. In this way, starting from the initial point at $t = 0$, a time of arrival can be assigned to each discrete point along each path. The time of arrival to the target for each UAV is stored in variable t_curr_j.

Taking a constant time step, linear interpolations between successive discrete points are performed, and the position of each UAV is calculated for a specific time step. Subsequently, the distances between all UAVs are calculated in each time step and in case that a distance is less than a predefined safety distance d_{safe}, a penalty is added to term f_4.

Term f_5 is calculated as

$$f_5 = \sum_{j=1}^{N} \left(t\_\max - t\_curr_j\right) / t\_\max \tag{18}$$

where t_max is the time of arrival of the last UAV. As the main objective is to obtain feasible paths, weights in Eq. 10 were experimentally determined in order term $w_2 f_2$ dominate the rest.

5 The Optimization Procedure

5.1 Differential Evolution Algorithm

In this work, Differential Evolution (DE) [40, 41] is used as the optimization tool. DE is an extremely simple to implement EA, which has demonstrated better convergence performance than other EAs. Differential Evolution algorithm represents a type of Evolutionary Strategy, especially formed in such a way, so that it can effectively deal with continuous optimization problems, often encountered in engineering design, being a recent development in the field of optimization algorithms. The classic DE algorithm evolves a fixed size population, which is randomly initialized. After initializing the population, an iterative process is started and at each iteration (generation), a new population is produced until a stopping condition is satisfied. At each generation, each element of the population can be replaced with a new generated one. The new element is a linear combination between a randomly selected element and a difference between two other randomly selected elements. Below a more analytical description of the algorithm's structure is presented.

Given an objective function

$$f_{obj}\left(X\right) : \mathrm{R}^{n_{param}} \to \mathrm{R}, \tag{19}$$

the optimization target is to minimize the value of this objective function by optimizing the values of its parameters (design variables)

$$X = \left(x_1, x_2, \ldots, x_{n_{param}}\right), \; x_j \in \mathrm{R}, \tag{20}$$

where X denotes the vector composed of n_{param} objective function parameters (design variables). These parameters take values between specific upper and lower bounds

$$x_j^{(L)} \le x_j \le x_j^{(U)}, \; j = 1, \ldots, n_{param}. \tag{21}$$

The DE algorithm implements real encoding for the values of the objective function's parameters. In order to obtain a starting point for the algorithm, an initialization of the population takes place. Often the only information available is the boundaries of the parameters. Therefore the initialization is established by randomly assigning values to the parameters within the given boundaries

$$x_{i,j}^{(0)} = r \cdot \left(x_j^{(U)} - x_j^{(L)} \right) + x_j^{(L)}, \;\; i = 1, \ldots, n_{pop}, \;\; j = 1, \ldots, n_{param}, \qquad (22)$$

where r is a uniformly distributed random value within range [0, 1]. DE's mutation operator is based on a triplet of randomly selected different individuals. A new parameter vector is generated by adding the weighted difference vector between the two members of the triplet to the third one (the donor). In this way a perturbed individual is generated. The perturbed individual and the initial population member are then subjected to a crossover operation that generates the final candidate solution

$$x_{i,j}^{\prime(G+1)} = \begin{cases} x_{C_i,j}^{(G)} + F \cdot \left(x_{A_i,j}^{(G)} - x_{B_i,j}^{(G)} \right) \; if \;\; (r \le C_r \vee j = k) \; \forall j = 1, \ldots, n_{param} \\ x_{i,j}^{(G)} \qquad\qquad otherwise\,, \end{cases} \qquad (23)$$

where $x_{C_i,j}^{(G)}$ is called the "donor", G is the current generation,

$$\begin{array}{l} i = 1, \ldots, n_{pop}, \;\; j = 1, \ldots, n_{param} \\ A_i \in [1, \ldots, n_{pop}]\,, \;\; B_i \in [1, \ldots, n_{pop}]\,, C_i \in [1, \ldots, n_{pop}] \\ A_i \ne B_i \ne C_i \ne i \\ C_r \in [0, 1]\,, \;\; F \in [0, 1+]\,, \;\; r \in [0, 1] \;, \end{array} \qquad (24)$$

and k a random integer within [1, n_{param}], chosen once for all members of the population. The random number r is seeded for every gene of each chromosome. F and C_r are DE control parameters, which remain constant during the search process and affect the convergence behaviour and robustness of the algorithm. Their values also depend on the objective function, the characteristics of the problem and the population size.

