Robotica (2018) volume 36, pp. 904–924. © Cambridge University Press 2018
doi:10.1017/S0263574718000139

Constrained coverage path planning: evolutionary and classical approaches

S. M. Ahmadi†, H. Kebriaei†,‡,* and H. Moradi†,§

†*School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran*
‡*School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*
§*Intelligent Systems Research Institute, SKKU, Suwon, South Korea*

(Accepted January 22, 2018. First published online: February 21, 2018)

SUMMARY
The constrained coverage path planning addressed in this paper refers to finding an optimal path traversed by a unmanned aerial vehicle (UAV) to maximize its coverage on a designated area, considering the time limit and the feasibility of the path. The UAV starts from its current position to assess the condition of a new entry to the area. Nevertheless, the UAV needs to comply with the coverage task, simultaneously and therefore, it is likely that the optimal policy would not be the shortest path in such a condition, since a wider area can be covered through a longer path. From the other side, along with a longer path, the UAV may not reach to the target in due time. In addition, the speed of UAV is assumed to be constant and as a result, a feasible path needs to be smooth enough to support this assumption. The problem is modeled as an Epsilon-constraint optimization in which a coverage function has to be maximized, considering the constraints on the length and the smoothness of the path. For this purpose, a new genetic path planning algorithm with adaptive operator selection is proposed to solve such a complicated constrained optimization problem. The proposed approach has been compared to some classical approaches like, a modified version of the Artificial Potential Field and a modified version of Dijkstra's algorithm (a graph-based approach). All the methods are implemented and tested in different scenarios and their performances are evaluated via the simulation results.

KEYWORDS: Coverage, Path planning, Unmanned aerial vehicle, Genetic algorithm, Artificial potential field, Dijkstra's algorithm

1. Introduction

Monitoring a designated area while path planning between two points in that area can be considered as a typical problem of monitoring and surveillance.[1,2] For instance, consider a lifeguard unmanned aerial vehicle (UAV), which is called UAV1 from now on, monitoring a designated area over a seashore to warn human lifeguards about swimmers who are in danger. When a swimmer is going to enter the designated area from a neighboring area, the UAV that is responsible for monitoring the neighboring area informs the UAV1 about the expected crossing point. In this situation, UAV1 has to move toward the expected crossing point at the border of two adjacent areas in order to take over and assess the condition of the newly entered swimmer. If the hand over time available to UAV1 to reach the crossing point is greater than the shortest path time, then UAV1 has extra time to cover and monitor a wider portion of its designated area while moving toward the incoming swimmer. Thus, UAV1 may try to maximize the covered area while moving toward the incoming swimmer. This can be realized as an example for coverage-based path planning (CBPP) problem.

The classical motion planning techniques typically seek for a collision free path between a starting point and an endpoint that satisfies certain optimization criteria.[3,4] The performance of the classical

* Corresponding author. E-mail: kebriaei@ut.ac.ir
E-mail: s.m.ahmadi@ut.ac.ir, moradih@ut.ac.ir

methods such as cell decomposition, potential field, visibility-based, and PRM highly depends on the specifications of the environment. For example, the Artificial Potential Field (APF)[5] is a real-time path planning method. This approach has also some limitations such as: falling into local minima, non-smooth movement, oscillations in narrow passages, and oscillations due to obstacles.[6] Researchers have proposed new versions of potential field method to avoid local minima.[7−11] APF has also got many successful results in solving the path planning problem.[12−15] Another category of classical path planning approaches is the graph-based path planning. In this category, an environment is represented by a graph in which, the shortest path algorithms can be used to find the optimal path. For instance, Dijkstra's algorithm[16] finds a shortest path in a graph with non-negative weights. The graph-based approaches have got acceptable results in on-line path planning problems.[17,18]

The path planning problem can be molded also as a constrained optimization problem. However, for large-scale environments and complex objective functions, the classical approaches like linear or non-linear programming methods are not well-suited to solve such a problem. In such cases, evolutionary and AI-based methods have been applied to the path planning problem like artificial neural network, fuzzy logic, particle swarm optimization, genetic algorithm (GA), and hybrid methods.[19−24] GA has been applied to solve path planning problems adaptively in dynamic and uncertain environments.[20,25,26]

Different variants of GAs have been developed to solve single and multi-objective optimization (MOO) problems. Single objective methods can find the minimum distance collision free path, while multi-objective methods are utilized to find an optimal path considering multiple performance criteria. In ref. [25], a single objective optimization is proposed that combines the length of the path and a penalty factor to find the minimum distance collision free path. In ref. [27], this method is improved to cancel the penalty factor. A multi-objective method is proposed in ref. [20] with three objective functions, i.e., length, smoothness, and safety, to evaluate the optimality of a path. These objective functions are introduced in refs. [28-30].

Coverage planning for bounded regions has received much attention in the past two decades because of its application in mine hunting, vacuum cleaning, harvesting, lawn-mowing, and automated painting problem.[31] In ref. [31], various coverage patterns such as *Back and Forth*, *Spike*, *Squarel*, and *Spiral* have been described for two-dimensional (2D) spaces. These patterns have got some successful results in various environments.[32,33] However, the path planning for simultaneous rescue and coverage tasks is not studied in the aforementioned research works.

In this paper, we consider the scenario of monitoring a rectangular environment by a UAV. The aim is to determine a path between the current position of the UAV and an arbitrary goal point that satisfies the hand over time and smoothness constraints, as well as monitoring the maximum area with greater priorities while traversing the path. Such a problem is called CBPP. CBPP is a topic related to large area search and target tracking. In a large area search, instead of covering the whole environment via a fixed coverage pattern, UAVs tries to maximize the coverage area.[34] Target detection and tracking encompasses a variety of decisional problems such as surveillance, monitoring, coverage planning, search, patrolling, and pursuit-evasion along with others.[35] In such a view, CBPP trades-off between coverage and take over/tracking tasks.

We propose different approaches to solve this problem and compare their results: Two methods are based on the *GA* from the evolutionary approaches, one based on *APF* from the classical approaches and another approach based on *Dijkstra's algorithm* from graph-based approaches. To evaluate and compare the results, we define three criteria, i.e., "*Length*," "*Smoothness*," and "*Coverage*," to measure the optimality of the path. The length and the smoothness of the path are considered as the constraints, while the coverage is considered as the objective function that needs to be maximized.

The evolutionary methods work off-line, and the two other proposed methods work on-line. Most of the off-line algorithms correspond to global path/trajectory planning, where the coordinates are predefined before flying. On the other hand, on-line path/trajectory planning corresponds to the local planning approach class, in which decisions are to be made based on new information during operation. The proposed off-line method based on GA gives an initial plan when a UAV faces two simultaneous tasks. By neglecting the speed of a targeted swimmer with respect to the UAV (the speed of swimmer is considered to be much slower than the UAV) and assuming that during the operation, i.e., monitoring and moving toward the goal, no other event happens, the optimal policy given by the off-line method works well. On the other hand, since the off-line method has processing time limitation, the on-line method has to be chosen when the path needs to be updated during the tasks due to new observations.

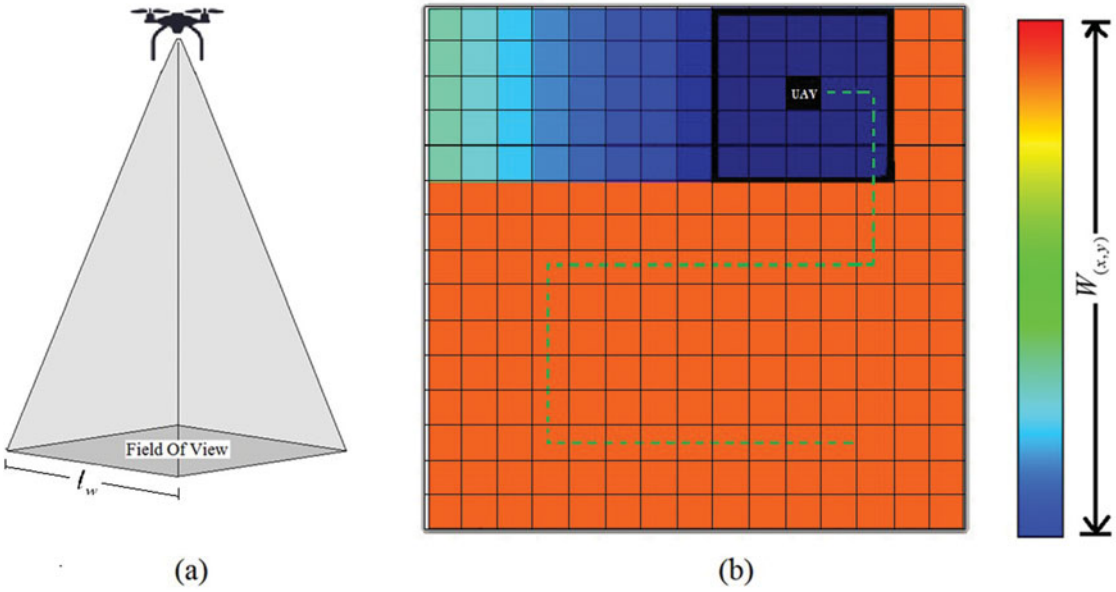

Fig. 1. (a) UAV has a field of view. (b) Priority weights, orange cells have not been visited recently.

The organization of the paper is as follows: In Section 2, we describe the problem statement and introduce the optimization problem. To solve the optimization problem based on the genetic programming approach, we introduce a new chromosome encoding method for path planning namely action-based encoding (ABE) and compare its results with a common chromosome encoding named node-based encoding (NBE),[20] which are described in Sections 3 and 4, respectively. Also, we propose a modified APF to solve the problem with a critical parameter to deal with the coverage task (Section 5). In Section 6, the environment is represented by a graph and appropriate edge – weights are determined to solve CBPP problem by Dijkstra's algorithm. In experiments (Section 7), the results of four different methods, i.e., ABE, NBE, APF, and Dijkstra's algorithm are compared. Section 9 concludes the paper with discussions on the results and giving further research insights.

