Path Planning for Single Unmanned Aerial Vehicle by Separately Evolving Waypoints

Peng Yang, *Student Member, IEEE*, Ke Tang, *Senior Member, IEEE*, Jose A. Lozano, *Member, IEEE,* and Xianbin Cao, *Senior Member, IEEE*

***Abstract*—Evolutionary Algorithms (EAs) based Unmanned Aerial Vehicle (UAV) path planners have been extensively studied for their effectiveness and flexibility. However, they still suffer from a drawback that the high quality waypoints in previous candidate paths can hardly be exploited for further evolution as they evolve a path as a whole. Due to this drawback, the previous planners usually fail when encountering lots of obstacles. In this paper, a new idea of separately evaluating and evolving waypoints is presented to solve this drawback. By using this new idea, the high quality waypoints can be highly exploited. For the evaluation phase, a set of new evaluation functions are derived from the existing objectives and constraints functions to evaluate each waypoint. Basically, the derivation can be made only if the original functions are separable on waypoints. For the evolution phase, JADE, one state-of-the-art variant of Differential Evolution (DE) is employed to drive the further evolution for waypoints. In order to further improve the performance of the proposed planner, the waypoints are encoded in a rotated coordinate system with an external restriction. To test the capabilities of the new planner on planning obstacle-free paths, 5 scenarios with increasing numbers of obstacles are constructed. 3 existing planners and 4 variants of the proposed planner are employed as compared planners to show the effectiveness and efficiency of the proposed planner. The results verify the ability of the proposed planner and the idea of separate evolution in solving scenarios with large number of obstacles.**

***Index Terms*—Evolutionary Algorithm, Path planning, Unmanned Aerial Vehicle.**

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are aircrafts without human pilots onboard. Due to their great advantages in terms of crew endurance, UAVs have been entrusted in high-threatened missions so that human lives can be completely kept away from dangers [1], [2]. In the past few decades, autonomous path planning technique has become increasingly important to the UAV, as the conventional remotely piloted techniques can hardly offer sufficient accuracy and timeliness for complex missions nowadays [3], [4].

The path planning problem for a UAV can be formulated as an optimization problem that finds a feasible path from the start to destination for a UAV to follow on [4]. In the literature, a path is usually represented as a set of segments by a sequence of waypoints. These segments can be line segments [3], B-spline Curves [1] or Bezier curves [5]. Hence, path planning, in general, is to find out a sequence of waypoints as well as the segments linking each pair of adjacent waypoints to optimize various objectives subject to a number of constraints. Generally, the curve-based representations can ensure the smoothness of candidate paths, while their computational costs are high as they introduce external local controls for generating paths. In this paper, line segments based paths are adopted for its simplicity and efficiency.

Path planning problem is not a problem that only emerges in the context of UAVs. In fact, it is much more intensively investigated in the domain of Robotics, where path planning is usually referred to as motion planning [6], [7]. However, path planning for UAVs involves two domain-specific challenges that may not be encountered in a different context (e.g., motion planning for a robot). First, as UAVs fly above the ground and can change their altitude during flight, they in essence work in in a 3-D space. In contrast, motion planning for robots usually considers 2-D space as robots move on the ground. The additional degree of freedom significantly enlarges the mission space, and thereby the solution space of UAVs path planning problem. Second, a fixed-wing UAV cannot hover and have to always keep a rather high cruise speed. Such a requirement induces additional complicated constraints to the path planning problem. On the contrary, motion planning for robots can be immune with this requirement, because robots can slow down and even stop whenever necessary.

The UAV path planning problems can be further categorized into 3 types, i.e., off-line planning, on-line planning and cooperative planning. If the global information about the environment is at hand, the problem is called off-line planning [1], [4], [9], [18]-[29], [35]. If the circumstance is partially known or completely unknown in advance, the path will be planned on-line [1], [9], [20], [22], [28]. In case a mission is too complex to accomplish by a single UAV, a team of UAVs are

This work was supported in part by the 973 Program of China under Grant 2011CB707006, the National Natural Science Foundation of China under Grant 61175065, the Program for New Century Excellent Talents in University under Grant NCET-12-0512, and the Science and Technological Fund of Anhui Province for Outstanding Youth under Grant 1108085J16.

Peng Yang and Ke Tang are with the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications, School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui 230027, China (e-mail: trevor@mail.ustc.edu.cn ; ketang@ustc.edu.cn).

J. A. Lozano is with the Intelligent Systems Group, University of the Basque Country, San Sebastian-Donostia 20018, Spain (e-mail: ja.lozano@ehu.es).

Xianbin Cao is with the School of Electrical Information Engineering, Beihang University, Beijing 100191, China (e-mail: xbcao@buaa.edu.cn).

called for, and hence cooperative planning is studied [1], [4], [20], [28]. All these three types of problems have been proved to be NP-Complete [10]. Among them, off-line planning is probably the most commonly adopted approach for UAV path planning. Besides, on-line planning and cooperative planning can be viewed as extended versions of off-line planning problems. Hence, this paper focuses on off-line path planning problems. For the sake of brevity, UAV off-line path planning will be referred as path planning in the rest of the paper.

In the literature, UAV path planning problems have attracted a large variety of optimization applications, including A* [10], [11], Mixed-Integer Linear Programming [12], [13], Nonlinear Programming [14], Voronoi Diagram [15], [16] and Evolutionary Algorithms (EAs) [1], [4], [9], [18]-[29], [35]. One of the difficult problems in the field of the UAV path planning is to plan a feasible path in a scenario with obstacles. The difficulty of this problem results from the fact that the feasible space decreases rapidly with the increase of number of obstacles. To be detailed, on one hand, as the number of obstacles increases, the feasible space for each waypoint decreases. On the other hand, the increased obstacles lead to much narrower and more zigzag passageways for the UAV, and thus more waypoints are required to keep a path sufficiently flyable, i.e., smooth and safe. As a result, it becomes more difficult to find a feasible path in which all its waypoints are at feasible positions.

Although the existing EA-based planners are found to be more flexible and effective than the other approaches on planning obstacle-free paths, they still usually fail when the number of obstacles becomes quite large. The reason of the failure for the existing EA-based planners is that a path is feasible only if all its waypoints are at feasible positions. However, when searching for a feasible path, it is unlikely that the feasible positions for all waypoints can be obtained simultaneously (e.g., in the same iteration). Instead, it is highly possible that one candidate path consists of good positions for some waypoints, while the other waypoints are in bad positions. In other words, waypoints in a candidate path may be of different qualities. Nevertheless, existing EA-based approaches are unable to identify such differences as they regard the whole candidate path rather than a single waypoint as the unit of evaluation and evolution. Consequently, all waypoints of a "bad" path will be regarded as "bad" waypoints and vice versa. Eventually, the lack of capability to exploit high quality waypoints leads the existing EA-based planners to an inefficient search when lots of obstacles exist.

During the investigation of the UAV path planning problems, it has been noticed that most of the commonly used objective and constraint functions are separable on waypoints. This fact enlightens us that if we can explicitly decompose those evaluation functions and design a new evolution strategy that can be used to evolve each single waypoint, the waypoints can be evaluated and evolved separately and thus high quality waypoints can be exploited to improve the performances of the whole candidate paths. Inspired by the above considerations, a new EA-based path planner is proposed. Instead of searching for a sequence of feasible waypoints simultaneously, the proposed path planner evaluates and evolves each waypoint separately. For the evaluation phase, a set of new objective and constraints functions for single waypoints are derived from the existing functions which are used to evaluate the whole paths. For the evolution phase, a state-of-the-art Differential Evolution (DE), JADE [30] is employed to evolve each single waypoint. A widely used multi-criteria handling method is also used to select the evaluated waypoints for selection. In this way, the planner can be better focused on seeking good positions for waypoints, and information about previous good positions of waypoints can be better exploited. To further enhance the performance of the proposed planner, a recently proposed 3-D coordinate system [29] is also employed to encode the waypoints.

Lastly, a set of detailed simulations are carried out. In the simulations, the proposed planner is compared with 7 compared planners on 5 scenarios with different numbers of obstacles. The obstacles are represented as ranges of missiles and mountains where the UAV is forbidden to fly through. By randomly setting missiles on ground, the numbers of obstacles are set as 7, 15, 30, 60 and 120 for the 5 scenarios, respectively. The simulation results show that the proposed idea can significantly improve the ability of path planners in scenarios with lots of obstacles. The proposed planner can outperform all compared planners when there are lots of obstacles.

The rest of the paper is organized as follows: Section II describes the evaluation functions of the key factors of UAV path planning in detail. The proposed path planner is then introduced in Section III. In Section IV, we test the effectiveness of the proposed planner by comparing with 7 planners in 5 scenarios with different numbers of obstacles. Lastly, the conclusions of this work and expectation of further research is discussed in Section V.

II. Problem Description

When planning a path for an UAV, quite a few important factors need to be taken into consideration, such as the maneuverability of the UAV, the environment of the mission space, safety and cost of the path. These factors are involved either in the form of objective functions that need to be maximized/minimized, or in the form of constraints that a path must comply with. Since the purpose of this paper is not to construct a new set of realistic evaluation functions, we directly employ or derive some existing representative functions in the literature [1], [4], [35] to include several key factors in UAV path planning. Detailed technical justifications of the chosen functions could be found in the corresponding references, i.e., [1], [4], [35]. Generally, these factors restrict the paths in a geometric manner. Specifically, the factors to be considered can be categorized into two types based on the way they restrict the paths. The first type of factors require only the waypoints for evaluation. That is, those factors can be evaluated by checking the locations of waypoints as well as the geometric relations in between. Examples are maximal turning angle, maximal slope, minimal path length, minimal flight altitude and map limited. The other factors are relevant to the segments as well as waypoints since waypoints are not sufficient to

determine the real states of an UAV. In other words, the segments may be infeasible even when the corresponding waypoints are in feasible locations. Examples are minimal risks of kill, minimal risks of radar detection and the terrain limited. For the first type of factors, we directly borrow the existing functions. For the second type of factors, however, the existing functions have only considered the states of waypoints and regard those states as the behaviors of the corresponding segments. In this paper, we try to modify those second type of functions and approximate the real behaviors of a segment. The approximation is to first divide each segment into N_d piecewise parts and then evaluate the N_d dividing points (the waypoint is also regarded as one dividing point). Suppose $(dx_{ij}, dy_{ij}, dz_{ij})$ indicates the j^{th} dividing point on the segment between $(i-1)^{\text{th}}$ and i^{th} waypoint, where $i = 2,3,\dots,N_w$, $j = 1,2,\dots,N_d$, it can be calculated as:

$$(dx_{ij}, dy_{ij}, dz_{ij}) = (x_{i-1}, y_{i-1}, z_{i-1}) + j \cdot \big((x_i, y_i, z_i) - (x_{i-1}, y_{i-1}, z_{i-1})\big)/N_d \quad (1)$$

N_d reflects the trade-off between the computational cost and the accuracy of approximation. Generally, the larger N_d is, the higher accuracy the approximation will have, while the efficiency will fall. The value of N_d is problem dependent, and will be discussed in Section IV.

At the end of this section, the scheme of selecting the final solution according to these constraints and objectives will be presented.

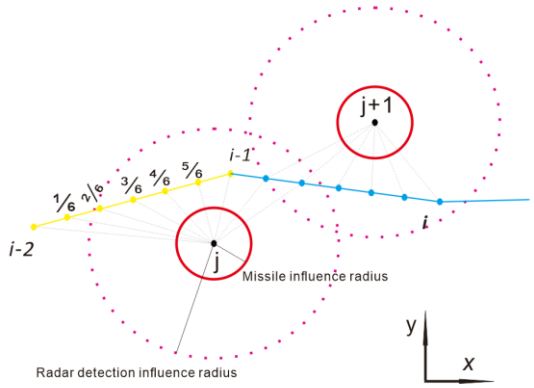

Fig. 1. a 2-D illustration of the probability of kill and risks of radar detection imposed on each dividing point. In this figure, $N_d = 6$.