The population for the next generation is selected between the current population and the final candidates. If each candidate vector is better fitted than the corresponding current one, the new vector replaces the vector with which it was compared. The DE selection scheme is described as follows (for a minimization problem)

$$X_i^{(G+1)} = \begin{cases} X_i^{\prime(G+1)} \;\; if \;\; f_{obj}\left(X_i^{\prime(G+1)} \right) \le f_{obj}\left(X_i^{(G)} \right) \\ X_i^{(G)} \qquad\qquad otherwise\,. \end{cases} \qquad (25)$$

A new scheme [42] to determine the donor for mutation operation is used, for accelerating the convergence rate. In this scheme, donor is randomly selected (with uniform distribution) from the region within the "hyper-triangle", formed by the three members of the triplet. With this scheme the donor comprises the local information of all members of the triplet, providing a better starting-point for the mutation operation that result in a better distribution of the trial-vectors. As it is reported in [42], the modified donor scheme accelerated the DE convergence rate, without sacrificing the solution precision or robustness of the DE algorithm.

The random number generation (with uniform probability) is based on the algorithm presented in [43], which computes the remainder of divisions involving integers that are longer than 32 bits, using 32-bit (including the sign bit) words. The corresponding algorithm, using an initial seed, produces a new seed and a random number. In each different operation inside the DE algorithm that requires a random number generation, a different sequence of random numbers is produced, by using a different initial seed for each operation and a separate storage of the corresponding produced seeds. By using specific initial seeds for each operation, it is ensured that the different sequences differ by 100,000 numbers.

5.2 Radial Basis Function Network for DE Assistance

Despite their advantages, EAs ask for a considerable amount of evaluations. In order to reduce their computational cost several approaches have been proposed, such as the use of parallel processing, the use of special operators and the use of surrogate models and approximations. Surrogate models are auxiliary simulations that are less physically faithful, but also less computationally expensive than the expensive simulations that are regarded as "truth". Surrogate approximations are algebraic summaries obtained from previous runs of the expensive simulation [44, 45]. Such approximations are the low-order polynomials used in Response Surface Methodology [46, 47], the kriging estimates employed in the design and analysis of computer experiments [48], and the various types of Artificial Neural Networks [45]. Once the approximation has been constructed, it is typically inexpensive to use.

DE has been demonstrated to be one of the most promising novel EAs, in terms of efficiency, effectiveness and robustness. However, its convergence rate is still far from ideal, especially when it is applied in optimization problems with time consuming objective functions. In order to enhance the convergence rate of DE algorithm, an approximation model is used for the objective function, based on a Radial Basis Functions Artificial Neural Network [49]. In general a RBFN (Fig. 5), is a three layer, fully connected feed-forward network, which performs a nonlinear mapping from the input space to the hidden space ($\mathrm{R}^L \rightarrow \mathrm{R}^M$), followed by a linear mapping ($\mathrm{R}^M \rightarrow \mathrm{R}^1$) from the hidden to the output space (L is the number of input nodes, M is the number of hidden nodes, while the output layer has a single node).

The corresponding output *yy(xx)*, for an input vector xx=[xx_1, xx_2,…,xx_L] is given

$$yy\,(xx) = \sum_{i=1}^{M} w_i \cdot \varphi_i\,(xx). \tag{26}$$

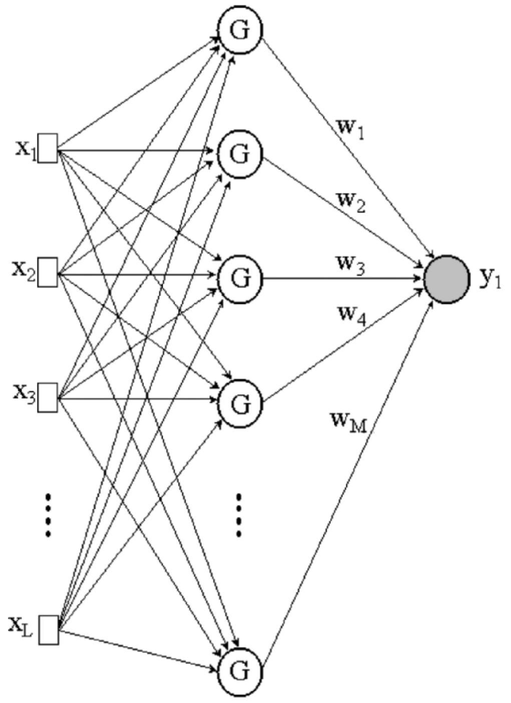

Fig. 5. A Radial Basis Function Artificial Neural Network

where $\varphi_i(xx)$ is the output of the i^{th} hidden unit

$$\varphi_i\,(xx) = G\left(\|xx - cc_i\|\right),\ i = 1,\ldots,M. \tag{27}$$

The connections (weights) to the output unit (w_i, i=1,...,M) are the only adjustable parameters. The RBFN centers in the hidden units cc_i, i=1,...,M are selected in a way to maximize the generalization properties of the network. The nonlinear activation function G in our case is chosen to be the Gaussian radial basis function

$$G\,(u,\sigma) = \exp\left(-u^2/\sigma^2\right), \tag{28}$$

where σ is the standard deviation of the basis function.

