Soft Comput
DOI 10.1007/s00500-016-2045-x

METHODOLOGIES AND APPLICATION

A novel path planning algorithm based on plant growth mechanism

Yaoming Zhou[1] · Yongchao Wang[1] · Xuzhi Chen[1] · Lei Zhang[2] · Kan Wu[3]

Abstract We propose a bio-inspired computing algorithm based on plant growth mechanism and describe its application in path planning in this paper. The basic rules of the algorithm include phototropism, negative geotropism, apical dominance, and branch in plant growth. The starting point of the algorithm is the seed germ (first bud) and the target point of the algorithm is the light source. The discretization of the plant growth process is used to realize computation in computer. The plant growth behavior in each iteration is assumed to be the same. The algorithm includes six steps: initialization, light intensity calculation, random branch, growth vector calculation, plant growth and path output. Several two-dimensional path planning problems are used to validate the algorithm. The test results show that the algorithm has good path planning ability and provides a novel path planning approach.

Keywords Path planning · Plant growth · Bud · Light intensity · Branch

Communicated by V. Loia.

✉ Kan Wu
wukan_ntu@163.com; WuKan@ntu.edu.sg

[1] School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China

[2] Hang Zhou Sysmagic Technology CO., LTD, Hangzhou 311121, China

[3] School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798, Singapore

1 Introduction

Bio-inspired computing algorithms are important parts of natural computing. They are inspired and developed by simulating the characteristics of living entities that the survival of living entities and the success of some species in a given environment rely on optimization and search to overcome the constraints imposing by the environment. It is well known that bio-inspired computing algorithms are capable of solving many complicated engineering problems whose solutions cannot effectively be obtained using classical (often, gradient-based) optimization algorithms. Bio-inspired computing algorithms have been widely used in path planning now. For example, path planning of a mobile robot or an unmanned aerial vehicle can be done by the ant colony optimization (Garcia et al. 2009; Tan et al. 2007; Duan et al. 2008; Chen et al. 2008), particle swarm optimization (Gong et al. 2011; Foo et al. 2009; Fu et al. 2012), artificial bee colony (abbreviated as ABC) (Xu et al. 2010; Bhattacharjee et al. 2011; Mernik et al. 2015; Li et al. 2014), artificial fish swarm (Peng et al. 2013; Ma and Lei 2010), differential evolution (abbreviated as DE) (Aydin and Temeltas 2004; Duan et al. 2010; Mo et al. 2012), firefly algorithm (Liu et al. 2012; Wang et al. 2012; Li et al. 2014; Liu et al. 2015), shuffled frog leaping algorithm (Hassanzadeh et al. 2010; Zhang et al. 2012; Ni et al. 2014; Alejandro et al. 2015), foraging algorithm (Jati et al. 2012; Liu et al. 2013; Liang et al. 2013; Hossain and Ferdous 2015) or artificial immune algorithm (Wang et al. 2007; Luh and Liu 2008; Deepak et al. 2012; Das et al. 2012). The path is planned by first establishing the objective function of path planning and then searching for the optimal solution through the bio-inspired computing algorithms.

The bio-inspired computing algorithms described above are set up through simulating the features of animals and have

Published online: 02 February 2016

obtained good results in path planning. As the living things in the bottom layer of nature, plants have more simple and efficient characteristics from the perspective of evolutionary philosophy. The path planning algorithm built through simulating the features of plants is expected to obtain better results. There are some bio-inspired computing algorithms constructed based on plants, such as plant growth simulation algorithm (Li et al. 2004; Li and Su 2007; Rao and Narasimham 2008), saplings growing up algorithm (Karc 2007), plant propagation algorithm (Salhi and Fraga 2011; Sulaiman et al. 2014), invasive weed optimization algorithm (Mohanty and Parhi 2014a, b; Mohanty et al. 2014) and Physarum algorithm (Atsushi et al. 2006; Miyaji and Onishi 2007; Atsushi et al. 2007; Zhang et al. 2014).

Although the source of inspiration is different in bio-inspired computing algorithms described above, they still have many similarities. The main relationships between various bio-inspired computing algorithms are: application of random variables, ability of dealing with uncertain and non-differentiable cost functions, simultaneous application of more than one computational agent for searching the domain of problem, existing a kind of communication scheme between computational agents (e.g., the social term in particle swarm optimization, pheromone trail in ant colony optimization, dancing of artificial bees in ABC, etc.), application of the objective function itself rather than its derivative for performing the search, and elimination of weak solutions at every iteration. In brief, it can be said that all of these algorithms perform a kid of memory stochastic search, which aims to optimize a certain objective function. More precisely, all algorithms of this type model the behavior of a (colony of) certain living entity by performing a kind of optimization. The procedure of modeling is always iterative and makes use of random variables, but the method is not wholly stochastic and has a kind of memory to remember the good solutions of previous iterations and provide the fittest agents of the colony with a more chance to survive and reproduce (Bayat 2014).

