Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 560184, 16 pages
doi:10.1155/2012/560184

Research Article

Using Genetic Algorithms for Navigation Planning in Dynamic Environments

Ferhat Uçan[1,2] and D. Turgay Altılar[2]

[1] *Center of Research for Advanced Technologies of Informatics and Security (TÜBİTAK BILGEM), 41470 Kocaeli, Turkey*
[2] *Computer Engineering Department, Istanbul Technical University, 34469 Istanbul, Turkey*

Correspondence should be addressed to Ferhat Uçan, ferhat.ucan@bte.tubitak.gov.tr

Received 25 April 2012; Revised 29 July 2012; Accepted 31 July 2012

Academic Editor: Tzung P. Hong

Navigation planning can be considered as a combination of searching and executing the most convenient flight path from an initial waypoint to a destination waypoint. Generally the aim is to follow the flight path, which provides minimum fuel consumption for the air vehicle. For dynamic environments, constraints change dynamically during flight. This is a special case of dynamic path planning. As the main concern of this paper is flight planning, the conditions and objectives that are most probable to be used in navigation problem are considered. In this paper, the genetic algorithm solution of the dynamic flight planning problem is explained. The evolutionary dynamic navigation planning algorithm is developed for compensating the existing deficiencies of the other approaches. The existing fully dynamic algorithms process unit changes to topology one modification at a time, but when there are several such operations occurring in the environment simultaneously, the algorithms are quite inefficient. The proposed algorithm may respond to the concurrent constraint updates in a shorter time for dynamic environment. The most secure navigation of the air vehicle is planned and executed so that the fuel consumption is minimum.

1. Introduction

Navigation planning requires producing a flight plan to describe a proposed aircraft flight. It involves two safety-critical aspects: minimum fuel consumption and compliance with air traffic control requirements. Navigation planning involves creating a flight plan to guide a point-like object from its initial position to a destination waypoint [1]. Along the way, there may be a set of regions to visit and a set of regions to avoid. Planners wish to reach the destination economically and by minimum risk of mid-air collision. Fuel consumption involves fuel flow rate estimations, so that the relation of fuel flow with air temperature, flight altitude, true airspeed, and gross weight can be defined with an accurate formula [2]. Safety regulations require aircraft to carry fuel beyond the minimum needed to fly from origin to destination, allowing for unforeseen circumstances or for diversion to another airport if the planned destination becomes unavailable. Furthermore, under the supervision of air traffic control, aircraft flying in controlled airspace must follow predetermined routes known as airways, even if such routes are not as economical as a more direct flight [3, 4]. The basis of the flight profile is the route that the aircraft is to fly from the departure airport to the destination airport. Flight planning function provides for the assembly, modification, and activation of this route data known as a flight plan. Flight plans are normally constructed by linking data stored in the navigation database.

In this paper, by using the flight map and waypoint information, the aim is to calculate the most secure, shortest flight path and the guidance information in order to execute the flight plan by using flight data. For the real-time navigation planning problem, the change in the cost criteria like distance and security is related with the flight level. Flight level is a standard nominal altitude of the air vehicle [4]. This altitude is calculated from the international standard pressure datum, the average sea-level pressure, and therefore is not necessarily the same as the aircraft's true altitude either above mean sea level or above ground level.

Traditional methods used for the standard route planning problem are Dijkstra, Floyd Warshall, A-Star, and

Bellman Ford's algorithm [5]. When the constraints of the problem are dynamic and the environment is not stable, these methods are not valid. The effect of the constraints like security and distance may be modeled with the classical optimization techniques, but in a dynamic environment, restarting to find the solution at each graph update increases the operation complexity. Moreover, when new waypoints or routes are inserted or some of the existing routes are deleted, the classical algorithms cannot compensate this situation without starting the solution from the beginning. The proposed evolutionary method considers multiobjectives and compensate dynamic conditions due to the evolutionary operators and the problem-specific fitness function. The proposed evolutionary navigation planning approach reduces the number of operations in dynamic environments; because it does not restart the solution at each update, it trends to extend best-fit individuals into the new generations.

2. Evolutionary Methods Used for Path Planning in Literature

In recent years, evolutionary algorithms have been successfully applied to real-time task and path planning problems. The evolutionary based techniques are attractive for solving large-scale complex path planning problems because gradient information about objective functions and constraints are not needed during search for optimal solutions. Gradients usually do not exist for all feasible solutions in the search space. Another crucial advantage of evolutionary techniques is that they can eventually give the global optimal solutions for large-scale path planning problems.

Nikolos et al. made a research for 3D path planning of unmanned air vehicles [6]. In their study they used evolutionary algorithms for calculating the path curve according to the earth's surface in a 3-D environment. They realized that, in order to obtain better results, the route must be curved instead of the combination of the straight lines. This may be a good idea, but in some cases especially if you provide guidance, the route must be divided into parts of segments.

Hui et al. studied the importance of artificial intelligence in game playing [7]. In their paper, they investigated the use of artificial intelligence in game development. Research is done on how artificial intelligence can be applied in games and the advantages it brings along. As the fields of artificial intelligence in game development are too wide to be covered, the focus of their project is placed on certain areas. Two programs are implemented through this project: an intelligent camera system and path finding in a 3D application. The path-finding problem in game theory is very different from the path-planning problem in the avionics, because security concept is essential in the dynamic navigation problem.

Misra and Oommen presented the first learning-automaton based solution to the dynamic single-source shortest-path problem [8]. It involves finding the shortest path in a single-source stochastic graph topology where there are continuous probabilistic updates in the edge-weights. The important contribution of their algorithm was that all the edges in a stochastic graph are not probed, and even if they are, they were not all probed equally often.

Li et al. developed an improved genetic algorithm of optimum path planning for mobile robots [9]. They introduced an obstacle avoidance algorithm to generate the initial population in order to improve the path planning efficiency. Domain heuristic knowledge-based crossover, mutation, refinement, and deletion operators are specifically designed to fit path planning for mobile robots.

Tu and Yang proposed a novel genetic algorithm-based approach to path planning of a mobile robot [10]. The major characteristic of the proposed algorithm was that the chromosome has a variable length. The locations of target and obstacles were included to find a path for a mobile robot in an environment that was a 2D workplace discredited into a grid net.

Wei et al. proposed a gene-constrained genetic algorithm to solve shortest-path problem [11]. In this genetic algorithm, gene was constrained to ensure that each chromosome represents a feasible path without loop during the whole process of search. Contrasting with other genetic algorithm for shortest-path problem, their algorithm improved the searching capacity with a more accurate solution and more rapid speed of convergence.

Mahjoubi et al. proposed a path planning-method which uses genetic algorithm to find the feasible and suitable paths in an environment with static and dynamic obstacles [12]. To increase the speed of calculations, dimension of the search space was reduced by developing a new method to represent the environment. Their representation method was based on detecting the corners of circumferential polygons of all obstacles as representatives of the environment.

Inagaki et al. proposed an algorithm that employs fixed-length chromosomes [13]. The chromosomes in the algorithm are sequences of integers, and each gene represents a node identifier that is selected randomly from the set of nodes connected with the node corresponding to its locus number.

The most familiar dynamic path-planning solution techniques are Ramalingam Reps [14], Franciosa et al. [15], and Frigioni et al. [16]. The solution by Franciosa et al. can be used only for semidynamic case. The Ramalingam Reps solution was found successful concerning run-time; Frigioni's is better when the number of segments to be updated had to be minimized. The existing fully dynamic algorithms process unit changes to topology one modification at a time, but when there are several such operations occurring in the environment simultaneously, the algorithms are quite inefficient. The problems are worse in large topologies which have a large number of nodes and edges, where a large number of topology modifications occur continuously at all times. In such cases, the existing algorithms may fail to determine the shortest path information in a time critical manner. The proposed algorithm may respond to the concurrent weight updates in a shorter time especially for dynamic environments [17].

There are four possible edge operations (insertion/deletion and increase/decrease); it has been shown that edge-insertion is equivalent to edge-weight decrease and edge-deletion is equivalent to edge-weight increase. If all edge operations are allowed, the problem is referred to as the fully dynamic problem. If only edge insertion/weight decrease or edge deletion/weight increase is allowed, the problem is referred to as the semi-dynamic problem. The solution by Franciosa et al. can be used only for semi-dynamic case. The Ramalingam-Reps algorithm processes only one change at a time. But in the dynamic navigation problem, a large number of topology modifications occur continuously and this case is not handled by Ramalingam Reps algorithm. Frigioni algorithm cannot be used for the environments where the edge-weights change stochastically. The proposed algorithm should work with uncertain graphs by means of multi objective fitness function. All these three algorithms (Franciosa, Ramalingam Reps and Frigioni) solve dynamic, single-source shortest path problem. They do not consider path security or fuel consumption parameters, since none of these algorithms solve the dynamic flight navigation problem.

In order to model and solve the path planning problem for different environments, many researches have been done recently [18]. If the domain is air vehicle routing and navigation planning-usually 3D graph-based methods are used. Genetic and evolutionary computation algorithms can be used to solve dynamic navigation planning problem [19, 20]. Particle swarm optimization algorithm also can be used to find optimal path planning in 3D [21, 22]. All of these algorithms could not solve uncertain and stochastically changing graph problem in navigation. This is due to problem-specific multiobjective constraints for navigation planning. Minimum fuel consumption is the main aim in order to achieve this, the flight route for the air vehicle must be short and secure. The flight route segments are defined with flight altitude levels the vertical navigation should also be considered for the minimum fuel consumption. Dubins, Pythagorean hodograph, and Cornu-spiral are all algorithms which are based on curve to solve path planning problem [23–26]. The most advantage of those algorithms is non-discontinuous path planning because of curve path construction [24]. But all these algorithms fail to solve edge insertion and deletion issues for stochastically changing topologies and they do not propose solution for minimum fuel consumption calculation under distance, security, and altitude constraints.