The selection of RBFN centers plays an important role for the predictive capabilities and the generalization of the network. There are several strategies that can be adopted concerning the selection of the radial-basis functions centers in the hidden layer, while designing a RBFN. Haykin refers to the following [49]: a) Random selection of fixed centers, which is the simplest approach and the selection of centers from the training data set is a sensible choice, given that the latter is adequately representative for the problem at hand. b) Self-organized selection of centers, where appropriate locations for the centers are estimated with the use of a clustering algorithm whose assignment is to partition the training set in homogeneous subsets. c) Supervised selection of centers, which is the most generalized form of a RBFN since the location of the centers undergo a supervised learning process along with the rest of the network's free parameters.

The standard process is to select the input vectors in the training set as RBFN centers. In this case results $M{=}NR$, where NR is the number of training data. For large training sets (resulting in large M values) this choice is expected to increase storage requirements and CPU cost. Additionally, the $M{=}NR$ choice could lead to over-fitting and/or bad generalization of the network. The proposed solution [49, 45] is the selection of $M{<}NR$ and consequently the search for sub-optimal solutions, which will provide a better generalizing capability to the network.

As far as training is concerned, there are two different approaches, the direct and the iterative learning. In our case the first approach was adopted. The direct learning process is based on a matrix formulation of the governing equations of RBF network. The presentation of the network with the NR input patterns allows the formulation of a $(NR \times M)$ matrix H, which becomes square in the special case when $NR{=}M$. Each line in the interpolation matrix H corresponds to a learning example and each column to a RBFN center. The output unit values result in the form of the matrix product:

$$H\,(NR \times M)\,w\,(M \times 1) = yy\,(NR \times 1)\,, \tag{29}$$

where yy is the desired output vector as provided by the training dada set, and w is the synaptic weights vector, which consists of M unknowns to be computed.

A possible way for inverting H is through the Gram-Schmidt technique. H is first decomposed as

$$H = Q\,R, \tag{30}$$

with Q and R being $(NR \times M)$ and $(M \times M)$ matrices respectively, where R is upper triangular and

$$Q^{\mathrm{T}}Q = diag\,(1, 1, \ldots, 1)\,. \tag{31}$$

After the computation of Q and R matrices, the weights vector can be computed using back-substitution in

$$R(M \times M)w(M \times 1) = Q^T(M \times NR)yy(NR \times 1) \tag{32}$$

There are several reasons why one should choose RBFN as the approximation model; Haykin [49] offers comparative remarks for RBFNs and Multilayer Perceptrons (MLPs). However, the main reason for choosing RBFNs is their compatibility with the adopted local approximation strategy, as it is described in the subsequent section. The use of relatively small numbers of training patterns i.e. small networks, helps creating local range RBFNs. That in turn allows the inversion of matrix H to use almost negligible CPU time and the approximation error is kept very small. We should keep in mind that the computing cost associated with the use of neural networks is the cost of training the networks, whereas the use of a trained network to evaluate a new individual adds negligible computation cost [45].

5.3 Using RBFN for Accelerating DE Algorithm

In each DE generation, during the evaluation procedure, each trial vector must be evaluated and then compared with the corresponding current vector, in order to select the better-fitted between them to pass to the next generation. The concept is to replace the costly exact evaluations of trial vectors with fast inexact approximations, and at the same time maintain the robustness of the DE algorithm. During the evaluation phase, each trial vector is pre-evaluated, using the approximate model. If it is pre-evaluated as lower-fitted (higher objective function in minimization problems) than the corresponding vector of the current population, then no further exact evaluation is needed and the current vector is transferred to the next generation, while the trial vector is abandoned. In case the trial vector is pre-evaluated as better fitted than the corresponding current vector, then an exact re-evaluation takes place after the pre-evaluation, along with a new comparison between the two vectors. If the trial vector is still better-fitted than the current vector, then the trial vector passes to the next generation. Otherwise the current vector is the one that will pass to the next generation. Additionally, a small percentage of the candidate solutions, are selected with uniform probability to be exactly evaluated, without taking into account their performance provided by the approximation model. In the first two generations, all vectors are exactly evaluated. According to the afore mentioned procedure, only exactly evaluated trial vectors have the opportunity to pass to the new generation, so the current population always comprises exactly evaluated individuals. In this way, one part of the comparison (the current vector) is always an exact-evaluated vector, and this enhances the robustness of the procedure.

The result of each evaluation (exact or inexact), along with the corresponding chromosome, are stored in a database. In order to have a local approximation model, only the best-fitted individuals of database entries are used in each generation to re-train the RBFN. In this way the approximation model evolves with the population and uses only the useful information for approximating the objective function. The surrogate model predictions replace exact and costly evaluations only for the less-promising individuals, while the more-promising ones are always exactly evaluated.