2. Statement

Consider a lifeguard UAV monitors a designated area according to a coverage pattern shown in Fig. 1(b). In an arbitrary time, when a new swimmer is going to pass in the designated area, the UAV faces two tasks; the incompletely covered area needs to be covered and a new swimmer needs to be monitored. To assure the security of the swimmer, the UAV should reach the swimmer in a limited time, which is called the hand over time. Let v_{UAV} be the speed of the UAV which is assumed to be fixed. The speed of the swimmer is in fact much slower than the speed of the UAV. Therefore, the UAV reaches the swimmer before he/she makes a notable movement. Based on this fact, we assumed the swimmer as a fixed point.

Based on the above scenario, we can consider that the UAV traverses a path between two fixed points. The aim is to find a path with maximum coverage, as well as acceptable length and smoothness. Furthermore, since the vertical movement of the UAV is ignored, the environment is considered to be a 2D grid space with fixed width and height. The UAV starts from a fixed point (S) to the goal point (G) as shown in Fig. 3. To find the optimal path between these two points, three criteria are defined for the optimization problem.

2.1. Coverage

The common task of a UAV is to visit the environment uniformly as much as possible according to a pattern like the *Back and Forth* pattern (Fig. 1(b)). At each cell, the UAV is able to cover a square region, with sizes l_w by l_w, as shown in Fig. 1(a).

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

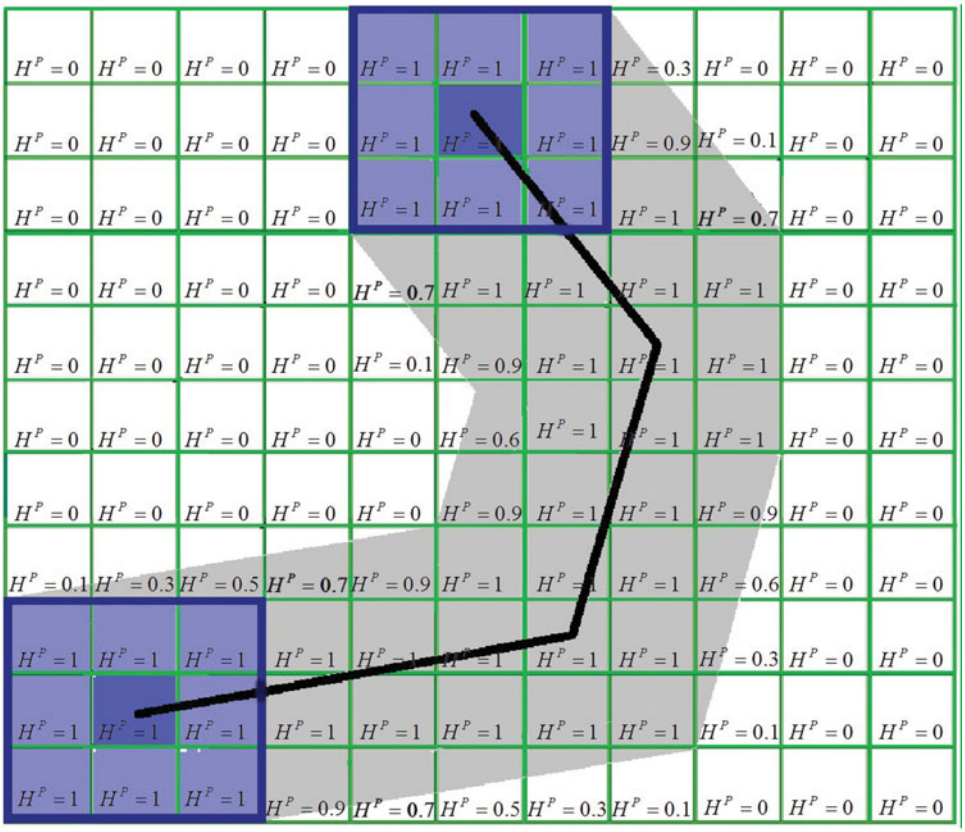

Fig. 2. H^P for a sample path.

Let us assume that the UAV reaches the coordinate (x, y) at time t_v. If the current time is given by t_c, the priority weight of the cell corresponding to the coordinate (x, y) is denoted by

$$W_{(x,y)}(t_c, t_v) = \frac{t_c - t_v(x, y)}{t_c} \tag{1}$$

where $t_v(x, y)$ is the last visited time of cell (x, y).

As a special case, if a cell has not been visited yet, its priority weight is equal to one.

$$\begin{cases} H^P_{(x_i,y_j)} = 0 & \text{Cell has not been visited through path } P \\ 0 < H^P_{(x_i,y_j)} < 1 & \text{Cell has been visited partially through path } P \\ H^P_{(x_i,y_j)} = 1 & \text{Cell has been visited once or more through path } P \end{cases} \tag{2}$$

As shown in Fig. 2, when the UAV traverses a path P, each cell could be at three situations: visited, partially visited, and unvisited. H^P is a measure to indicate the portion of a cell which is visited. The coverage value corresponding to path P is defined by the summation of priority weights of the cells as follows:

$$f_c(P) = \sum_{\forall(x_i,y_j)} W_{(x_i,y_j)} H^P_{(x_i,y_j)} \tag{3}$$

The function f_c is considered as the main objective function, in which the optimal path maximizes the function while satisfying the length and smoothness constraints, simultaneously.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

2.2. Length

Path P is assumed as a concatenated line with n junctions and n–1 segments (Fig. 3). The length of P can be calculated as follows:

$$f_l(P) = \sum_{i=1}^{n-1} d_i \tag{4}$$

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{5}$$

where (x_i, y_i) is the coordinate of a junction and d_i is the length of the segment between two consecutive junctions, i and i+1.

Let define the maximum hand over length between the UAV and the swimmer as follows:

$$l_{\text{hover}} = v_{\text{UAV}} \times t_{\text{hover}} \tag{6}$$

where t_{hover} is the hand over time. Therefore, for any feasible path P, we need to satisfy the constraint $f_l(P) \leq l_{\text{hover}}$.

2.3. Smoothness

Although the absolute value of the velocity of the UAV is considered to be fixed, nevertheless, in the sharp turns, the direction of velocity changes significantly. Therefore, the UAV needs to change the acceleration vector accordingly which may result in applying a big force to the UAV which is not desired. To avoid this situation, the smoothness of the path is an essential factor that needs to be taken into account in the path planning. The smoothness function is defined as follows:

$$f_s(P) = \pi - \min_{i=2}^{n-1}(\theta_i) \tag{7}$$

where $\theta_i \in [0, \pi]$ is the angle between two line segments which is shown in Fig. 3. In fact, by minimizing $f_s(P)$, we have a smoother path. Therefore, we put the constraint $f_s(P) \leq \delta$ to reach to a desired smoothness of the path.

To show that minimizing the smoothness function $f_s(P)$ results in minimizing the acceleration at turning points, consider the velocity vectors shown in Fig. 3. Hence, we can write the following:

$$\begin{aligned} |a_i| &= \frac{|\Delta v_i|}{\Delta t} = \frac{|v_i - v_{i-1}|}{\Delta t} = \frac{\sqrt{|V_{\text{UAV}}|^2 + |V_{\text{UAV}}|^2 + 2|V_{\text{UAV}}|^2 \cos(\theta_i)}}{\Delta t} \\ &= \frac{|V_{\text{UAV}}|\sqrt{2 + 2(2\cos^2(\theta_i/2) - 1)}}{\Delta t} = \frac{2\,|V_{\text{UAV}}|\cos\theta_i/2}{\Delta t} \quad \forall\, 0 \leq \theta_i \leq \pi \end{aligned} \tag{8}$$

From the other side, minimizing the smoothness function $f_s(P)$ is equivalent to maximizing $\min_i \theta_i$ such that $f_s(P) = \pi - \min_i \theta_i \leq \delta$ or $\pi - \delta \leq \min_i \theta_i \leq \pi$. Therefore, we have

$$0 \leq \cos(\min_i \frac{\theta_i}{2}) \leq \sin(\delta/2) \tag{9}$$

Therefore, the acceleration $|a_i|$ is minimized by minimizing $f_s(P)$.

In what follows, the evolutionary, classical, and the graph-based methods are implemented to solve the CBPP problem.

3. Genetic Algorithm

GAs have been fairly successful in solving discrete, non-linear, and ill-behaved optimization problems. In our case, the optimization criteria: *Coverage, Length,* and *Smoothness* are some non-linear and

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

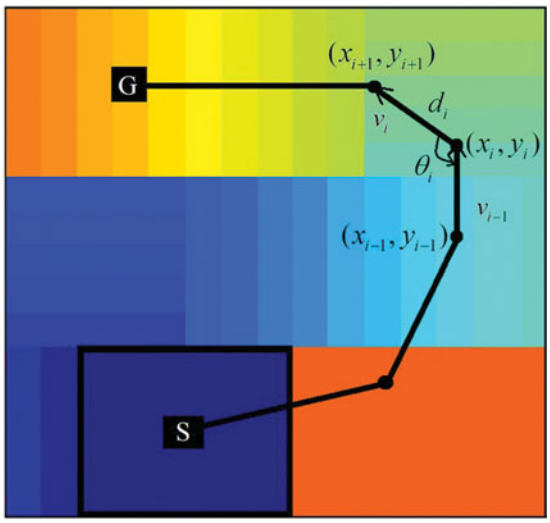

Fig. 3. Simple path with elements θ_i, x_i, y_i, and d_i.

non-differentiable functions of the path defined in a discretized environment. Therefore, the use of GA seems to be reasonable for solving such a complex optimization problem.

GA is initialized by a set of random population. The population is evaluated by the fitness function. In each generation, the higher reproduction probabilities are assigned to the individuals with higher fitness. This procedure is repeated until a given termination condition (number of iterations or improvement of solutions) is met.

Traditionally, GAs use binary strings as a chromosome and two basic genetic operators *Crossover* and *Mutation* to produce offsprings.[36] The grid-based representation is a trick to simplify genetic encoding, calculation of distance, and addressing the path planning process. In this paper, relying on the grid-based representation, a new encoding method is introduced namely *ABE*, which is discussed in this section.