The path-planning strategy could be either static or dynamic depending on whether the path-planning problem is to create a path in static or in dynamic environment [27]. Navigation planning solutions will attempt to for flight routes that are minimum in length, maximum in security and fully consistent with the physical constraints of the aircraft. Navigation planning can be divided in two forms. Former is pregenerated to show the flight route between source and destination waypoints that will be taken by the aircraft while it navigates between two waypoints and the latter is the controller that guides the aircraft between two waypoints while tracking a route that was pregenerated. In [28] a concurrent constraint programming was used as the main tool for the design and the implementation of a navigation planner. It is a very high level and complex heuristic path planner that takes into account the obstacle avoidance, shortest and best flight path, and weighed regions. But it fails to solve fully dynamic navigation problem. Most path planning follows an approach where the path planning, trajectory smoothing, and flight stability are separated into separate layers [29–31]. However, most of these do not deal with three-dimensional path planning. It is not an easy task to control the aircraft in a three-dimensional environment while at the same time executing the path algorithm. Our proposed method solves three-dimensional fully dynamic navigation planning problem, and our flight execution subsystem provides guidance in following the desired flight path under changing wind and speed conditions and platform dynamics.

Lin et al. designed a route guidance system for finding the shortest driving time which is their application on virtual maps of square matrix with appropriate to be used on handheld devices [32]. But their proposed solution fails to solve multiobjective navigation problem of air vehicles. Hasan et al. produced a different solution for the shortest-path problem using genetic algorithm [33]. They employed a chromosome-coding scheme using node indices and distance weights. Our proposed study presents a route guidance system and an evolutionary approach applied on this routing system to find the most secure flight path with minimum fuel consumption and shortest arrival time to the destination waypoint. The proposed guidance system provides the driving advice for the drivers considering not only the distances, but also the altitude and security values of the roads. Thus, it computes the optimum flight path instead of the shortest path.

Several approaches have been developed for evolutionary algorithms to address dynamic environments such as maintaining diversity during the run, increasing diversity after a change, using memory schemes to reuse stored useful information, and applying multipopulation and speciation schemes to search in different regions of the search space applying multipopulation and speciation schemes to search in different regions of the search space [34, 35]. The proposed evolutionary method considers multiobjectives like distance, security, and traffic, due to the objective function. The method responds to the dynamic environment situations and may offer an appropriate solution approach. It reduces the number of operations in dynamic environments; because it does not restart the solution at each update, it trends to protect and extend best-fit individuals according to the changed conditions.

3. System Definition

The evolutionary navigation planning system consists of two subsystems: mission planning subsystem and mission execution subsystem. In the mission planning subsystem, the flight transition of the air vehicle from a departure waypoint to a descent waypoint is planned by the dynamic evolutionary algorithm. The constraints of the mission planning

subsystem are distance, security, and altitude. The mission execution subsystem provides lateral and vertical guidance algorithms and fuel flow rate calculations so that the air vehicle passes through the flight legs and desired route in a real-time environment. Real-time flight data is taken from the Aerosim flight simulator. Mission execution subsystem uses these real-time data in order to calculate pitch and bank angle commands that feed the pilot. Mission execution subsystem executes the planned flight path from the desired departure waypoint to the each arrival waypoint owing to the lateral and vertical navigation guidance functions. The constraints of the fuel flow-rate calculation problem are gross weight, air temperature, flight altitude, and true airspeed. The two subsystems developed use 3-D graphs for the solution. The block diagram of the system and the data flow is shown in Figure 1.

4. Navigation System

Flight management system (FMS) is a computer system which handles all the navigation and flight functions of the air vehicle. FMS gathers all of the information generated by electronic equipments and sensors on a screen. So the workload of the pilot decreases. FMS provides autonomous guidance by means of inertial navigation unit and global positioning system [36]. The navigation functions of the FMS may be used to fly published airways, to route directly to a waypoint, to follow a flight plan, or to execute a mission pattern. FMS permits loading of flight plan database and it also permits the pilot to generate a new flight plan by using the waypoint database or modifying the existing plan due to the dynamic environment.

The navigation planning system takes the position coordinates of the waypoints as an input. Coordinate values are given as longitude and latitude. Latitude is the angle from a point on the earth's surface to the equatorial plane, measured from the center of the sphere. Longitude is the angle between the two geographical poles to another meridian that passes through an arbitrary point. The longitude and latitude components specify the position of any location on the planet but do not consider altitude or depth. The altitude constraint is considered for the legs of the flight plan. A flight leg is a route between two combined waypoints. The constraints of the flight leg are defined as a vector consists of the distance, the security value, and the altitude of the flight leg.

A primary flight display provides flight and navigation information. The primary flight display contains an attitude indicator, which gives the pilot information about the aircraft's attitude information, pitch and roll angles, and the orientation of the aircraft with respect to the horizon. The mechanical gyroscope is a separate device whose information is simply displayed on the primary flight display. Unlike mechanical instruments, this information can be dynamically updated as required; the stall angle, for example, can be adjusted in real time to reflect the calculated critical angle of attack of the aircraft in its current configuration. The primary flight display may also show an indicator of the aircraft's future path, as calculated by the proposed dynamic navigation planning algorithm, making it easier for pilots to anticipate aircraft movements and reactions. There are airspeed and altitude indicators next to the pitch and roll indicators. The airspeed indicator displays the speed of the aircraft in knots, while the altitude indicator displays the aircraft's altitude above sea level. These measurements are conducted through the aircraft's pitot system, which tracks air pressure measurements. The vertical speed indicator, next to the altitude indicator, indicates to the pilot how fast the aircraft is ascending or descending or the rate at which the altitude changes. This is usually represented with numbers in "thousands of feet per minute." At the bottom of the primary flight display is the heading display, which shows the pilot the magnetic heading of the aircraft. This functions much like a standard magnetic heading indicator, turning as required.

The most critic parameters of a primary flight display are pitch, roll, heading indicators, speed, and altitude indicators. The proposed algorithm and control system uses these data in order to find the best route for the air vehicle. The primary flight display consists of a vertical situation display with flight instruments, a flight director, and essential engine instruments and mode-selectable horizontal situation displays. The flight director commands calculated by the proposed control system show steering commands to capture desired aircraft roll and pitch. Flight director commands are satisfied by flying the aircraft into the intersection formed by the command bars.

For the execution test of the developed algorithm, air navigation process of the air vehicles from a departure waypoint to a descent waypoint is simulated. The simulation environment used to verify the navigation algorithms is shown in Figure 2.

The simulator represents a simulation of the aircraft in open-loop flight [37]. That is, all aircraft control inputs are set to fixed values, independent of the aircraft states. The lateral dynamics of the aircraft is stabilized by adding a wing leveler. This is implemented using proportional and integral feedback from bank angle to ailerons. The flight path guidance law is tested with an accurate nonlinear dynamic model of the Aerosonde [37, 38]. In the assessment of this work, both vehicle guidance and camera pointing algorithms are based on GPS data; the camera-pointing algorithm relies on aircraft attitude information. Some navigation algorithms are used for the guidance of air vehicles. These algorithms include the distance between two waypoints, the course angle between two waypoints, the intersection point of two radials, the cross-track error, and along-track distance calculations. The air vehicle is supposed to approach the desired flight path on a smoothly bank to capture the desired path without overshoot and exceeding the maximum bank angle. This may be achieved by modelling of the Helmsman behavior [39, 40].

The parameters like initial position, wind speed, initial velocity, initial altitude, initial engine speed, and sample time can be modified before the navigation simulation. The algorithm developed here can save the flight state information and current position and can continue to the flight to a different waypoint.

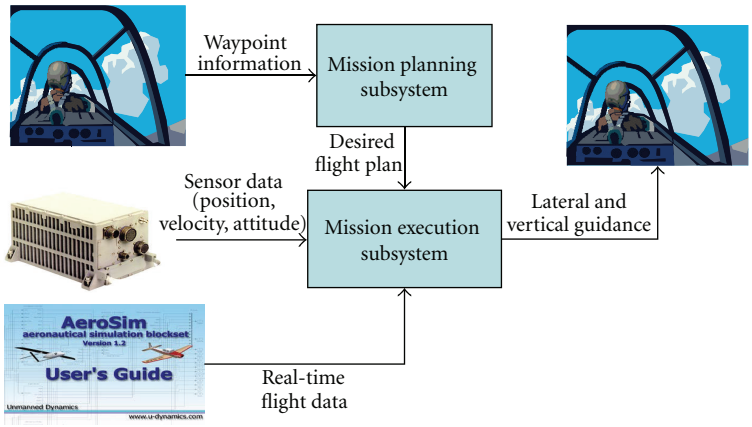

Figure 1: System block diagram.