6 Simulation Results

The same artificial environment was used for all the test cases considered, with different starting and target points. The (experimentally optimized) settings of the Differential Evolution algorithm were as follows: *population size* = 50,

$F = 0.99$, $C_r = 0.85$. The algorithm was defined to terminate after 700 generations, although feasible solutions can be reached in less than 30 generations. The large number of generations was used in order to compare the convergence behavior between the original DE algorithm and the RBFN assisted one. For the 4 test cases presented here, 3 free-to-move control points were used for each B-Spline path, resulting in a total number of control points equal to 5 for each B-Spline curve (along with the fixed starting and target points). For 3 different paths (corresponding to 3 UAVs) and 3 free-to-move control points for each path, a total number of 27 design variables are needed ($seg\_length_{k,j}$, $seg\_angle_{k,j}$ and $c_{k,j}$, for each path$j$ and each control point k).

Figures 6 to 9 present simulation results for the four different test cases, using the RBFN assisted DE. For all test cases safety distance d_{safe} was set equal to 12.5% of the length of each side of the rectangular terrain. For all test cases, term f_4 of the cost function converged to zero, indicating no violation of the safety distance constraint. Concerning the time intervals between the first and the last arrival to the target, for all the test cases considered this time interval was kept less than about 3% of the flight duration (0.71% for the 1st case, 3.08% for the 2nd case, 1.33% for the 3rd case and 1.41% for the 4th case). As it can be observed, term f_3 of the fitness function managed to produce uniform distribution of UAVs around the target for all cases considered. Even for the fourth test case a uniform distribution of UAV paths around the target was achieved, although the target point was positioned very close to an obstacle (island coast).

As it has been already stated, the main reason for introducing the RBFN surrogate model was to speed-up the optimization procedure. However, as

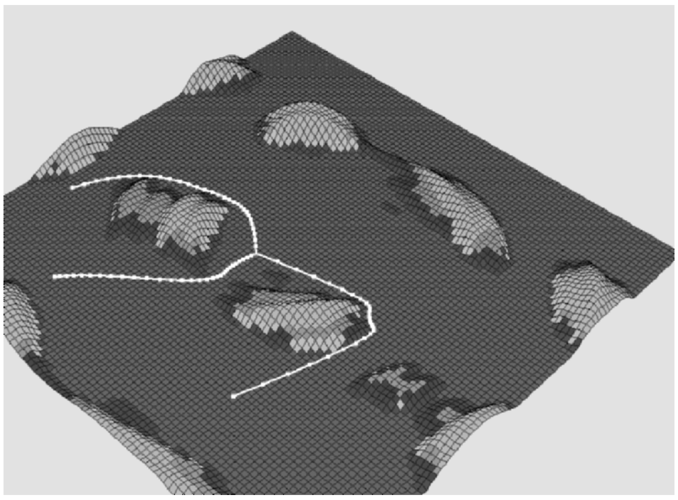

Fig. 6. The first test case for the coordinated UAV path planning

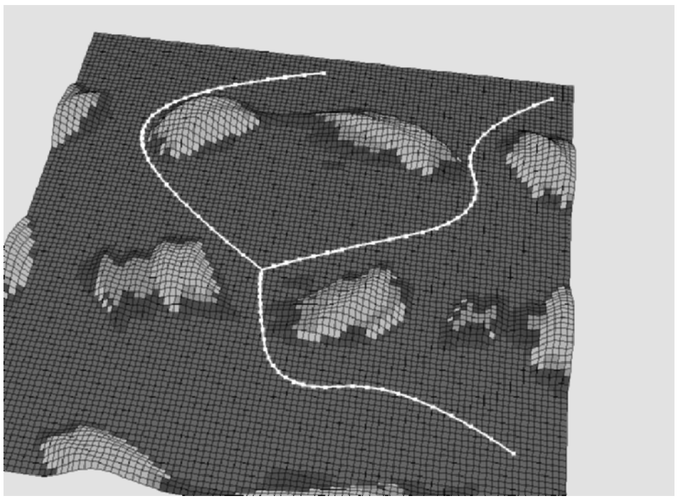

Fig. 7. The second test case for the coordinated UAV path planning

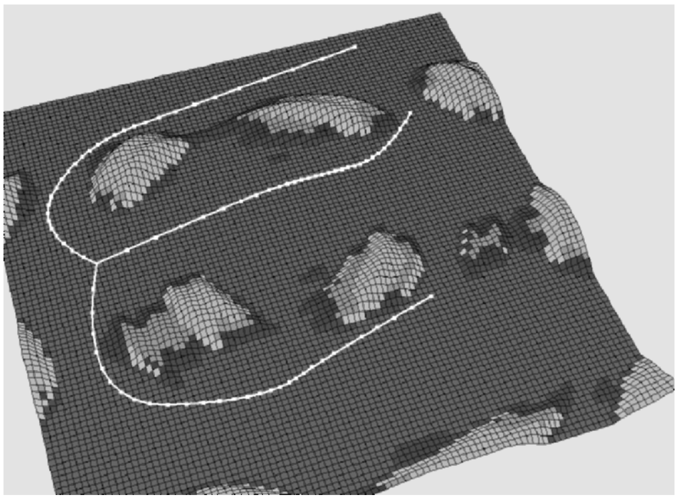

Fig. 8. The third test case for the coordinated UAV path planning

it was observed, the introduction of RBFN assistance resulted in a deeper convergence (better final value of fitness function), compared to the original DE. Both algorithms (the original DE and the RBFN assisted DE) were used in order to solve the path planning problem for the aforementioned four test cases, using the same parameters. In order to compare the effect of RBFN