Another encoding method in grid-based representation is the *NBE*. *NBE* is also utilized to solve CBPP in Section 4 and is compared to *ABE* in Section 7. Each encoding mechanism needs some specific operators. We proposed seven new *ABE* operators in Section 3.2 and reviewed seven *NBE* operators (Section 4). The objective functions are adopted to build up an appropriate fitness function in Section 3.3.

In a few first generations, GA is in the phase of random walk, and then step by step it continues until the population converges to a near optimal set. Therefore, operators at the first and at the final generations have different performance. Adaptive operator selection (AOS)[37] is a performance-based operator selection method, which helps GA to select operators optimally at each generation (Section 3.4).

3.1. Chromosome encoding and initialization

Representation of chromosomes is a critical aspect in GAs. A chromosome is usually represented by a string of the same elements. In our model, we construct a chromosome as a sequence of actions and call it ABE.

Each cell is connected to its four neighboring cells: "*Up*," "*Down*," "*Left*," and "*Right*" cells as shown in Fig. 4(a). In each cell, one of those actions is chosen by the UAV to go along a path.

Consider a path started from S (x_s, y_s) and ended at G (x_g, y_g) as shown in Fig. 4(b). For this sample path, the chromosome is represented by a string that contains first letter of actions: "**UUUUUURRUUUULLLL**."

The shortest distance, between start and goal point is defined as

$$d_{\min} = |d_x| + |d_y| \tag{10}$$

where $d_x = x_g - x_s$, $d_y = y_g - y_s$.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

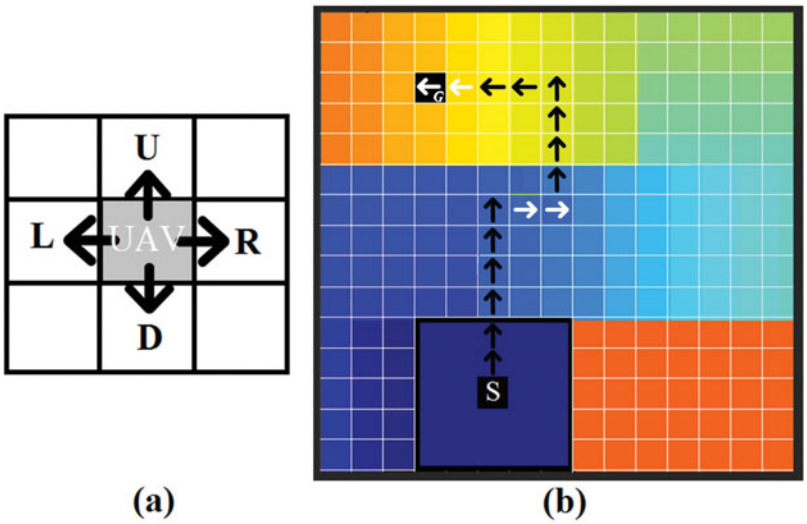

Fig. 4. Sample path in *ABE*. $(x_s, y_s) = (6,3)$ and $(x_g, y_g) = (4,13)$. White arrows are opposite actions.

Shortest path: The path has exactly $|d_x|$ and $|d_y|$ horizontal and vertical moves, respectively. Therefore, in this case, the length of the path is equal to the distance, i.e., $l = d_{\min}$.

Opposite actions: The path may contain opposite actions of types: *U/D*[(1)] or *R/L*.[(2)] If a path has opposite actions, its length is greater than the length of the shortest path, i.e., $l > d_{\min}$. For example, in Fig. 3(b), $d_x = -2$, $d_y = 10$, this path contains two *R/L* opposite actions.

Random organize path: A random organize path is constructed by $|d_x|$ number(s) of *R* or *L* actions, $|d_y|$ number(s) of *U* or *D* actions and random number(s) of opposite actions *R/L* or *U/D*.

The initial population is a set of random organize paths with n_{pop} numbers.

3.2. Genetic operators

Input and output of the *ABE* operators are *organized paths.* Such operators may change the number and type of *opposite actions* and may change the sequence of actions.

Based on this idea, we introduce seven operators as shown in Fig. 5.

(a) *Reversion:* Randomly selects two actions and reverses sequence of actions that are between them.
(b) *Inversion:* Randomly selects two actions and exchanges them.
(c) *Crossover:* If two paths have an intersection, exchanges the parts after intersecting point.
(d) *Add U/D:* Adds a *U/D* pair randomly placed in the path sequence.
(e) *Add R/L:* Adds an *R/L* pair randomly placed in the path sequence.
(f) *Improvement:* Adds a pair of opposite action (*U/D* or *R/L*) randomly placed in the path sequence in which, the offspring have a better fitness value.
(g) *Deletion:* Deletes a pair of *U/D* or *R/L* to increase the fitness.

3.3. Evaluation

MOO is an approach to the multi-criteria decision making.[38,39] One of the most popular MOO methods is $\epsilon-$*constraint*. This method reformulates the problem by keeping only one objective function and constraining the rest of the measures. To exploit this method in our problem, we keep *Coverage* as the main objective function and consider *Length* and *Smoothness* measures as the optimization

[(1)]Up/Down.
[(2)]Right/Left.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

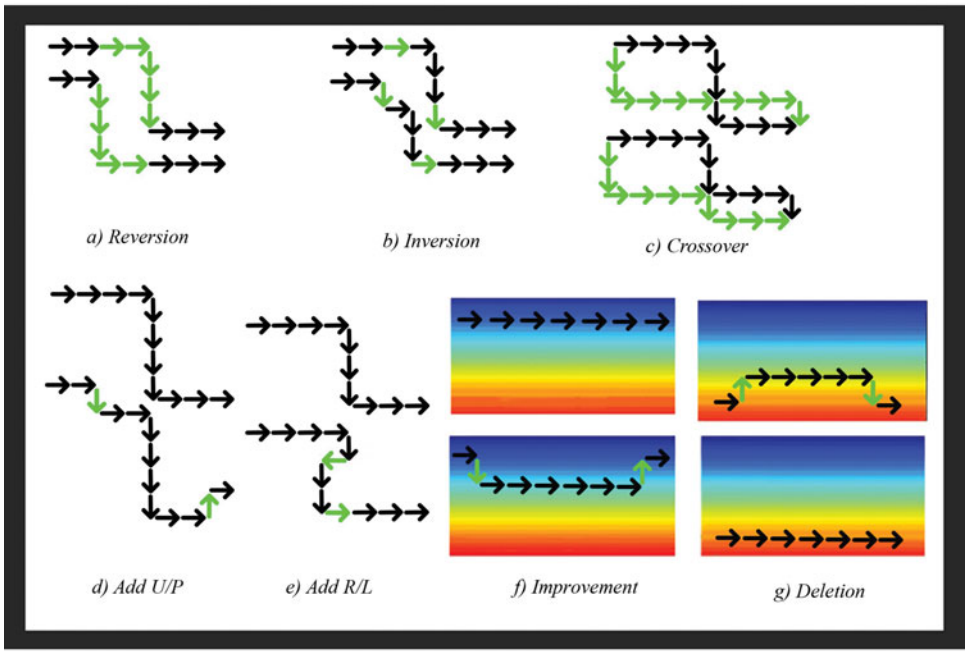

Fig. 5. *ABE* operators.

constraints. Therefore, the optimization problem is defined as follows:

$$\begin{cases} \max : f_{\text{coverage}}(P) \\ \text{subject to} : f_l(P) < \varepsilon_l \\ \text{subject to} : f_s(P) < \varepsilon_s \end{cases} \tag{11}$$

where ε_l and ε_s are upper bounds of *Length* and *Smoothness* constraints, respectively.

Based on this, there are two path types, i.e., feasible and infeasible. A *Feasible path* (P^f) satisfies all the constraints and an *infeasible path* (P^i) violates at least one of them.

We impose two kinds of penalty factors in the fitness function to distinguish between feasible (P^f) and infeasible (P^i) paths as follows:

$$\begin{cases} F(P^f) = f_{\text{coverage}}(P^f) \\ F(P^i) = f_{\text{coverage}}(P^i) - C_l(f_l(P^i) - \varepsilon_l) - C_s(f_s(P^i) - \varepsilon_s) \end{cases} \tag{12}$$

where C_l and C_s are the penalty factors. Therefore, considering $F(g)$ as the value of the fitness function at gth generation, the generation g is evaluated as follows:

$$f_R = \frac{1}{g_{\max}} \sum\nolimits_{g=1}^{g_{\max}} \frac{F(g)}{F(g_{\max})} \tag{13}$$

3.4. Adaptive operator selection

Multi-operator genetic algorithm (MO-GA), which is used in this research, takes the advantage of different crossover and mutation operators. In the case of MO-GA, it is well-posed to use an adaptive operation selection approach to select the operators optimally in each generation.