A. Objective Functions

1) *Minimal Path Length:* For military missions, shorter path are always preferred to longer ones, because shorter paths usually consume less fuel and have lower chance of encountering some unexpected threats, e.g., gusty wind and undetected enemy. Hence, the total length of the path needs to be minimized. This consideration leads to the objective function Path Length Ratio (PLR) [1], [35] given by (2),

$$f_1 = \frac{\sum_{i=2}^{N_w} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}}{\sqrt{(x_{N_w} - x_1)^2 + (y_{N_w} - y_1)^2 + (z_{N_w} - z_1)^2}} \quad (2)$$

where (x_i, y_i, z_i), $i = 2,3,\dots,N_w$, denotes the position of the i^{th} waypoint in the 3-D mission space, N_w is the total number of waypoints of a path (including the starting point and the destination). Here, the path length ratio is used instead of the absolute path length. [1] has given the reason that they are equivalent and the former one is more admissible.

2) *Minimal Probability of Kill:* If a UAV is within the range of the hostile missiles, it is at risk. Intuitively, paths with lower probability of kill (PKill) are safer than those with higher ones. For each dividing point, the k^{th}, $k = 1,2,\dots,M$ (the number of missile), hostile missile imposes a certain probability of kill on the UAV only if that point is inside the region defined by the missile's maximal risk distance (seen in Fig. 1), denoted as R^k_{PKmax}. The distance between a dividing point and the k^{th} missile is calculated as:

$$dis^k_{ij} = \sqrt{(dx_{ij} - mx_k)^2 + (dy_{ij} - my_k)^2 + (dz_{ij} - mz_k)^2} \quad (3)$$

where (mx_k, my_k, mz_k) is the given location of the k^{th} missile. At last, the PKill of the whole path can be calculated as:

$$f_2 = \sum_{i=2}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{M} \mathrm{PK}^k_{ij} \quad \text{with}$$

$$\mathrm{PK}^k_{ij} = \begin{cases} \frac{(R^k_{PKmax})^4}{(R^k_{PKmax})^4 + (dis^k_{ij})^4} & \textbf{\textit{if}}\ dis^k_{ij} \le R^k_{PKmax} \\ 0 & \textbf{\textit{otherwise}} \end{cases}. \quad (4)$$

This formulation (4) is derived from the one suggested in [35], which is a simplified version of the real PKill in [1] where the impact of the altitude of the UAV is neglected.

3) *Minimal Risks of Radar Detection:* UAV can always keep stealthy until enemy radars detect it. The risks of radar detection (RRD) should be as small as possible. The RRD is technically fourth power of the distance between the dividing point and the radar. [4] suggested a simplified version of the real RRD. We modify it by evaluating N_d dividing points for each segment. The derived function is as follow,

$$f_3 = \sum_{i=2}^{N_w} \sum_{j=1}^{N_d} \sum_{k=1}^{R} \mathrm{RD}^k_{ij} \quad \text{with}$$

$$\mathrm{RD}^k_{ij} = \begin{cases} \left(\frac{\delta}{dis^k_{ij}}\right)^4 & \textbf{\textit{if}}\ dis^k_{ij} \le R^k_{RRDmax} \\ 0 & \textbf{\textit{otherwise}} \end{cases} \quad (5)$$

where

$$dis^k_{ij} = \sqrt{(dx_{ij} - rx_k)^2 + (dy_{ij} - ry_k)^2 + (dz_{ij} - rz_k)^2} \quad (6)$$

where δ is a scale of the intensity of the radar, (rx_k, ry_k, rz_k) is the location of the k^{th} radar, R is the number of radars, and R^k_{RRDmax} represents the maximal risk distance of missile.

4) *Minimal Flight Altitude:* A UAV may need to fly at a low altitude to keep mass threats to the enemy on ground. The formulation of Flight Altitude (FA) is directly borrowed from [1], as follow,

$$f_4 = \sum_{i=2}^{N_w} \mathrm{FA}_i \ \ with\ \mathrm{FA}_i = \begin{cases} 0 & \boldsymbol{if}\ \ z_i \le \mathrm{map}(x_i, y_i) \\ (z_i - \mathrm{map}(x_i, y_i))/N_w & \boldsymbol{otherwise} \end{cases} \quad (7)$$

where $\mathrm{map}(x, y)$ is a function that returns the elevation of the location (x, y).

B. Constraints Functions

1) *Maximal Turning Angle:* Subject to the maneuverability of a UAV, a path should be sufficiently smooth. This requires the turning angle of the UAV at a waypoint to be kept small. The turning angle is defined as the angle between its previous direction and the current direction in the horizontal direction. That is,

$$g_1 = 0\ where\ g_1 = \sum_{i=2}^{N_w - 1} c_i^1\ \ with\ c_i^1 = \begin{cases} 1 & \boldsymbol{if}\ \theta_i > \theta_{max} \\ 0 & \boldsymbol{otherwise} \end{cases} \quad (8)$$

where θ_{max} is the upper limit of the turning angle, θ_i is the turning angle at the i^{th} waypoint $(x_i, y_i, z_i), i = 2,3, \dots, N_w$. [4] suggested a formulation of θ_i as follow,

$$\theta_i = \arccos\left(\frac{(x_i - x_{i-1}, y_i - y_{i-1}) \cdot (x_{i+1} - x_i, y_{i+1} - y_i)^T}{\|(x_i - x_{i-1}, y_i - y_{i-1})\| \cdot \|(x_{i+1} - x_i, y_{i+1} - y_i)\|}\right) \quad (9)$$

where $\|x\|$ means the norm of vector x.

2) *Limited UAV Slope:* Similar to the turning angle, the slope characterizes the change of flying direction in the vertical direction, i.e., the diving or climbing angle. The slope is the included angle between the horizontal and the direction from the current waypoint towards the next one. For each waypoint (x_i, y_i, z_i), $i = 2,3, \dots, N_w$, [4] suggested its slope as:

$$r = \frac{z_i - z_{i-1}}{||(x_i - x_{i-1}, y_i - y_{i-1})||} \quad (10)$$

Similarly, the slope should be in the range of the maximal diving or climbing angle. For a feasible path, this constraint can be depicted as (11),

$$g_2 = 0\ \ where\ g_2 = \sum_{i=2}^{N_w} c_i^2\ \ with\ c_i^2 = \begin{cases} 0 & \boldsymbol{if}\ \alpha \le r \le \beta \\ 1 & \boldsymbol{otherwise} \end{cases} \quad (11)$$

3) *Terrain Limited:* A UAV should fly above the rugged terrain and avoid collisions against the mountains. The height of the UAV should always be higher than the terrain below it. We derive the following formulation from [1]. The dividing points are also used as the segments may be in the mountains. This constraint can be depicted as follow:

$$g_3 = 0 \ \ \text{where}$$

$$g_3 = \sum_{i=2}^{N_w} \sum_{j=1}^{N_d} c_{ij}^3\ \ with\ c_{ij}^3 = \begin{cases} 1 & \boldsymbol{if}\ dz_{ij} \le \mathrm{map}(dx_{ij}, dy_{ij}) \\ 0 & \boldsymbol{otherwise} \end{cases} \quad (12)$$

4) *Map Limited:* Before executing a mission, a certain related mission space is usually investigated. Conversely, the areas outside the mission space is ordinarily unknown and may conceal unexpected dangerous, e.g., unknown hostile army. Thus, UAV should always fly in the mission space to keep away from uncertainties. Commonly, the mission space is assumed as a cube. For a feasible path, it should be inside of the cube. [1] suggested this constraint as follow,

$$g_4 = 0 \ \ \text{where}$$

$$g_4 = \sum_{i=2}^{N_w} c_i^4\ \ with\ c_i^4 = \begin{cases} 0 & \boldsymbol{if}\ \mathrm{InRange}(x_i, y_i) \\ 1 & \boldsymbol{otherwise} \end{cases} \quad (13)$$

$$\mathrm{InRange}(x_i, y_i) = (l_x \le x_i \le h_x) \wedge (l_y \le y_i \le h_y) \quad (14)$$

where l_x and h_x are the lower and higher bounds for x coordinate system, and l_y and h_y are the lower and higher bounds for y coordinate.

C. Selection of the Final Solution

The problem above described appears to be a Multi-objective Optimization Problem (MOP) at the first sight as there are several conflicting objective functions to optimize. Commonly, a MOP outputs a set of Pareto optimal solutions, which will be presented to human experts to determine the final solution to be followed on by the UAV. However, this does not fit the context of UAV path planning as no human expert is onboard to make such a choice. Hence, a final solution should be selected for the UAV. Usually, a common practice in the context of UAV is to integrate different objectives. In the literature, some previous work solve this problem by using weighted sum [4], [9], [19]-[21]. However, those weighted parameters appear very difficult to fine-tune as different objectives are in different scales. In this paper, we adopt a more intuitive scheme proposed in [1]. This scheme considers different human preferences to the objectives. Detailedly, this scheme takes two cases into account:

1) For all the feasible paths such that f_2=0, i.e., there is no chance for the UAV to be destroyed. The path with the smallest f_1, i.e., path length, is selected as the final output. If there exist more than one path sharing the same value of f_2 and f_1, we randomly select one of them as the final best output. This is because the objectives Minimal Probability of Radar Detection and Minimal Flight Altitude looks equally important to the UAV.

2) For all the feasible paths such that $f_2 > 0$, i.e., it is probably that the UAV will be destroyed. We first calculate the relative f_2^i of each i^{th} path as $rf_2^i = f_2^i / \min_i(f_2^i)$. Then, the paths with $rf_2^i \ge$V (for example, with V =1.05) are discarded, which makes the non-discarded j^{th} paths have a value of f_2^j that is insignificantly larger than the minimum. At last, the path with $\min_j(f_1^j)$ is selected. Therefore, the final output has a f_2 that is a bit larger than the minimum, as well as a reasonable path length f_1.

III. The Proposed Planner

Most of these EA-based approaches adopt a similar iterative search framework. That is, a candidate solution to the path planning problem is encoded as a real-valued vector that represents the positions of all the waypoints, and the optimal path is iteratively searched in the corresponding real space. At each iteration, a number of candidate paths are generated and evaluated with respect to the objective functions and constraints. Only those paths with higher fitness will be maintained, based on which new candidate paths will be generated by applying some search operators to the maintained paths. The search process terminates when the optimal (or sufficiently good) solution is obtained or a given time budget is reached. The above-described EA-based approaches have shown to be very effective for UAV path planning when obstacles in the mission space are few. However, when the obstacles increase, those planners usually fail. This results from the common disadvantage that the high quality waypoints in previous candidate paths appear very difficult to exploit. To be specific, a path is optimal only if all its waypoints are at optimal positions. When searching for the optimal path, it is unlikely that the optimal positions for all waypoints can be obtained simultaneously (e.g., in the same iteration). Instead, it is highly possible that one candidate path consists of good positions for some waypoints, while the other waypoints are assigned good positions in another candidate path. In other words, waypoints in a candidate path may be of different quality. However, existing EA-based approaches are unable to identify such differences. Intuitively speaking, all waypoints of a "bad" path will be regarded as "bad" waypoints and vice versa. Such a search behavior will make it difficult to exploit high quality waypoints in previous candidate paths and eventually lead to an inefficient search.