Clearly, every bio-inspired computing algorithm has its own advantages and disadvantages, and actually, it is pointless to look for the best algorithm (Coello 1999). For example, PSO is fast but sometimes leads to a solution outside the region defined by the boundary values of variables sometimes. However, it is important to note that the effectiveness of a certain algorithm strictly depends on the problem it is going to solve. In other words, it may happen that a certain algorithm be very successful in dealing with a problem while it is rather unsuccessful in dealing with another one. For this reason, researchers apply different algorithms to a certain problem to find the best method suited to solve it (Bayat 2014).

The aim of this paper is to propose a path planning algorithm inspired by the plant growth mechanism (abbreviated as PGPP). One main difference between the proposed algorithm and other bio-inspired computing algorithms is that the features of phototropism and negative geotropism are applied to guide the optimized direction instead of the objective function. In this way, only the value of the next waypoint needs to be optimized in each step of the optimization process rather than the objective function depending on all the waypoints. Therefore, this approach effectively reduces the complex of calculation and improves the computational efficiency. The other difference between the proposed algorithm and others is that, in this algorithm, a collision detection is introduced. The collision detection in each step of the optimization process increases the effectiveness of the optimum solution, which helps to avoid the problem occurs in the optimization process based on the objective function when the overall objective function value is excellent while several waypoints overstep the planning space. Moreover, unlike some bio-inspired computing algorithms, the number of computational agents is not (uniformly) constant from the beginning to end. In fact, at every iteration, the number of computational agents is duplicated in an appropriate manner and then the weakest agents are subjected to death.

The rest of this paper is organized as the following. Firstly, some basic principles and assumptions involved in PGPP are explained. Secondly, the basic processes of PGPP are proposed. Finally, several two-dimensional path planning cases are introduced to validate PGPP.

2 Basic principles and assumptions

PGPP is established through extracting information processing mechanism in the plant growth process. PGPP takes the plant bud as the basic computation unit to search for the optimal path. Different growth direction and speed of buds in PGPP form group search capability. PGPP defines the seed germ (first bud) as the starting point and the light source as the target point. PGPP discretizes the plant growth period and replaces it with iterative computations. During each iteration, the plant growth behavior is assumed to remain the same. Along with the plant growth, the path reaching the light source point first is the best growth path. The path is planned accordingly.

PGPP extracts phototropism, negative geotropism, apical dominance, and branch in plant growth as the basic rules.

Phototropism is the phenomenon of directional bending growth that occurs only towards the shining of one direction light source and represents the plant adaptive mechanism to poor illumination. It is induced by the changes of auxin concentration (abbreviated as AC) in light side and backlight side (Hao and Kang 2005). Phototropism is abstracted in PGPP as: the bud growth direction depending on the environment light intensity in a limited reference area. The light intensity of the computation point is inversely proportional to the square of the distance between the computation point and the target point. Furthermore, to consider the shadow of the obstacle in the reference range, the light intensity of the points under it needs a correction. The calculated light intensity is used to determine the growth direction and growth rate of the bud.

Negative geotropism is the characteristic that the front end of plant grows toward the sky and away from the ground (Hao and Kang 2005). It is the complement of the phototropism mechanism. The gravity vector introduces the effect of negative geotropism into PGPP. The line between the starting point and the target point is defined as the vertical direction. The gravity vector points from the target point to the starting point. Negative geotropism and the phototropism both have impacts on the growth direction of the bud.

Apical dominance is the restraining effect on the buds of lateral branches in growth and germination caused by the apical bud of the main branch, including the impact on the growth angle of the lateral branches (Hao and Kang 2005). It is due to the effect of two aspects. One is the polarity of auxin transport. The other is the different response of bud to AC, as shown in Fig. 1. The polarity transport makes the buds of lateral branches have too high AC, which will restrain the growth of buds. Meanwhile, the apical bud of the main branch has the suitable AC to promote its growth, and the apical dominance occurs. The amount of auxin polarity transport is determined by the difference between the branches' ages.