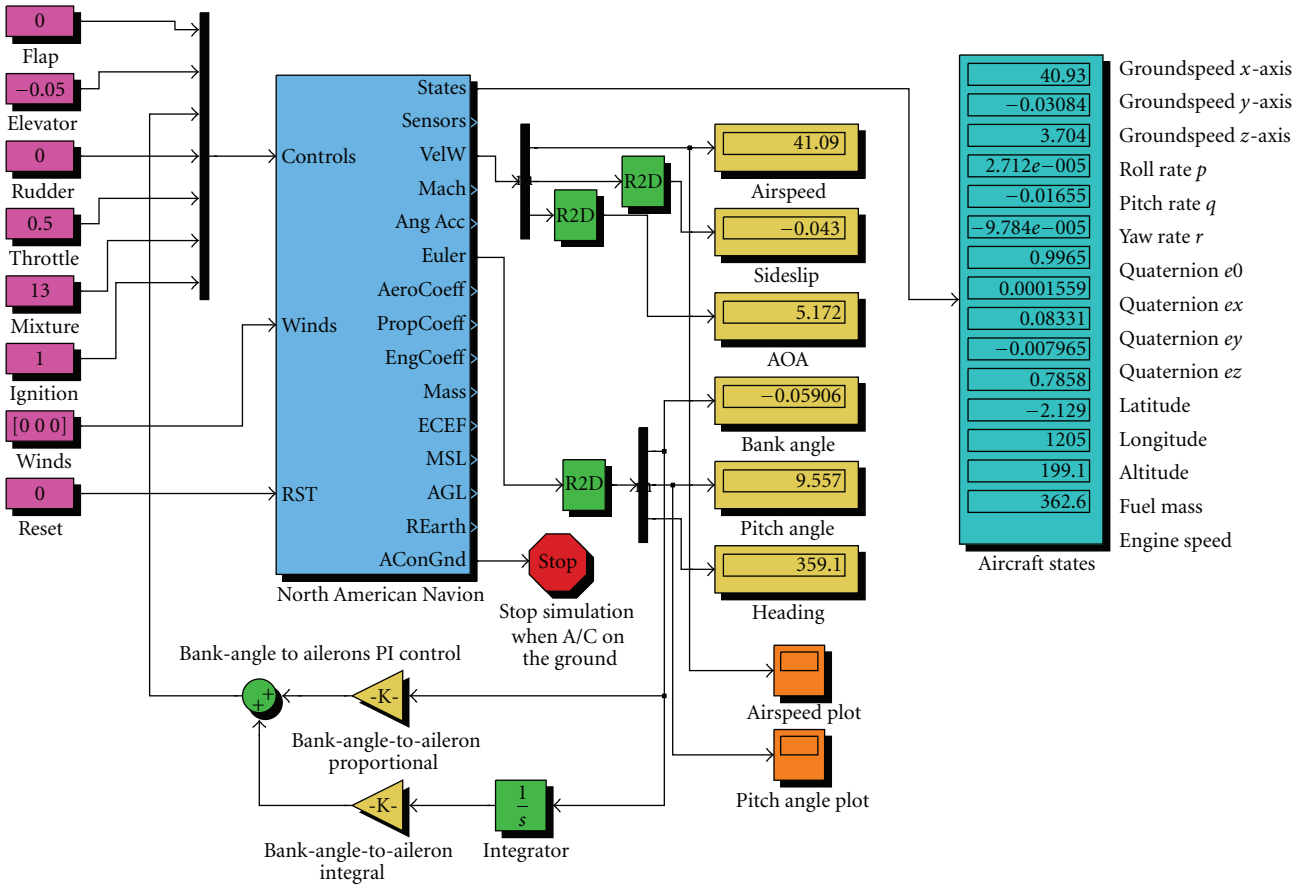

Figure 2: Aerosim smulator block diagram.

The AeroSim blockset provides a complete set of tools for developing nonlinear 6-degree-of-freedom aircraft dynamic models. The Simulink blocks include the nonlinear equations of motion, linear aerodynamics based on component buildup, piston-engine propulsion, aircraft inertia model including weight variation due to fuel consumption, atmosphere models including standard atmosphere, wind gusts and von Karman turbulence, and Earth models which provide Earth radius, gravity, and magnetic field components at current aircraft location. In addition the AeroSim blockset provides basic analog sensor and nonlinear actuator models and unit conversion blocks for translation between metric and English units, as well as transformations between various reference frames (wind, body, navigation,

and Earth-centered frame). In addition to the individual blocks, several prebuilt aircraft models are available, which can be customized through parameter files.

5. Lateral and Vertical Guidance

The general functions used in lateral and vertical navigation are geometric distance, bearing, and coordinate calculation functions. The first basic function calculates the distance between two waypoints and bearing angle of the flight route formed by the combination of two waypoints.

The inputs of the basic function of navigation are the longitude and latitude values of the source and destination waypoints. The outputs of the function are the distance of the flight leg and bearing angle of the flight route. The function calculates the distance value in radians. In order to convert the distance to units of length, the distance value is multiplied by the radius of the Earth. According to the WGS-84 ellipsoid model, the radius of the earth is 6378.137 kilometers. The function initially calculates the unit position vectors of two waypoints in Earth-centered Earth-fixed coordinate system. Then the cross-multiplication of two vectors is taken. So the angle and distance between two position vectors can be calculated.

The navigation planning system takes the position coordinates of the waypoints as an input. Coordinate values are given as longitude and latitude. Latitude is the angle from a point on the earth's surface to the equatorial plane, measured from the center of the sphere. Longitude is the angle between the two geographical poles to another meridian that passes through an arbitrary point. The longitude and latitude components specify the position of any location on the planet but do not consider altitude or depth. The altitude constraint is considered for the legs of the flight plan. A flight leg is a route between two combined waypoints. The constraints of the flight leg are defined as a vector consists of the distance, the security value, and the altitude of the flight leg:

$$\begin{aligned}\mathbf{P_1} &= \cos(L_1)\cos(\lambda_1)\mathbf{i} + \cos(L_1)\sin(\lambda_1)\mathbf{j} + \sin(L_1)\mathbf{k},\\ \mathbf{P_2} &= \cos(L_2)\cos(\lambda_2)\mathbf{i} + \cos(L_2)\sin(\lambda_2)\mathbf{j} + \sin(L_2)\mathbf{k}.\end{aligned} \tag{1}$$

By using the two equations above, the distance between source and target waypoints is calculated as shown below:

$$\text{dist}\,12 = \tan^{-1}\left(\frac{|\mathbf{P_1}\times\mathbf{P_2}|}{\mathbf{P_1}\cdot\mathbf{P_2}}\right). \tag{2}$$

The bearing angle between the source and destination waypoints is calculated by using the formula below:

$$\begin{aligned}\boldsymbol{\eta}_{P_1P_2} &= \frac{\mathbf{P_1}\times\mathbf{P_2}}{|\mathbf{P_1}\times\mathbf{P_2}|}, \qquad \boldsymbol{\eta}_{P_2P_1} = \frac{\mathbf{P_2}\times\mathbf{P_1}}{|\mathbf{P_2}\times\mathbf{P_1}|},\\ \psi_{12} &= \tan^{-1}\left(\frac{-\boldsymbol{\eta}_{P_1P_2}}{P_1\boldsymbol{\eta}_{P_1P_2} - P_1\boldsymbol{\eta}_{P_1P_2}}\right).\end{aligned} \tag{3}$$

The distance and bearing angle of a flight leg are shown in Figure 3.

Another function used for lateral guidance is the position calculation function. This function calculates the longitude and latitude of the waypoint from a given position vector, distance, and bearing values. In order to find the second waypoint's position, firstly tangent unit vector in the bearing angle direction to great circle is calculated. L_1 is the latitude; λ_1 is the longitude of the first waypoint. ψ_{12} is the input bearing angle. The coordinates of the new waypoint are calculated by the rotation of the first waypoint and the unit vector:

$$\begin{aligned}\mathbf{U}_\psi &= -\sin(L_1)\cos(\lambda_1)\cos(\psi_{12}) - \sin(\lambda_1)\sin(\psi_{12})\mathbf{i}\\ &\quad - \left(\sin(L_1)\sin(\lambda_1)\cos(\psi_{12}) - \cos(\lambda_1)\sin(\psi_{12})\right)\mathbf{j}\\ &\quad + \cos(L_1)\cos(\psi_{12})\mathbf{k},\\ \mathbf{P_2} &= \cos(\text{dist}\,12)\mathbf{P_1} + \sin(\text{dist}\,12)\mathbf{U}_\psi,\\ L_2 &= \tan^{-1}\left(\frac{P_2}{\sqrt{1-(P_2)^2}}\right), \qquad \lambda_2 = \tan^{-1}\left(\frac{P_2}{P_2}\right).\end{aligned} \tag{4}$$

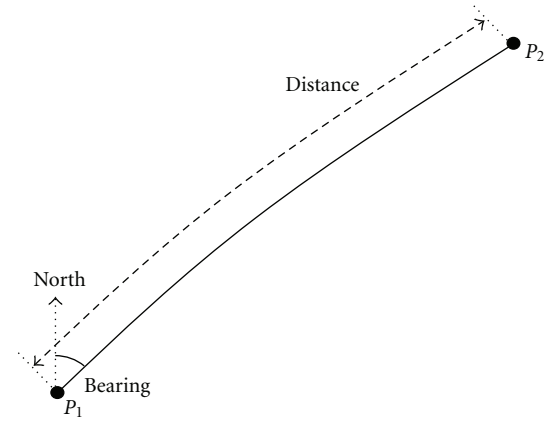

Figure 3: Distance and bearing of a flight leg.

Third basic function for lateral navigation is the calculation of the coordinates of the intersection of two flight legs. For lateral navigation, when the air vehicle is routed by parallel deviation from a flight plan, the intersection point of two segments is calculated for navigation planning. An intersection point of two flight segments is shown in Figure 4. P_0 is the intersection of the P_1-P_2 route and P_3-P_4 route.

L_1, λ_1 are the coordinates of P_1, similarly L_2, λ_2 are the coordinates of P_2, L_3, λ_3 are the coordinates of P_3, and L_4, λ_4 are the coordinates of P_4. The longitude and latitude values of the intersection point P_0 are L_0, λ_0.

$$\begin{aligned}\boldsymbol{\eta}_{P_1P_2} &= \mathbf{P_1}\times\mathbf{P_2}, \qquad \boldsymbol{\eta}_{P_3P_4} = \mathbf{P_3}\times\mathbf{P_4},\\ \mathbf{P_{01}} &= \frac{\boldsymbol{\eta}_{\mathbf{P_1P_2}}\times\boldsymbol{\eta}_{\mathbf{P_3P_4}}}{\left|\boldsymbol{\eta}_{\mathbf{P_1P_2}}\times\boldsymbol{\eta}_{\mathbf{P_3P_4}}\right|}, \qquad \mathbf{P_{02}} = \frac{\boldsymbol{\eta}_{\mathbf{P_3P_4}}\times\boldsymbol{\eta}_{\mathbf{P_1P_2}}}{\left|\boldsymbol{\eta}_{\mathbf{P_3P_4}}\times\boldsymbol{\eta}_{\mathbf{P_1P_2}}\right|}.\end{aligned} \tag{5}$$

In order to calculate the coordinates of the intersection point, firstly the normal vectors of the two surfaces are found. These two surfaces intersect at two points. These two points are found by the formula shown above. By using Pnk(i),

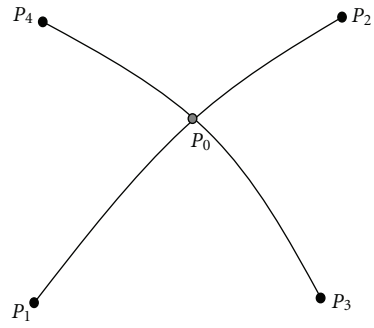

Figure 4: Intersection of two flight paths.

