Evolutionary Route Planner for Unmanned Air Vehicles

Changwen Zheng, Lei Li, Fanjiang Xu, Fuchun Sun, *Member, IEEE*, and Mingyue Ding, *Senior Member, IEEE*

***Abstract*—Based on evolutionary computation, a novel real-time route planner for unmanned air vehicles is presented. In the evolutionary route planner, the individual candidates are evaluated with respect to the workspace so that the computation of the configuration space is not required. The planner incorporates domain-specific knowledge, can handle unforeseeable changes of the environment, and take into account different kinds of mission constraints such as minimum route leg length and flying altitude, maximum turning angle, and fixed approach vector to goal position. Furthermore, the novel planner can be used to plan routes both for a single vehicle and for multiple ones. With Digital Terrain Elevation Data, the resultant routes can increase the surviving probability of the vehicles using the terrain masking effect.**

***Index Terms*—Evolutionary computation, route planning, unmanned air vehicles (UAVs).**

I. Introduction

Route planning is to generate a space path between an initial location and the desired destination that has an optimal or near-optimal performance under specific constraint conditions [6]. Such a mission is required in diverse applications, such as autonomous robotics, navigation and guidance of air, naval and ground forces, intelligence transportation systems, as well as the space-oriented applications [9], [12], [22]–[24]. However, in this paper, we concentrate our attention mainly on route planning problem of unmanned air vehicles (UAVs).

Compared with the route-planning problem in other applications, route planning for UAVs has the following attributes [3], [23].

1) *Stealth*: The air vehicles are usually required to carry out missions in threatened environments. In such a circumstance, stealth means safety. It is very important to minimize the probability of detection by hostile radar. There are two ways to achieve this. One is absorbing incoming radar radiation as much as possible and/or reflecting it in a direction different than the ambient direction, so that little is reflected back to the original radar site. The other is flying along a route which keeps away from perceived threats and/or has a lower altitude to avoid radar detection utilizing the masking of terrain.
2) *Physical Feasibility*: The physical feasibility of a route refers to the physical limitations from the use of UAVs. They include the following constraints.
    - *Maximum route distance*: It determines the elapsing time between the start and goal points and finally it depends on the fuel supply of aircraft.
    - *Minimum route leg length*: Due to inertia of motion, aircraft will fly straightly along a route for a certain distance before initiating a turn. We call it route leg length. In order to decrease navigational error, we must make the route leg length larger than a minimum threshold.
3) *Performance of Mission*: Each flight has its special mission. This depends on the application. In order to complete the mission, some requirements must be met when we design a route. These requirements include:
    - *Maximum turning angle*: This constrains the route allowing only to turn an angle less than or equal to a pre-specified threshold. Aircraft usually does not wish to make severe turns in some flight scenarios. For example, aircraft in tight formation cannot make severe turns without a greater risk of collision.
    - *Maximum climbing/diving angle*: This has the same definition as maximum turning angle but in altitude direction. It can be either positive or negative, which depends on its moving direction (it is positive if it is climbing). In order to decrease the risk of collision, a sharp climb or dive should be avoided.
    - *Minimum flying height*: As mentioned before, in order to reduce the probability of being detected by the hostile radar, the aircraft should fly along a low penetration route so as to enhance terrain-masking effect. But on the other hand, this definitely increases the probability of crashing into the ground. For this reason, we must maintain a minimum flying height for the entire route.
    - *Specific approaching angle to goal point*: In some applications such as attacking operation, an optimal direction is fixed *a priori*. Thus the aircraft should go along a specific direction to approach the goal point.
4) *Cooperation*: The route-planning algorithm must be compatible with the cooperative nature envisioned for the use of UAVs. In the future operation of UAVs, a flight mission

Manuscript received February 26, 2004; revised June 18, 2004. This paper was recommended for publication by Associate Editor H. Zhuang and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported in part by the China Postdoctoral Science Foundation under Grant 2004035424.

C. Zheng, L. Li, and F. Xu are with the General Software Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China (e-mail: cwzheng@hotmail.com; lil@admin.iscas.ac.cn; xufj@intec.iscas.ac.cn).

F. Sun is with the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: chunsheng@mail.tsinghua.edu.cn).

M. Ding is with the Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: mding@imaging.robarts.ca).

Digital Object Identifier 10.1109/TRO.2005.844684

might involve multiple UAVs. In such an application, the ability to coordinate the arriving time of each UAV will be vital in many missions [4], [13].

5) *Real-Time Implementation*: The flight environments of the UAVs are usually constantly changing. Therefore, our route-planning algorithm must be computationally efficient. The replanning ability of trajectory is critical for adapting to unforeseen threats.

It is demonstrated that an optimal solution of the route-planning problem is NP-complete in nature if we search it in an exhausting way [6], [22]. In order to speed up this procedure, a more advanced searching approach should be used. Nowadays, a variety of methods have been proposed, such as A* algorithm, dynamic programming, simulated annealing, and artificial potential field. Although a large number of approaches have been developed, there are a number of difficulties left in UAV-oriented applications.

1) *Space Representation:* Most traditional route planning algorithms can be considered either graphic-based (e.g., roadmap) or a grid-based (e.g., cell decomposition) in nature [7], [9], [12], [14]. They assume that a complete representation of configuration space (C-space) has been calculated before searching a path in a search map [8]. Building a search map and searching a path in such a map are the main computational complexities of this kind of approach. However, as the environment is varying, the C-space must be updated when new environment information is received, which makes this configuration very time-consuming. Usually, the complexity of this approach grows exponentially with the number of degrees of freedom of the vehicles.

   Local methods such as potential field approaches [11], [19] are more efficient in implementation time, but they often get trapped in local minimums. Probabilistic roadmap approaches [10], [21] incorporate local and global methods and offer a tradeoff between computation and route quality. However, these algorithms cannot incrementally update an existing roadmap when the workspace changes slightly. Thus, unless other mechanisms are added, the algorithms are unsuitable for online applications. Furthermore, these approaches are inefficacy in circumstance where obstacles and threat areas overlap or intersect with each other [20].

2) *Incorporation of Route Constraints:* Most conventional route-planning algorithms generate a minimum cost route based on a predetermined cost function. Such a route may not represent a desirable solution for many mission scenarios, which require several constraints on the resultant route (as detailed above). Szczerba *et al.* [23] tried to incorporate these constraints into the route-planning process using a modified A* algorithm (SAS). However, they modeled the planning environment into a two-dimension (2-D) C-space, thus their approach could not represent all constraints in UAVs, such as minimum flying height and maximum climbing/diving angle. Although they claimed that their approach could be extended to route planning in a higher dimensional case, it would become very memory and time consuming. Another shortcoming of this technique is that, when it speeds up the search process, it impairs the optimum of resultant route.

3) *Inefficacy in Seeking the Stealthy Route:* The route planning in a large geographical area is a typical large-scale optimization problem. The computational cost grows exponentially as the search space expands. In order to speed up the planning process and reduce the memory requirement, most conventional route-planning algorithms project the three-dimensional (3-D) environment to 2-D C-space. However, this 2-D C-space cannot embody all 3-D information of the environment (such as the terrain features) completely, which will impair us to generate an optimal route. Although some of them have calculated the areas where the perceived threats have intervisibility to the aircraft flying at a specified height [23], they can do nothing for unperceived or pop-up threats. There are some algorithms which model the planning environment with 3-D workspace [1], [17], [26], but they are inefficient in responding to mission-dependent route constraints.