This paper simplifies the relationship between the AC and the growth rate. The red curve in Fig. 1 is the actual growth rate curve. The black line is the simplified growth rate curve.

Branch means that new branches germinate from the main branch to absorb the sunlight better in flank (Hao and Kang 2005). Branching is used to realize the diversity of the path search. It ensures PGPP finds the optimal path rather than a suboptimum one. When the branch grows to the branching age, PGPP judges whether to branch in a random way. The growth way of bud in the new branch depends on the light intensity distribution of the bud in the original branch. The light intensity of the new bud in the new branch is chosen randomly from the first five largest light intensity values of the original bud.

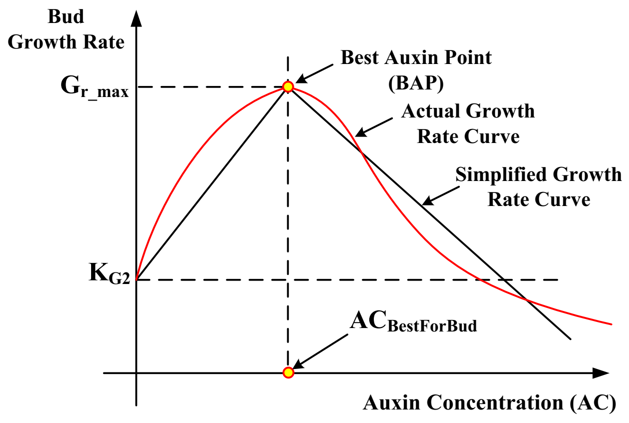

Fig. 1 Relationship between AC and growth rate

3 Basic processes

PGPP mainly includes six steps. The flowchart is presented in Fig. 2.

Step 1: Initialization. Initialize variables such as the plant, light intensity. Import the map data. Set the starting point as the growth point of the first bud.

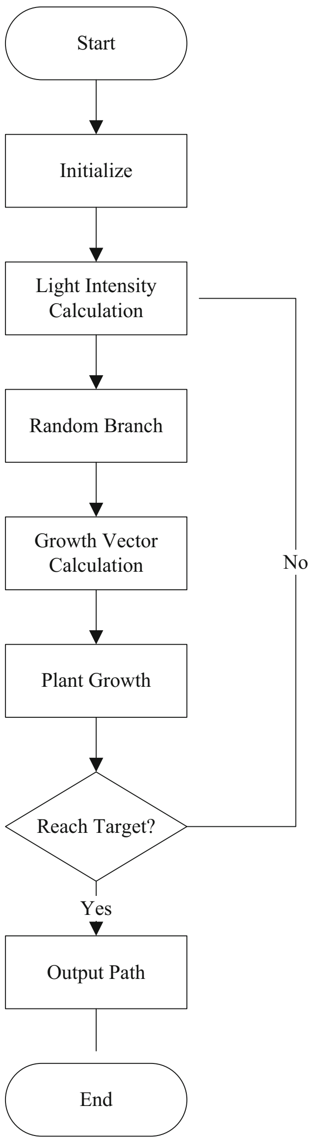

Fig. 2 PGPP flowchart

Step 2: Light intensity calculation. Calculate light intensity in the reference range.

Step 3: Random branch. Generate new branch randomly if the main branch meets the branch condition.

Step 4: Growth vector calculation. Calculate all the buds' AC and the light intensity growth vector. Then calculate the weighted sum of the light intensity growth vector, the gravity growth vector and the last period growth vector to get the new growth vector.

Step 5: Plant growth. Each bud carries out cell division according to its growth vector until a new cell reaches the target. If any cell reaches the target, the path search is over, and then head to the next step. Otherwise, return to step 2 as soon as all buds have finished the growth of this period.

Step 6: Path output. The path planning is over. The optimal path can be set up from the target point to starting point through looking for the parent cell of cell constantly.

This paper uses a two-dimensional path planning problem as an example to illustrate the PGPP algorithm. Steps 2–5 are described in detail as the key process. The calculations below are based on the two-dimensional area divided by grids. The interior of each grid is assumed to have the same property.

3.1 Light intensity calculation

In this step, we calculate each bud's light intensity to obtain the light intensity distribution in the reference range. One bud is taken as an example to explain the light intensity calculation process. The reference range of the bud is a half circle in the growth vector direction, as shown in Fig. 3. Obstacles in the reference range can lead to appearance of shadows, and this can have an effect on the light intensity on the buds.