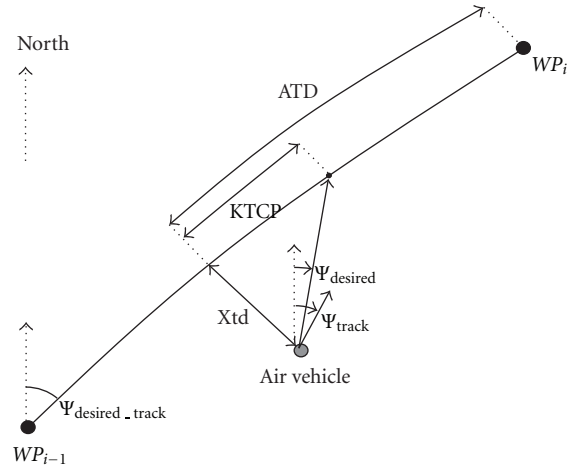

Figure 5: Track and bearing angles.

the latitude and the longitude of the intersection point are calculated. Pnk(i) shows the ith element of the Pnk vector:

$$L_0 = \tan^{-1}\left(P_{01}(3), \sqrt{1-(P_{01}(3))^2}\right),$$
$$\lambda_0 = \tan^{-1}(P_{01}(2), P_{01}(1)). \tag{6}$$

When a deviation from the flight plan occurs, the pitch and bank angle commands are calculated in order to fit the desired flight leg. Beside this, when the route switching points are reached, these angles are again calculated in order to put the air vehicle in the next flight route. For the calculation of the lateral guidance command, firstly the projection of the air vehicle on the desired route is found. KTCP distance shown in Figure 5 is the distance between the projection of the air vehicle and the point that the air vehicle will pass on the route. The coordinates of the point that the air vehicle will pass on the flight leg are found by (7). The bearing angle between the current position of the air vehicle and the point RP is calculated by (3). This angle is called desired track angle. ATD is the along-track distance, the distance between the vertical projection of the air vehicle and the waypoint to be reached. Xtd is the vertical distance to the flight leg:

$$\psi_{\text{track}} = \tan^{-1}\left(\frac{V_{\text{GS\_E}}}{V_{\text{GS\_N}}}\right). \tag{7}$$

Then the track angle is found by calculating the angle between the ground speed components, east and north. The track angle error is found by subtracting track angle from the desired track:

$$\psi_{\text{error}} = \psi_{\text{desired}} - \psi_{\text{track}}. \tag{8}$$

By using the track angle error found by (8), lateral-rotation speed command is found. The fd_lat parameter is chosen according to the simulations. And finally by using (10), the lateral navigation bank-angle command is calculated. This command provides holding the air vehicle in the desired route:

$$\dot{\psi}_{\text{commanded}} = (2\pi f_{\text{d\_lat}})\psi_{\text{error}}, \tag{9}$$

$$\phi_{\text{commanded}} = \tan^{-1}\left(\frac{\left(V_{\text{GS\_N}}^2 + V_{\text{GS\_E}}^2\right)^{1/2} \cdot \dot{\psi}_{\text{commanded}}}{g}\right). \tag{10}$$

6. Mission Planning Subsystem

Route planning and navigation control based on evolutionary programming concepts can be designed as a general, flexible and adaptive technique. By integrating the planning process in evolutionary algorithms, definition of the different optimization criteria, dynamic update of the constraints, domain specific evolutionary operators, and control of the dynamic obstacles may be handled. Evolutionary algorithms, in comparison to the classical optimization methods, are more effective for discontinuous and noisy objective functions [41].

Genetic algorithms imitate the evolutionary process in order to solve the optimization problems. Instead of developing one solution candidate, genetic algorithms form a set of individuals. The set, which contains probable solution candidates, is defined as population in genetic algorithm terminology. Population occurs from arrays called vector, chromosome, or individual. Each element of an individual is called gene. In evolutionary programming method, individuals in the population are determined by the operators of the evolutionary algorithm. In the problems like path planning, the permutation representation is used and the operators differ from the operators of the basic genetic algorithm. The most important parameters of the genetic algorithm are cross-over rate, mutation rate, and the number of individuals in a population. In order to declare cross-over and mutation rates, different values are tested, and by this way the most

appropriate values are found for these parameters. Chromosome number in the population is determined according to the nodes of the graph topology.

In the proposed heuristic approach, variable-length chromosomes are used for representing the routes. Variable-length chromosomes are used in dynamic path-planning systems in order to cover the whole search space [42]. Chromosomes are encoded by permutation-encoding method. Each gene of a chromosome represents a node in the graph. Evolutionary parameters of the mission planning system are listed in Table 1.

In a genetic algorithm, crossover takes two parents and replaces a randomly chosen part of one parent with another, randomly chosen part of the other. This is often very destructive to the structure and functionality of the child program. It is, however, the means by which valuable code can be transferred between programs and is also the theoretical reason why genetic programming is an efficient and successful search strategy. Even though it often produces unfit children, it does produce parent-superior fitness occasionally and those individuals often possess the critical improvements that allow evolution to progress to the next round of fitness improving generations. Mutation takes one parent and replaces a randomly selected chunk of that parent with a randomly generated sequence of code. One of the advantages of this operator is that it maintains diversity in the population, since any of the function/terminal set can be inserted into the program, whereas crossover can only insert code present in the current generation's population. Through the repeated application of these operators to the selected parents of the old generation, a new generation is formed, some of the members of which will hopefully be more fit than the best of the last generation. At this point a new population is available to be evaluated for fitness. The cycle will continue, until either a single member of the population is found which satisfies the problem within the level of error designated as acceptable by the success criteria or the number of generations exceeds the limit specified.

Cross-over operator developed for navigation planning problem exchanges pieces of routes. An identical intermediate node is chosen. The first part of the route connects the initial node to the intermediate node. The second part of the route connects the intermediate node to the target node. Crossover may generate infeasible chromosomes that violate the loop constraint. Repair operator makes a postprocessing operation and removes the cycles from the infeasible individuals [43].

Mutation operator increases the variation in the population. Mutation avoids local optima by changing the genes of the potential chromosome. Two-point mutation is applied. The genes in the region between the mutation points are modified with a different route. Finally two new individuals with different genotype are formed. The fitness of the individual chromosome is calculated by using the distance, height difference, and security values of the segments in the path. Height difference is the difference of altitudes of a segment and the altitude of the previous segment. The height difference is not considered for the first segment of the route. The sample crossover and repair operations are shown in Figures 6 and 7. For postprocessing local search is included in the procedure in order to eliminate the individual that have very low fitness values.

Table 1: Evolutionary algorithm parameters.

Selection method	Roulette wheel
Chromosome length	Variable
Cross-over rate	70–80%
Mutation rate	5–10%
Representation	Permutation based
Stopping criteria	Individual similarity
Population size	Proportional with the number of edges

The flowchart of the mission planning subsystem is shown in Figure 8. At the first step the initial individuals are generated randomly and represented by chromosomes. In order to evaluate the initial population, the multi-objective fitness function is used. The individuals are divided into pairs in the parent selection phase. Cross-over operator is applied to the pairs. After crossover if necessary the repair operator is applied in order to remove the cycles in the flight path. Mutation operator is applied to some of the individuals for diversity. After recombination phase, the fitness function is used again to evaluate the parents and individuals. The individuals which have higher fitness values are kept alive. This procedure is continued again till the stopping criterion is satisfied.

7. Experimental Results

Three most important parameters of the proposed evolutionary method are cross-over rate, mutation rate, and number of generations. For each parameter different flight simulations are planned and executed, and the following graphs are formed by the average values of the experimental results. The experiments are done with 50 waypoints city map. We produced a connected graph and selected a source waypoint for the start of the flight and a destination waypoint for arrival. The experiments are repeated with updated constraints and edges. At each measurement, different edges are added to the graph or some edges are removed from the graph. The experiments are repeated for 100 different maps in the same topology.

A first set of experiments is done for determining the appropriate cross-over rate. The mutation rate, number of individuals, and the stop criteria are fixed in this set. The mutation rate in this set is 10%. The stopping criterion is individual similarity. The result of the first experiment set is shown in Figure 9. As cross-over rate increases up to 80%, the error percentage decreases. Error percentage is calculated by finding the ratio of the fitness of the proposed solution and the real most secure, shortest path solution. Experimental results show that for this kind of navigation planning problem, the most suitable cross-over rate is 70%.