4) *Incompatibility With the Cooperation*. The focuses of most exiting route-planning algorithms are concentrated on the route-planning problem for a single vehicle [3], [7], [23]. As using UAVs in a cooperative sense to accomplish a mission is one vision of their use, how to make the algorithm compatible with the cooperative nature envisioned for the UAVs is an important issue.

   There are a few approaches proposed to address the cooperative route-planning problem for multiple UAVs in recent years [4], [13], [14]. Chandler *et al.* [4] developed a Voronoi-diagram-based approach for UAVs' cooperative route planning. They significantly emphasize developing a route under the dynamic environments of the UAVs. The problem of simultaneous arrival of multiple UAVs at a target location was resolved by varying each UAV's velocity along its route. McLain *et al.* [13], [14] also presented a cooperative route-planning approach for multiple UAVs based on the Voronoi diagram. They use the combinations of path length and velocity to ensure simultaneous arrival. In order to avoid any collisions between UAVs, those approaches assumed that individual UAVs fly at different preassigned altitudes. More over, as they modeled the planning environment into a 2-D C-space, those approaches have the common shortcoming of this configuration.

5) *Inefficiency*: Szczerba *et al.* [23] demonstrated that no existing route planner could provide solutions in 30 s with the constraints given above. The SAS algorithm generates a 2-D route within a minute, not including the time for computing the C-space [23].

Evolutionary computation technique has been proven to be an efficient and effective way of dealing with NP-hard problem. Also, it can escape from local minima. As evolutionary approaches look for a solution in parallel, where individual candidates interact through genetic operators to generate possibly better solutions, they can easily be implemented on a massively parallel machine to achieve superlinear speed-up with the number of processors. Creative application of the evolutionary computation concept rather than dogmatic imposition of a standard algorithm proves to be more effective in solving specific types of real problems [8], [15], [25]. Using genetic algorithms (GAs), Pellazar [18], Yi *et al.* [26], and Nikolos *et al.* [17] each presented a route planner for aircrafts. However, they did not use domain-specific knowledge and could not handle various constraints on the resultant route.

Motivated by these observations, a novel route planner for UAVs in dynamically changing environments is presented in this paper. Combining the concepts of evolutionary computation with problem-specific representation of candidates and genetic operators, our approach overcomes the deficiencies of existing route planning algorithms. The main characteristic of our evolutionary route planner is that the individual candidates are evaluated with respect to the workspace so that the computation of the C-space is avoided. With digital terrain elevation data (DTED), our approach can find a near-optimal route that can increase the surviving probability efficiently by using the masking of terrain. Furthermore, our novel approach can be used to plan routes both for a single vehicle and for multiple ones. When it is used for cooperative route planning for multiple vehicles, candidate routes of each vehicle form their own subpopulation and evolve only in their own subpopulation, whereas the interactions among all subproblems is reflected by the means of the definition of fitness function. The experimental results illustrated that our algorithm can involve different kinds of mission constraints and generate a desired solution in real time.

The rest of this paper is organized as follows. Section II provides some basic definitions used in our approach. Section III gives a description of the representation used for encoding routes. Section IV presents the evaluation function of the route planner. Section V describes the genetic operators of the evolutionary route-planning algorithm. Section VI describes the evolutionary route-planning algorithm. The experimental results are given in Section VII, and our summary and discussion are presented in Section VIII.

II. Preliminaries

A. Workspace and Flight Route

As mentioned in Section I, most traditional route-planning algorithms need to build a search map before searching a route in the map. However, as the flight environment is constantly varying, the search map must be updated when new environment information is received, which makes this configuration very time-consuming and impedes its real-time in-flight application.

In our evolutionary route planner, all of the candidate routes are evaluated in workspace so that the computation of C-space can be avoided. We input the planning environment information as follows: 1) as most DTED is presented by grid, we can input the DTED of the environment using a 2-D grid directly and 2) other sources of planning environment, such as threats and weather conditions, were recorded in an environment information look-up (EILU) table. In the route-planning process, the DTED in workspace is constant whereas the information recorded in the EILU table is always changing and cannot be accurately known in advance. Once new information of the environment becomes available (for example, a perceived pop-up threat), the EILU table will be updated, and this process can be done in real time.

In our algorithm, the flight route consists of straight-line segments, e.g., a sequence of segments connecting the waypoints from the starting point to the goal point.

B. Cooperative Route Planning for Multiple UAVs

Consider a mission scenario where a group of UAVs is required to drop munitions on a known target location. The UAVs are equipped with a set of on-board sensors, through which they can sense their surroundings. There are a number of threats in the flight environment, some of them are known *a priori*, whereas others "pop-up" or become known only when a UAV maneuvers into its proximity. In order to maximize the probability of success, it is desirable to have multiple UAVs arrive at the target area simultaneously [2]. Our problem is to generate routes for the UAVs in real time, which takes into account each vehicle's exposure to the threats and enable the UAVs to arrive at their goal location simultaneously. The generated routes may be not optimal for an individual vehicle, but they should be optimal or near optimal for a team.

Assume that there are N UAVs participating in the flying mission. Each vehicle flies along its route with velocity constraints $v \in [V_{\min}, V_{\max}]$, which determines a range of estimated time of arrival (ETA). Suppose that the range of ETA of vehicle i is $[t^i_{\min}, t^i_{\max}]$. Considering that less time of exposure to threats is better, the ETA of the team is defined as

$$\min\left(\bigcap_{i=1}^{N}\left(\left[t^i_{\min}, t^i_{\max}\right]\right)\right). \tag{1}$$

III. Chromosome Structure

For any evolutionary computation technique, a chromosome representation is needed to describe each individual in the population of interest. The representation scheme determines how the problem is structured in the algorithm and the genetic operators that are used. Each individual or chromosome is made up of a sequence of genes from a certain alphabet. An alphabet can consist of binary digits, floating-point numbers, integers, symbols (i.e., A, B, C, D), or matrices. In early GAs, the binary digit alphabet was used. Since then, problem representation has been extensively addressed. It has been shown that more natural representations can get more efficient and better solutions. Michalewicz [15] has performed extensive experiments comparing real-valued and binary GAs and shown that the real-valued GA is more efficient in terms of CPU time. His experiments also showed that a real-valued representation moves the problem closer to the problem representation that offers higher precision with more consistent results across replications.

In our method, a real-valued problem-specific chromosome representation is used, e.g., a chromosome corresponds to a flight route. As illustrated in Fig. 1, each node specified by the coordinates (x_i, y_i, z_i) of intersection points between line segments s_i and s_{i+1}. Each node also contained a link to the next node of the same chromosome, and a state variable b_i, providing

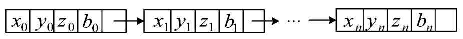


Fig. 1. Linked list chromosome representing a flight route. Each node contains x, y, and z coordinates of the nodes, together with a state variable b, which provides information on feasibility of the point and the following route segment. Here, the point (x_0, y_0, z_0) is the starting point and the point (x_n, y_n, z_n) is the goal point (note that the starting and the goal points of all routes for a vehicle are fixed).

information such as whether or not: 1) the point is feasible (i.e., without constraints violation) and 2) the route segment connecting the point to the next point is feasible (i.e., without intersecting constraints violation areas). A route is feasible if and only if it has only feasible nodes and route segments. Thus, a route (or chromosome) can be either feasible or infeasible.