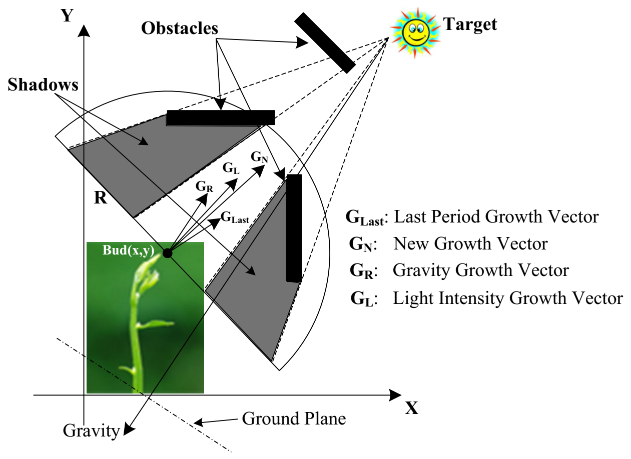
Y
Obstacles
Target
Shadows
R
G_L
G_N
G_R
G_{Last}
Bud(x,y)
G_{Last}: Last Period Growth Vector
G_N: New Growth Vector
G_R: Gravity Growth Vector
G_L: Light Intensity Growth Vector
X
Gravity
Ground Plane

Fig. 3 Schematic diagram of light intensity calculation

However, the obstacles outside the reference range need not be considered.

Light intensity calculation mainly includes four steps: coordinate transformation, initial light intensity calculation, shadow correction, and light intensity distribution calculation.

(A) Coordinate transformation

To simplify the bud light intensity calculation, we adopt the polar coordinate system, which uses the current bud as the origin, and uses the line between bud and the target as the polar axis. First, move the point (x, y) in Cartesian coordinate system to obtain the point (x_{bo}, y_{bo}) through Eq. (1). Let (x_b, y_b) represent the coordinate of current bud in Cartesian coordinate system.

$$\begin{cases} x_{bo} = x - x_b \\ y_{bo} = y - y_b \end{cases} \tag{1}$$

Then transfer Cartesian coordinate system to polar coordinate system through Eq. (2).

$$\begin{cases} \rho_o = \sqrt{x_{bo}^2 + y_{bo}^2} \\ \theta_o = \begin{cases} \arctan \frac{y_{bo}}{x_{bo}} \ (x_{bo} > 0) \\ \pi + \arctan \frac{y_{bo}}{x_{bo}} \ (x_{bo} < 0) \end{cases} \end{cases} \tag{2}$$

At last, rotate the polar coordinate system to get the polar axis starting from bud to target. The rotation angle is α, which is defined in Eq. (3). Here (x_T, y_T) is the coordinate of target point in Cartesian coordinate system.

$$\alpha = \begin{cases} \arctan \frac{y_T - y_b}{x_T - x_b} \ (x_T - x_b > 0) \\ \pi + \arctan \frac{y_T - y_b}{x_T - x_b} \ (x_T - x_b < 0) \end{cases} \tag{3}$$

The polar coordinate system is obtained from Eq. (4).

$$\begin{cases} \rho = \rho_o \\ \theta = \theta_o - \alpha \end{cases} \tag{4}$$

(B) Initial light intensity calculation

Calculate the initial light intensity (the effect of shadow is not considered in this step) of all the blank grids in the reference range. The reference range is the semi-circle in which the symmetry axis is polar axis, and the radius is R, as shown in Fig. 3. It also meets Eq. (5).

$$(\rho, \theta) \in \begin{cases} 0 \le \rho \le R \\ -\frac{\pi}{2} \le \theta \le \frac{\pi}{2} \end{cases} \tag{5}$$

The light intensity is inversely proportional to the square of the distance between the grid and target point. The initial light intensity formula is presented in Eq. (6) to reduce the light intensity difference because of distance change between the bud and target point, and avoid the growth rate increasing

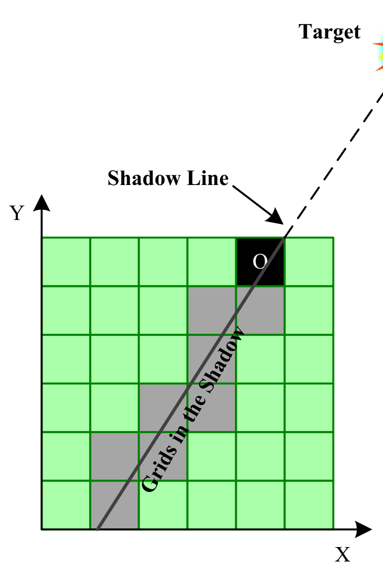

Fig. 4 Schematic diagram of shadow rule

inconsequently in the late phase. Here $Li(x, y)$ is initial light intensity of (x, y) in Cartesian coordinate system. K_{L1} is the coefficient of light intensity.