During the investigation of the UAV path planning problems, it has been noticed that most of the commonly used objective and constraint functions are separable on waypoints. This fact enlightens us that if we can explicitly decompose those evaluation functions and design a new evolution strategy that can be used to evolve each single waypoint, the waypoints can be evaluated and evolved separately and thus high quality waypoints can be exploited to improve the performances of the whole candidate paths. Inspired by the above considerations, a new EA-based path planner is proposed. Instead of searching for the feasible path as a whole, the proposed path planner evaluates and evolves each waypoint separately. Detailedly, at each generation, for each path j, its waypoints are separately evolved in an ascending order, i.e., the $(i+1)^{\text{th}}$ waypoint will be evolved after the i^{th} one has been evolved, $i = 2,3,\dots,N_w-1$. For the i^{th} waypoint of the j^{th} path, its offspring is generated by referring to the i^{th} waypoints of all the other candidate paths, $i = 2,3,\dots,N_w-1$, $j = 1,2,\dots,N_p$. In this way, the information of the other i^{th} waypoints in the population can be explicitly exploited to improve the quality of the currently being evolved i^{th} waypoint. After a new offspring is produced, it is asked to compete with its parent for survival based on their fitness values. To evaluate the waypoints, a set of new evaluation functions are derived from the commonly used functions introduced in Section II. The evolution of each single waypoint is performed by a state-of-the-art Differential Evolution (DE), JADE. The diagram of the evolution of waypoints is illustrated in Fig. 2. The evolution of each waypoint, i.e., the inner loop in Fig. 2, is only executed once at each generation here. Ideally, it can be executed any fixed times, say N. As the larger N is, more local information of the waypoint can be used and the new produced waypoint may be with better quality. However, the efficiency of optimization will drop as more time budgets are required for the local improvements. For simplification and efficiency, here we set N=1. A widely used multi-criteria handling method is also used to select the evaluated waypoints. To further enhance the performance of the proposed planner, a recently proposed 3-D coordinate system [29] is also employed to encode the waypoints.

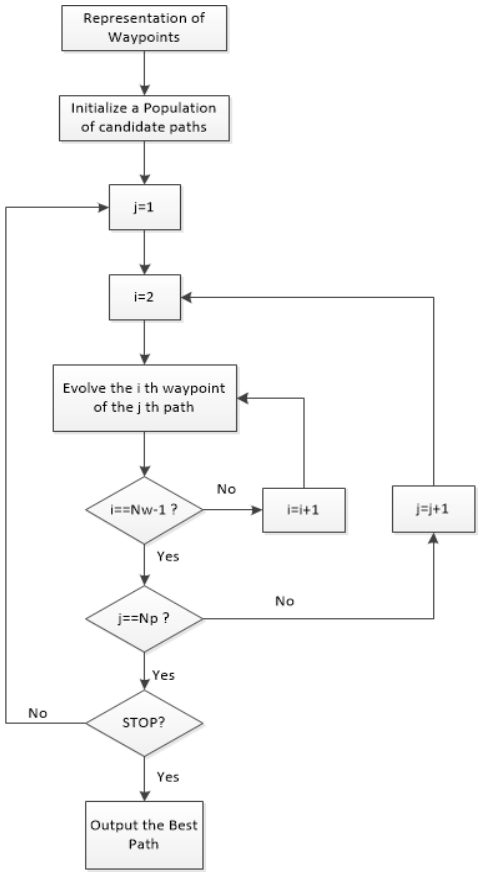

Fig. 2 The framework of the proposed path planner. N_w is the number of waypoints of a path and N_p is the number of paths.

In the EAs framework, a path planner usually consists of several key components, i.e., evaluation, reproduction, selection and path representation. To detailedly introduce the proposed planner, each key component is described one by one in this section.

A. New Evaluation Functions

Each waypoint should be evaluated before evolution. However, the above-mentioned evaluation functions cannot be adopted in our framework as they can only be used for evaluating the global states of a path. Fortunately, those

commonly used evaluation functions are found separable on waypoints. The reason is that those objectives restrict the flight of the UAV at the geometric level, where most objects concerned are the points, segments and angles, which are all separable on points, i.e., waypoints. To be specific, the behavior of a waypoint usually depends on, except for itself, one or two neighbor waypoints. For example, the *Minimal Path Length* and *Maximal Turning Angle* involve three waypoints, i.e., the previous neighbor, the current waypoint and its successive neighbor, while the rest functions require the information about the current waypoint and its previous neighbor. In the framework of the proposed planner, as the waypoints of a new candidate path are produced in sequence, the knowledge of the previous neighbor can easily be obtained as they are produced earlier, while the information about the successive neighbor is unknown. Considering this, the two rules are given as below:

1) For the evaluation function involving two waypoints, its local version is calculated relevant to the previous neighbor and the current waypoint.

2) For the evaluation function involving three waypoints, its local version is calculated relevant to the previous neighbor, the current waypoint and the destination.

The idea behind the second rule is driven by: *suppose all the previous waypoints have been determined and the current waypoint is the last intermediate waypoint, where should it be*? Although this idea sounds a bit greedy and the transcribed local versions are only approximations of their global ones, it works well as we will see later in the simulation results.

Based on the two rules, the new evaluation functions, i.e., local versions, are introduced as follows.

1) *Minimal Path Length:* As seen in Fig. 3, for the i^{th} waypoint (x_i, y_i, z_i), the PLR is calculated as follows:

$$f'_{1,i} = \frac{\sqrt{(x_i-x_{i-1})^2+(y_i-y_{i-1})^2+(z_i-z_{i-1})^2}}{\sqrt{(x_{N_w}-x_{i-1})^2+(y_{N_w}-y_{i-1})^2+(z_{N_w}-z_{i-1})^2}} + \frac{\sqrt{(x_{N_w}-x_i)^2+(y_{N_w}-y_i)^2+(z_{N_w}-z_i)^2}}{\sqrt{(x_{N_w}-x_{i-1})^2+(y_{N_w}-y_{i-1})^2+(z_{N_w}-z_{i-1})^2}} \quad (15)$$

2) *Minimal Probability of Kill:* Given the location of the k^{th} missile (mx_k, my_k, mz_k), the PKill of the i^{th} waypoint (x_i, y_i, z_i) can be calculated as:

$$f'_{2,i} = \sum_{j=1}^{N_d}\sum_{k=1}^{M} \text{PK}_{ij}^k \quad (16)$$

where PK_{ij}^k can be calculated following (4).

3) *Minimal Risks of Radar Detection:* Given the k^{th} radar (rx_k, ry_k, rz_k), the RRD of the i^{th} waypoint (x_i, y_i, z_i) can be calculated as:

$$f'_{3,i} = \sum_{j=1}^{N_d}\sum_{k=1}^{R} \text{RD}_{ij}^k \quad (17)$$

where RD_{ij}^k can be calculated following (5).

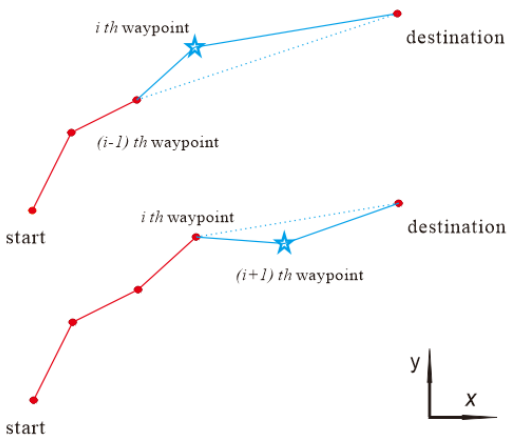

Fig. 3 A 2-D illustration of the i^{th} and $(i+1)^{\text{th}}$ waypoints' PLR, where the circles indicate the evolved waypoints and the start means the current waypoint to be served.

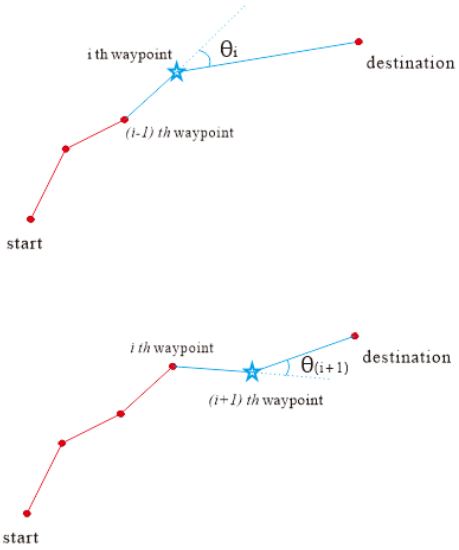

Fig. 4 A 2-D illustration of the i^{th} and $(i+1)^{\text{th}}$ waypoints' turning angle, where the circles indicate the evolved waypoints and the start means the current waypoint to be served.

4) *Minimal Flight Altitude:* For the i^{th} waypoint (x_i, y_i, z_i), the FA is calculated as follows:

$$f'_{4,i} = \text{FA}_i \quad (18)$$

where FA_i can be calculated following (7).

5) *Maximal Turning Angle:* As seen in Fig. 4, the turning angle of the i^{th} waypoint can be calculated as follow,

$$\theta_i = \arccos\left(\frac{(x_i-x_{i-1}, y_i-y_{i-1})\cdot(x_{N_w}-x_i, y_{N_w}-y_i)^T}{\|(x_i-x_{i-1}, y_i-y_{i-1})\|\cdot\|(x_{N_w}-x_i, y_{N_w}-y_i)\|}\right). \quad (19)$$

The constraint of waypoint (x_i, y_i, z_i) can be written as:

$$g'_{1,i} = 0 \text{ where } g'_{1,i} = \begin{cases} 1 & \textbf{\textit{if}}\ \theta_i > \theta_{max} \\ 0 & \textbf{\textit{otherwise}} \end{cases} \quad (20)$$

6) *Limited UAV Slope:* For each waypoint (x_i, y_i, z_i), $i = 2,3,\dots,N_w\text{-}1$, its slope can be calculated as (10), and the constraint of waypoint (x_i, y_i, z_i) can be written as:

$$g'_{2,i} = 0 \; where \;\; g'_{2,i} = \begin{cases} 0 & \boldsymbol{if}\ \alpha \le r \le \beta \\ 1 & \boldsymbol{otherwise} \end{cases} \tag{21}$$

7) *Terrain Limited:* This constraint of the i^{th} waypoint (x_i, y_i, z_i), $i = 2,3,\dots,N_w\text{-}1$, can be transribed as:

$$g'_{3,i} = 0 \;\text{ where}$$
$$g'_{3,i} = \sum_{j=1}^{N_d} c_{ij}^3 \;\; with\ c_{ij}^3 = \begin{cases} 1 & \boldsymbol{if}\ dz_{ij} \le \max(dx_{ij}, dy_{ij}) \\ 0 & \boldsymbol{otherwise} \end{cases} \tag{22}$$

8) *Map Limited:* The penalty of the i^{th} waypoint (x_i, y_i, z_i), $i = 2,3,\dots,N_w\text{-}1$, on this constraint can be calculated as follows,

$$g'_{4,i} = 0 \;\text{ where}$$
$$g'_{4,i} = c_i^4 \;\; with\ c_i^4 = \begin{cases} 0 & \boldsymbol{if}\ \text{InRange}(x_i, y_i) \\ 1 & \boldsymbol{otherwise} \end{cases} \tag{23}$$

Notice that, for the new evaluation functions transcribed by the first rule, the behaviors of the last segment, from the $(N_w - 1)^{\text{th}}$ waypoint to the destination, should be laid on the waypoint the $(N_w - 1)^{\text{th}}$ waypoint as well. This is reasonable as the $(N_w - 1)^{\text{th}}$ waypoint determines the last two segments.