An initial population of chromosomes can be randomly generated. All chromosomes of each vehicle form their own subpopulation. The maximum length of a chromosome, i.e., the number of nodes, is an input parameter of the system. Therefore, each chromosome has a random number of intermediate nodes and each intermediate node has coordinates generated randomly. Note that the start node (x_0, y_0, z_0) and the end node (x_n, y_n, z_n) are always the same for the chromosomes in the same subpopulation.

IV. Evolutionary Evaluation Function

An evaluation function is used to define a fitness value to each candidate route. The fitness value of a route is a measurement of its "goodness". Because the individuals in each subpopulation evolve internally, the definition of evaluation function is crucial for reflecting the interactions among subproblems. This section provides an evaluation method for chromosome that takes account of the cost of the route and the mission scenario constraints, and also combines the cooperative and coordinative requirements among the vehicles.

A. Cost of Flight Route

The cost function of flight route is defined as follows:

$$C = \sum_{i=1}^{n} \left(w_1 l_i^2 + w_2 h_i^2 + w_3 f_{TAi}\right) \tag{2}$$

where l_i is the length of the ith segment which penalizes the length of the route to prevent the aircraft from wandering too far away from the line connecting the start and target points, h_i is the average altitude above the sea level of the ith route segment which minimizes the aircraft's altitude and causes the algorithm to seek a lower altitude penetration route enhancing the terrain masking effect, and f_{TA} penalizes penetration routes that come dangerously close to "known" ground threat sites. In our situation, f_{TA} is expressed as follows:

$$f_{TA} = \sum_{j=1}^{N_{\text{Site}}} \frac{K_j}{(R_{Sj})^4} \tag{3}$$

where K_j is a scale which reflects the intensity of the jth known threat, R_{Sj} is the aircraft's slant range to the threat site, and N_{Site} is the number of distinct threat sites. For the threat sites whose locations are not precisely perceived, we assume that they are uniformly distributed in the penetration area. The cost function reduces the aircraft's altitude above sea level and enhances the terrain masking effect. Therefore, the second term of (2) will become the primary penalty term at this situation [1]. The weighting coefficients w_1, $w_{2,}$, and w_3 control the effects of flying over terrain obstacles, threat sites, and flying around them on the cost function.

B. Constraint Expression in Route Planning

Because a route consists of a series of route segments, the constraints on route can be converted to that on route segments.

1) *Minimum Route Leg Length*: Assume that the minimum route leg length is l. Then this constraint can be written as

$$l_i \geq l, \qquad (i = 1, \ldots, n). \tag{4}$$

2) *Maximum Route Distance*: Suppose the maximum route distance as L; we have

$$\sum_i l_i \leq L. \tag{5}$$

3) *Minimum Flying Height*: Record the minimum altitude above the terrain of the ith route segment as H_i, the minimum flying height constraint as H. Then the minimum flying height constraint can be expressed as

$$H_i \geq H, \qquad (i = 1, \ldots, n). \tag{6}$$

4) *Maximum Turning Angle*: Record the coordinates of the ith way point as (x_i, y_i, z_i), set $a_i = (x_i - x_{i-1}, y_i - y_{i-1})^T$, and assume that the maximum turning angle is θ. Then the maximum turning angle constraint can be written as

$$\frac{a_i^T a_{i+1}}{|a_i||a_{i+1}|} \geq \cos(\theta), \quad i = 2, \ldots, n-1 \tag{7}$$

where $|a|$ is the length of the vector a.

5) *Maximum Climbing/Diving Angle*: Assume that the maximum climbing/diving angle is β, then the maximum climbing/diving angle constraint can be expressed as:

$$\frac{|z_i - z_{i-1}|}{|a_i|} \leq \tan(\beta), \quad i = 2, \ldots, n. \tag{8}$$

6) *Simultaneous Arrival at Goal Locations*: As discussed in Section II-B, this is one of the primary mission requirements in multi-UAV route planning. According to (1), in order to arrival at goal locations simultaneously, the intersection of the ETA ranges of different vehicles should not be empty. When examining a route p_i with ETA range of $[t_{\min}^i, t_{\max}^i]$, we select p_j from each of the other subpopulations. Assuming that the ETA range of p_j is $[t_{\min}^j, t_{\max}^j]$, we have the constraint

$$\begin{cases} t_{\min}^i \leq t_{\max}^j, \\ t_{\max}^i \geq t_{\min}^j, \end{cases} j = 1, \ldots, N; \qquad j \neq i. \tag{9}$$

In the first generation, p_j is selected randomly from the other subpopulations. After that, p_j is the route with the best current performance in its subpopulation.

7) *No Collision Between Vehicles*: When examining route p_i on this constraint, we select $p_j, j = 1, \ldots, N\ j \neq i$, from each of the other subpopulations using the above method.

Supposing that all vehicles fly along the selected routes with the same ETA and the minimal distance between the vehicles is d during the course, then this constraint can be expressed as

$$d \geq d_s \tag{10}$$

where d_s is the minimal safe distance. Note that, when checking this constraint, the velocity constraint can be neglected and the ETA can be given arbitrarily.

A specific approaching angle constraint is not listed here but will be merged into the genetic operator, which will be illustrated in Section V.

C. Evaluation Function

From the discussion above, we can conclude that flight route planning is to minimize the cost function of (2) under a number of constraints using genetic operators. If it is route planning for a single vehicle, only the first five constraints are involved. While if it is cooperative route planning for multiple UAVs, all seven constraints should be included.

Michalewicz and Schoenauer [16] have discussed different constraint-handling methods used in GAs and suggested using the static penalty function method. The traditional penalty function approach involves a number of penalty parameters that must be set right to obtain feasible solutions. This is time-consuming work and usually requires extensive experimentation to set up appropriate parameters.

In our approach, we use a penalty function approach that does not contain any penalty parameters, but the following criteria are enforced when two routes are compared with each other [5].

1) Any feasible route has better fitness than any infeasible route in the same subpopulation.
2) Between two feasible routes from the same sub population, the one with less cost has better fitness.
3) Between two infeasible routes from the same sub population, the one with smaller constraint violation has better fitness.

As all candidate routes evolve only in their own subpopulation, we need not compare two routes from difference subpopulations.

In traditional penalty function approaches, penalty parameters are needed to make the constraint violation values of the same order as the objective function value. In our approach, penalty parameters are not needed because, in any of the above three cases, routes are never compared in terms of both cost function and constraint violation information. Moreover, the idea of comparing infeasible routes only in terms of constraint violation has a practical implication. In order to evaluate any route, it is a usual practice to check its feasibility first. If the route is infeasible (that is, at least one constraint is violated), it is not needed to compute its cost function value. It does not make sense to compute the cost function value of an infeasible solution, because the route simply cannot be flown in practice.