A second set of experiments is done for determining the appropriate mutation rate. The cross-over rate, number of individuals, and the stop criteria are fixed in this set. The cross-over rate in this set is 70%. The stopping criterion is

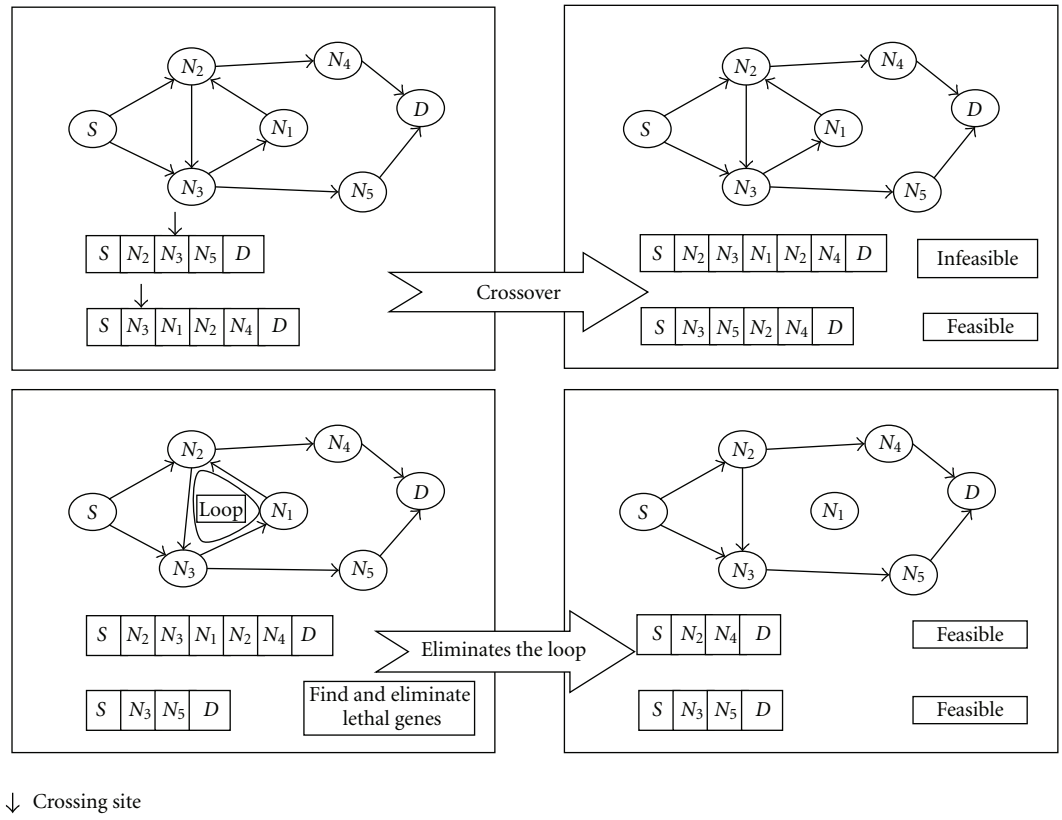

Figure 6: Crossover and repair operators.

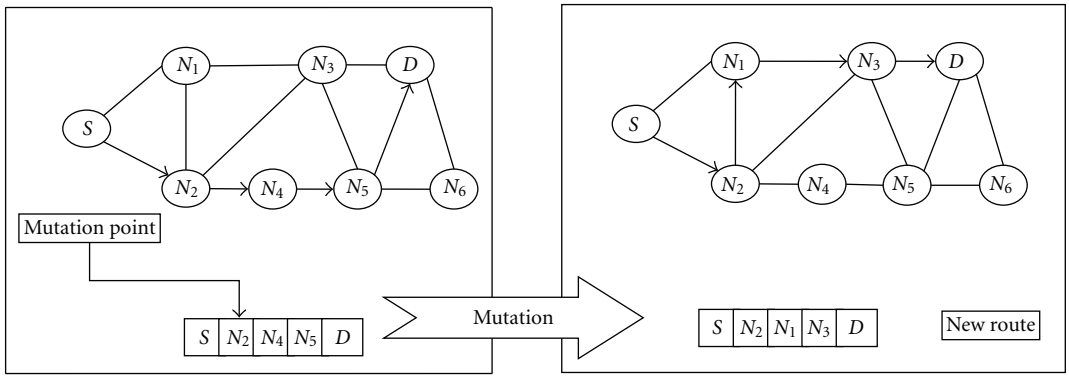

Figure 7: Mutation operator.

individual similarity. The result of the second experiment set is shown in Figure 10. As mutation rate increases up to 10%, the error percentage decreases. Experimental results show that for this kind of navigation planning problem, the most suitable mutation rate is 10%. If mutation rate is chosen at higher values, it may lead to loss of good solutions. If mutation rate is too small, then it is more difficult to reach different points of the search space, and usually the algorithm may converge to local optima in these cases.

In order to show the best candidate of each generation, a third set of experiments is done. The cross-over rate is fixed to 70%; the mutation rate is fixed to 10%. For each population number of individuals is 100 in this set. The result of the third experiment set is shown in Figure 11. In each generation the individual, who has highest fitness value, becomes nearer to the real solution, but when the global optimum is reached, further generations do not produce better individuals, and by means of the stopping criteria, genetic algorithm reproduction loop is terminated.

The run-time performance and the complexity of the proposed algorithm are compared with deterministic methods. When segment insertion/deletion/cost update operations occur, the proposed algorithm does not start calculations from scratch, and it converges to the solution in a

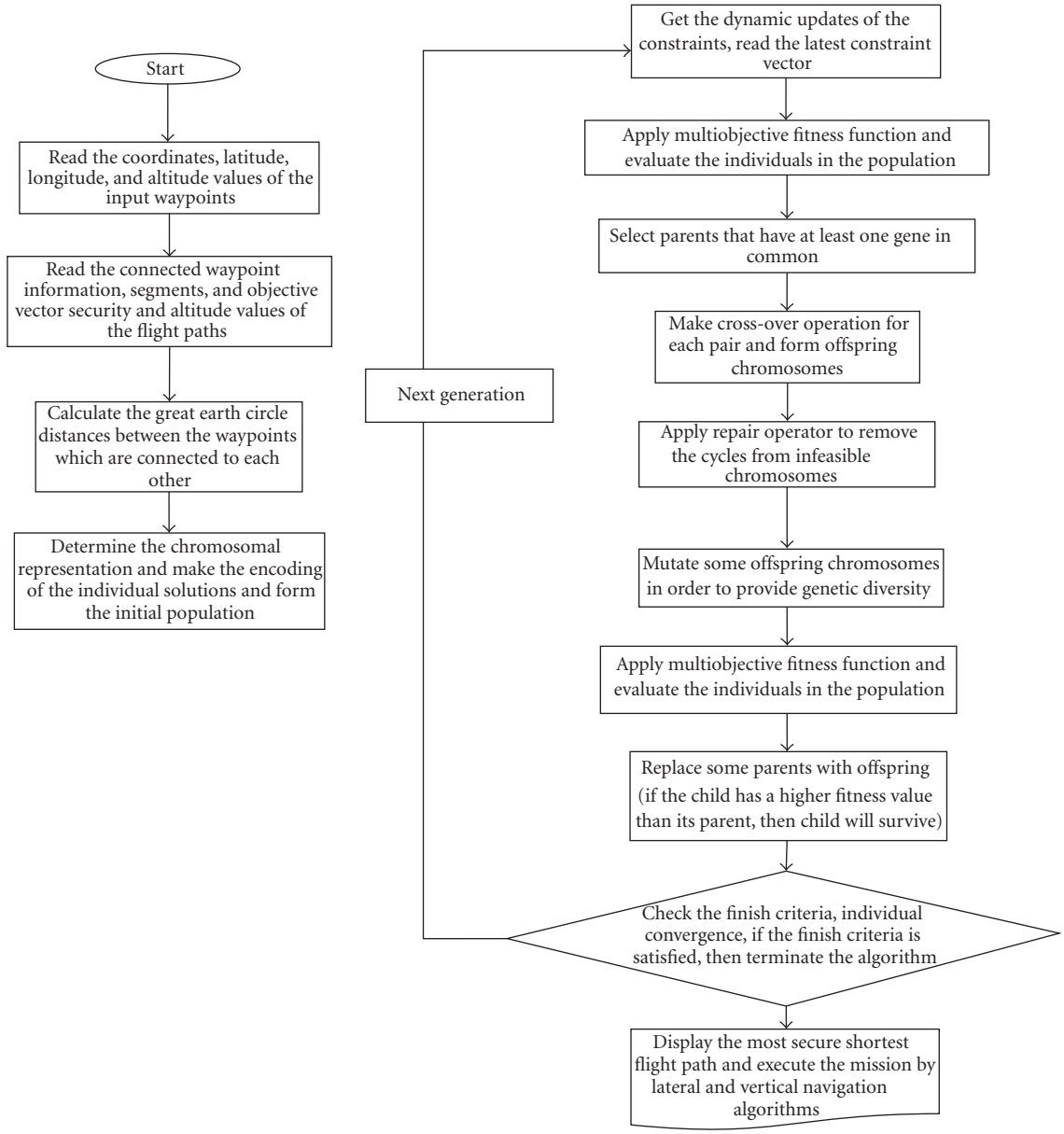

Figure 8: Mission planning subsystem.

shorter time than Dijkstra. Figure 12 shows the performance of the proposed algorithm at each dynamic change. Figure 12 also shows the higher performance of the proposed algorithm especially for strongly connected graphs. When the number of nodes increases, the computation time difference between the Dijkstra algorithm and the proposed algorithm in dynamic scheme also increases.

8. Navigation Simulations

In the simulations a part of USA waypoint database, including the longitude and latitude coordinates of the south region states, is used. Transition of the air vehicles between waypoints is planned. For the visual simplicity, the numbers are used instead of the names of the states, and only 15 states are chosen. Finding the most secure, smoothest and the fastest transition of the air vehicles from an initial waypoint to a target waypoint is aimed. In the city map shown in Figure 13, the links between the cities are represented by vectors of distance, security, and altitude. As the flight plan is generated, the altitudes of the waypoints are also considered and 3-D graph solution is planned.

The proposed evolutionary method aims to find the shortest, most secure and the smoothest route for the initial constraints. As the problem considers distance, height, and security conditions, the problem is multi-objective optimization problem. The initial solution of the dynamic system for the initial conditions is shown in Figure 14.

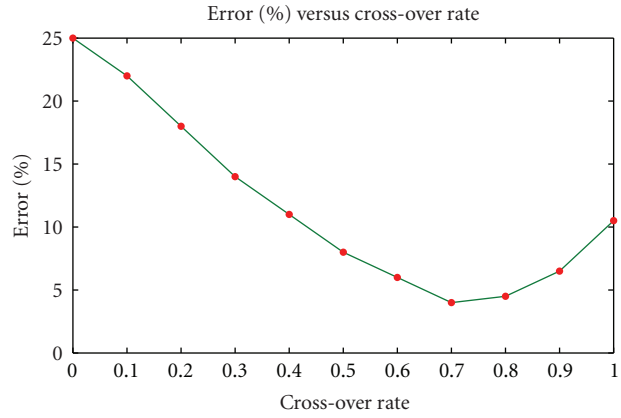

Figure 9: Effect of the cross-over rate.