AOS has been used in MO-GAs. In AOS, the optimal probability of applying an operator is learned by some adaptive rules. This method examines the impact of using different operators by a credit assignment mechanism to prepare a reward for each operator. Operator selection parameters are updated at each generation by using this reward. In this method, the difference between fitness of offspring and parent is defined as a credit.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

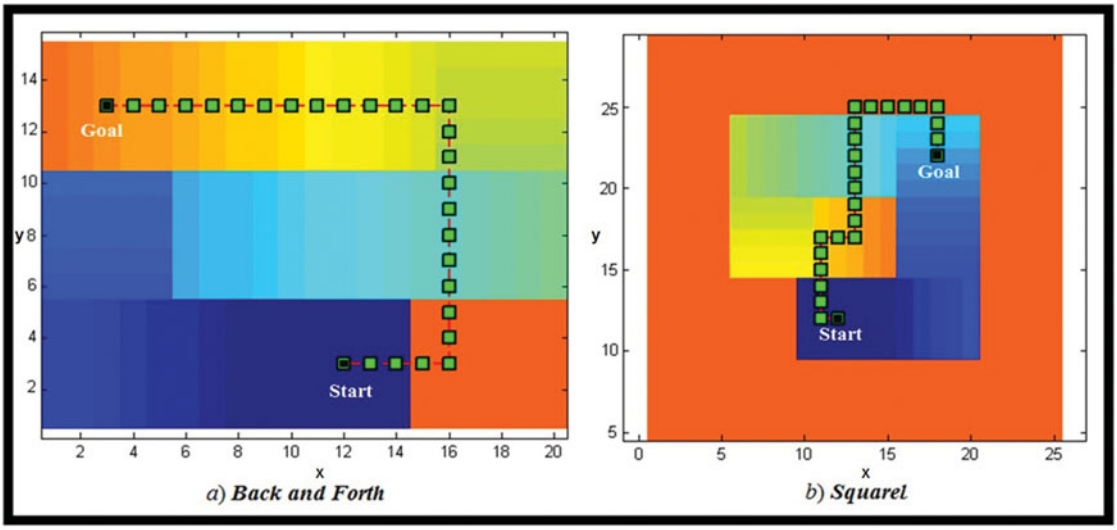

Fig. 6. Sample ABE path in *Back and Forth* ($\varepsilon_l = 28\varepsilon_s = \pi/2$) and *Squarel* ($\varepsilon_l = 28\varepsilon_s = \pi/2$) .

Based on the credit assigned to an operator, the probability of selection for each operator is adjusted based on the Adaptive Pursuit[40] as follows:

$$\hat{q}_{i,t+1} = (1-\alpha)\hat{q}_{i,t} + \alpha r_t \tag{14}$$

$$\begin{cases} i^* = \text{argmax}\left(\hat{q}_{i,t}, i = 1, ..., K\right) \\ s_{i^*,t+1} = s_{i^*,t} + \beta\left(1 - (K-1)p_{\min} - s_{i^*,t}\right) \\ s_{i,t+1} = s_{i,t} + \beta\left(p_{\min} - s_{i,t}\right) \end{cases} \tag{15}$$

where $\hat{q}_{i,t}$ is the estimate of ith operator reward at step t, α is the adaptation rate, i.e., $0 < \alpha \leq 1$, and r_t is the instant reward. K is the count of operators and i^* is argument of an operator that has maximal reward estimate at step t that we call it winner operator. $S_{i,t}$ is the operator probability of ith operator at step t, also $S_{i^*,t}$ is operator probability of winner operator at step t, β is the learning rate, i.e., $0 < \beta \leq 1$, and $p_{\min}$ is the minimum probability to select an operator, i.e., $0 < p_{\min} < 1/K$. For example, Fig. 6 shows the snapshot of *ABE* in two coverage patterns.

4. Node-Based Genetic Algorithm

NBE is a well-known approach in grid-based representation.[20] This method attaches a number to each grid and constructs a chromosome using a string of these numbers. As shown in Fig. 7, a path is a concatenated line started from S toward G. A sequence of the nodes from (v_1) to (v_n) constructs a chromosome. In this chromosome, v_1 and v_n are fixed and the other nodes are variable. For example, the sample path in Fig. 7 is represented by a chromosome shown as "**186-176-118-43-35**."

ABE enforces the UAV to move through four sides of each cell, but *NBE* does not have this limitation, therefore, grid representation helps it to find the nodes and picks them to create a path. The population is initialized with chromosomes which have a random number of intermediate nodes. We use seven types of genetic operators, i.e., *Crossover*, *Mutation*, *Deletion*, *Repair_line*, *Improvement*, *Smooth*, and *Swap*,[20,25,41] which are shown in Fig. 8. The operators change the intermediate nodes of a chromosome both to explore the environment and to reproduce more effective chromosomes. For example, Fig. 9 shows snapshots of *NBE* in the two coverage patterns shown in Fig. 6.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

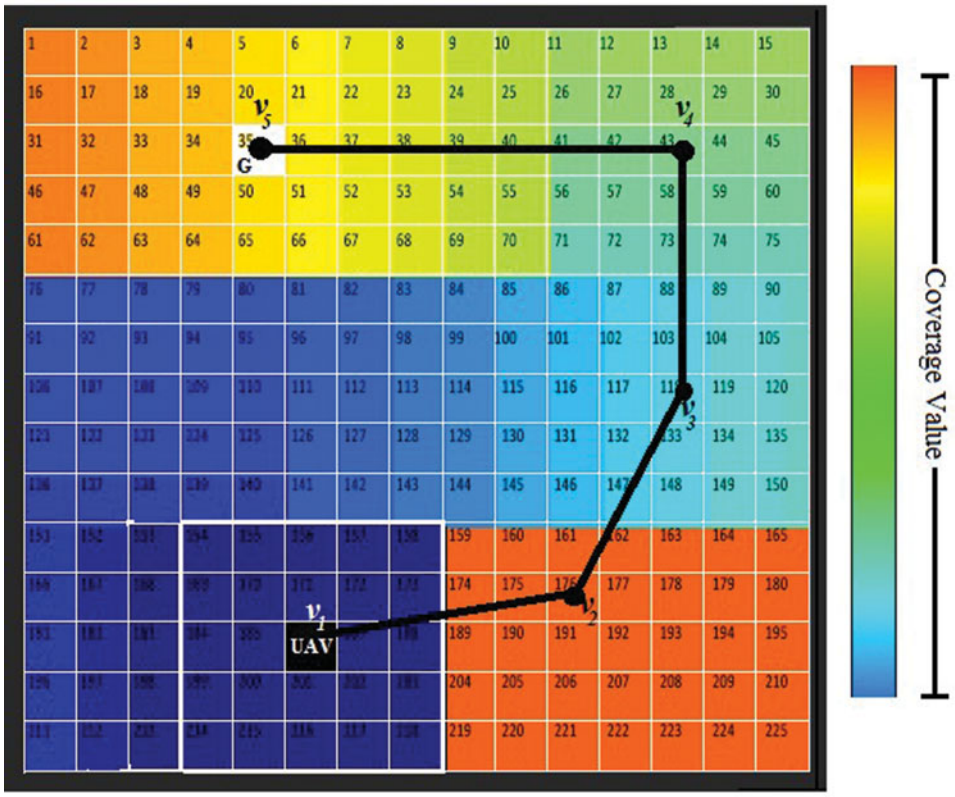

Fig. 7. A sample path in the orderly numbered NBE.

5. Artificial Potential Field Method

In the APF approach, it is assumed that the UAV moves under an artificial force field. The artificial field includes two kinds of the forces. The first one that we call it *priority weight force* is designed to maximize the desired objective function. The second force, which is called toward the goal force, guarantees that the UAV finally end up to the goal point.

5.1. Priority weight force

Naturally, a higher priority needs to be assigned to the areas those have been visited lately. So, we define the *priority weight force* as follows:

$$f_{pw}(x_u, y_u) = \sum_{(x_p, y_p) \in N} k_p f(x_p, y_p) \tag{16}$$

where

$$f(x_p, y_p) = \frac{W_{(x_p, y_p)}}{|d_{pu}|} \frac{\vec{d}_{pu}}{|d_{pu}|} \tag{17}$$

and

$$\vec{d}_{pu} = (x_p - x_u)\vec{i} + (y_p - y_u)\vec{j} \tag{18}$$

(x_p, y_p) represents the coordinates of a grid cell, $W_{(x_p, y_p)}$ is its *priority weight*, (x_u, y_u) shows the coordinates of the location of the UAV, and N is a set of neighboring grids of (x_u, y_u). $\vec{i}$, $\vec{j}$ are the unit vectors pointing to $+x$ and $+y$ directions, respectively, and k_p is an indicator value to select the greater force as below:

For two grid cells, p and q with coordinates (x_p, y_p) and (x_q, y_q) and indicators k_p and k_q: if $\angle \vec{d}_{pq} = \pm\pi$ and $|f(x_p, y_p)| \geq |f(x_q, y_q)| \rightarrow k_p = 1, k_q = 0$. $\vec{d}_{pq} = (x_p - x_q)\vec{i} + (y_p - y_q)\vec{j}$. This would

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

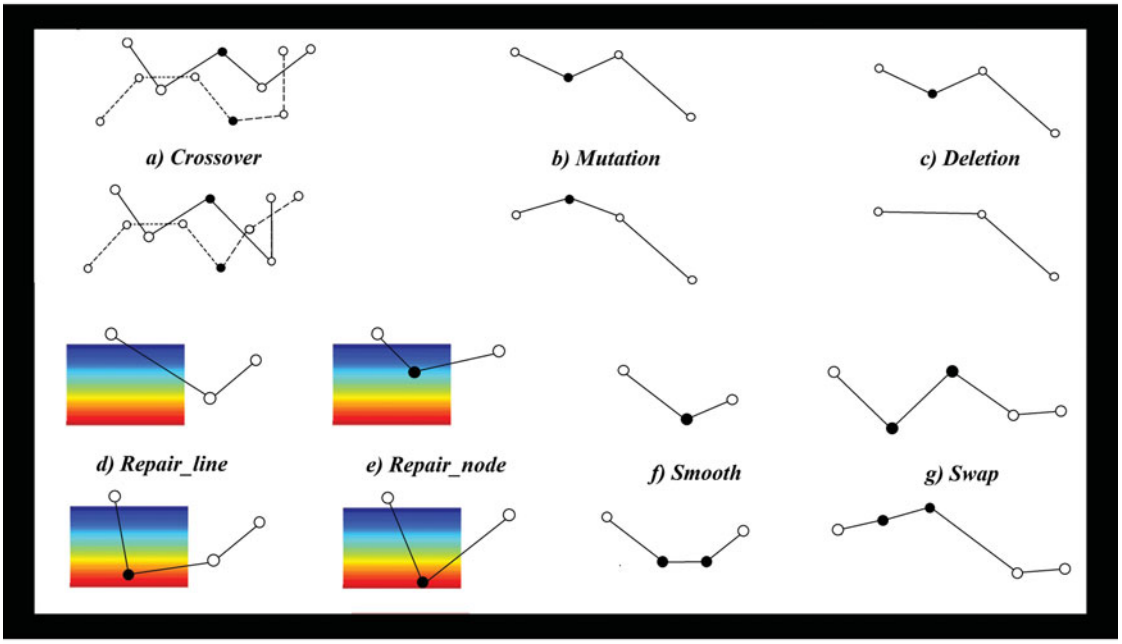

Fig. 8. *NBE* operators.