Motivated by these arguments, we introduce the following evaluation function of route p, where infeasible routes are compared based on only their constraint violation:

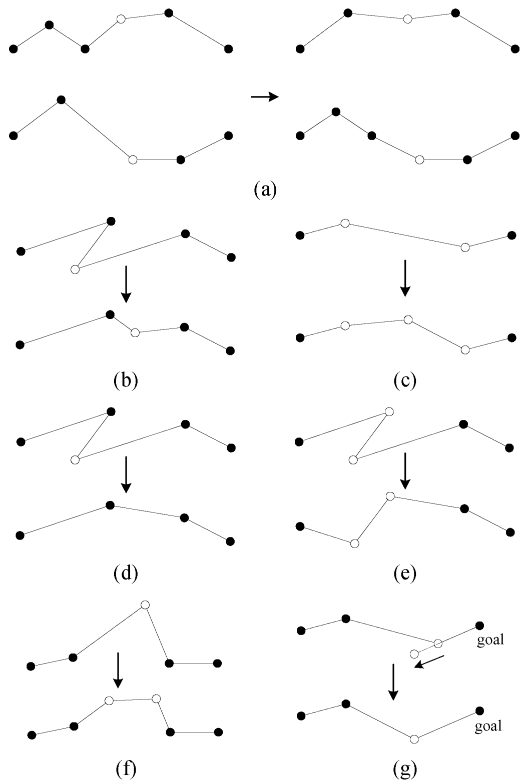


Fig. 2. Evolutionary operators. (a) Crossover. (b) Perturb mutator. (c) Insertion mutator. (d) Deletion mutator. (e) Swap mutator. (f) Smooth mutator. (g) Fixed vector mutator. In the six mutators, the upper part of each diagram represents a subroute before the operator is applied, whereas the lower shows a possible outcome after application of the operator. In the crossover diagram, the left part represents two subroutes of the parents before the operator is applied, whereas the right part shows the possible outcome after application of the operator.

$$F(p) = \begin{cases} C(p), & \text{if } p \text{ is feasible} \\ C_{\max} + \text{constraint violation values}, & \text{if } p \text{ is unfeasible} \end{cases} \tag{11}$$

where $C(p)$ is the cost function value of p defined in (2). The parameter $C_{\max}$ is the cost function value of the worst feasible solution in its subpopulation. Thus, the evaluation value of solution not only depends on the amount of constraint violation, but also on the population of solutions at hand. However, the evaluation value of a feasible solution is equal to its cost function value. In order to avoid any bias from any particular constraint, all constraints should be normalized when computing the constraint violation values [(4) is normalized as $l_i/l - 1 \geq 0$, $i = 1, \ldots, n$, for example]. If no feasible solution exists in a subpopulation, $C_{\max}$ is set to zero.

V. Evolutionary Operators

Our evolutionary route planner uses seven types of operators: one crossover operator and six mutation operators. These operators manipulate the route intermediate nodes to evolve chromosomes into possibly better ones. These operators are sufficient to generate an arbitrary route but may not all be applicable or needed in all situations. The application of each operator is controlled by a probability. Note that all operators only change the intermediate nodes of a route. Now we describe these seven operators, which are also illustrated in Fig. 2.

A. Crossover

Crossover recombines two parent routes into two offspring routes. This operator is applied on both feasible and infeasible routes. The parent routes are divided randomly into two parts respectively and recombined: the first part of the first route with the second part of the second route, and the first part of the second route with the second part of the first route. Note that there can be different number of nodes in the two parent routes.

B. Perturb Mutator

The perturb mutator is used to impose a random change to a node's coordinates in a route, which can be either feasible or infeasible. Given a route, the operator randomly selects an intermediate node and changes the coordinates of this node randomly. When the selected route is feasible, the change of coordinates should be within the local feasible area.

C. Insert Mutator

The insert mutator operates on a feasible or infeasible route by inserting randomly generated new nodes into a route segments. This operator is more likely applied on a route segments above a mountain, which violates the minimum flying height constraint or a route segments near a thread site.

D. Delete Mutator

Delete mutator removes nodes from a route, which can be either feasible or infeasible. If the route is infeasible, nodes for deletion are selected randomly in the chromosome. Otherwise, the operator decides whether a node should be deleted based on some heuristic knowledge. In the case where there is no knowledge supporting the deletion of a node, its selection for deletion is decided randomly with a small probability.

E. Swap Mutator

The swap mutator exchanges the coordinates of randomly selected adjacent nodes in a chromosome to eliminate two consecutive sharp turns. The probability for selecting a node n_i and the next node n_{i+1} is proportional to the sharpness of the two turns (measured by angles between the route segments) at the two nodes. This operator is applied on infeasible routes only.

F. Smooth Mutator

The smooth mutator smoothes turns of a route by "cutting corners," i.e., for a selected node the operator inserts two new nodes on the two route segments connected to that node respectively and deletes that selected node. The nodes with sharper turns are more likely to be selected. This operator is also applied on infeasible routes only.

G. Fixed Vector Mutator

The fixed vector mutator is designed to meet the constraint of specific approaching angle to the goal position and just applied on the node which lies before the goal point. If this constraint is required, all routes are initiated according to the fixed approach vector. During the evolutionary process, if the operator is applied to a chromosome, the coordinates of the $(n-1)$th node are modified according to

$$\begin{cases} x'_{n-1} = x_n + r(x_{n-1} - x_n) \\ y'_{n-1} = y_n + r(y_{n-1} - y_n) \\ z'_{n-1} = z_n + r(z_{n-1} - z_n) \end{cases} \tag{12}$$

where $(x'_{n-1}, y'_{n-1}, z'_{n-1})$ is the coordinates of the $(n-1)$th node after the operation, and $r > 0$ is a random real number.

VI. Evolutionary Route-Planning Algorithm

A. Description of the Evolutionary Route-Planning Algorithm

In our planner, a chromosome represents a flight route, consisting of line segments, as a sequence of intersection points (see Section III for details on the chromosome structure). All candidate routes of each vehicle form their own subpopulation (If it is route planning for a single vehicle, there is only one subpopulation). Each subpopulation consists of P (feasible or infeasible) flight routes determined by the same start and goal position. Each individual is evaluated by using the evaluation criterion described in Section IV.

In the evolutionary loop, a set of $S(\leq P)$ individuals from each subpopulation is selected for evolutionary crossover and mutation. A roulette wheel of P slots is used to implement the selection process. The slot sizes are linearly decreasing according to the rank of the individuals' fitness value [15]. In this way, we need not generate a different roulette wheel at each generation. The chance for a chromosome to be selected is not necessarily proportional to its fitness value but is still better than the chance for a worse chromosome.

An evolutionary operator is selected on the basis of a probability distribution. The crossover operator transforms two individuals (parents) into two new offspring by combining parts from each parent. The mutation operator operates on a single individual and creates an offspring by mutating that individual (see Section V for details on evolutionary operators). Note that the last mutation operator is adopted and applied to the $(n-1)$th node if and only if the specific approaching angle constraint is required. The other five mutation operators are applied to the other nodes or route segments. The newly generated individuals are added to their parents' original subpopulation and evaluated on the basis of the fitness function. The worst S individuals of the extended subpopulation are then removed to reduce the population to its original size. In this scheme of evolution, the new individuals may only make it to the next generation if they have a better fitness value than the worst individuals in the original subpopulation. The process terminates after some number of generations, which can be fixed either by the user or determined dynamically by the program itself, and the best chromosomes of each subpopulation presents the near-optimum route found.