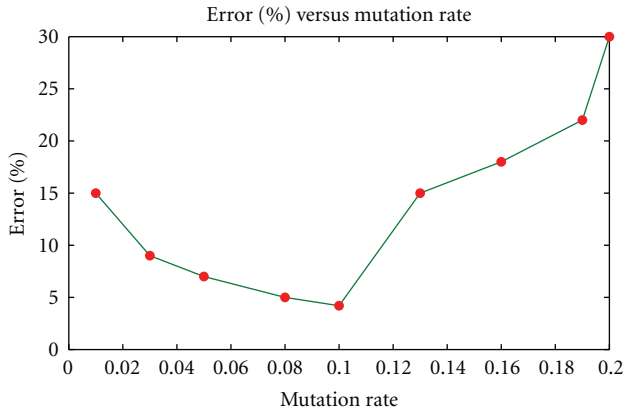

Figure 10: Effect of the mutation rate.

When the links between City 8-City 9 and City 8–City 11 are disabled dynamically, the new transition flight plan for the air vehicle is shown in Figure 15.

When the link between City 1–City 3 is disabled dynamically, the proposed route computed by the algorithm is shown in Figure 16. The algorithm proposes that a flight plan passes through City 8 considering smoothness and security. The link between the cities, City 4–City 8, is suitable for security and distance metrics.

The system converges to the solution in the dynamic environments without processing from scratch. Analytical approaches repeat the calculations from scratch because an update in the constraints change the matrix used to find the solution [44, 45]. This property is an important characteristic of the proposed algorithm. Evolutionary methods may succeed this by problem specific fitness function and selection operators [46]. The fitness function used in the proposed algorithm is

$$\sum_{i=1}^{\text{segmentcount}} \left(\frac{A}{\text{distance}} + \frac{B}{(100 - \text{security})} + \frac{C}{\text{altitude\_difference}} \right). \tag{11}$$

The fitness of the individual chromosome is calculated by using the distance, height difference, and security values of the segments in the path. Height difference is the difference of altitudes of a segment and the altitude of the previous segment. The height difference is not considered for the first segment of the route. The problem is multi-objective in our case, because security objective may subject to distance constraints, and distance constraint depends on a statistical distribution function. The effects of metrics like security and distance to the solution may be modeled with the classical optimization techniques, but in the dynamic environments like battle scenarios, at each condition change, the algorithm starts the solution from the scratch and increase the operation complexity and calculation time, so they are infeasible [47]. Furthermore, the insertion/deletion of the nodes or segments from the graph changes the matrix of the topology, so they have to start the solution from scratch. The proposed

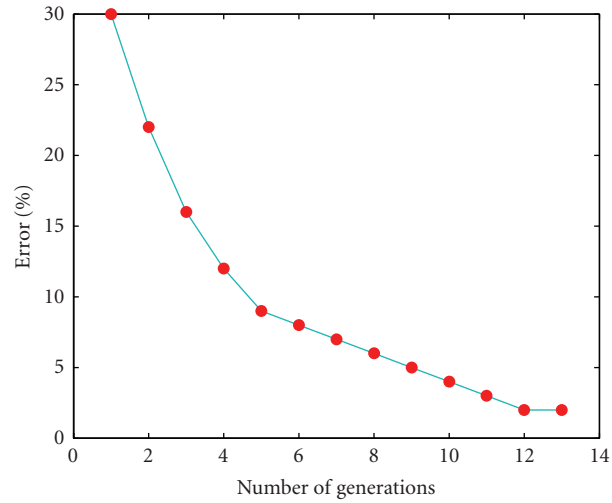

Figure 11: Effect of the number of generation count.

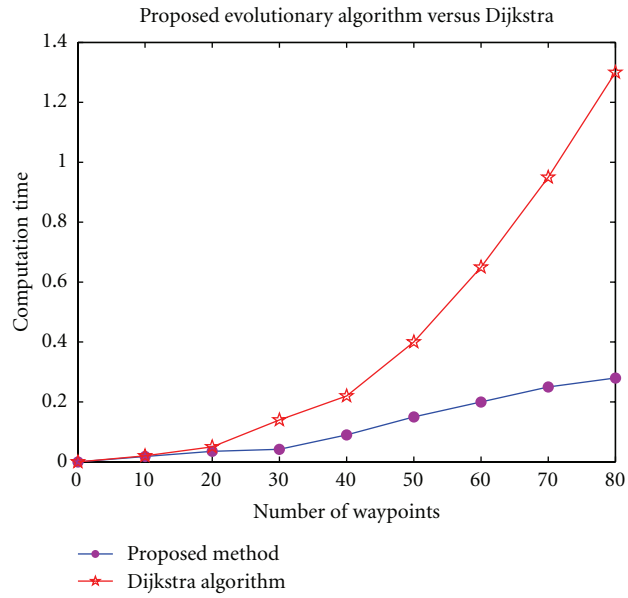

Figure 12: Run-time performance of the proposed algorithm.

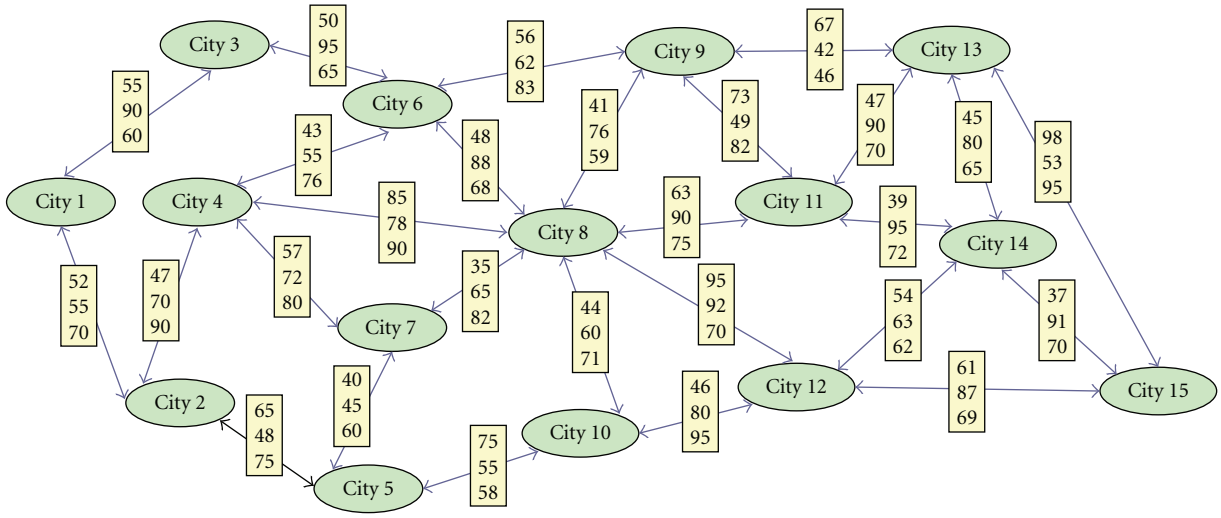

Figure 13: Initial city map.

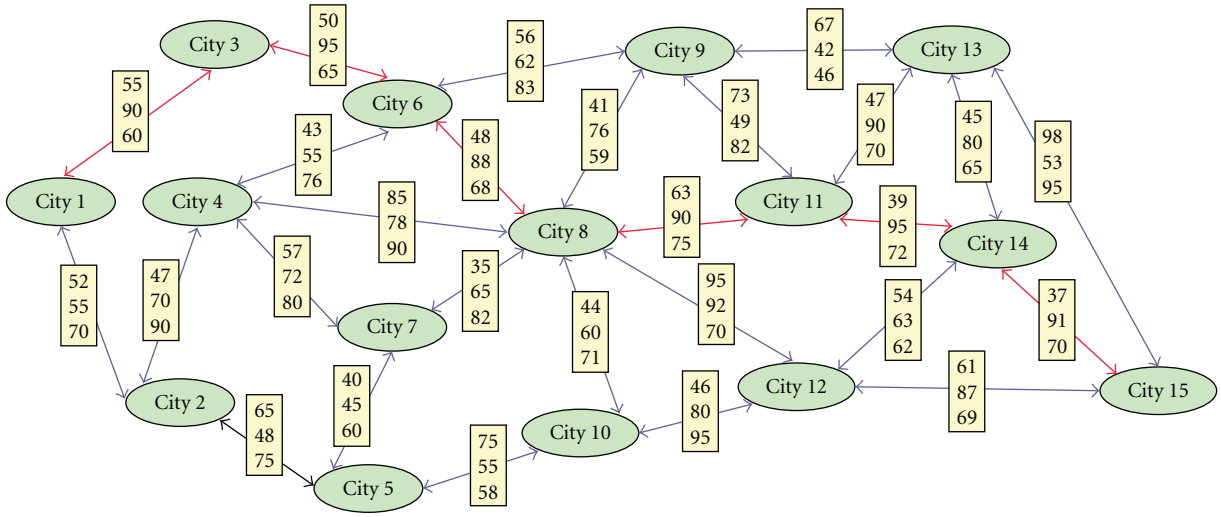

Figure 14: Initial transition plan.