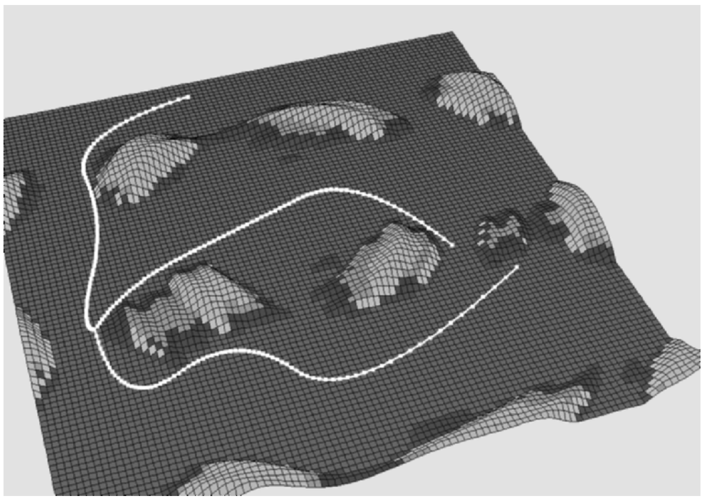

Fig. 9. The fourth test case for the coordinated UAV path planning

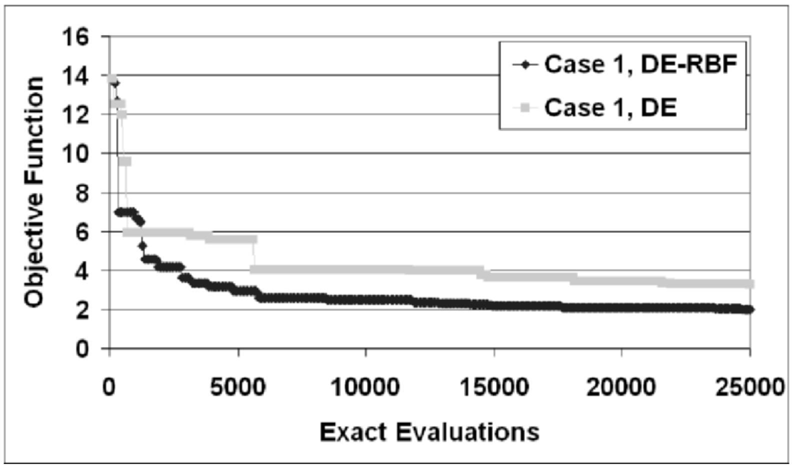

Fig. 10. Convergence histories for the first test case, with and without the use of the RBFN assistance

surrogate model, the convergence histories for the four test cases of the DE algorithm (with and without the RBFN assistance) are presented in Fig. 10 to 13. As it can be observed, the adoption of the approximation model resulted in a considerable reduction in the number of exact evaluations for a specific fitness value. This reduction, for all cases considered, reached or exceeded 80% of the number of exact evaluations with respect to the case without

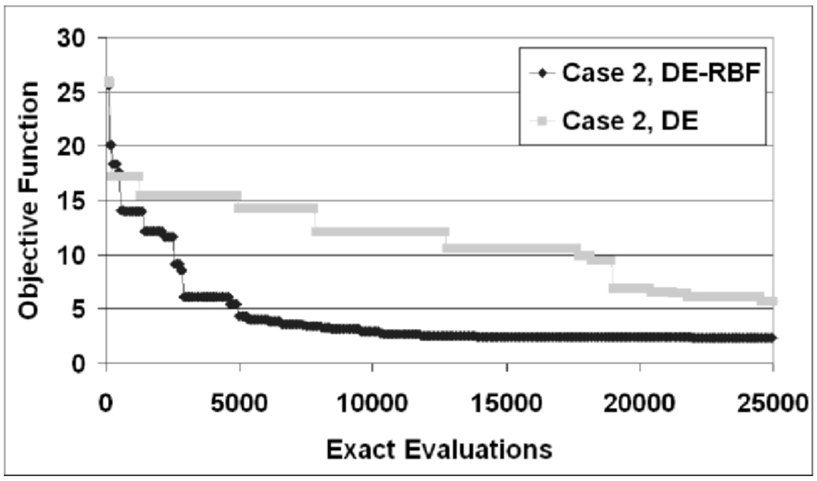

Fig. 11. Convergence histories for the second test case, with and without the use of the RBFN assistance