The evolutionary route-planning algorithm is described as follows.

1) Initialize a group of routes with size of P for each vehicle. If the fixed approach vector constraint is required, all routes are initiated according to their vehicles' approach vector respectively.
2) Evaluate each candidate route.

3) If the termination condition is not reached, go to Step 4. Otherwise, select the best individual from each subpopulation as the near-optimal route and terminate execution.
4) Select S individuals from each subpopulation for crossover and mutation.
5) Apply crossover operator to the selected individuals based on the crossover probability.
6) Select a mutation operator based on operator selection probabilities and probabilistically apply it to the selected individuals. Note that if the specific approaching angle constraint is required, all six mutation operators are used. Otherwise, just the first five mutation operators participate in the evolutionary computation.
7) Add the newly generated individuals to their parents' original subpopulation.
8) Evaluate the newly generated candidates.
9) Remove the worst S individuals from each subpopulation to return the population to its original size. The newly created individuals may only survive to the next generation if they are better than the worst individual in their subpopulation.
10) Go back to Step 3.

B. In-Flight Route Replanning Using the Evolutionary Route Planner

Our evolutionary route-planning algorithm could be used for both off-line route planning and in-flight route replanning. At the beginning, the evolutionary route planner is used to find an optimal or near optimal route for each vehicle with the same ETA. Assume that the vehicles have a range of view by their on board sensors. When the vehicles fly along the planned routes toward their goal positions, they sense the environment with their on board sensors. If new information of the environment is perceived by any of the vehicles (a pop-up threat, for example), it notifies the new information to all of its teammates, and all of the vehicles update their EILU tables. As the vehicles are in flight, the time is too limited to generate near-optimal routes for the vehicles. In such an urgent situation, the planner can be used to find a feasible route for each vehicle firstly. Then the vehicles move a specific step along their feasible routes respectively. In the same time the planner updates the values of all potential routes due to the new starting position and carries through the evolutionary process until the near-optimal routes are found. This process is repeated until the vehicles reach their goal positions.

The route replanning algorithm using the evolutionary planner is described as follows.

1) Use the evolutionary route planner to obtain the optimal or near optimal route for each vehicle.
2) If the vehicles reach their goal positions, terminate execution.
3) The vehicles fly along the current optimal routes and sense the battle environment.
4) If new information of the environment is perceived by any of the vehicles, go to Step 5. Otherwise, go back to Step 2.
5) Notify the new information to all of the other vehicles.
6) All vehicles update their EILU tables and their current positions.
7) Generate a feasible route for each vehicle using the evolutionary planner.
8) The vehicles move a specific step along their feasible routes, respectively.
9) Update the start nodes of all potential routes and carry through the evolutionary process until the near-optimal routes are found.
10) Go back to Step 3.

VII. Experimental Results

The evolutionary route-planning algorithm was implemented in a Visual C++ 6.0 programming environment on a Pentium IV PC running Windows 2000. No commercial EA tools were used. The experiments were conducted using a DTED with a resolution of 100 m $\times$ 100 m and different sets of synthetic threat data were tested. In all experiments, the same set of parameter values shown below were used.

- All genetic operators' usage rates are set to be equal. That is 1/7 when the approaching angle constraint is required and 1/6 when it is not required.
- $P = 60$ and $S = 30$. Each subpopulation comprises of 60 individuals. Thirty of all 60 individuals are selected for reproduction at each generation.
- If there is not additional explanation, the coefficients w_1, w_2, and w_3 in the cost function are 1.0, 1.0, and 1.0. The coefficients of the constraints are set as: L is 2.5 times the length of the straight-line distance between the start and goal locations. l, H, θ, and β are 1200 m, 30 m, 45°, and 30°, respectively.
- All simulations are terminated after 120 generations.

A. Route Planning for a Single Vehicle

Figs. 3–6 show the experiment results of route planning for a single vehicle using the evolutionary planner in different planning environment. In the figures of the route projections on the XY plane, filled circles at the endpoints of the route represent the start and goal locations. The lighter larger circles represent areas of perceived threats. In the route vertical profile figures, the curves on the top are the profiles of the route and the shaded areas at the bottom are the terrain contours under the routes.

Fig. 3 shows the route generated for a single vehicle in a 3-D environment. In this figure, Fig. 3(a)–(c) displays the snapshots of the evolution process. For each snapshot, only the best 20 routes are displayed. Fig. 3(d) shows the best route after generation 120; Fig. 3(e) shows the vertical profiles of the best route and Fig. 3(f) is its 3-D show.

Figs. 4–6 display the planning results of experiment in three different threat environments. Part (b) of these figures show the best current performance of the evolutionary process versus generations count. From them, we can see that different generations of evolution are needed to obtain the near-optimal result. The summit of (b) is the point that the first feasible route is found. In general, 30 generations are sufficient to find a feasible route in the single vehicle scenario and 80 generations are sufficient to obtain a near-optimal one. Almost all of the routes converge together after 120 iterations.

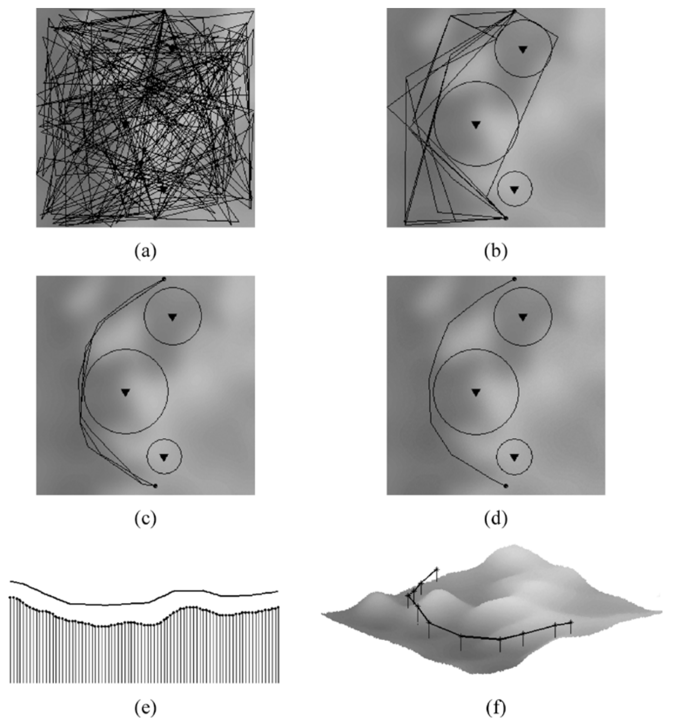


Fig. 3. Results of route planning for a single vehicle. (a) Generation 0: routes are initiated. (b) Generation 20: evolution has taken 0.312 s. (c) Generation 60: evolution has taken 1.468 s. (d) The best route of generation 120: evolution has taken 3.648 s. (e) The vertical profiles of (d). (f) The 3-D show of (d).