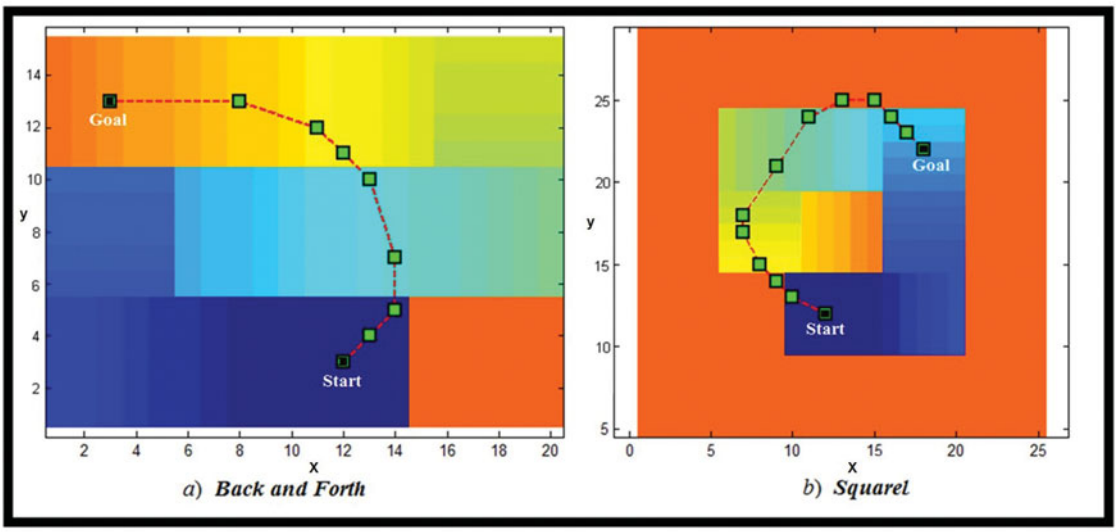

Fig. 9. Sample NBE path in *Back and Forth* ($\varepsilon_l = 28.\ \varepsilon_s = \pi/2$) and *Squarel* ($\varepsilon_l = 28.\ \varepsilon_s = \pi/2$) .

help the approach to avoid local minima, because the sum of two forces with opposite directions is less than one of them. For example, in Fig. 10 $\angle \vec{d}_{pq} = \pi$ and $|f(x_p, y_p)| > |f(x_q, y_q)|$, so $k_p = 1, k_q = 0$.

Since the range of *priority weight force* $f_{pw}(x_p, y_p)$ changes by the size of the grid neighbors N, we normalize this force to become comparable with the *toward the goal force*

$$F_{pw}(x_u, y_u) = \frac{f_{pw}(x_u, y_u)}{f_{pw}^{\max}} \tag{19}$$

where $f_{pw}^{\max}$ is the maximum value for $f_{pw}(x_u, y_u)$ at the grid neighbors N. Thus, we have $0 \leq |F_{pw}(x_u, y_u)| < 1$.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

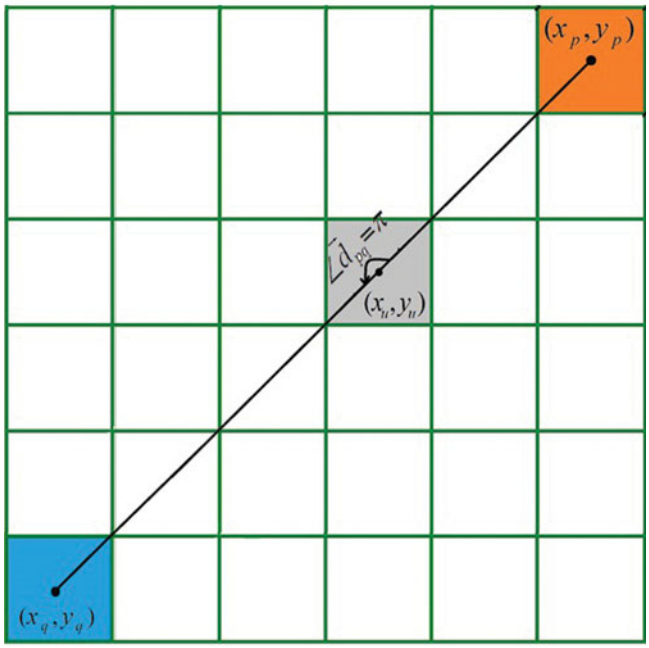

Fig. 10. Priority weight direction example: $|f(x_p, y_p)| = 1/2\sqrt{2}$, $|f(x_q, y_q)| = 0.1/3\sqrt{2}$.

5.2. Toward the goal

The UAV may not reach the goal by impact of *Priority weight force*. Therefore, we need an attraction force toward the goal to guarantee the path ends up at G. We set a uniform force toward the goal at all of the environment as follows:

$$F_{tg}(x_u, y_u) = \frac{\vec{d}_{gu}}{|d_{gu}|} \tag{20}$$

$$\vec{d}_{gu} = (x_g - x_u)\vec{i} + (y_g - y_u)\vec{j} \tag{21}$$

where (x_g, y_g) are the coordinates of the goal point and (x_u, y_u) are coordinates of the location of the UAV. $\vec{i}$ and $\vec{j}$ are the unit vectors at $+x$ and $+y$, respectively, and $|d_{gu}|$ is the magnitude of vector $\vec{d}_{gu}$.

The total force is determined by the weighted combination of the two forces as below:

$$F(x_u, y_u) = (2 - C)F_{tg}(x_u, y_u) + CF_{pw}(x_u, y_u) \tag{22}$$

where C is the coverage parameter, i.e., $(0 \leq C < 1)$.

The total force has two parts. The first part is the effect of the *toward the goal force*, and the second is effect of the *priority weight force*. If at each point (x_u, y_u), the magnitude of $(2 - C)F_{tg}(x_u, y_u)$ is greater than the magnitude of $CF_{pw}(x_u, y_u)$, the distance between the UAV and G point will be decreased step by step, and finally the path ends at the G point. This is proved as follows:

Proof:

$$\begin{aligned} &|F_{pw}(x_u, y_u)| < |F_{tg}(x_u, y_u)| = 1 \\ &C|F_{pw}(x_u, y_u)| < (2 - C)|F_{tg}(x_u, y_u)| \\ &|CF_{pw}(x_u, y_u)| < |(2 - C)F_{tg}(x_u, y_u)| \end{aligned}$$

By increasing the coverage parameter C, the path covers areas with higher priority weights. For example, Fig. 11 shows the snapshot of *APF* in the two coverage patterns used in the previous two methods.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

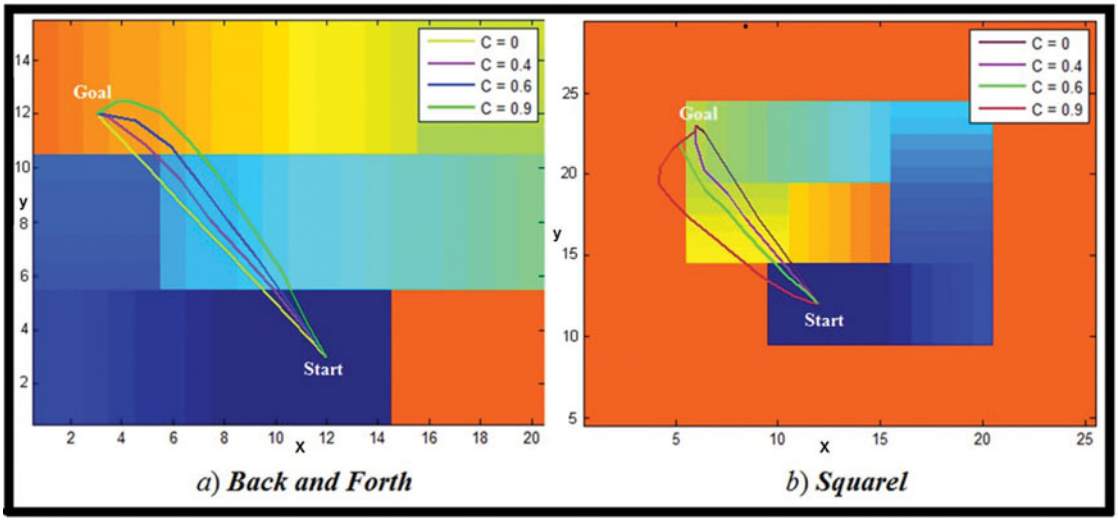

Fig. 11. Sample *APF* path in *Back and Forth* and *Squarel.*

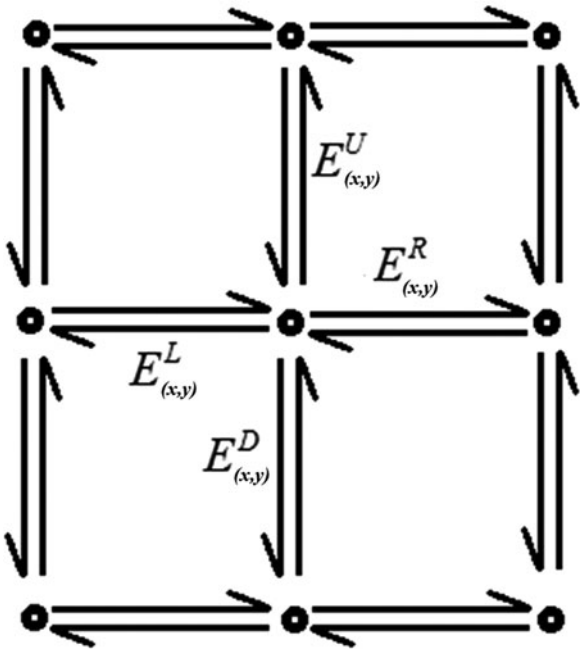

Fig. 12. Sample graph.