B. Reproduction and Selection

In the proposed planner, the reproduction component is independent of the other components, e.g., selection, evaluation and representation. Ideally, any strategy of reproduction can be adopted to drive the evolution. Here, JADE [30], one state-of-the-art variant of Differential Evolution (DE), is employed to evolve waypoints. DE is arguably one of the most powerful stochastic real-parameter optimization algorithms and its variants have been widely used in solving many real world problems [31]. DE shares the same framework with traditional EAs. Within this framework, DE employs a differential mutation operator that creates trial vectors (individuals) by adding the weighted difference vector between two individuals to a third one. This novel mutation strategy turns out to be very efficient. In recent years, DE has been extensively studied and lots of variants have been proposed [31]. Among them, JADE [30] is undoubtedly one of the state-of-the-art variants. JADE is an adaptive version of DE that requires very few parameters to be tuned.

It is very easy to implement JADE into the proposed path planner. For intuition, we first list the framework of the proposed path planner in Table I and then explain them in details. In the pseudo code, each waypoint is denoted as $x_{j,i}^t$, where $i = 2,3,\dots N_w - 1$; $j = 1,2,\dots N_p$, which means the i^{th} waypoint of the j^{th} path at the t^{th} generation. And $x_{j,i,m}^t$, where $m = 1,2,3$, means the m^{th} coordinate value of waypoint $x_{j,i}^t$, i.e., the x, y, z coordinate, respectively. μ_{CR}^i and μ_F^i are adaptive parameters for updating CR and F, i.e., two key parameters for crossover and mutation, for the i^{th} waypoints. Steps 9 to 12 describe the mutation scheme: firstly, three distinct i^{th} waypoints are randomly selected from the whole population at the t^{th} generation, denoted as x_{qbest}, x_{r1} and x_{r2}, respectively. Specifically, x_{qbest} must be selected from the top $q\%$ of the i^{th} waypoints. The value of q is usually chosen from $[5, 20]$. The term $x_{1:N_p,i}^t$ indicates all the i^{th} waypoints at the t^{th} generation. Strictly, the selected waypoint x_{qbest} may not be the best reference for generating offspring for the waypoint $x_{j,i}^t$. This is because the good behavior of x_{qbest} is referred to a path different from the j^{th} path and a good waypoint of one path may not be good in the other path. Nevertheless, this mutation scheme is still reasonable: at the early stage of the search process, candidate paths are quite diverse. Although the feasibility of the segment from $x_{j,i-1}^t$ to x_{qbest} cannot be guaranteed, the location of x_{qbest} is at least in the good (or even feasible) regions. This information provides a bias for the generated offspring towards the feasible regions. This stage can be seen as the coarse tuning. As the optimization goes on, waypoints in each order will gradually converge and candidate paths will get closer to each other. At this stage, the information of x_{qbest} can be used to fine tune the waypoints and gradually drive the segments to feasibility.

With this mutation scheme, a new potential waypoint $v_{j,i}^t$ is generated by step 12. After that, the crossover scheme is from steps 13 to 20 with respect to the three coordinates. The evaluation and ranking is at step 21, where the parent waypoint and offspring waypoint are asked to compete for survival. As introduced in the subsection III.A, the evaluations of the parent and offspring waypoints are in relation to their common previous waypoint $x_{j,i-1}^t$, and the destination if necessary. The ranking of the parent and its offspring in terms of their evaluation values will be introduced later in this subsection. At step 24, S_{CR}^i and S_F^i record the value of CR and F of successful reproduction where offspring is better than its parent. The update scheme is shown in step 29 and 30, where $\text{mean}_A(S_{CR}^i)$ is the ordinary arithmetic mean and $\text{mean}_L(S_F^i)$ is the Lehmer mean that is

$$\text{mean}_L(S_F^i) = \frac{\sum_{F \in S_F^i} F^2}{\sum_{F \in S_F^i} F} \;. \tag{24}$$

Parameter c is used to control the adaptation of μ_{CR}^i and μ_F^i. The authors of [30] suggest that c works well if it is chosen within the range of [0.05, 0.2]. In this work, we set c as 0.1. After μ_{CR}^i and μ_F^i are updated, the parameters CR and F are adaptively generated in step 8, where $\text{randn}(\mu_{CR}^i, 0.1)$ is the Gaussian distribution with mean μ_{CR}^i and standard deviation 0.1. $\text{randc}(\mu_F^i, 0.1)$ represents the Cauchy distribution with mean μ_F^i and scale parameter 0.1. Step 32 indecates the next generation starts.

The parent waypoint and new reproduced waypoint are evaluated at step 21. After evaluation, they will compete for

Table I: The Framework of The Proposed Path Planer

Begin
  Set $t = 1$; $\mu_{CR}^i = 0.5$; $\mu_F^i = 0.5$; $i = 2,3,\dots N_w - 1$.
  Uniformly generate N_p candidate paths, of which each waypoint is denoted as $x_{j,i}^t$. $j = 1,2,\dots N_p$.
  repeat until a fixed number of generations runs out
    Set $S_F^i = \phi$; $S_{CR}^i = \phi$;
    For j = 1 to N_p
      For i = 2 to $N_w - 1$
        $CR = \text{randn}(\mu_{CR}^i, 0.1), F = \text{randc}(\mu_F^i, 0.1)$;
        Randomly choose x_{qbest} from the q% "best" waypoints of $x_{1:N_p,i}^t$.
        Randomly choose $x_{r1} \neq x_{j,i}^t$ from $x_{1:N_p,i}^t$.
        Randomly choose $x_{r2} \neq x_{r1} \neq x_{j,i}^t$ from $x_{1:N_p,i}^t$.
        $v_{j,i}^t = x_{j,i}^t + F \cdot (x_{qbest} - x_{j,i}^t) + F \cdot (x_{r1} - x_{r2})$.
        Generate m_{rand} = randint(1,3);
        For m = 1 to 3
          If $m = m_{\text{rand}}$ or rand(0,1) < CR
            $u_{j,i,m}^t = v_{j,i,m}^t$;
          Else
            $u_{j,i,m}^t = x_{j,i,m}^t$;
          End
        End
        If $f(x_{j,i-1}^t, x_{j,i}^t) \leq f(x_{j,i-1}^t, u_{j,i}^t)$
          $x_{j,i}^{t+1} = x_{j,i}^t$;
        Else
          $x_{j,i}^{t+1} = u_{j,i}^t$; $CR \rightarrow S_{CR}^i$, $F \rightarrow S_F^i$
        End
      End
    End
    For i = 2 to $N_w - 1$
      $\mu_{CR}^i = (1-c) \cdot \mu_{CR}^i + c \cdot \text{mean}_A(S_{CR}^i)$;
      $\mu_F^i = (1-c) \cdot \mu_F^i + c \cdot \text{mean}_L(S_F^i)$;
    End
    t = t + 1;
  End
End

survival by comparing their fitness. However, it is not intuitive for such a comparison as each waypoint receives a vector of fitness values rather than a scalar. To deal with this difficulty, most previous work try to combine the fitness vector into a scalar with some weight parameters [4], [9], [19]-[21]. However, those weight parameters appear very difficult to fine-tune as different constraints and objectives are in different scales. In this paper, a multi-criteria handling method [33] based on the priorities is adopted to select the best waypoint, which has already been used in [1], [29] and [34]. In fact, we have already introduced a priorities based selection scheme in Section II. However, it cannot be adopted here as it is used to select the best path for output rather than a temporarily better waypoint. The first step of the waypoint selection scheme is to place all these 8 constraints and objectives in different priority levels, which reflects the human preferences. To be specific, 4 constraints are placed in the highest level as they must be satisfied. PL and PKill that should be firstly minimized are placed in the second level and RRD and FA are placed in the lowest level. Then, a waypoint a is said to dominate waypoint b, only if one of the following situations happen:

1) a and b are all feasible and a dominates b based on the criteria in second level.

2) a and b are all feasible and a cannot dominate b based on the criteria in second level, but a dominates b based on the criteria in lowest level.

3) a is feasible but b is not.

4) a and b are all infeasible, while a dominates b based on the criteria in the highest level.

If waypoint a dominates waypoint b, a is selected as the survivor, and vice versa. If a and b cannot dominate each other, we will keep the parent waypoint alive.

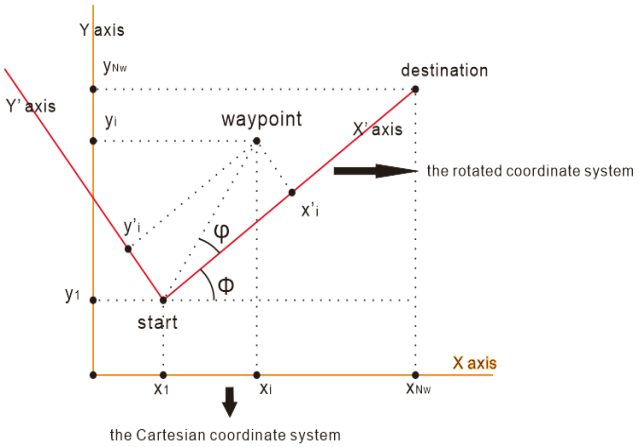

Fig. 5.The adopted coordinate system is a rotation of Cartesian coordinate.

C. Representation of Waypoints

In most existing work, the waypoint is usually represented as a 3-D coordinate within a Cartesian coordinate system or a polar coordinate system in previous work. Recently, [29] discussed the shortage of these two coordinate systems, either generating very large search spaces or appearing very difficult for local controls, e.g., mutation and crossover. To solve these problems, [29] proposed a new coordinate system. The new coordinate system (x', y', z'), is actually a rotation of the Cartesian coordinate system (x, y, z), where its x' axis lies along the horizontal direction from the start to the destination and y' keeps being orthogonal to x' axis, and z' axis stays the same with z axis. A 2-D illustration of the relation between these two coordinate systems is shown in Fig. 5. For any waypoint (x_i', y_i', z_i'), i=1,2,…N_w, in rotated coordinate system, its codification in Cartesian coordinate system, (x_i, y_i, z_i), is mathematically defined as follows:

$$\begin{cases} x_i = x_1 + \cos(\varphi + \phi) \cdot \sqrt{x_i'^2 + y_i'^2} \\ y_i = y_1 + \sin(\varphi + \phi) \cdot \sqrt{x_i'^2 + y_i'^2} \\ z_i = z_i' \end{cases} \tag{25}$$

where φ is the angle included by the direction from start to waypoint and x' axis, and ϕ is the angle between x' axis and x axis. According to (25), for example, the codifications of the start and destination in the rotated coordinate system are $(0, 0, z_1)$ and $(\sqrt{(x_{N_w} - x_1)^2 + (y_{N_w} - y_1)^2}, 0, z_{N_w})$, respectively.

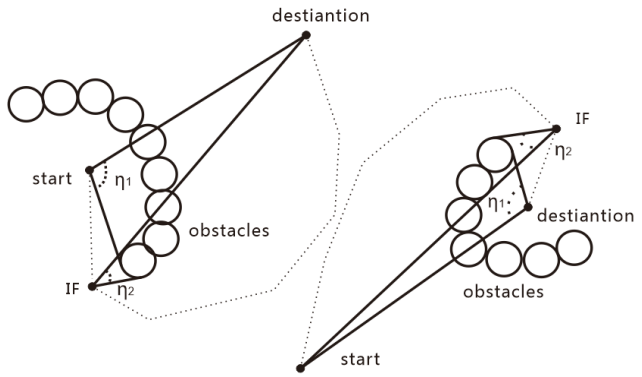

Fig. 6.The inserted fixed point can remedy the limitation of the external restriction. The left figure shows the situation where the start is crowded by obstacles. While the right figure shows the situation at the destination. The solid lines are LOSs from the UAV to the destination or the edge of obstacles. The dotted lines are the possible paths for the UAV.