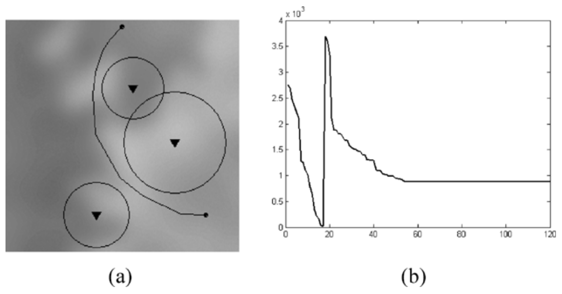


Fig. 4. Result of route planning for a single vehicle. (a) Projections on the XY plane. (b) Performance versus generation diagram.

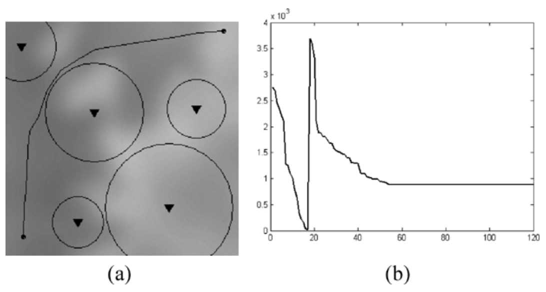


Fig. 5. Result of route planning for a single vehicle. (a) Projections on the XY plane. (b) Performance versus generation diagram.

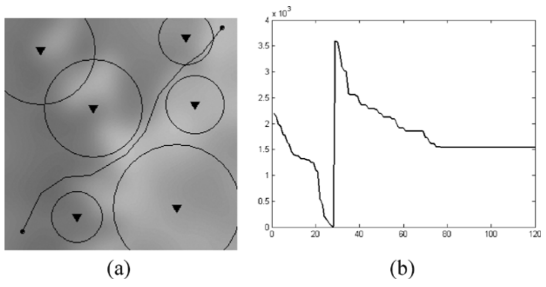


Fig. 6. Result of route planning for a single vehicle. (a) Projections on the XY plane. (b) Performance versus generation diagram.

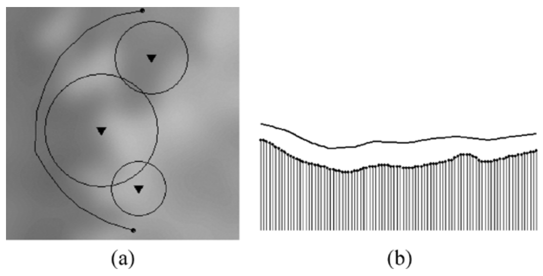


Fig. 7. Result of experiment with more emphasis on the altitude. (a) Projection on the XY plane. (b) Vertical profiles.

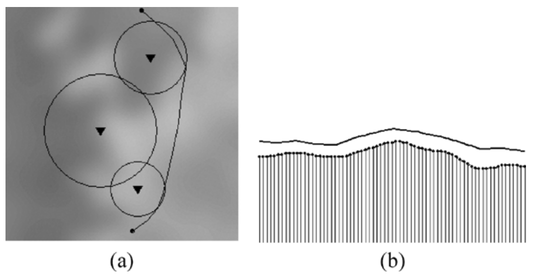


Fig. 8. Result of experiment with smaller maximum route distance. (a) Projection on the XY plane. (b) Vertical profiles.

It is an important characteristic of our algorithm that the evolutionary route planner can find feasible routes for the vehicles very quickly (less than 0.5 s for a single vehicle). In many situations, the air vehicle is required to carry out an instant movement and even 1 s to plan a route is too long (escaping from a threat or carrying out an urgent rescue mission, for example). In such a circumstance, time is much more important than the route's optimality. With the evolutionary route planner, the vehicle can move a specific step along the first feasible route found. In the same time, the planner updates the start nodes of all potential routes and continues the evolutionary process until the best route is found.

Figs. 7–9 show the planning results of experiment with different parameters in the same environment. In Fig. 7, the evolutionary algorithm gives more emphasis on the second factor of the route cost function [$w_2 = 3.0$ in (2)], to seek a lower altitude penetration route for enhancing terrain masking. Note that the resultant route penetrates through the low area of the southwest. In Fig. 8, the evolutionary algorithm is only allowed to plan a route with maximum route distance equal to 1.4 times the straight-line distance between the start and goal, severely restricting how far away from the threat areas the route could go. Note that the generated route cuts through the threat circle, and goes along the high area of the southeast. In Fig. 9, the algorithm is requested to generate a route with a fixed approach vector of (1, 1, 0) into the goal (coming into the goal evenly from the southwest). From these figures, it can be seen that our algorithm can accommodate different optimization criteria.

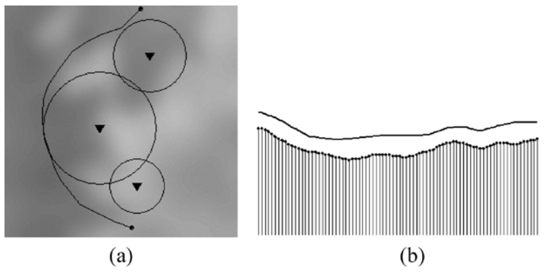


Fig. 9. Result of experiment with fixed approach vector. (a) Projection on the XY plane. (b) Vertical profiles.

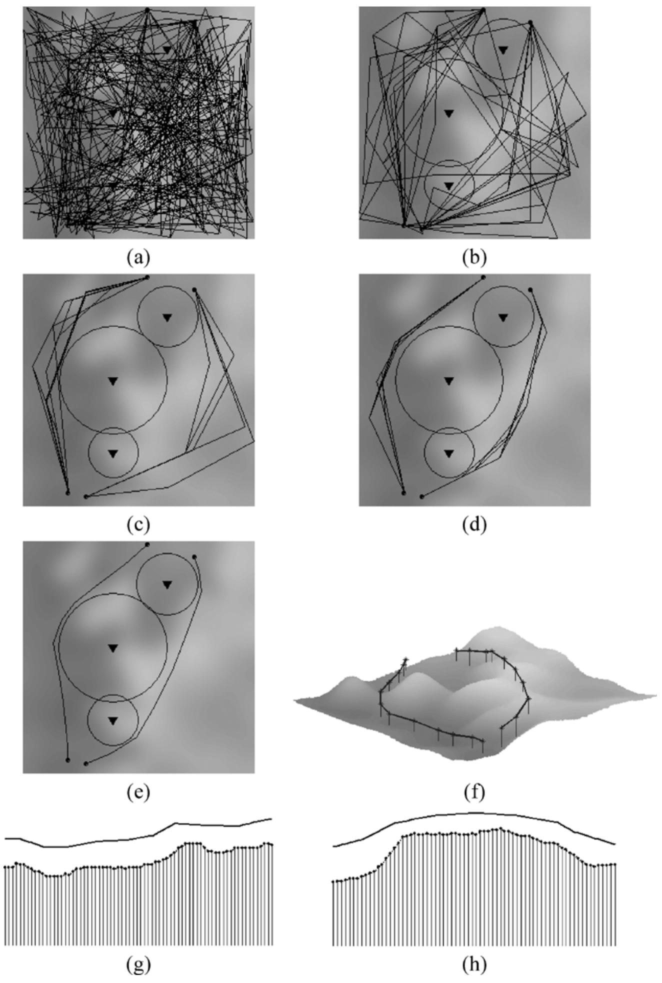


Fig. 10. Result of cooperative route planning for two vehicles. (a) Generation 0: the initiated routes. (b) Generation 20: evolution has taken 0.646 s. (c) Generation 40: evolution has taken 1.602 s. (d) Generation 80: evolution has taken 3.924 s. (e) The best routes at generation 120: evolution has taken 6.318 s. (f) The 3-D show of the best routes. (g), (h) Vertical profiles of the best routes.