6. Dijkstra

Shortest path problem is a well-known topic in the graph theory. E. W. Dijkstra in 1959 proposed an algorithm to solve the shortest path called *Dijkstra*'s *Algorithm*.[16] This method is modified to find a suboptimal path in a graph with non-negative weights. This method can be used in the path planning problems as well. There are three critical elements in a graph, i.e., "Vertex," "Edge," and "Weight."

In our grid environment, each cell is connected to its four neighbor cells, i.e., "Up," "Down," "Left," and "Right" cells. Each cell is a vertex and connection between cells is assumed as the edge (Fig. 12).

The weight values are defined to take into account the coverage value in the path planning problem. Figure 13 shows covered cells by move at four sides. The value of coverage toward each of four

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

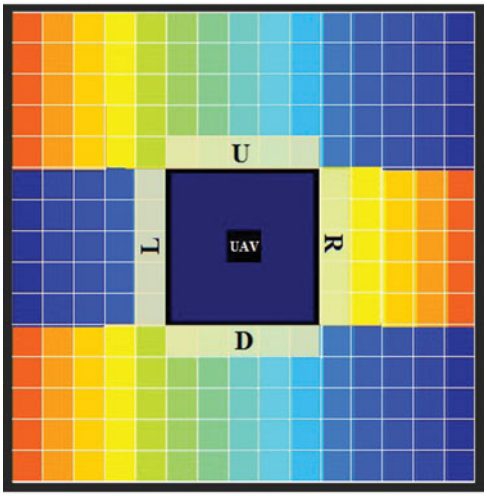

Fig. 13. Covered cells by *Right*, *Left*, *Up*, and *Down* movement.

directions is considered to be the sum of the weights corresponding to the covered cells as follows:

$$\begin{cases} W^R_{(x,y)} = \sum_{i=y-m}^{y+m} W_{(x+m+1,i)} \\ W^L_{(x,y)} = \sum_{i=y-m}^{y+m} W_{(x-m-1,i)} \\ W^U_{(x,y)} = \sum_{i=x-m}^{x+m} W_{(i,y+m+1)} \\ W^D_{(x,y)} = \sum_{i=x-m}^{x+m} W_{(i,y-m-1)} \end{cases} \tag{23}$$

where $W_{(x,y)}$ is the priority weight of cell (x, y). m is defined as $m = (l_w - 1)/2$, where l_w is the side length of the field of view (Fig. 1(a)). $W^R_{(x,y)}$ is sum of the priority weights of the covered cells by *Right* move, i.e., $0 \leq W^R_{(x,y)} \leq l_w$.

The weights of the four edges (Fig. 12) are defined as below:

$$\begin{cases} E^R_{(x,y)} = K - W^R_{(x,y)} \\ E^L_{(x,y)} = K - W^L_{(x,y)} \\ E^U_{(x,y)} = K - W^U_{(x,y)} \\ E^D_{(x,y)} = K - W^D_{(x,y)} \end{cases} \tag{24}$$

where the constant K is an upper bound for coverage value, i.e., $K > l_w$. By this definition, the shortest path in the graph *G(V, E)* is equivalent to the higher coverage value in the path planning problem. So, *Dijkstra's algorithm* finds such a path with higher coverage value. For example, Fig. 14 shows the snapshot of *Dijkstra's algorithm* in the two coverage patterns used in the previous three methods.

7. Experiment

In this experiment, the proposed methods: MO-GA (*ABE* and *NBE*), *APF,* and *Dijkstra's algorithm* are tested in different coverage patterns – *Back and Forth* and *Squarel* – and their performances are compared via simulation results. The aim is to find the optimal path from a start point *S* to a different arbitrary goal points *G*. To ensure the robustness of the results to variations of the tuning parameters, we have implemented the methods along with different values for parameters and used t-test method to show the statistical significance of the results.

Let consider f_l and f_s as the cost functions and f_c as the profit function of the methods as introduced before. Also, f_R is added to compare generation results in GA.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

Table I. Parameters and evaluation functions.

Method	Variable parameters	Fixed parameters	Cost functions	Profit functions
ABE	ε_s, N_l	$p_{\min} = 0.05$, pop $= 50$ $g_{\max} = 100, \alpha = 0.9$ $\beta = 0.9, C_s = C_l = 1000$	f_l, f_s	f_c, f_R
NBE	N_l		f_l, f_s	f_c, f_R
APF	C	–	f_l, f_s	f_c
Dijkstra	–	–	f_l, f_s	f_c

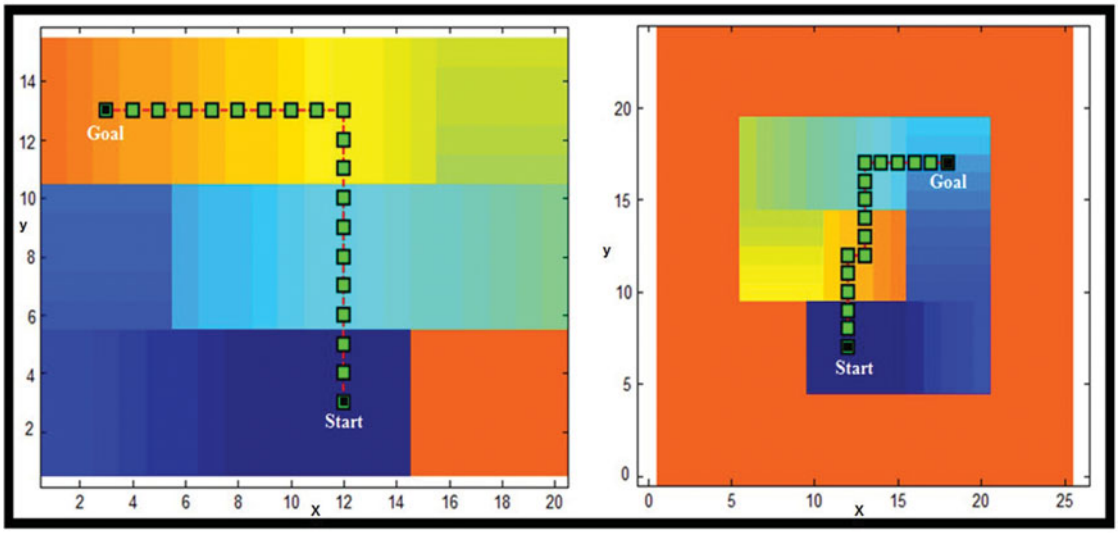

Fig. 14. Sample *Dijkstra's algorithm* path in *Back and Forth* and *Squarel*.

Since only in GA, there is a mechanism to set upper bounds on the cost functions, first we solve the problem by *APF* and *Dijkstra's algorithm*, and then we obtained the corresponding cost of the resulted paths. Finally, we set those costs as the upper bounds on the cost functions in the GA methods.

We divide the parameters into two types: fixed and variable parameters. Fixed parameters may be changed based on the specification of environment, so those are set to appropriate values by trial and error and have same values in all experiments. However, the variable parameters are set to appropriate values at each experiment.

ε_l is defined as follows:

$$\varepsilon_l = (1 + N_l)d_{\min} \tag{25}$$

$$d_{\min} = \begin{cases} |x_g - x_s| + |y_g - y_s| & \text{for } ABE \\ \sqrt{(x_g - x_s)^2 + (y_g - y_s)^2} & \text{for } NBE \end{cases} \tag{26}$$

where (x_s, y_s) and (x_g, y_g) are the coordinates of start and goal point, respectively. $N_l \geq 0$ determines the upper bound of *Length* with respect to the shortest path. Fixed and variable parameters are presented in Table I.

7.1. Results

First, we slip the variable parameters to get different cost and profit results. Increasing the parameter N_l produces longer paths with a wider covered area. Also, finding an optimal solution among the longer paths needs more computation. As we expected, Fig. 15 shows higher N_l, produces higher f_l and f_c and lower f_R. The results for NBE and ABE methods are shown in Figs. 16 and 17. Increasing N_l causes same effects in both NBE and ABE. Increasing ε_s provides more relaxation on the angels. This means that the path solutions with sharper turns are likely to be selected. It should be mentioned

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

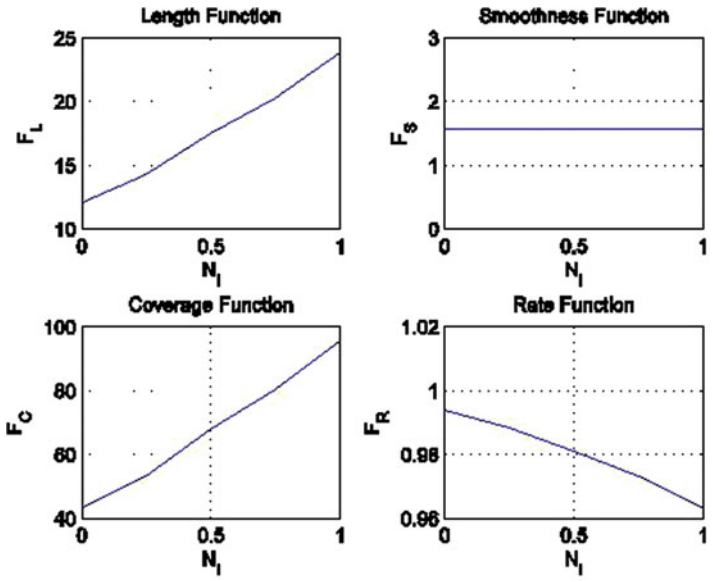

Fig. 15. ABE results in two cost and two profit functions.