Within the rotated coordinate system, an external restriction is imposed on the encoded paths. This restriction forces the x' coordinates of waypoints along the paths to be monotone increasing. With this restriction, the search space can be significantly reduced. To be specific, as the waypoints along the x′ axis will not intersect, the search space can be explicitly equally divided into $N_w - 2$ subspaces along the x′ axis. And within each subspace, the $N_w - 2$ corresponding intermediate waypoints will be generated. Consequently, the whole search space has been reduced for $(N_w - 2)^{N_w-2}$ times. Some other researchers [35] have also noticed the advantage of this restriction, and a quite similar rotated coordinate system has been adopted. Although this advantage is attractive, this restriction compromises the flexibility of the planners as the UAV cannot go backward. In relation with this shortage, the researchers briefly mentioned in [35] that there are very few cases where a UAV needs to go backward to bypass the obstacles. In fact, such cases only happen at the beginning of the flight and at the end of the path. The cause of this case is that the angle η_1, included by the x′ axis and the line-of-sight (LOS) between the start/destination and the edge of the obstacles, is larger than 90°. From this point of view, we can easily remedy this limitation by artificially inserting an Intermediate Fixed point (IF) somewhere safe, so that the new angle η_2 at the IF is smaller than 90°, as seen in Fig. 6. The angle η_2 is defined as the included angle between the LOS from the IF to start/destination and the LOS from the IF to the edge of the obstacles. After that, the original path planning problem can be solved as two sub-problems from the start to the IF and from the IF to the destination, as illustrated with the dot line in Fig. 6. The use of IF is not a new idea as it has been used in [1] to control the B-spline curves. The proper location for IF is usually very easy to obtain. Although the artificial insertion slightly decreases the autonomous capacity of the proposed planner, it is still worthwhile, considering its contribution to the reduction of the search space.

Note that the waypoints are encoded in rotated coordinate system through the whole search process. However, since the new evaluation functions require the Cartesian coordinate encoded waypoints, it is necessary to generate a Cartesian coordinate copy of those waypoints according to (25) as the inputs of the evaluation phase.

IV. Simulation Results

Ideally, by evolving waypoints separately, the waypoints with better quality can be better exploited to guide the evolution. To verify its actual ability, the proposed planner is asked to handle different scenarios with increasing obstacles. In each scenario, the proposed planner is compared with 7 compared planners from different viewpoints. The superiorities of the proposed planner over the compared planners are shown based on the effectiveness and efficiency. To test how the evaluation accuracy influences the proposed planner, the impacts of the number of dividing points, i.e., N_d, is also analyzed and tested. The sensitive analysis is also given for a proper choice of the only EA-related parameter, i.e., N_w. Lastly, we clarify that the proposed planner is insensitive to the quality of the initialized solutions.

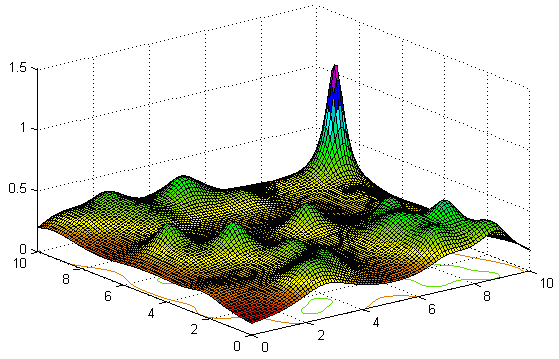

Fig. 7.The landscape of Modified Foxholes Shekel problem consists of some mountains and vallevs. which is close to a real terrain.

A. Scenarios Description

In the field of path planning for UAVs, there are no widely accepted benchmark problems. Hence, we have designed 5 scenarios with different numbers of obstacles for the simulation. Detailedly, the scenarios consist of three key components, i.e., terrain, obstacles and the start as well as the destination. The terrain here is represented as the landscape of a variant of the well-known Foxhole Shekel optimization problem (seen in Fig. 7), formulated as (26).

$$h(\boldsymbol{x}) = \sum_{i=1}^{30} \frac{0.1}{\sum_{j=1}^{2}(x_j - a_{ij})^2 + c_i} \tag{26}$$

where parameters $\boldsymbol{a}$ and $\boldsymbol{c}$ are employed to vary the landscape. The reason of adopting this terrain is that the landscape appears very rugged and the local optima can be imaged as “mountains” in real life, which is similar to the real terrain. The mission space is limited within the space of $[0,10] \times [0,10] \times [0,1.5]$. The obstacles are the zones that are dangerous and even prohibited for the UAV to fly through. In our scenarios, the obstacles are depicted as the range of hostile missiles and mountains. The number of obstacles is varied by randomly setting the missiles on the ground in the range of $[1,9] \times [1,9]$.

Specifically, the number of missiles in the 5 scenarios is set as 7, 15, 30, 60 and 120, respectively. For each missile, a coupled radar is set aside the missile. The diameters of the range of missiles and radar detections are set 0.5 and 1.5, respectively. The start position of mission is set at (0.5, 0.5, $h([0.5, 0.5])$) and the destination is set at (9.6, 9.6, $h([9.6, 9.6])$).

B. Compared Algorithms

In the literature, there are quite a few related work focusing on planning obstacles-free paths. In this simulation, we select 3 recently proposed EA-based planners as the first group of compared algorithms, denoted as planner A, B and C, respectively. The aim of this group of comparisons is to show the effectiveness and efficiency of the proposed planner. Planner A [1] was based on Genetic Algorithms (GAs). The candidate paths were first initialized in Polar coordinate system and then evolved in the Cartesian coordinate system. The evolution was processed by a single-point crossover and Gaussian mutation. The immigrants were also included. Planner B [29] encoded the candidate paths in the rotated coordinate system. Within such a codification, the evolution process was driven by a simple Estimation of Distribution Algorithm (EDA), i.e., UMDA_c. [34] suggested a set of comparison measures for UAV path planning. By using these measures, a lot of EA-based variants, including GAs, DEs and Particle Swarm Optimizations (PSOs) were compared. Among them, two DE-based approaches, i.e., D14 and D15 in [34], were found most effective. As D14 and D15 perform generally the same, we thus simply employ D15 as the compared planners C. Planner C encodes the candidate paths in Cartesian coordinate system. A DE/rand/1/bin reproduction strategy is used to evolve the candidate paths. All these three planners employ the same selection strategy with the proposed planner.

Despite of the first comparison group, two variants of the proposed planner were also employed as compared planners. The purpose of this comparative study is to show how the proposed separate evolution idea improves the performance of path planning. We denote these 2 planners as Planner D and E, respectively. Both these 2 planners use the same selection strategy and EA, i.e., JADE, with the proposed planner. Specifically, Planner D encodes the waypoints in the rotated coordinate system with external restriction as the proposed planer does, while it excludes the proposed separate evolution strategy. Instead, it evolves the whole candidate path as the existing work does. Planner E evolves the waypoints separately as the proposed planner does, but it encodes the waypoints in the ordinary Cartesian coordinate system.

There are two kinds of parameters for planners, i.e., non-EA-related parameters and EA-related parameters. One typical non-EA-related parameter is the number of waypoints in a path, i.e., N_w. In the UAV path planning problems, a candidate path is usually represented as a sequence of waypoints. This candidate path is in fact an approximation to a real flight. From this point of view, more waypoints can keep the candidate path closer to a real flight. However, the search space will be too large and both the effectiveness and efficiency of the planner will fall. To balance this trade-off, there is no widely acknowledged criterion for choosing an optimal N_w. Instead, the existing planners usually select a rather small N_w that sufficiently guarantees the feasibility of candidate paths. This idea is also used in this paper to set N_w. In our simulation, the increasing obstacles in 5 scenarios lead to increasingly narrower and more zigzag feasible passageway for the UAV. To keep the path sufficiently smooth and safe, N_w should be increased for the scenarios with more obstacles. Thus, by testing several different possible values, we find some feasible N_w, i.e., N_w =7, 10, 12, 15 and 20, for the proposed planner in the corresponding scenarios with 7, 15, 30, 60 and 120 obstacles, respectively. Taking the scenario with 7 obstacles as an example, we tested the proposed planner with $N_w = 4, 7, 10$. We found that $N_w = 4$ cannot guarantee good performances of the proposed planner, while $N_w = 10$ requires much more computational time. Hence, we set $N_w = 7$ for that scenario. Generally speaking, some other values of N_w can also be used as long as the feasibility of candidate paths and the computational efficiency can be guaranteed.

Relating to N_w, another non-EA-related parameter is N_d, i.e., the number of dividing points in each segment. Recall that the purpose of using dividing points is to detect the violations of segments regarding the missiles, radars and mountains. If the interval between two adjacent dividing points is smaller than the range of missiles, radars and mountains, the violations of segment are highly possible to be detected. This geometric relation can be depicted as follow,

$$\frac{PL}{(N_w-1)\cdot N_d} < D \tag{27}$$

where PL is the path length and D is the minimal diameter of the range of missiles, radars or mountains. The ranges of mountains are usually larger than 0.5, i.e., the diameter of the range of missiles. Hence, we set $D = 0.5$. The smallest N_w, i.e., 7, and the largest feasible path length, which is 1.5 times of the distance between start and destination, are also considered. The value 1.5 is the preference of Minimal Path Length Ratio, as shown in Table III. According to (27), we have $N_d > 4$. Generally, N_d reflects the trade-off between accuracy of evaluations and computational cost. The larger N_d is, the higher accuracy of evaluations we can get, while the efficiency will fall. In this simulation, we simply set $N_d = 6$ for all the planners. Furthermore, we also test the proposed planner with $N_d = 12$ and $N_d = 18$ to see how N_d influences the planner.

The EA-related parameters of planners A and C are those suggested in the original work [2] and [34]. In [29], the EA-related parameters of planner B, i.e., population size, are problem-dependent. For the purpose of unifying the population sizes in different scenarios, they are set as 200 in this paper. The EA-related parameters of Planners D and E are set the same with the proposed planner. As all the components of the proposed planner are parameterless, there is actually only one EA-related parameter to be fine-tuned, i.e., the population size N_p. After a set of parameter sensitive analyses (which will be discussed later), N_p is set to 10. For intuition, the parameter settings of these 8 planners are listed in Table II, where New_6,

New_{12} and New_{18} are the proposed planner with $N_d = 6, 12, 18$, respectively.

TABLE II. PARAMETER SETTINGS

Planners	Parameters
A	N_p=30, N_s=12, P_c=0.75, P_m=0.008, C_{ms}=0.1, C_{mb}=0.5; N_d=6
B	N_p=200, N_s=100; N_d=6
C	$N_p = 100$, $F_{min} = 0.1$, $F_{max} = 0.8$; N_d=6
D	N_p=10; N_d=6
E	N_p=10; N_d=6
$\boldsymbol{New_6}$	N_p=10; N_d=6
$\boldsymbol{New_{12}}$	N_p=10; N_d=12
$\boldsymbol{New_{18}}$	N_p=10; N_d=18

C. Performances Measures

Through the whole simulations, the best path of each planner in each scenario is output when 100 generations run out. All the results are obtained by repeating N_r=25 runs on the Matlab 2012b software on a windows-8 personal computer with i3-2350 @ 2.30GHz CPU and 2GB RAM.