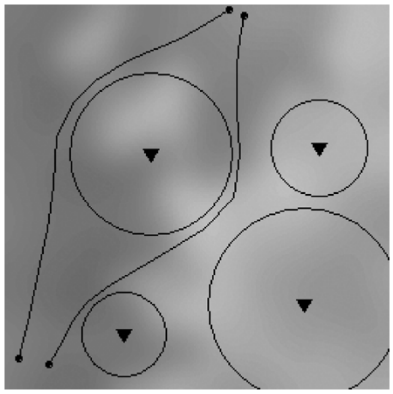


Fig. 11. Cooperative route planning for two vehicles.

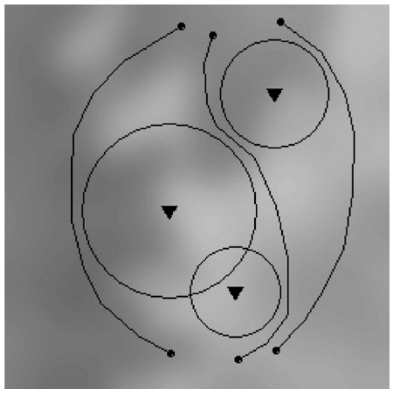


Fig. 12. Cooperative route planning for three vehicles.

B. Cooperative Route Planning for Multiple Vehicles

Figs. 10–12 display the experimental results of cooperative route planning for multiple vehicles in different experiment environments. In these experiments, the velocity constraints of vehicles are assumed to be $v_{\min} = 121.00$ m/s and $v_{\max} = 148.00$ m/s. The minimal safe distance d_s is set to be 600 m. Fig. 10 shows the routes generated for two UAVs in a 3-D environment. In this figure, Fig. 10(a)–(d) displays the snapshots of the evolution process. For each snapshot, only the best 20 routes of each subpopulation are displayed. Fig. 10(e) displays the best routes after generation 120; Fig. 10(f) shows the 3-D show of the best routes; Fig. 10(g) and (h) shows the vertical profiles of the best routes. The length of the route for UAV 1 is 114 829.20 m and that for UAV 2 is 114 545.39 m. The UAV 1 flies along its route at a speed of 148.00 m/s and UAV 2 at 147.63 m/s. Two vehicles will arrive at their goal location with the same ETA of 775.87 s. Fig. 11 shows the cooperative route-planning result for two vehicles in another environment, and Fig. 12 displays the cooperative route-planning result for three vehicles.

Table I shows the time and generations that the evolution takes to generate the first feasible and near-optimal routes in the experiments. As shown in Table I, the computation time and generation number are most strongly related to the number of threats and vehicles.

The threats number affects the complexity of solution in two ways. First, as the number of threats increases, the vehicles need to take more turns to fly around these threats. Therefore, the routes have more segments to be checked. Second, the number of threats also has an impact on the computation time of the cost function.

The vehicles number also impacts the complexity of the problem in two ways. First, the vehicles number determines the number of subpopulations. Second, an increase in the vehicles number results in more constraint examination of simultaneous arrival and no collision between vehicles on the routes. As the candidate routes of each vehicle form their own subpopulation and evolve internally, this mechanism is very suitable for parallel computation, which will decrease the run time efficiently.

TABLE I
TIME AND GENERATIONS FOR FEASIBLE AND NEAR-OPTIMAL ROUTE IN THE EXPERIMENTS

Experiments	Vehicles number	Threat number	Sub-population	First feasible route		Near-optimal route	
				Generations	Time (Secs)	Generations	Time (Secs)
1 (Fig. 4)	1	4	1	19	0.302	55	1.262
2 (Fig. 5)	1	5	1	21	0.313	60	1.502
3 (Fig. 6)	1	6	1	24	0.375	74	2.065
4 (Fig. 10)	2	3	1	22	0.702	76	3.721
			2	23	0.712	76	3.721
5 (Fig. 11)	2	4	1	26	0.804	79	3.894
			2	28	0.826	80	3.951
6 (Fig. 12)	3	3	1	29	0.980	85	4.847
			2	29	0.980	87	4.984
			3	31	1.007	86	4.915

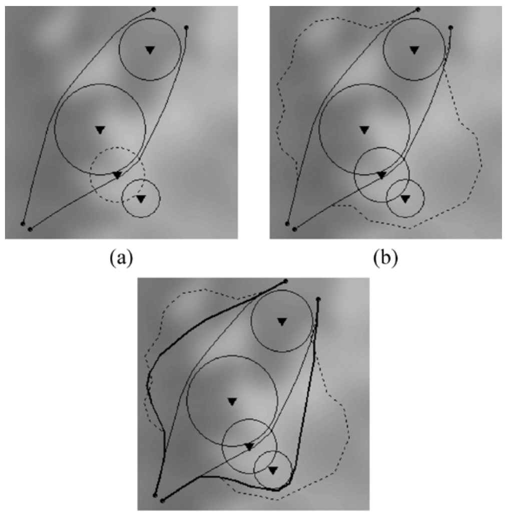


Fig. 13. Online route replanning experiment. (a) Initial best routes. (b) Replanned feasible routes. (c) Replanned near-optimal routes.

The roughness of the terrain has a much slighter impact on the run time of solution. There are two reasons for this. The first is that the computation complexity of route evaluation is not related to the terrain roughness directly. The second is that the constraints of maximum climbing/diving angle, minimum route leg length, and flying altitude prevent the routes from following the terrain too closely, which weaken the impact of the terrain roughness on the routes.

C. Online Route Replanning Experiment

The results of on-line route replanning for two vehicles using the evolutionary route planner are illustrated in Fig. 13. There are four threats in the planning area, one of them is pop-up (marked by dashed circle). The starting points of the two vehicles are in the southwest corner, and the goal points are close to the northeast corner of the area. At the beginning, the evolutionary route planner generates two best routes for the UAVs with the known environment information, which is shown in Fig. 13(a). The vehicles fly along their routes respectively and sense the threat. During the vehicles' motion, the fourth unknown threat appears in the view range of the left vehicle, the threat becomes known and the EILU tables of the two vehicles are updated. Then the evolutionary planner generates two feasible routes for the vehicles with the new information, which are represented by dashed lines in Fig. 13(b). The vehicles move a specific step along their feasible routes, and the planner updates the start node of all potential routes and carries through the evolutionary process. The new near-optimal routes are then generated and are represented by overstriking lines in Fig. 13(c). The evolution takes 0.817 s to generate a feasible route for each of the vehicles and takes another 3.162 s to gain the near-optimal ones.