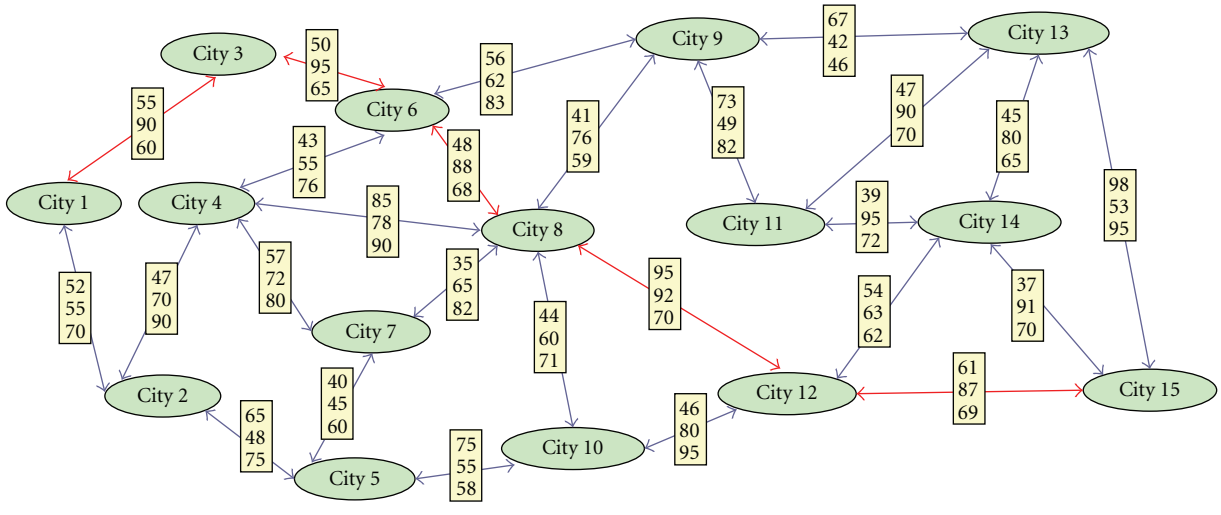

Figure 15: Updated transition plan.

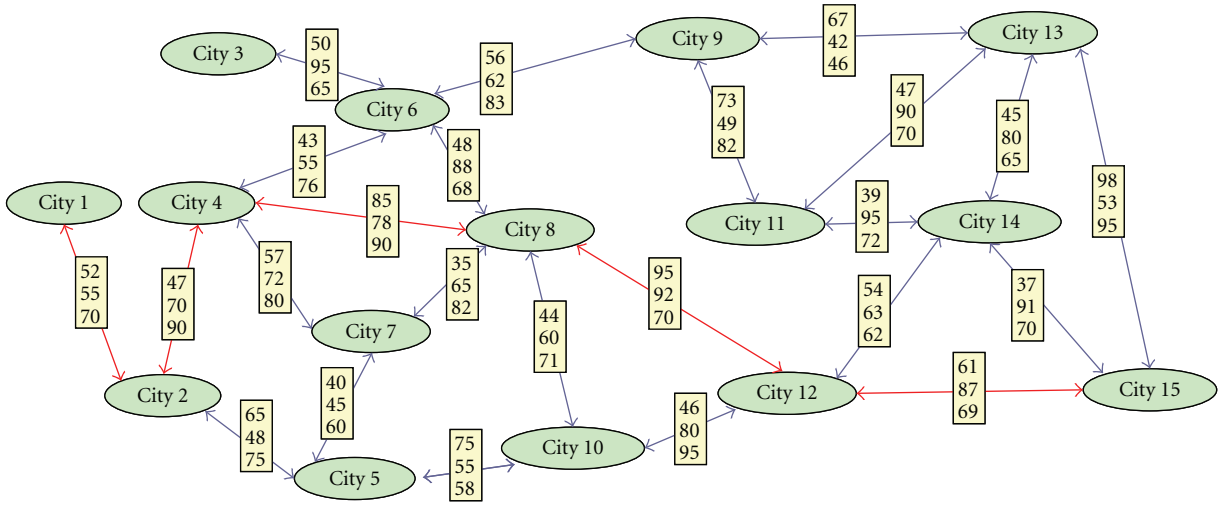

Figure 16: Dynamic transition plan.

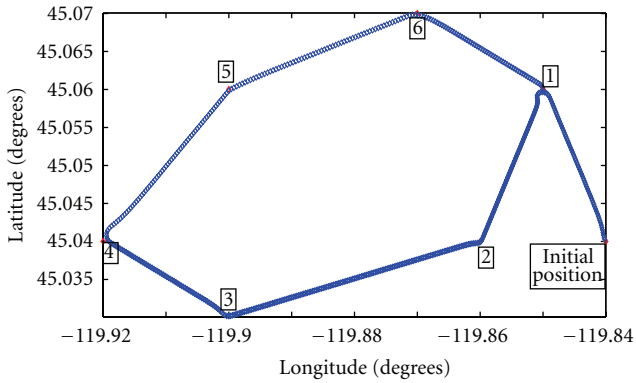

Figure 17: Flight route execution simulation.

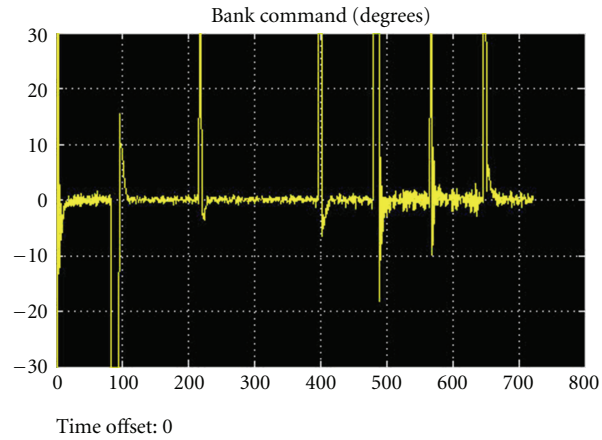

Figure 18: Lateral navigation bank-angle command simulation result.

algorithm responds to the concurrent modifications and the best-fit individuals according to the modified conditions are spread over the population. Military transition is a mission critic problem, and in dynamic environments the aim is to reach the best solution in an acceptable time. This time changes according to the graph density and the number of nodes. The coefficients A, B, and C may be modified according to which objective will have more effect on the calculation of the flight path. These coefficients may be given as an input to the algorithm.

By the execution of the flight plan formed by the mission planning subsystem, the most secure, shortest navigation path is processed by mission execution subsystem. The simulation results of the flight plan with six waypoints are shown in the Figures 17 and 18. The graph related with the execution of the flight plan is shown in Figure 17; the commanded bank angle for the execution of this flight path is shown in Figure 18. In this example, the air vehicle follows the desired flight path under (north 10 meters/second, east 10 meters/second) wind. The wind speed is 14.14 meters/second. The initial position is 45° 04′ 47″ latitude

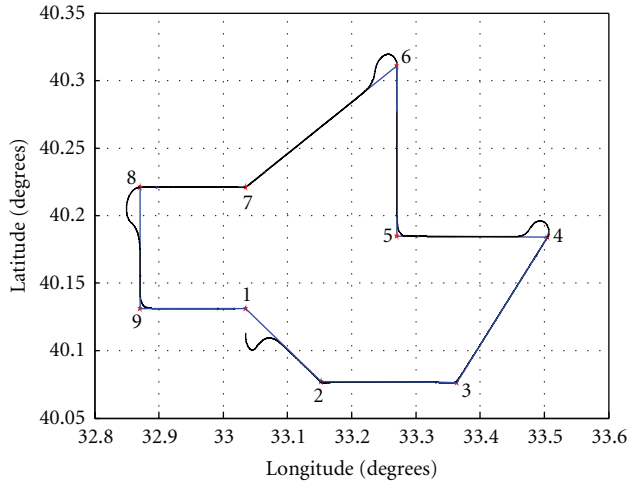

Figure 19: Mission execution 2D graph with nine waypoints.

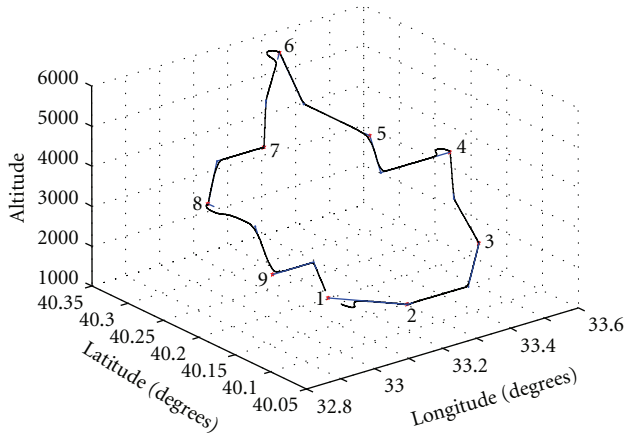

Figure 20: Mission execution 3D graph with nine waypoints.

and 19° 84′ 06″ longitude. The initial ground speed is 70 meters/second, and the initial altitude is 1985 meters.

Another flight plan execution simulation is shown in the Figures 19 and 20. For the second example simulation, the flight plan for nine waypoints is found by evolutionary mission planning subsystem (see Table 2). In this example, the air vehicle follows the desired flight path under (north 10 meters/second, east 10 meters/second) wind. The wind speed is 14.14 meters/second. The initial position is 40° 06′ 47″ latitude and 33° 02′ 06″ longitude. The initial ground speed is 70 meters/second, and the initial altitude is 1985 meters.

In Figure 19, the flight path of the air vehicle is shown by the 2-D graph. 3-D graph structure of this simulation is shown in Figure 20.

9. Conclusion

Navigation planning is an optimization problem that requires finding and executing the most feasible flight path between source and destination waypoints. Deterministic algorithms like Floyd, Dijkstra, or heuristic methods like neural network and A-Star may be used for the solution of navigation planning problem in static schemes. Dynamic path planning algorithms like Frigioni, Franciosa, and Ramalingam Reps may be used in dynamic environments, but they are constrained by some limitations. When several concurrent changes occur in the environment simultaneously, these algorithms are quite inefficient. When the segment costs and conditions change stochastically and continuously, these algorithms fail to reach the actual underlying average solution. And these algorithms run when the costs of the links in the graph are clear. But in a flight plan scenario, the altitude of the paths and the security of the links may be approximately known. So the algorithm should work with uncertain graphs.