To compare the final outputs, generally, most of the previous work lack of a statistical analysis. This situation makes the comparison among UAV planners far from rigorous. Fortunately, some researchers [34] have noticed this gap and suggested several metrics for statistical comparison in UAV path planning domain. In this paper, we adopt one of them, i.e., the Statistical Front-Dominance Ranking Procedure (SFDRP) metric to measure the ability of the 8 planners. SFDRP measures the performances of two planners, for example, planner A and B, to see if they are statistically different by comparing their final outputs in terms of corresponding objectives and constraints. The term $\left(\diamondsuit_{A_l}^{B_{1:N_r}}\right)$ counts the numbers of the best path of the l^{th} run obtained by planner A is dominated by each of the N_r best paths obtained by planner B and vice versa $\left(\diamondsuit_{B_l}^{A_{1:N_r}}\right)$. To be detailed, $\diamondsuit_{A_l}^{B_{1:N_r}} = \sum_{m=1:N_r} I_c(A_l \prec B_m)$ and $\diamondsuit_{B_l}^{A_{1:N_r}} = \sum_{m=1:N_r} I_c(B_l \prec A_m)$, where $I_c(\cdot)$ is the indicator function that returns 1 if the input condition is true and 0 otherwise, and $A \prec B$ means B dominates A. Then, the non-parametric Wilcoxon rank sum test is applied to the vectors $\left[\diamondsuit_{A_1}^{B_{1:N_r}} + 1, \diamondsuit_{A_2}^{B_{1:N_r}} + 1, \ldots, \diamondsuit_{A_{N_r}}^{B_{1:N_r}} + 1\right]$ and $\left[\diamondsuit_{B_1}^{A_{1:N_r}} + 1, \diamondsuit_{B_2}^{A_{1:N_r}} + 1, \ldots, \diamondsuit_{B_{N_r}}^{A_{1:N_r}} + 1\right]$. If this test finds a statistically significant difference, the median of each vector can be used to infer which planner dominates the other one. To illustrate the results of the statistical test, [34] suggests a type of graphic presentation (as seen in Fig. 8). In each cell of each graphic we represent when a planner in the Y axis is better (less dominated, in white), equivalent (no statistically different, in gray) or worst (more dominated, in black) than a planner in the X axis.

Despite of the statistical analysis of the outputs, we have also constructed the comparisons on the average of the convergence speed and elapse time of each planner. Briefly, we first calculate the generation when all the constraints are satisfied, denoted as G_c, the generation when PKill and PRL are satisfied, denoted as G_s, and the generation when RRD and FA are satisfied, denoted as G_t. And the elapse time ET of each planner in each run is noted. Then we average each metric above with respect to those runs where the corresponding indices are satisfied, denoted as $\widetilde{G_c}$, $\widetilde{G_s}$, $\widetilde{G_t}$ and $\widetilde{ET}$, respectively. If the objectives/constraints of any planner have never been satisfied through all 25 runs, the corresponding $\widetilde{G_c}/\widetilde{G_s}/\widetilde{G_t}$ will be noted as N/A. Lastly, the number of successful runs out of total 25 runs is also calculated, denoted as SR. Generally, a constraint or an objective is said to be satisfied if its value is less than its preference. If all constraints and objectives are satisfied, it is said a successful run. The preferences listed in Table III. As seen in Table III, no constraint violation is allowed. The PLR should be less than 1.5 as we have introduced earlier. The preference of PKill is set to be 0 so that no risk of kill is permitted. That is, the zones within the range of hostile missiles are actually obstacles that are prohibited to fly through. The UAV should fly no higher than 0.5 above the terrain. Since that the diameter of range of missiles is also 0.5, the preference of flight altitude in fact prevents the UAV from flying above the missiles. Consequently, the UAV can only bypass the obstacle from its flank. The preference of RRD should be related to the scale of the intensity of the radars.

TABLE III. PREFERENCES OF CRITERIA

Constraints				
Name	*Maximal Turning Angle*	*Limited UAV Slope*	*Terrain Limited*	*Map Limited*
Priority Level	1^{st}	1^{st}	1^{st}	1^{st}
Preference	0	0	0	0
Objectives				
Name	*Minimal Path Length Ratio*	*Minimal Risks of Kill*	*Minimal Risks of Radar Detection*	*Minimal Flight Altitude*
Priority Level	2^{st}	2^{st}	3^{st}	3^{st}
Preference	1.5	0	30	0.5

D. Results and Analyses

First, the results of all the 25 runs of those 8 planners in each scenario have been statistically analyzed. Those results are shown by means of graphic representations in Fig.8. Among all the planners, New_6, New_{12} and New_{18} perform the best when the number of obstacles increases. Within these three new planners, New_6 has slightly better results. However, we cannot claim any superiority of $N_d = 6$ over $N_d = 12$ and $N_d = 18$ as they have different evaluation accuracy. In [34], the DE-based planner (Planner C) is empirically better than the GA-based planner. This also happens when the obstacles are few (see Fig. 8(a)). However, when encountering more obstacles, GA-based Planner A performs significantly better than Planner C, which seems to be in contradiction with the conclusion in [34]. In fact,

the reason behind these distinct results is comprehensible. In [34], all compared planners are encoded in Cartesian coordinate system, and the selected DEs outperform GAs. While the candidate paths of Planner A [1] here are first generated in Polar coordinate system and then evolved freely in Cartesian coordinate system. The Polar coordinate system has been acknowledged to be able to reduce the search space. Hence, when the obstacles increase and the feasible space decreases, Planner A overtakes Planner C in terms of the reduced search space. It can be inferred from this pair of results that the choice of the encoded coordinate system has an important impact on the planners especially when there are large number of obstacles. Similar conclusion can be obtained by comparing New_6 and Planner E. The only difference between these two planners is the employed coordinate systems. Apparently, New_6 outperforms Planner E in all scenarios. In relation to this pair of comparison, the advantage of New_6 can be owed to the rotated coordinate system that it can significantly reduce the search space. Note that the superiority of New_6 is not only based on the employed rotated coordinate system. As seen in Fig. 8, Planner E performs significantly better than Planner D in the latter four scenarios. As the basis planners of New_6, the performances of Planner D and Planner E reflect the real contributions of the rotated coordinate system and separate evolution to New_6. From the results, it can be inferred that the rotated coordinate system contributes less, comparing to the proposed idea of separate evolution.

Besides statistical analyses, the convergence speed, runtime and successful rate of the 8 planners are listed in Table IV-VIII. Planner New_6, Planner D and Planner E consume the least $\widetilde{ET}$ due to the small population size, i.e., $N_p = 10$. New_{12} and New_{18} also have the same population size, while they are more computationally expensive. This is because they evaluate more dividing points for each segment, which elevates the evaluation accuracy while compromises the efficiency. Planners A, B and C spend much more computational time than the others due to

TABLE IV. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE OF 8 PLANNERS ON 7-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
New_6	5.84	6.60	7.04	127.74	100
New_{12}	7.44	10.40	11.44	208.18	100
New_{18}	8.72	12.88	13.42	360.16	96
A	5.75	8.48	8.48	363.55	100
B	4.92	5.8	6.04	1549.37	100
C	24.96	27.20	27.52	1392.68	100
D	28.08	30.12	31.24	122.04	100
E	16.71	18.40	18.74	128.08	92

TABLE V. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE OF 8 PLANNERS ON 15-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
New_6	8.44	10.44	11.08	218.99	100
New_{12}	10.52	20.52	20.52	360.35	100
New_{18}	10.36	15.76	16.72	530.34	100
A	13.24	19.71	23.70	577.38	92
B	18.52	24.64	28.20	2811.79	100
C	83.00	86.00	89.50	2233.14	8
D	62.16	63.56	66.83	195.36	72
E	38.75	43.09	43.74	204.32	92

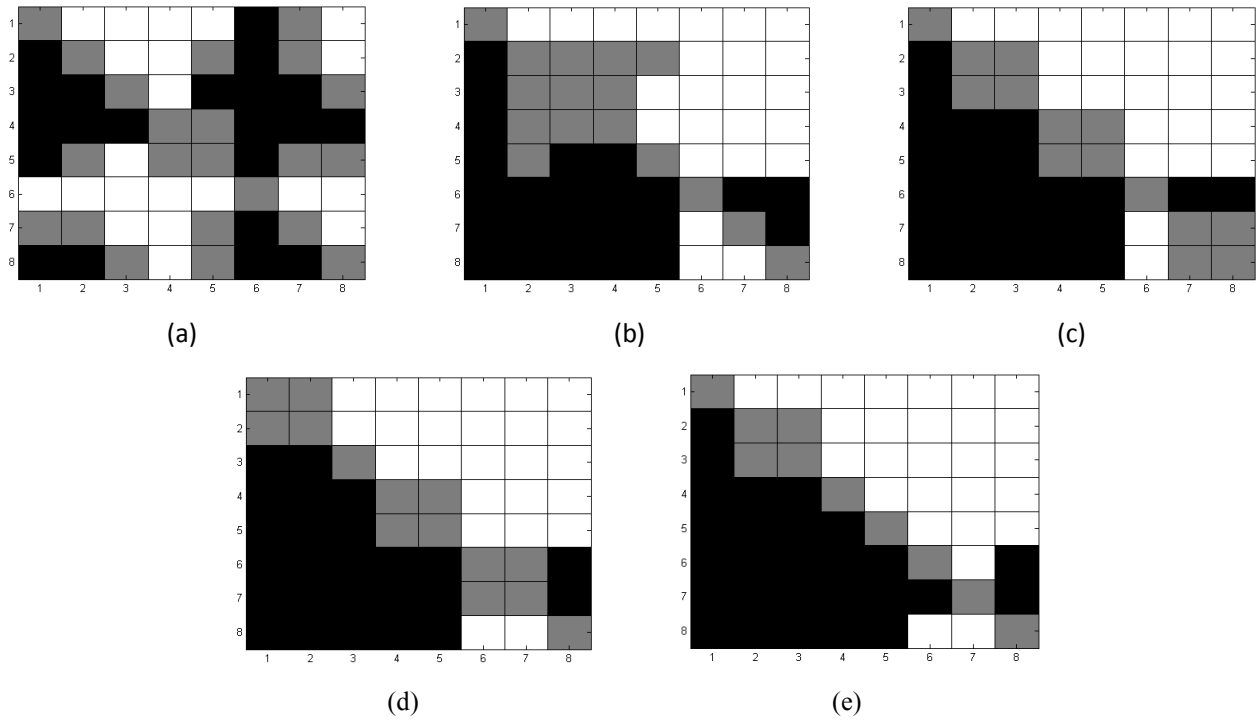

Fig. 8.The statistical analyses of the 8 planners in scenarios with 7, 15, 30, 60 and 120 ADUs are shown in (a)-(e), respectively. In the axes, 1-8 are with respect to the Planners New_6, New_{12}, New_{18}, A, B, C, D and E. In each cell of each graphic we represent when the planners in the Y axis is better (less dominated, in white), equivalent (no statistically different, in gray) or worst (more dominated, in black) than the planners in the X axis.

TABLE II. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE OF 8 PLANNERS ON 30-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
New_6	11.52	16.68	20.44	250.64	100
New_{12}	12.28	23.32	28.58	424.99	96
New_{18}	11.84	22.88	29.38	649.72	96
A	8.92	32.40	29.23	664.71	52
B	38.36	50.36	61.52	3564.82	92
C	N/A	N/A	N/A	2546.61	0
D	80.67	84.75	84.00	238.14	8
E	57.00	64.80	65.00	247.90	20

TABLE III. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE OF 8 PLANNERS ON 60-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
New_6	13.36	18.32	36.92	303.55	96
New_{12}	12.92	22.04	46.04	481.16	96
New_{18}	12.76	19.80	45.92	769.26	100
A	16.21	41.71	40.00	891.92	12
B	77.13	88.73	N/A	4512.69	0
C	N/A	N/A	N/A	3075.40	0
D	N/A	N/A	N/A	288.53	0
E	N/A	N/A	N/A	295.40	0

the larger population sizes. However, those larger population sizes cannot remain them effective in scenarios involving lots of obstacles due to their poorly exploitation of high quality waypoints. Specifically, Planner C deteriorates rapidly when obstacles increase. Planner A has the ability of generating paths satisfying all 4 constraints rapidly in all scenarios. This is because the polar coordinate system essentially restricts the turning angle and slope and Planner A actually has only two constraints, i.e., *Map Limited* and *Terrain Limited*, to satisfy.