VIII. CONCLUSION AND FUTURE WORK

This paper has proposed an evolutionary algorithm for solving 3-D route-planning problems for unmanned air vehicles. The evolutionary planner utilizes domain-specific knowledge for making decisions and can accommodate different optimization criteria. During the planning process, the computation of the C-space is not required as the candidate routes are evaluated with respect to the workspace. The implementation and experiments described in this paper have demonstrated that our evolutionary route planner is an efficient robust algorithm that is able to handle different kinds of mission parameters and generate routes with computation times that are acceptable for real-time in-flight applications.

When the evolutionary planner is used for cooperative route planning for multiple vehicles, candidate routes of each vehicle form their own subpopulation and evolve internally, whereas the interactions among all subproblems is reflected by the means of the definition of fitness function of each subpopulation. Such a mechanism is very suitable for parallel computation, which will decrease the run time of solution efficiently.

An important issue in our future work is how to enhance the performance of our evolutionary route planner. A possible way is to make the evolutionary algorithm capable of adapting its parameter values based on the states of evolution. Currently, all operators have the same probabilities of application. However, different operators may have different impacts at different stages of the evolution process. For example, at early stage of evolution, most routes are infeasible; the roles of *Swap* should be very significant. As the evolution process goes on, their importance should shrink; at the same time, the significance of other operators should increase. Another possible way is to further incorporate domain-specific knowledge in important components/processes of the evolutionary route planner. Although we have incorporated domain knowledge in both chromosome representation and evolutionary operators, there are other components/processes, such as the initialization process, which may benefit from more knowledge. For example, rather than random initialization, an initial population may consist of some feasible routes created by mutating the straight line between start and goal locations and some routes generated randomly.

Acknowledgment

The authors wish to thank the anonymous referees for their very valuable comments and constructive suggestions which helped improve the quality of the paper.

References

[1] S. J. Asseo, "Terrain following/terrain avoidance path optimization using the method of steepest descent," in *Proc. Nat. Aerosp. Electron. Conf.*, Dayton, OH, 1988, pp. 1128–1136.
[2] R. Beard, T. McLain, M. Goodrich, and E. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.
[3] S. Bortoff, "Path planning for UAVs," in *Proc. Amer. Control Conf.*, Chicago, IL, 2000, pp. 364–368.
[4] P. Chandler, S. Rasmussen, and M. Pachter, "UAV cooperative path planning," in *Proc. AIAA Guid., Navigat. Control Conf.*, Denver, CO, 2000, Paper 2000-4370 [CD-ROM].
[5] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, pp. 311–338, Mar. 2000.
[6] J. F. Gilmore, "Autonomous vehicle planning analysis methodology," in *Proc. Assoc. Unmanned Veh. Syst. Conf.*, Washington, DC, 1991, pp. 503–509.
[7] J. Goldman, "Path planning problems and solutions," in *Proc. Nat. Aerosp. Electron. Conf.*, Dayton, OH, 1994, pp. 105–108.
[8] C. Hocaoğlu and A. C. Sanderson, "Planning multiple paths with evolutionary speciation," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 169–192, Jun. 2001.
[9] Y. K. Hwang and N. Ahuja, "Gross motion planning—A survey," *ACM Comput. Surveys*, vol. 24, pp. 219–291, Sep. 1992.
[10] L. E. Kavraki, P. Švestka, J. -C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
[11] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, 1988, pp. 1778–1784.
[12] J. -C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
[13] T. McLain, P. Chandler, and M. Pachter, "A decomposition strategy for optimal coordination of unmanned air vehicles," in *Proc. Amer. Control Conf.*, Chicago, IL, 2000, pp. 369–373.
[14] T. McLain, P. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," in *Proc. Amer. Control Conf.*, Arlington, VA, 2001, pp. 2309–2314.
[15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.
[16] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, pp. 1–32, Jan. 1996.
[17] I. K. Nikolos *et al.*, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Trans. Syst. Man. Cybern. B, Cybern.*, vol. 33, no. 12, pp. 898–912, Dec. 2003.
[18] M. Pellazar, "Vehicle route planning with constraints using genetic algorithms," in *Proc. Nat. Aerosp. Electron. Conf.*, Dayton, OH, 1994, pp. 111–118.
[19] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
[20] D. Sent and M. H. Overmars, "Motion planning in environments with danger zones," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 1488–1493.
[21] P. Švestka and M. H. Overmars, "Motion planning for car-like robots using a probabilistic learning approach," *Int. J. Robot. Res.*, vol. 16, pp. 119–143, Apr. 1997.
[22] R. J. Szczerba, "Threat netting for real-time, intelligent route planners," in *Proc. IEEE Symp. Inf., Decision, Contr.*, Adelaide, Australia, 1999, pp. 377–382.
[23] R. J. Szczerba, P. Galkowski, I. S. Glickstein, and N. Ternullo, "Robust algorithm for real-time route planning," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 7, pp. 869–878, Jul. 2000.
[24] C. W. Warren, "A technique for autonomous underwater vehicle route planning," *IEEE J. Ocean. Eng.*, vol. 15, no. 3, pp. 199–204, Jul. 1990.
[25] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 18–28, Apr. 1997.
[26] M. Yi, M. Ding, and C. Zhou, "3D route planning using genetic algorithm," in *Proc. SPIE Int. Symp. Multi-Spectral Image Process.*, Wuhan, China, 1998, pp. 92–95.

Changwen Zheng received the B.S. degree in mathematics from Huazhong Normal University, Wuhan, China, in 1992, and the Ph.D. degree in control science and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2003.

He is currently engaged in postdoctoral research with the General Software Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China. His current research interests include route planning, evolutionary computation, neural networks, and computer communication networks.

Lei Li received the B.S. and Ph.D. degrees in computer science and engineering from the National University of Defense Technology, Changsha, China, in 1994 and 1999, respectively,

He is currently the Vice Director of the Institute of Software, Chinese Academy of Sciences, Beijing, China. His research interests include computer operation systems, network security and management, neural networks, artificial intelligence, and e-government.

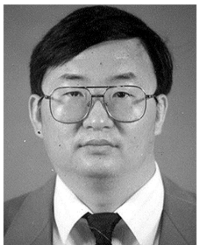
Fanjiang Xu received the B.S. and M.S. degrees in computer science and engineering from the National University of Defense Technology, Changsha, China, in 1994 and 1997, respectively.

He is currently the Vice Director of the General Software Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China. His research interests include network security and management, artificial intelligence, and route planning.

Fuchun Sun (S'94–M'98) received the B.S. and M.S. degrees from the Naval Aeronautical Engineering Academy, Yantai, China, in 1986 and 1989, respectively, and the Ph.D. degree from Tsinghua University, Beijing, China, in 1998.

He was with the Department of Automatic Control, Naval Aeronautical Engineering Academy, for over four years. From 1998 to 2000, he was a Postdoctoral Fellow with the Department of Automation, Tsinghua University. He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include intelligent control, neural networks, fuzzy systems, variable structure control, nonlinear systems, and robotics.

Dr. Sun was the recipient of the Doctoral Dissertation Prize of China in 2000.



Mingyue Ding (M'96–SM'03) graduated from Beijing University of Aerospace and Aeronautics in 1982, received the M.S. degree from the Chinese University of Electronic and Technology, Chengdu, China, in 1985, and the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 1988.

He is currently a Professor and Vice Director of the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology. His current research interests include route planning, medical image processing, industry detection, and three-dimensional reconstruction.