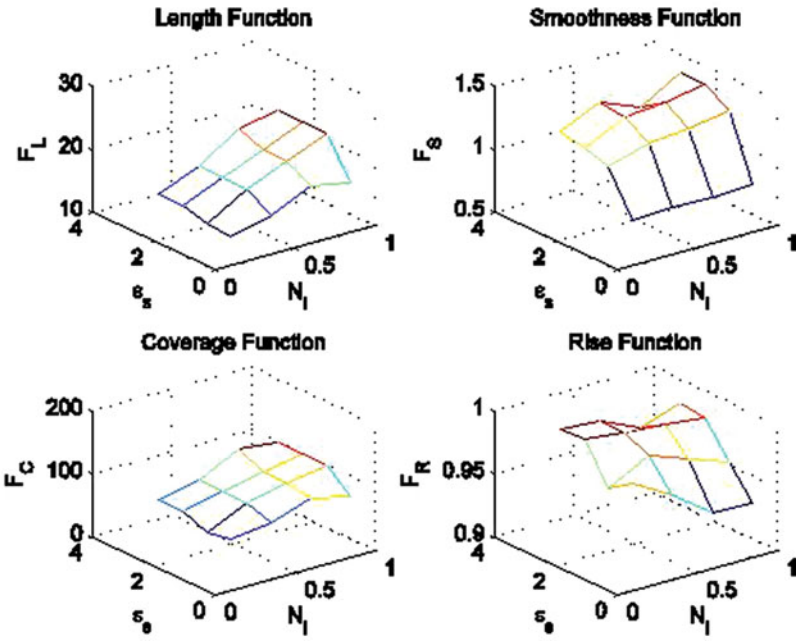

Fig. 16. NBE results in two cost and two profit functions.

that increasing $\varepsilon_s > \pi/2$ had no effect on the results, so it is deduced that the optimal solution does not have any angle with $\theta_i < \pi/2$. Coverage factor C is a single design parameter in *APF*. Increasing this parameter builds stronger effect for priority weight forces. Therefore, the paths will be longer with sharper turns and with wider covered area. It should be mentioned, this method does not have any generation in its construction, so we assumed the same results for all hypothetical generation by setting $f_R = 1$.

7.2. Comparison

Figure 18 shows average ranges of the cost and the profit functions. $F_L^{\min}$ and $F_C^{\min}$ are average of *Length* and *Coverage* of shortest paths, respectively. *ABE* and *NBE* from the GA methods have the

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

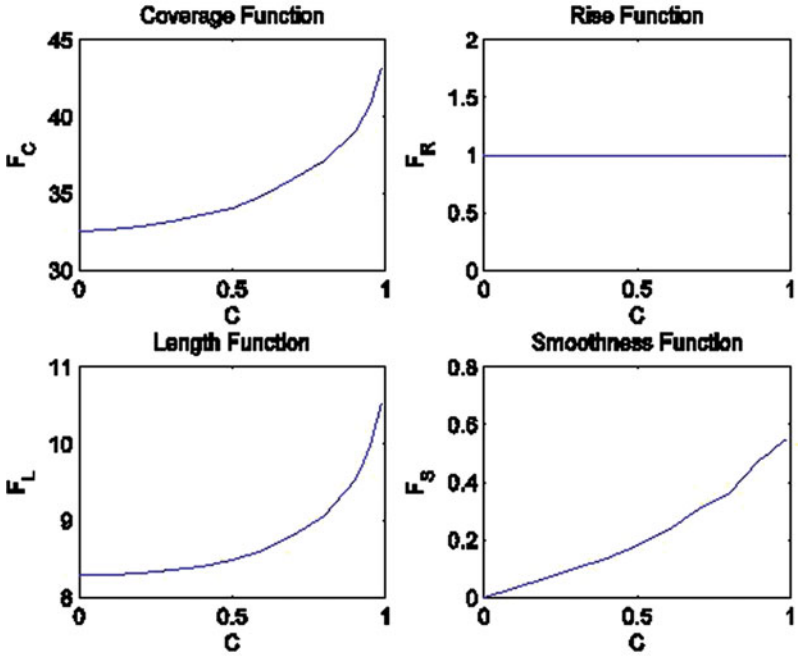

Fig. 17. APF results in two cost and two profit functions.

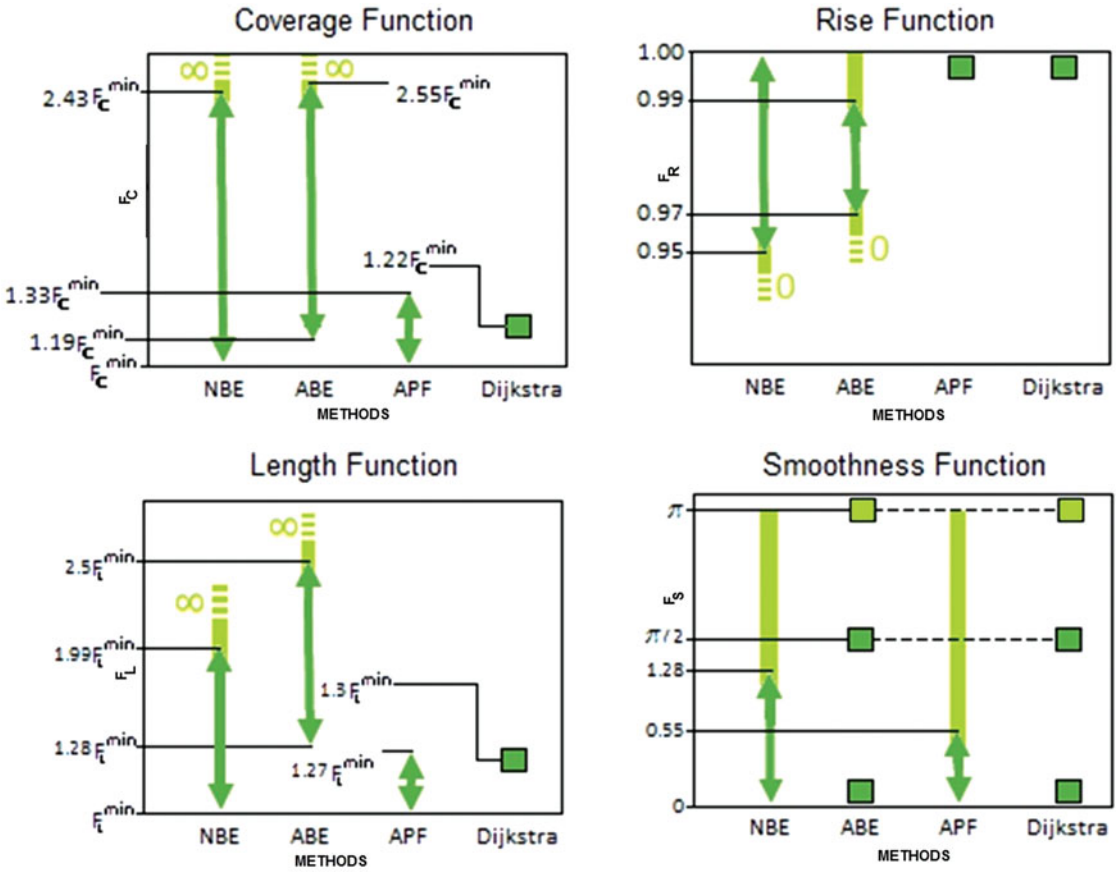

Fig. 18. Bounds of cost and profit functions in four methods. Light green: Range of functions. Strong green: Range of generated data by slipping the parameters. Squares show separate points.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

Table II. p-values from the t-test and comparison between ABE and NBE at five points.

P-value	Point 1	Point 2	Point 3	Point 4	Point 5
p_{F_C}	1E−2	1E−2	1E−2	2E−2	3E−2
p_{F_R}	2E−4	6E−3	9E−6	2E−7	1E−7

p_{F_C} and p_{F_R} are p-values for F_C and F_R results at five points as shown in Fig. 18, respectively.

Table III. Comparison between Dijkstra's algorithm with ABE and NBE.

Method	f_C	f_R	p_{fc}	p_{f_R}
Dijkstra	39.73	1.0	–	–
ABE	38.67	0.99	0.75	3E-17
NBE	47.84	0.99	0.03	8E-26

p_{f_C} and p_{f_R} are p-values for f_C and f_R results at five points, respectively.

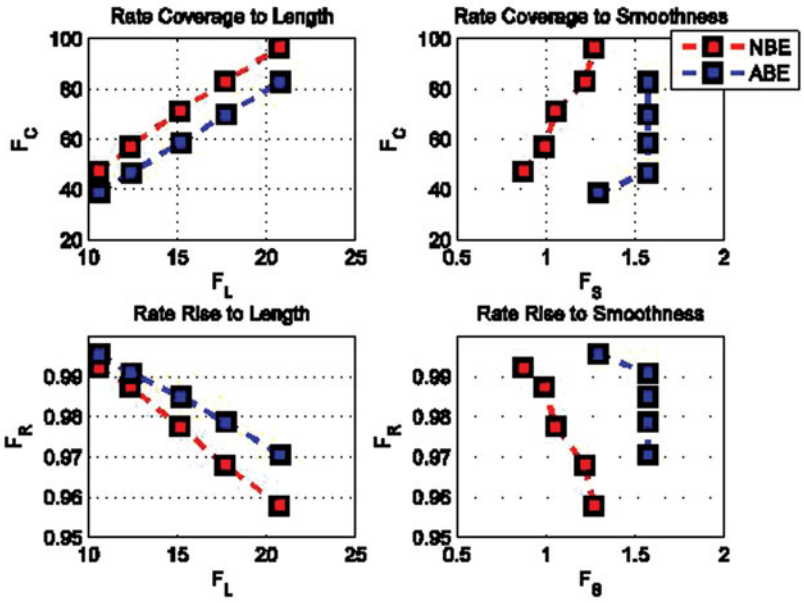

Fig. 19. Comparison between ABE and NBE.

same structure. To achieve results with equal costs, first we get *ABE* results and then set ε_l and ε_s in *NBE*.

As shown in Fig. 19, *Length* costs are equal in both methods, but *Smoothness* is better in *NBE*. At these costs, the *Coverage* profit in *NBE* gets better values for all the cases, and the *Rise* profit has better values in *ABE*.

All p-values from the t-test, as shown in Table II, satisfy standard constraint ($p < 0.05$) .

Dijkstra's algorithm has no parameter in its construction, so similar to the previous comparison, first, we get *Dijkstra's algorithm* results and then set its costs as upper bound for *ABE* and *NBE*.