As a result, New_6 keeps high stability on both efficiency and effectiveness in all scenarios. On one hand, its $\widetilde{ET}$ is acceptable and its convergence speed is very fast. Note that the $\widetilde{ET}$ is the total runtime for 100 generations. Thus, it is easy to know that New_6 spends the least runtime to obtain the feasible solution by calculating $\frac{\widetilde{G_t}\times\widetilde{ET}}{100}$. On the other hand, New_6 keeps very high SR for all scenarios and is statistically the best.

As analyzed above, New_{12} and New_{18} also have very good

TABLE IV. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE OF 8 PLANNERS ON 120-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
New_6	19.56	26.29	48.55	408.88	88
New_{12}	16.64	24.92	53.95	560.63	80
New_{18}	17.44	26.38	51.15	854.05	80
A	17.16	N/A	N/A	1116.74	0
B	N/A	N/A	N/A	5378.61	0
C	N/A	N/A	N/A	3845.90	0
D	N/A	N/A	N/A	391.78	0
E	N/A	N/A	N/A	386.24	0

TABLE V. COMPARISON OF THE CONVERGENCE SPEED, RUNTIME AND SUCCESS RATE AMONG PLANNERS A, D AND E ON 60-MISSILES SCENARIO

Planner	$\widetilde{G_c}$	$\widetilde{G_s}$	$\widetilde{G_t}$	$\widetilde{ET}$ (s)	SR (%)
A	8.92	32.40	29.23	664.71	52
D	80.67	84.75	84.00	238.14	8
D_{30}	75.20	81.57	91.25	777.99	16
E	57.00	64.80	65.00	247.90	20
E_{30}	50.80	59.46	64.46	715.12	96

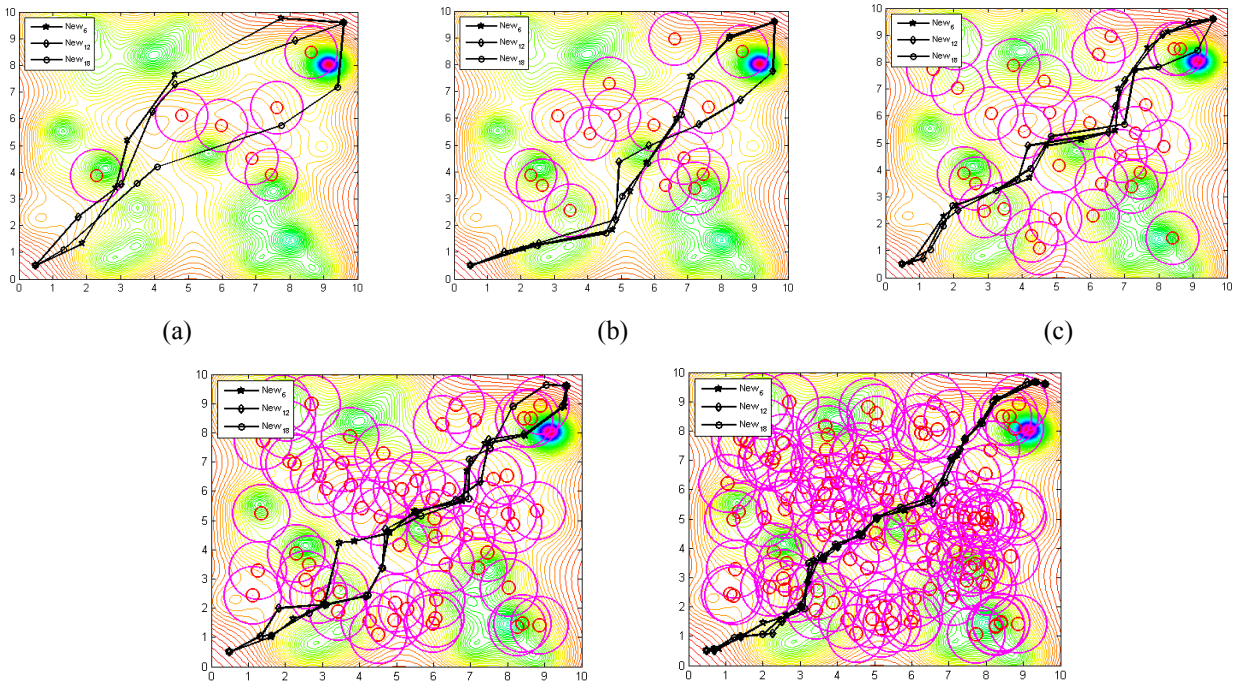

Fig. 9.The best paths of New_6, New_{12} and New_{18} in 5 scenarios are shown in (a)-(e), respectively. The star-line represents the best path of Planner New_6, the diamond-line indicates the best result of Planner New_{12} and the circle-line is the best path of Planner New_{18}.

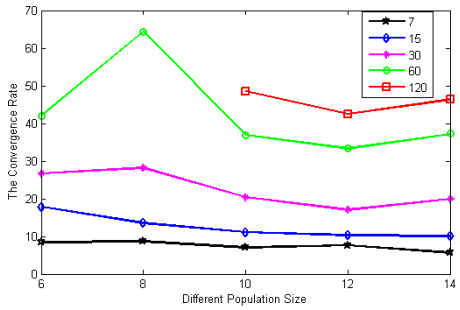

Fig. 10(a). $\widetilde{G_t}$ of the proposed planner with $N_p = 6, 8, 10, 12, 14$ on 5 scenarios. Here, the proposed planner cannot produce feasible path with $N_p = 6$ and 8 on the last scenario, thus the corresponding lines miss

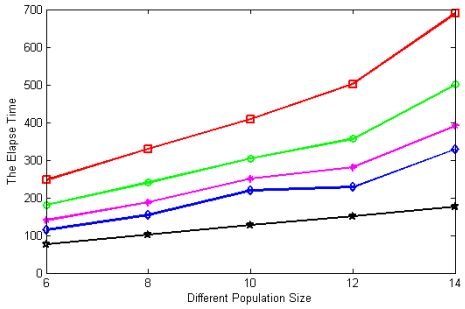

Fig. 10(b). $\widetilde{ET}$(s) of the proposed planner with $N_p = 6, 8, 10, 12, 14$ on 5 scenarios.

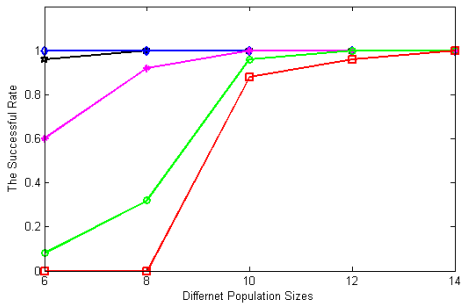

Fig. 10(c). SR of the proposed planner with $N_p = 6, 8, 10, 12, 14$ on 5 scenarios.

final outputs. It is difficult to tell how N_d impacts the proposed planner by statistical analyses. In other words, the evaluation of candidate paths are essentially related to N_d. To illustrate the impacts of N_d, the best paths, out of 25 runs, of New_6, New_{12} and New_{18} in the 5 scenarios are shown in Fig. 9 (a)-(e) in a 2-D view, respectively. In these figures, the ranges of missiles and radars are represented by groups of concentric circles. With the preferences in Table III, the paths are forbidden to go through the smaller circles and are better to be out of the bigger circles. The terrain is depicted in color contour line where higher places are darker. The paths of New_6, New_{12} and New_{18} are presented as star-lines, diamond-lines and circle-lines, respectively. The stars, diamonds and circles are the corresponding waypoints. As seen, the best paths always keep smooth and avoid all the obstacles, i.e., missiles and mountains, in all scenarios. No significant differences between the best paths of three planners can be observed in Fig. 9, which implies that $N_d = 6$ is sufficient for planners to detect the radars, missiles and mountains. Specially, one interesting phenomenon is that the paths of the three planners get closer when obstacles increase. This is because the increasing obstacles reduce the feasible space and thus the effective planners have fewer choices for producing best paths.

Both Planner D and Planner E can overtake Planner C. While the superiority of Planner A over Planner D and Planner E may be because Planner A uses a larger population size. To verify this viewpoint, we have made another comparison among Planner A, Planners D and E with the same population size, i.e., $N_p = 30$. As the purpose of this comparison is to show whether a larger population size will influence the performances of Planner D and Planner E or not, we simply give just one example on the scenario with 30 obstacles for illustration. The results are shown in Table IX. As seen, when N_p increases from 10 to 30, the SR of Planner E improves significantly from 20% to 96%, which means Planner E_{30} is more stable than Planner A. Hence, it verifies the point that it is the larger population size that makes Planner A outperforms the original Planner E. For Planner D, although Planner D_{30} is slightly better than Planner D, it seems that the population size $N_p = 30$ is still not enough for a remarkable promotion. It can be observed that, due to the effective exploitation of waypoints, Planner E_{30} benefits much more than Planner D_{30} from the increased N_p. This comparison shows the advantages of the proposed idea of separate evolution. On the other hand, it again verifies that the separate evolution idea contributes more than the rotated coordinate system to the proposed planner.

Besides the comparisons among planners, we also carry out a parameter sensitive analyses of the new proposed planners. As most planners employ 6 dividing points, we thus analyze the parameter sensitivity based on New_6. There is only one parameter, i.e., N_p, that needs carefully fine-tuned. To find out a proper N_p, the performances of $\widetilde{G_t}$, $\widetilde{ET}$ and SR of New_6 with different values of N_p, i.e., 6, 8, 10, 12 and 14, on the 5 scenarios are recorded and shown in Fig. 10 (a)-(c). The reason of testing these three metrics is that $\widetilde{G_t}$ indicates the generation when the path has satisfied all objectives and constraints, i.e., the convergence speed, $\widetilde{ET}$ presents the real running time, while SR reflects the stability of the planner. And the reason of testing such small population sizes is because we believe the separately evolving strategy has highly effectively exploited the better waypoints during optimization and thus the population can be reduced. As seen in Fig. 10 (a)-(c), in each scenario, as the N_p increases, $\widetilde{G_t}$ keeps generally the same, and SR increases. However, $\widetilde{ET}$ also generally increases, which means that higher computation times will be required to produce relatively good outputs. These results show that such a small N_p appears very sensitive, hence we need to determine a proper N_p. To balance the trade-off between the effectiveness and efficiency, we suggest N_p=10.