The flight and mission planning is a fault-tolerant real-time system, so the algorithm should attach more importance to a constraint, like distance, time, security, or altitude. In the algorithms like Floyd and Dijkstra, the graph is represented by a matrix. But the proposed solution is not matrix-based. Owing to the reproduction loop, selection mechanism and the fitness function, the algorithm finds better results in a shorter computation time compared with the analytical algorithms. The proposed algorithm provides advanced search speed, quality, and flexibility in dynamic schemes. In addition to the flight planning algorithm, a flight execution algorithm is developed to guide the pilot follow the mission plan. The execution algorithm provides flight commands like pitch, bearing, track and bank angles, and desired ground speed to arrive the next waypoint on time and fuel flow rate.

The results support the view that evolutionary algorithms are an effective, robust search procedure for NP complete problems in the sense that, although they may not outperform highly tuned, problem-specific algorithms. Evolutionary algorithms can be easily applied to a broad range of NP-complete problems with performance characteristics no worse than the theoretical lower bound of an N^3 speedup.

Table 2: Coordinate and altitude values of the second flight plan waypoints.

WP number	Latitude (°)	Longitude (°)	Altitude (meters)
1	40,1310	33,0351	1985,7
2	40,0768	33,1524	1985,7
3	40,0764	33,3635	2985,7
4	40,1840	33,5050	3985,7
5	40,1847	33,2701	4985,7
6	40,3109	33,2706	5985,7
7	40,2211	33,0352	4985,7
8	40,2210	32,8706	3985,7
9	40,1309	32,8708	2985,7

References

[1] M. A. Al-Jarrah and M. M. Hasan, "HILS setup of dynamic flight path planning in 3D environment with flexible mission planning using Ground Station," *Journal of the Franklin Institute*, vol. 348, no. 1, pp. 45–65, 2011.

[2] Joint Planning and Development Office, "Concept of operations for the next generation air transportation system," Version 3.0, 2009.

[3] M. T. De Garmo, "Issues concerning integration unmanned aerial vehicles in civil airspace," Tech. Rep. MP 04W0000323, MITRE, Va, USA, 2004.

[4] R. Weibel and R. J. Hansman, "Safety considerations for operation of unmanned aerial vehicles in the national airspace system," MIT International Center for Air Transportation Report ICAT, 2005-01, ICAT, 2005.

[5] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple UAVs path planning algorithms: a comparative study," *Fuzzy Optimization and Decision Making*, vol. 7, no. 3, pp. 257–267, 2008.

[6] I. K. Nikolos, N. Tsourveloudis, and K. P. Valavanis, "Evolutionary algorithm based 3-D path planner for UAV navigation," in *Proceedings of the 9th Mediterranean Conference on Control and Automation (CDROM'10)*, Dubrovnik, Croatia, 2001.

[7] Y. C. Hui, E. C. Prakash, and N. S. Chaudhari, "Game AI: artifical intelligence for 3D path finding," in *Proceedings of*

the IEEE Region 10 Conference: Analog and Digital Techniques in Electrical Engineering (TENCON'04), pp. B306–B309, tha, November 2004.

[8] S. Misra and B. J. Oommen, "Stochastic learning automata-based dynamic algorithms for the single source shortest path problem," in *Proceedings of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE'04)*, pp. 239–248, May 2004.

[9] Q. Li, W. Zhang, Y. Yin, Z. Wang, and G. Liu, "An improved genetic algorithm of optimum path planning for mobile robots," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06)*, pp. 637–642, October 2006.

[10] J. Tu and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1221–1226, Taiwan, September 2003.

[11] W. Wu and Q. Ruan, "A gene-constrained genetic algorithm for solving shortest path problem," in *Proceedings of the 7th International Conference on Signal Processing Proceedings (ICSP'04)*, pp. 2512–2515, September 2004.

[12] H. Mahjoubi, F. Bahrami, and C. Lucas, "Path planning in an environment with static and dynamic obstacles using genetic algorithm: a simplified search space approach," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'06)*, pp. 2483–2489, Vancouver, Canada, July 2006.

[13] J. Inagaki, M. Haseyama, and H. Kitajima, "Genetic algorithm for determining multiple routes and its applications," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS '99)*, pp. I-137–I-140, June 1999.

[14] G. Ramalingam and T. Reps, "On the computational complexity of dynamic graph problems," *Theoretical Computer Science*, vol. 158, no. 1-2, pp. 233–277, 1996.

[15] P. G. Franciosa, D. Frigioni, and R. Giaccio, "Semi-dynamic shortest paths and breadth first search in digraphs," in *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 33–46, Lübeck, Germany, 1997.

[16] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Fully dynamic algorithms for maintaining shortest paths trees," *Journal of Algorithms*, vol. 34, no. 2, pp. 251–281, 2000.

[17] A. Elshamli, H. A. Abdullah, and S. Areibi, "Genetic algorithm for dynamic path planning," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pp. 677–680, May 2004.

[18] D. Catanzaro, M. Labbé, and M. Salazar-Neumann, "Reduction approaches for robust shortest path problems," *Computers and Operations Research*, vol. 38, no. 11, pp. 1610–1619, 2011.

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

[20] G. Xiao, F. Xiao, and C. Da, "A genetic-algorithm-based approach to UA V path planning problem," in *Proceedings of the 5th WSEAS International Conference on Simulation, Modelling and Optimization*, pp. 17503–19507, Corfu, Greece, August 2005.

[21] S. Li, X. Sun, and Y. Xu, "Particle swarm optimization for route planning of unmanned aerial vehicles," in *Proceedings of the IEEE International Conference on Information Acquisition (ICIA'06)*, pp. 1213–1218, Shandong, China, August 2006.

[22] G. Wang, Q. Li, and L. Guo, "Multiple UAVs routes planning based on particle swarm optimization algorithm," in *Proceedings of the 2nd International Symposium on Information Engineering and Electronic Commerce (IEEC'10)*, pp. 150–154, July 2010.

[23] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a dubins airplane," in *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07)*, pp. 2379–2384, New Orleans, La, USA, December 2007.

[24] R. Dai and J. E. Cochran, "Path planning for multiple unmanned aerial vehicles by parameterized cornu-spirals," in *Proceedings of the American Control Conference (ACC'09)*, pp. 2391–2396, St. Louis, Mo, USA, June 2009.

[25] M. Shanmugavel, A. Tsourdos, and B. A. White, "Collision avoidance and path planning of multiple UAVs using flyable paths in 3D," in *Proceedings of the 15th International Conference on Methods and Models in Automation and Robotics (MMAR'10)*, pp. 218–222, August 2010.

[26] M. Shanmugavel, A. Tsourdos, B. White, and R. Zbikowski, "Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs," *Control Engineering Practice*, vol. 18, no. 9, pp. 1084–1092, 2010.

[27] S. P. Bhat and P. Kumar, "A feedback guidance strategy for an autonomous mini-air vehicle," in *Proceedings of the National Conference on Control and Dynamical Systems*, IIT Bombay, Bombay, India, January 2005.

[28] S. Gualandi and B. Tranchero, "Concurrent constraint programming-based path planning for uninhabited air vehicles," in *Proceedings of the SPIE/Defense and Security Symposium on Unmanned Ground, Ocean, and Air Technologies*, pp. 12–16, April 2004.

[29] W. Ren and R. W. Beard, "Constrained nonlinear tracking control for small fixed-wing unmanned air vehicles," in *Proceedings of the American Control Conference (AAC'04)*, pp. 4663–4668, Boston, Mass, USA, July 2004.

[30] D. Kingston, *Implementation issue of real-time trajectory generation of small UAV [M.S. thesis]*, Birmingham Young University, 2004.

[31] R. Beard, D. Kingston, M. Quigley et al., "Autonomous vehicle technologies for small fixed-wing UAVs," *Journal of Aerospace Computing, Information and Communication*, pp. 92–108, 2005.

[32] C. H. Lin, J. L. Yu, J. C. Liu, and C. J. Lee, "Genetic algorithm for shortest driving time in intelligent transportation systems," in *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE'08)*, pp. 402–406, Busan, Korea, April 2008.

[33] B. S. Hasan, M. A. Khamees, and A. S. H. Mahmoud, "A heuristic genetic algorithm for the single source shortest path problem," in *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'07)*, pp. 187–194, May 2007.

[34] S. Yang, H. Cheng, and F. Wang, "Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 40, no. 1, pp. 52–63, 2010.

[35] S. Yang, "Population-based incremental learning with memory scheme for changing environments," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05)*, pp. 711–718, June 2005.

[36] J. A. Farrell and M. Barth, *The Global Positioning System & Inertial Navigation*, McGraw Hill, 1999.

[37] "Aerosim Blockset," Version 1.2, User's Guide, Unmanned Dynamics, LLC.

[38] S. Stolle and R. Rysdyk, "Flight path following guidance for unmanned air vehicles with pan-tilt camera for target observation," in *Proceedings of the 22nd Digital Avionics Systems Conference*, pp. 8.B.3/1–8.B.3/12, Indianapolis, Ind, USA, October 2003.

[39] K. Y. Pettersen and E. Lefeber, "Way-point tracking control of ships," in *Proceedings of the 40th IEEE Conference on Decision and Control (CDC'01)*, pp. 940–945, Orlando, Fla, USA, December 2001.

[40] R. Rysdyk, "UAV Path Following for constant line-of-sight," in *Proceedings of the 2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations Aerospace, Land, and Sea Conference*, paper no. 6626, San-Diego, Calif, USA, September 2003.

[41] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, London, UK, 1997.

[42] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller, "The Gambler's ruin problem, genetic algorithms, and the sizing of populations.," *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, 1999.

[43] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.

[44] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference (GECCO'06)*, pp. 1201–1208, July 2006.

[45] P. A. N. Bosman, "Learning, anticipation and time-deception in evolutionary online dynamic optimization," in *Proceedings of the Workshop on Evolutionary Algorithms for Dynamic Optimization (GECCO'05)*, Washington, DC, USA, 2005.

[46] C. Chitra and P. Subbaraj, "Multiobjective optimization solution for shortest path routing problem," *International Journal of Computer and Information Engineering*, vol. 4, no. 2, pp. 77–85, 2010.

[47] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*, Springer, Berlin, Germany, 2004.