$$Li(x, y) = \frac{K_{L1} \times \left[(x_T - x_b)^2 + (y_T - y_b)^2\right]}{(x - x_T)^2 + (y - y_T)^2} \tag{6}$$

(C) Shadow correction

The shadow rule used in the algorithm is shown in Fig. 4 when an obstacle exists in the reference range. The black grid represents the obstacle that the plant cannot pass. The black line represents the shadow line. The gray grids represent the area influenced by the shadow. The principle is that effect of shadow works in case the shadow line crosses the grid. The gray grids (x, y) need to be corrected, and they are described in Eq. (7). Here (x_o, y_o) is the coordinate of obstacle in Cartesian coordinate system.

$$\begin{aligned}
&K_b = \frac{y_T - y_b}{x_T - x_b} \\
&K_s = \frac{K_b}{|K_b|} \\
&x \in \left[\text{floor}\left[\frac{x_T(K_b y_b - K_b y_o + x_b) - x_o y_b K_b - x_o x_b + K_b y_T x_o}{K_b(y_T - y_o) + (x_T - x_o)} + 0.5\right],\right. \\
&\quad \left. x_o - 1\right] \\
&y \in \left[\text{floor}\left[(\frac{y_o - y_T}{x_o - x_T})(x - \frac{1}{2}K_s) + \frac{x_o y_T - y_o x_T}{x_o - x_T} + 0.5\right],\right. \\
&\quad \left.\text{floor}\left[(\frac{y_o - y_T}{x_o - x_T})(x + \frac{1}{2}K_s) + \frac{x_o y_T - y_o x_T}{x_o - x_T} + 0.5\right]\right]
\end{aligned} \tag{7}$$

The light intensity correction of these gray grids is presented as in Eq. (8). Here $Li_{\text{new}}(x, y)$ is the corrected light intensity of (x, y) in Cartesian coordinate system. K_{L2} is the coefficient of shadow.

$$Li_{\text{new}}(x, y) = Li(x, y) \times (1 - K_{L2}) \tag{8}$$

(D) Light intensity distribution calculation

We adopt the discretization way to obtain the light intensity distribution. The reference range is divided into 12 sectors Ω_i. The angle of each sector is 15°. Ω_i can be expressed by Eq. (9).

$$\Omega_i(\rho, \theta) \in \begin{cases} 0 < \rho < R \\ \frac{\pi}{2} - \frac{\pi}{12} \times i \le \theta \le \frac{\pi}{2} - \frac{\pi}{12} \times (i - 1) \quad i \in [1, 12] \end{cases} \tag{9}$$

$Li_i\, i \in [1, 12]$ expresses the sum of the grids' light intensity in each Ω_i, as shown in Eq. (10). The largest light intensity $Li_{\max}$ is taken as the optimal light intensity of the bud. It will be used in next calculation. $Li(\rho, \theta)$ is the light intensity of (ρ, θ) in the polar coordinate system.

$$Li_i = \sum Li(\rho, \theta)\ (\rho, \theta) \in \Omega_i \tag{10}$$

3.2 Random branch

Branch age is defined first. Once the bud in one branch completes one cell division, the branch age adds one. Cell division will repeat many times during each iteration. The new branch age and the original branch age are one when the new branch germinates. When the branch age meets the condition in Eq. (11), the branch probability $P_{\text{Bran-off}}$ determines whether to branch. The growth way of bud in the new branch depends on the light intensity distribution of the bud in the original branch. The light intensity of the new bud in the new branch is chosen randomly from the first five largest light intensity values of the original bud. In Eq. (11), Age_{Bud} is the bud age in one branch. $\text{Age}_{\text{Bran-off}}$ is the threshold of branch age.

$$\text{Age}_{\text{Bud}} \ge \text{Age}_{\text{Bran-off}} \tag{11}$$

3.3 Growth vector calculation

The bud's growth vector is the weighted sum of the light intensity growth vector, the gravity growth vector, and the last period growth vector, as defined in Eq. (12). Here $\mathbf{G}_{\text{New}}$ is the new growth vector. $\mathbf{G}_L$ is the light intensity growth vector. $\mathbf{G}_R$ is the gravity growth vector. $\mathbf{G}_{\text{Last}}$ is the last period growth vector. K_{n1}, K_{n2}, K_{n3} are the weighted coefficients.