In the above analyses, the advantages of the proposed idea of separate evolution has been discussed. It is shown that the proposed planner is more effective and efficient than the compared planners. Besides, it is also shown that the separate

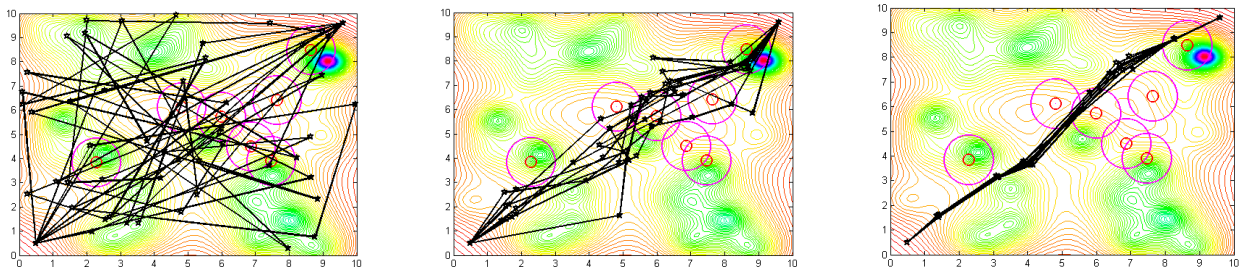

Fig. 11 the 1st, 40th and 80th generation of the procedure of Planner E on the first scenarios is recorded to illustrate the behavior of the proposed idea of separate evolution.

evolution makes the proposed planner more stable than the compared planners. This stability results from the fact that the separate evolution is insensitive to the quality of the initialized solutions. To illustrate this viewpoint, the process of Planner E on the first scenario is recorded. The reason of choosing Planner E instead of New_6 is to eliminate the influence of the rotated coordinate system. As seen in Fig.11, the initialized waypoints are scattered in the mission space and the candidate paths are far from feasible. Under the impact of the separate evolution, waypoints in each order will finally converge to a rather good state, respectively. This is because, for all the i^{th} waypoints, there will be one or several optimal locations, in terms of the global information of all the other waypoints. The process of the separate evolution can be regarded as a sub-problem that finding the optimal solution for the i^{th} waypoint of a path, regarding all the i^{th} waypoints as the candidate population. From this point of view, the separate evolution can have a better ability of convergence and thus is insensitive to the quality of the initialized population.

V. Conclusion And Future Work

The path planning technique is very important to the autonomy of Unmanned Aerial Vehicles (UAVs). In this field, three major tasks are required to be solved. This paper studies the off-line planning problem, which is the basic of the other two, i.e., on-line planning and cooperative planning. There have been quite a few research work proposed for the off-line planning problem. However, they are usually ineffective when the scenarios involve lots of obstacles. The reason behind those failures are that the information of better waypoints are not highly exploited during the optimization. This paper proposes a new idea to solve this shortage by separately evaluating and evolving the waypoints. To practice this idea into the UAV path planning, we first derive a set of new evaluation functions from the existing evaluation functions for the evaluations of single waypoints. This derivation is based on that those evaluation functions are separable on waypoints. For the purpose of separately evolving the waypoints, JADE is employed. A rotated coordinate system is also used to encode the waypoints for the reduction of the search space. To validate the proposed planner, we compare it with 7 planners from different viewpoints on 5 scenarios with increasing obstacles. From the simulation results, the effectiveness and efficiency of the proposed planner are shown. The advantages of the separate evolution idea are also discussed.

Although the advantages of the new proposed planner has been shown, the EA-based off-line planners still required to be further studied. For example, our work has to pre-define the number of waypoints, i.e., N_w, which is still difficult for a UAV to decide on-the-fly. Hence, a strategy that can autonomously choose the number of waypoints for UAV is to be studied.

As the on-line planning and cooperative planning are based on off-line planning, we may further extend our work onto those two problems as they are more practical in real life missions. As a matter of fact, the path planning for single UAV is usually regarded as the cornerstone of cooperative path planning for multiple UAVs. The cooperation constraints, e.g., time cooperation, distance cooperation, are usually independent from the other objectives and constraints. Hence, ideally, we can extend our proposed planner to a cooperative planner by introducing external cooperation constraints as [1] does.

References

[1] E. Besada-Portas, L. de la Torre, J.M. de la Cruz, B. Andres-Toro, Evolutionary trajectory planner for multiple UAVs in realistic scenarios, *IEEE Transactions on Robotics.* 26 (2010) 619–634.

[2] Sarris, Zak, and STN ATLAS. Survey of UAV applications in civil markets (June 2001). *The 9 th IEEE Mediterranean Conference on Control and Automation (MED'01)*. 2001.

[3] C. Goerzen, Z. Kong, B. Mettler. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems.* 57(1-4). 2010. Pages 65-100.

[4] C. Zheng, L. Li, F. Xu, F. Sun, M. Ding, Evolutionary route planner for unmanned air vehicles, *IEEE Transactions on Robotics* 21 (2005) 609–620.

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

[6] Latombe, J.: Robot Motion Planning. *Kluwer Academic*, Boston (1991)

[7] C. Yang, H. He and X, An, Motion Planning of Intelligent Vehicles. *In proceedings of IEEE International Conference on Vehicular Electronics and Safety,* 13-15 Dec. 2006. pp. 333-336. Beijing.

[8] N. Dadkhah, B. Mettler. Survey of Motion Planning Literature in the Presence of Uncertainty: Considerations for UAV Guidance. *Journal of Intelligent and Robotic Systems.* 65(1-4). 2012. Pages 233-246.

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

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

[11] K. I. Trovato. A* Planning in Discrete Configuration Spaces of Autonomous Systems. *PhD thesis*, Amsterdam University, 1996.

[12] A. G. Richards, J. P. How. Aircraft Trajectory Planning with Collision Avoidance Using Mixed-Integer Linear Programming. *In Proceeding of the American Control Conference* (May 2002).
[13] J. J. Ruz, O. Arévalo, de la Cruz, J.M, and G. Pajares. Using MILP for UAVs Trajectory Optimization under Radar Detection Risk. *In Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation* (Prague, September 2006).
[14] A.Raghunathan, V.Gopal, D.Subramanian, L.Biegler and T.Samad. Dynamic Optimization Strategies for 3D Conflict Resolution of Multiple Aircraft. *AIAA Journal of Guidance, Control and Dynamics*, 27 (4). (2004), 586-594.
[15] P. Bhattacharya and M. L. Gavrilova. Voronoi diagram in optimal path planning, *in Proc. IEEE Int. Symp. on Voronoi Diagrams in Science and Engineering*, 2007, pp. 38-47.
[16] R. W. Beard. Coordinate system target assignment and intercept for unmanned air vehicles, *IEEE Trans. on Robotics and Automation*, vol. 18, no. 6, pp. 911-922, Dec. 2002.
[17] C. Hocaoglu and A.C. Sanderson. Planning multiple paths with evolutionary speciation, *IEEE Trans. on Evolutionary Computation,* Jun 2001, vol.5, pp. 169-191.
[18] S. Mittal and K. Deb. Three-dimensional offline path planning for UAVs using multi-objective evolutionary algorithms, *in Proc. IEEE Congr. Evol. Comput*., 2007, vol. 7, pp. 3195–3202.
[19] I. Hasircioglu, H. R. Topcuoglu, and M. Ermis. 3-d path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms, *in Proc. Genet. Evol. Comput. Conf.*, 2008, pp. 1499–1506.
[20] I. K. Nikolos, N. C. Tsourveloudis, andK. P.Valavanis. Evolutionary algorithm based path planning for multiple UAV cooperation, *in Advances in Unmanned Aerial Vehicles*. Berlin, Germany: Springer-Verlag, Jan. 2007, pp. 309–340.
[21] R. Zhang, C. Zheng, and P.Yan. Route planning for unmanned air vehicles with multiple missions using an evolutionary algorithm, *in Proc. IEEE 3rd Int. Conf. Nat. Comput*., 2007, pp. 1499–1506.
[22] De La Cruz, J.M., E. Besada-Portas, L. de la Torre, B. Andres-Toro, and J. A. Lopez-Orozco. Evolutionary path planner for UAVs realistic environments. *in 10th Annual Genetic and Evolutionary Computation Conference, GECCO 2008*, July 12, 2008 - July 16, 2008. 2008. Atlanta, GA, United states: Association for Computing Machinery.
[23] G. Sanders, T. Ray. Optimal offline path planning of a fixed wing unmanned aerial vehicle (UAV) using an evolutionary algorithm. *IEEE Congress on Evolutionary Computation,* 25-28 Sept. 2007. pp. 4410-4416. Singapore.
[24] C. Jia. UAV Mission Planning: Problem Modeling and Solution Methods. *PhD thesis*, 2011, National University of Singapore.
[25] A. Brintaki, I. Nikolos, Coordinate systemd UAV path planning using differential evolution, *Operational Research* 5 (2005) 487–502.
[26] S. Li, X. Sun, Y. Xu, Particle swarm optimization for route planning of unmanned aerial vehicles, in: *2006 IEEE International Conference on Information Acquisition*, pp. 1213–1218.
[27] Q. Ma, X. Lei, Application of improved particle swarm optimization algorithm in UCAV path planning, *in: International Conference on Artificial Intelligence and Computational Intelligence*, pp. 206–214.
[28] P.B. Sujit, R. Beard, Multiple uav path planning using anytime algorithms, *in: 2009 American Control Conference*, pp. 2978–2983.
[29] P. Yang, K. Tang and J. A. Lozano, "Estimation of Distribution Algorithms based Unmanned Aerial Vehicle Path Planner Using a New Coordinate System", *in: IEEE Congress on Evolutionary Computation,* July, 2014, Beijing, accepted
[30] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[31] Das, S. and Suganthan, P.N., Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Coput.,* vol 15, no 1, pp. 4-31, Oct. 2010.
[32] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
[33] C. M. Fonseca and P. J. Fleming. Multi-objective optimization and multiple constraint handling with evolutionary algorithms—Part I: Unified formulation, *IEEE Trans. Syst.,Man, Cybern. A, Syst. Humans*, vol. 18, no. 1, pp. 26–37, Jan. 1988.
[34] Besada-Portas, E., De La Torre, L., Moreno, A., & Risco-Martín, J. L. (2013). On the performance comparison of multi-objective evolutionary UAV path planners. *Information Sciences*, *238*, 111-125
[35] Zhang, Xiangyin, and Haibin Duan. "An Improved Constrained Differential Evolution Algorithm for Unmanned Aerial Vehicle Global Route Planning."*Applied Soft Computing* (2014).

Peng Yang received his B.Eng. degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2012. He is currently pursuing the Ph.D degree from the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China (USTC). His research interests include evolutionary computation, estimation of distribution algorithm and their real-world applications.

Ke Tang (S'05-M'07-SM'13) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2002, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007.

He is currently a Professor with the USTCBirmingham Joint Research Institute in Intelligent Computation and Its Applications, School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui, China. He is also an Honorary Senior Research Fellow with the School of Computer Science, University of Birmingham, Birmingham, U.K. He has authored and co-authored more than 40 refereed publications. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real-world applications.

Prof. Tang is an Associate Editor of the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE, an editorial board member of three international journals, and the Chair of the IEEE Task Force on Large Scale Global Optimization.

Jose. A. Lozano (M'04) received the B.S. degree in mathematics and the B.S. degree in computer science from the University of the Basque Country, San Sebastian-Donostia, Spain, in 1991 and 1992, respectively, and the Ph.D. degree in computer science from the University of the Basque Country in 1998.

Since 2008, he has been a Full Professor with the University of the Basque Country, where he leads the Intelligent System Group. He is the co-author of more than 50 ISI journal publications and the co-editor of the first book published about estimation of distribution algorithms. His current research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real- world applications.

Prof. Lozano is an Associate Editor of the IEEE Transactions on Evolutionary Computation and a member of the Editorial Board of Evolutionary Computation, Soft Computing, and other three journals.

Xianbin Cao (M'08–SM'10) received the B.S. degree in computer science and the M.S. degree in information and system from Anhui University, Hefei, China, in 1990 and 1993, respectively, and the Ph.D. degree in intelligent information processing from the University of Science and Technology of China (USTC), Hefei, in 1996.

He joined the Department of Computer Science and Technology, USTC, in 1996 and became an Associate Professor and a Professor in 1999 and 2005, respectively. He was the Vice Director of the department and the Artificial Intelligence Research Center from 2006 to 2009. Since 2005, he has been the Administrative Director of the Anhui Province Key Laboratory in Computing and Communication. He is currently a Professor with the School of Electronic and Information Engineering, Beihang University,
Beijing, China. He has published more than 100 books, book chapters, and papers in these areas since 1993. His current research interests include natural computation, intelligent transportation systems, and information security.