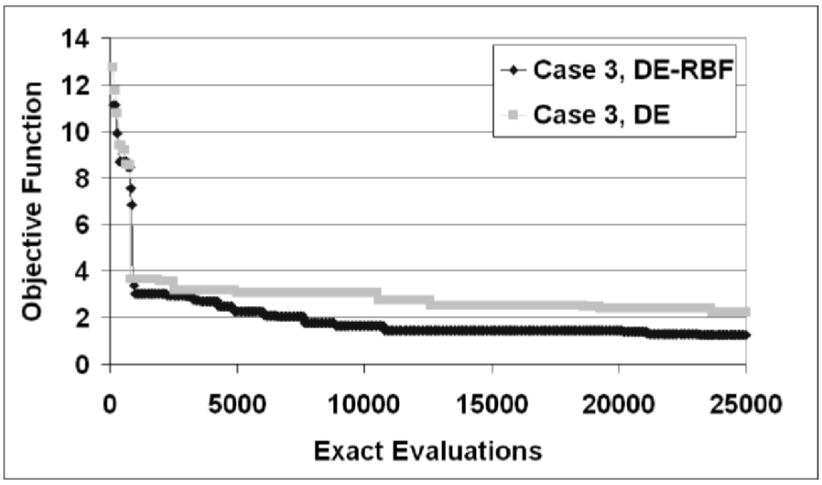

Fig. 12. Convergence histories for the third test case, with and without the use of the RBFN assistance

RBFN assistance. As the introduction of the RBFN has a minor effect on the computation time, this 80% reduction in the number of exact evaluations results in a speedup factor approximately equal to 5 to the whole computation procedure, which is very important for real world applications of such kind.

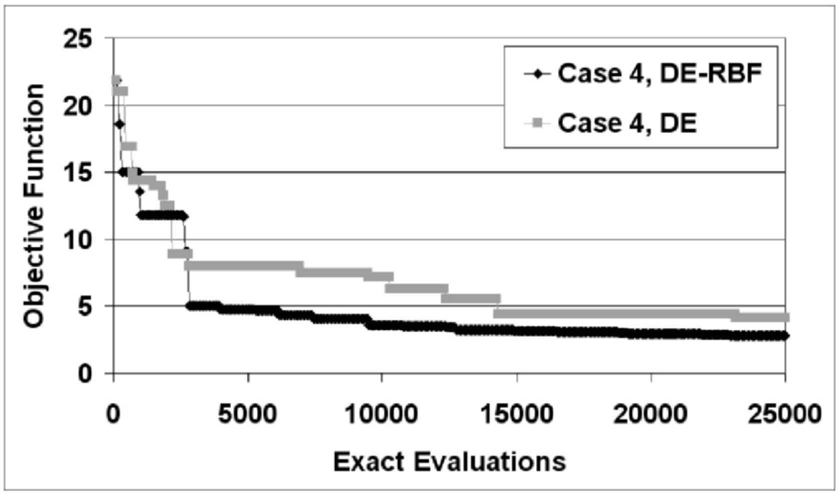

Fig. 13. Convergence histories for the fourth test case, with and without the use of the RBFN assistance

7 Conclusions

This work is an extension of a previous one, which used Differential Evolution in order to find optimal paths of coordinated UAVs, with the paths being modeled with straight line segments. Although very satisfactory results were achieved, the main drawback of the previous approach was the need of a large number of segments for complicated paths, resulting in a large number of design variables. However, as the number of design variables increases, the dimensionality of the optimization problem also increases; consequently, much more generations are needed for a converged solution, which is not always affordable for real world applications.

In this work an off-line path planner for UAVs coordinated navigation and collision avoidance in known static maritime environments was presented. The problem was formulated as a single-objective optimization one, with the objective function being the weighted sum of different terms, which correspond to various objectives and constraints of the problem. B-Spline curves were adopted in order to model the 2-D flight paths, as they provide the ability to produce complicated paths with a small number of control variables. In this way the number of design variables, and the dimensionality of the optimization problem, can be kept small. The velocity distribution along each flight path was also modeled using the B-Spline formulation. A Radial Basis Function Artificial Neural Network was introduced in the Differential Evolution algorithm (the optimizer) to serve as a surrogate model and decrease the number of costly exact evaluations of the objective function. The RBF Network managed to considerably reduce the DE computation time and to provide deeper convergence to the optimization procedure.

The path planner was tested in a simulated environment, and the simulation results demonstrated the ability of the algorithm to produce near optimal paths without violating the imposed constraints. The adoption of the B-Spline formulation provided the ability to keep the number of design variables as small as possible, and at the same time produce reasonable and smooth paths, without abrupt turns.

Future work will be focused on the development of an on-line path planner for coordination of a team of UAVs. The methodology that will be used for the planner will be a combination of this work and the work presented in [23], where an on-line path planner for a single UAV was presented, which is able to gradually produce B-spline paths in an unknown 3D environment.