In this case, the *Coverage* profit of *ABE* is very close to *Dijkstra's* and therefore, the p-values in the t-test could not satisfy standard constraint, nevertheless, *Dijkstra's algorithm* has greater mean value. *NBE* resulted greater values in the *Coverage*. The *Rise* function $f_R = 1$ in the *Dijkstra's algorithm* is better than others. The summary of comparative results is given in Table III.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press

8. Conclusion

The CBPP problem presented in this paper is suitable for applications such as monitoring swimmers. In such a scenario, a UAV starts from its location to an arbitrary goal location to visit areas that have not been visited recently while satisfying the smoothness and length constraints. The shortest path is not a good policy, because it cannot visit areas other than the one under the shortest path. Based on this idea, we defined two cost factors, i.e., *Smoothness* and *Length,* and one profit factor, i.e., *Coverage,* for each path. Thus, an optimal path has lower cost with higher profit.

Two evolutionary, i.e., ABE and NBE, and two classical, i.e., APF and Dijkstra's algorithm, methods were proposed, implemented, and compared. Evolutionary methods are off-line, and the result got after several generations, so we added *Rise* function to compare results with each other and with other classical methods. We introduced a new chromosome encoding, i.e., ABE, with seven genetic operators and proposed another chromosome encoding, i.e., NBE, with ϵ-constraint method and two penalty factors to combine cost and profit functions. In NBE, we can set exact upper bound for each cost, however, in ABE upper bound of *Smoothness* may be picked from three values, i.e., $\{0, \pi/4, \pi/2\}$.

We proposed two forces in APF and determined a parameter to balance between them. Dijkstra's algorithm has no parameter, but their edge weights are assigned, so that shortest path in graph is equivalent to better value in the *Coverage* function.

If we want to generate a path, close to the shortest path, on-line methods are helpful, since they generate paths with close to minimum travel time. On the other hand, to get better time-coverage performance, off-line methods should be used. Generally, the off-line methods better maximize *Coverage* profit and have wider range in both cost and profit functions. ABE has higher *Rise* profit than NBE since it has limited search space. However, its *Rise* profit is lower than other on-line methods.

In the future work, the on-line methods may be improved to have more parameters to result better cost and profit functions ranges, compared to the proposed approached. Furthermore, the off-line methods will be evaluated based on the processing time limit. ABE can be implemented for a lot of path planning problems and more genetic operators may be introduced. In addition, if we consider the effect of new discoveries of the UAV during the tasks, we need to update the path accordingly, and therefore we need to take the advantage of on-line method in such a case.

Acknowledgments

This research was in part supported by a grant from the Institute for Research in Fundamental Sciences (IPM) (No. CS 1396-4-33).

References

1. D. Jia, M. Wermelinger, R. Diethelm, P. Krüsi and M. Hutter, "Coverage Path Planning for Legged Robots in Unknown Environments," *Proceedings of the 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Lausanne (2016) pp. 68–73.
2. A. Bircher *et al*., "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Auton. Robots* **40**(6), 1059–1078 (2016).
3. J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Rob. Res.* **10**, 628–649 (1991).
4. C. Chen, "Path planning in distorted configuration space," *Robotica* **35**(7), 1585–1597 (2017).
5. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.* **5**, 90–98 (1986).
6. Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *Proceedings of the IEEE International Conference on Robotics and Automation* (1991) pp. 1398–1404.
7. J. Ren, K. McIsaac and R. V. Patel, "Modified Newton's method applied to potential field-based navigation for mobile robots," *IEEE Trans. Robot.* **22**, 384–391 (2006).
8. B. Xiao, L. Yu, S. Li and R. Chen, "Research of escaping local minima strategy for artificial potential field," *J. Syst. Simul.* **19**, 4495–4503 (2007).
9. K. S. Al-Sultan and M. D. S. Aliyu, "A new potential field-based algorithm for path planning," *J. Intell. Robot. Syst.* **17**, 265–282 (1996).
10. J. Lee, Y. Nam, S. Hong and W. Cho, "New potential functions with random force algorithms using potential field method," *J. Intell. Robot. Syst.* **66**, 303–319 (2012).

11. Y. Zhu, T. Zhang and J. Song, "An Improved Wall Following Method for Escaping from Local Minimum in Artificial Potential Field Based Path Planning," *Proceedings of the 48th IEEE Conference on Decision and Control. Held Jointly with 28th Chinese Control Conference CDC/CCC* 2009 (2009) pp. 6017–6022.
12. R. Raja, A. Dutta and K. S. Venkatesh, "New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover," *Robot. Auton. Syst.* **72**, 295–306 (2015).
13. J. Simon and G. Martinovic, "Navigation of mobile robots using WSN's RSSI parameter and potential field method," *Acta Polytech. Hung.* **10**, 107–118 (2013).
14. L. Yin, Y. Yin and C. Lin, "A new potential field method for mobile robot path planning in the dynamic environments," *Asian J. Control* **11**, 214–225 (2009).
15. P. Vadakkepat, K. Chen Tan and W. Ming-Liang, "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning," *Proceedings of the 2000 IEEE Congress on Evolutionary Computation* (2000) pp. 256–263.
16. E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.* **1**, 269–271 (1959).
17. Z. Zhang and Z. Zhao, "A multiple mobile robots path planning algorithm based on A-star and Dijkstra algorithm," *Int. J. Smart Home* **8**, 75–86 (2014).
18. C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su and W. Li, "Path Planning of Automated Guided Vehicles Based on Improved A-Star Algorithm," *Proceedings of the IEEE International Conference on Information and Automation* (2015) pp. 2071–2076.
19. A. K. Pamosoaji and K. S. Hong, "A path-planning algorithm using vector potential functions in triangular regions," *IEEE Trans. Syst. Man, Cybern.* **43**, 832–842 (2013).
20. J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. Evol. Comput.* **1**, 18–28 (1997).
21. V. Roberge, M. Tarbouchi and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, ," *IEEE Trans. Ind. Inform.* **9**, 132–141 (2013).
22. J. Yen, "A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation," *IEEE Trans. Syst. Man, Cybern.* **25**, 971–978 (1995).
23. H. Qu, S. X. Yang, A. R. Willms and Z. Yi, "Real-time robot path planning based on a modified pulse-coupled neural network model," *IEEE Trans. Neural Netw.* **20**, 1724–1739 (2009).
24. R. Moharam and E. Morsy, "Genetic algorithms to balanced tree structures in graphs," *Swarm Evol. Comput.* **32**, 132–136 (2016).
25. Y. Hu and S. X. Yang, "A Knowledge Based Genetic Algorithm for Path Planning of a Mobile Robot," *Proceedings of the IEEE International Conference on Robotics and Automation ICRA2004.* (2004) pp. 4350–4355.
26. D. Rathbun, S. Kragelund, A. Pongpunwattana and B. Capozzi, "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments," *Proceedings of the IEEE 21st Digital Avionics Systems Conference* (2002) vol. **2**, pp. 8D2-1-8D2-12.
27. Q. Li, W. Zhang, Y. Yin, Z. Wang and G. Liu, "An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots," *Proceedings of the 6th International Conference on IEEE Intelligent Systems Design and Applications ISDA2006* (2006) pp. 637–642.
28. I. Hasircioglu, H. R. Topcuoglu and M. Ermis, "3-D Path Planning for the Navigation of Unmanned Aerial Vehicles by Using Evolutionary Algorithms," *Proceedings of the 10th Annual Conference Genetic and Evolutionary Computation* (2008) pp. 1499–1506.
29. M. Ataei and A. Yousefi-Koma, "Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle," *Robot. Auton. Syst.* **67**, 23–32 (2015). doi:10.1016/j.robot.2014.10.007.
30. F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Comput.* **17**, 1283–1299 (2013).
31. H. Choset, E. Acar, A. Rizzi and J. Luntz, "Exact Cellular Decompositions in Terms of Critical Points of Morse Functions," *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA2000* (2000) pp. 2270–2277.
32. E. Semsch, M. Jakob, D. Pavlíček and M. Pěchouček, "Autonomous UAV Surveillance in Complex Urban Environments," *Proceedings of the IEEE/WIC/ACM International Jt. Conference, IET, Web Intelligence and Intelligent Agent Technologies WI-IAT2009* (2009) pp. 82–85.
33. W. H. Huang, "Optimal Line-Sweep-Based Decompositions for Coverage Algorithms," *Proceedings of the IEEE International Conference on Robotics and Automation ICRA2001* (2001) pp. 27–32.
34. P. DeLima and D. Pack, "Maximizing Search Coverage Using Future Path Projection for Cooperative Multiple UAVs with Limited Communication Ranges," **In**: *Optimization and Cooperative Control Strategies* (M. Hirsch, C. W. Commander, P. M. Pardalos and R. Murphey, eds.) (Springer, Berlin, Heidelberg, 2009) pp. 103–117.
35. C. Robin and S. Lacroix, "Multi-robot target detection and tracking: Taxonomy and survey," *Auton. Robots* 1–32 (2015).
36. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (MIT Press, Cambridge, MA, 1992).
37. L. Davis, "Adapting Operator Probabilities in Genetic Algorithms," *Proceedings of the 3rd International Conference on Genetics, Algorithms* (1989) pp. 61–69.
38. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16 (John Wiley & Sons, Hoboken, New Jersey, 2001).

39. A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.* **1**, 32–49 (2011).
40. S. M. Ahmadi, H. Kebriaei and H. Moradi, "Adaptive Operator Selection for Path Planning in Static Environments," *Proceedings of the IEEE SAI Intelligent Systems Conference IntelliSys2015* (2015) pp. 285–289.
41. R. Zhang, C. Zheng and P. Yan, "Route Planning for Unmanned Air Vehicles with Multiple Missions Using an Evolutionary Algorithm," *Proceedings of the 3rd International Conference on Natural Computation IEEE ICNC2007* (2007) pp. 23–28.

https://doi.org/10.1017/S0263574718000139 Published online by Cambridge University Press