7.1 Trends and challenges

The military market for UAVs has demonstrated a strong positive trend during the past decade, with the corresponding commercial market showing a similar behavior, although not so strong [1]. This trend is expected to continue, as the technology provides new solutions to the problems of autonomous navigation of UAVs, and new ideas are emerging about the roles and tasks that can be assigned to UAVs. The trend is supported by the large number of research teams that are working in the field. In particular, the field of UAV cooperation gained increased interest during the past years due to the advantages of using a team of UAVs instead of a single one to accomplish a complicated mission. However, because of the youth of the field, the research has been "scenario" oriented, and a rigorous formalism is still missing. Work is still needed in the direction of clarifying: a) the assumptions about the systems of cooperating UAVs, b) the terminology used to describe the various problems under consideration, c) the different categories of working scenarios, d) the objectives and constraints for each problem, e) the "best practices" that can be adopted for specific problems or sub-problems.

The research in the field of cooperating UAVs is highly interdisciplinary, and knowledge from different science and technology fields is needed, even from the beginning of the formulation of the problem under consideration. It would be highly beneficially for the researchers, especially for those working in more theoretical fields, to collaborate with possible users of the systems or methodologies under development. Some of the problems, relevant to UAV cooperation, that gain high interest are: a) the assignment of multiple tasks to a team of UAVs, b) the path planning problem of cooperating UAVs in the presence of various cooperation and mission constraints, c) information exchange and data fusion between the cooperating vehicles, d) cooperative sensing of targets, e) cooperative sensing of the environment, f) centralized versus distributed coordination methodologies, especially for cases with communication problems between the vehicles, g) on-line mission rescheduling and task reassignment for fault-tolerant systems.

References

1. Newcome, L.R.: Unmanned Aviation, a Brief History of Unmanned Aerial Vehicles. AIAA (2004)
2. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers (1991)
3. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
4. Gilmore, J.F.: Autonomous vehicle planning analysis methodology. Proceedings of the Association of Unmanned Vehicles Systems Conference. Washington, DC (1991) 503–509
5. Uny Cao, Y., Fukunaga, A.S., Kahng, A.B.: Cooperative Mobile Robotics: Antecedents and Directions. Autonomous Robots 4 (1997) 7-27
6. Fujimura, K.: Motion Planning in Dynamic Environments. Springer-Verlag, New York, NY, (1991)
7. Arai, T. and Ota, J. 1992. Motion planning of multiple robots. Proceedings of the IEEE/RSJ IROS (1992) 1761–1768
8. Shima, T., Rasmussen, S.J., Sparks, A.G.: UAV Cooperative Multiple Task Assignments using Genetic Algorithms. Proceedings of the 2005 American Control Conference, June 8-10, Portland, OR, USA (2005)
9. Shima, T., Rasmussen, S.J., Sparks, A.G.: UAV Team Decision and Control using Efficient Collaborative Estimation. Proceedings of the 2005 American Control Conference, June 8-10, Portland, OR, USA (2005)
10. Mitchell, J.W. and Sparks, A.G.: Communication Issues in the Cooperative Control of Unmanned Aerial Vehicles. Proceedings of the Forty-First Annual Allerton Conference on Communication, Control, & Computing (2003)
11. Schumacher, C.: Ground Moving Target Engagement by Cooperative UAVs. Proceedings of the 2005 American Control Conference, June 8-10, Portland, OR, USA (2005)
12. Moitra, A., Mattheyses, R.M., Hoebel, L.J., Szczerba, R.J., Yamrom, B.: Multivehicle reconnaissance route and sensor planning. IEEE Transactions on Aerospace and Electronic Systems, 37 (2003) 799–812
13. Bortoff, S.: Path planning for UAVs. Proceedings of the Amer. Control Conf., Chicago, IL, (2000) 364–368
14. Szczerba, R.J., Galkowski, P., Glickstein, I.S., and Ternullo, N.: Robust algorithm for real-time route planning. IEEE Transactions on Aerospace Electronic Systems 36 (2000) 869–878
15. Zheng, C., Li, L., Xu, F., Sun, F., Ding, M.: Evolutionary Route Planner for Unmanned Air Vehicles. IEEE Transactions on Robotics 21 (2005) 609–620
16. Beard, R.W., McLain, T.W., Goodrich, M.A., Anderson, E.P.: Coordinated target assignment and intercept for unmanned air vehicles. IEEE Transactions on Robotics and Automation, 18 (2002) 911–922
17. Vandapel, N., Kuffner, J., Amidi, O.: Planning 3-D Path Networks in Unstructured Environments. Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2005)
18. Dubins, L.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position. American Journal of Math. 79 (1957) 497–516.
19. Shima, T., Schumacher, C.: Assignment of cooperating UAVs to simultaneous tasks using Genetic Algorithms. AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco (2005)

20. Tang, Z., and Ozguner, U.: Motion Planning for Multi-Target Surveillance with Mobile Sensor Agents. IEEE Transactions on Robotics 21 (2005) 898-908
21. Martinez-Alfaro H., and Gomez-Garcia, S.: Mobile robot path planning and tracking using simulated annealing and fuzzy logic control. Expert Systems with Applications 15 (1988) 421–429
22. Nikolos, I.K., Tsourveloudis, N., and Valavanis, K.P.: Evolutionary Algorithm Based 3-D Path Planner for UAV Navigation. CD-ROM Proceedings of the 9th Mediterranean Conference on Control and Automation, Dubrovnik, Croatia (2001)
23. Nikolos, I.K., Valavanis, K.P., Tsourveloudis, N.C., Kostaras, A.: Evolutionary Algorithm based offline / online path planner for UAV navigation. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 33 (2003) 898–912
24. Mettler, B., Schouwenaars, T., How, J., Paunicka, J., and Feron E.: Autonomous UAV guidance build-up: Flight-test Demonstration and evaluation plan. Proceedings of the AIAA Guidance, Navigation, and Control Conference, AIAA-2003-5744 (2003)
25. Richards, A., Bellingham, J., Tillerson, M., and How., J.: Coordination and control of UAVs. Proceedings of the AIAA Guidance, Navigation and Control Conference, Monterey, CA, (2002)
26. Schouwenaars, T., How, J., and Feron, E.: Decentralized Cooperative Trajectory Planning of multiple aircraft with hard safety guarantees. Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA-2004-5141 (2004)
27. Flint, M., Polycarpou, M., and Fernandez-Gaucherand, E.: Cooperative Control for Multiple Autonomous UAV's Searching for Targets. Proceedings of the 41st IEEE Conference on Decision and Control (2002)
28. Gomez Ortega, J., and Camacho, E.F.: Mobile Robot navigation in a partially structured static environment, using neural predictive control. Control Eng. Practice 4 (1996) 1669–1679
29. Kwon, Y.D., and Lee, J.S.: On-line evolutionary optimization of fuzzy control system based on decentralized population. Intelligent Automation and Soft Computing 6 (2000) 135–146
30. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer Publications (1999)
31. Smierzchalski, R.: Evolutionary trajectory planning of ships in navigation traffic areas. Journal of Marine Science and Technology 4 (1999) 1–6
32. Smierzchalski, R., and Michalewicz Z.: Modeling of ship trajectory in collision situations by an evolutionary algorithm. IEEE Transactions on Evolutionary Computation 4 (2000) 227–241
33. Sugihara, K., and Smith, J.: Genetic Algorithms for Adaptive Motion Planning of an Autonomous Mobile Robot. Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, California (1997) 138–143
34. Sugihara, K., and Yuh, J.: GA-based motion planning for underwater robotic vehicles. UUST-10, Durham, NH (1997)
35. Nikolos, I.K., Brintaki, A.: Coordinated UAV Path Planning Using Differential Evolution. Proceedings of the 13th Mediterranean Conference on Control and Automation, IEEE, Limassol, Cyprus (2005)

36. Piegl, L., Tiller, W.: The NURBS Book. Springer (1997)
37. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide. Academic Press (1988)
38. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)
39. Holland, J.H.: Adaptation in Natural and Artificial Systems. The MIT Press (1992)
40. Storn, R., and Price, K.: DE - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Space. ICSI, Technical Report TR-95-012 (1995)
41. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution, a Practical Approach to Global Optimization. Springer-Verlag, Berlin Heidelberg (2005)
42. Hui-Yuan F., Lampinen J., Dulikravich G.S.: Improvements to Mutation Donor Formulation of Differential Evolution. Proceedings of EUROGEN 2003 conference on Evolutionary Methods for Design, Optimization and Control, Applications to Industrial and Societal Problems, CIMNE, Barcelona (2003)
43. Marse, K. and Roberts, S.D.: Implementing a portable FORTRAN uniform (0,1) generator. Simulation (1983) 41–135
44. Torczon, V., Trosset, M.W.: Using Approximations to Accelerate Engineering Design Optimization. NASA/CR-1998-208460, ICASE Report No. 98-33 (1998)
45. Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. Progress in Aerospace Sciences 38 (2002) 43–76
46. Myers, R.H., Montgomery, D.C.: Responce Surface Methodology: Progress and Product in Optimization Using Designed Experiments. Wiley – Interscience, New York (1995)
47. Shyy, W., Papila, N., Vaidynathan, R., Tucker, K.: Global Design Optimization for Aerodynamics and Rocket Propulsion Components. Prog. Aerospace Sci. 37 (2001) 59–118
48. Ratle, A.: Optimal Sampling Strategies for Learning a Fitness Model. Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), Washington DC, USA (1999)
49. Haykin, S.: Neural Networks, a Comprehensive Foundation. Second Edition, Prentice Hall (1999)