$$\mathbf{G}_{\text{New}} = K_{n1} \times \mathbf{G}_L + K_{n2} \times \mathbf{G}_R + K_{n3} \times \mathbf{G}_{\text{Last}} \tag{12}$$

To calculate the light intensity growth vector, we first calculate the AC. The bud AC can be obtained from Eq. (13). The first item of the right side of Eq. (12) is the part generated by light intensity, and the second item is the part generated by polarity transport.

$$\begin{cases} A_c = K_{a1} \times Li_{\max} + K_{a2}(\mathrm{Age}_{\max} - \mathrm{Age}) \ (K_{a1} \times LI \le A_{c\_\mathrm{BestForBud}}) \\ A_c = A_{c\_\mathrm{BestForBud}} + K_{a2}(\mathrm{Age}_{\max} - \mathrm{Age}) \ (K_{a1} \times LI > A_{c\_\mathrm{BestForBud}}) \end{cases} \quad (13)$$

In Eq. (13), $A_c$is the AC of current bud. K_{a1} is the light intensity coefficient of auxin. K_{a2} is the polarity transport coefficient of auxin. $\mathrm{Age}_{\max}$ is the maximum bud age in all buds. Age is the age of current bud.

After getting the AC of the bud, the growing rate can be calculated based on the relationship between the growth rate and AC shown in Fig. 1. When the AC is lower than the best auxin point (abbreviated as BAP), the growth rate of the bud will increase as the AC increases. When the AC is higher than the BAP, the growth rate of the bud will decrease as the AC increases. The relationship is shown in Eq. (14).

$$\begin{cases} |\mathbf{G_L}| = K_{G1} \times A_c + K_{G2} & \text{when} \quad A_c \in [0, A_{\mathrm{BestForbud}}] \\ |\mathbf{G_L}| = K_{G3} \times A_c + (G_{r\_\max} - K_{G3} \times A_{\mathrm{BestForbud}}) & \text{when } A_c \in [A_{\mathrm{BestForbud}}, \infty) \end{cases} \quad (14)$$

The direction of $\mathbf{G}_L$ is the symmetry axis of the sector in which the light intensity is largest and points to the target. K_{G1}, K_{G2}, K_{G3} are coefficients of growth rate. $G_{r\_\max}$ is the largest growth rate.

The gravity vector is a line pointing from target point to starting point. The gravity growth vector is the reverse of the gravity vector as defined in Eq. (15). Here (x_S, y_S) is the coordinate of starting point in Cartesian coordinate system.

$$\mathbf{G_R} = (x_T - x_S)\mathbf{i} + (y_T - y_S)\mathbf{j} \quad (15)$$

3.4 Plant growth

The plant growth is realized by the cell division. The bud will repeat cell division several times in the direction of growth vector to complete growth in each iteration. The cell division carries out in Cartesian coordinate system. Therefore, the growth vector in the polar coordinate system should be converted into Cartesian coordinate system first. Then the growth end point $(x_D,\ y_D)$ in this iteration can be calculated by Eq. (16). Here $\rho_G,\ \theta_G,\ \alpha_{Bo}$ are the parameters of $\mathbf{G}_{\mathrm{New}}$ in polar coordinate system.

$$\begin{cases} x_D = \mathrm{floor}(x_B + \rho_G \times \cos(\theta_G + \alpha_{Bo}) + 0.5) \\ y_D = \mathrm{floor}(y_B + \rho_G \times \sin(\theta_G + \alpha_{Bo}) + 0.5) \end{cases} \quad (16)$$

The cell division has three typical situations, as shown in Fig. 5. Figure 5a represents the situation that has no obstacle. Figure 5b, c represents the situations that have some obstacles. The blue grids are the current buds. The red ones are the growth end points in the current period. The black ones are the obstacles. The green ones are the processes of cell division. First, we calculate the coordinate difference in x axis and y axis between the current bud point and the growth end point. The axis in which the difference is bigger is taken as the priority growth direction until the difference in x axis and y axis is equal. Then the tilted way is adopted to grow, as shown in Fig. 5a. In this iteration, cell division repeats five times. Therefore, the branch age adds five after this iteration. If obstacle appears in the growth direction, the other direction becomes the grow direction, as shown in Fig. 5b. In this iteration, cell division repeats seven times. Therefore, the branch age adds seven after this iteration. If there is no available direction, as shown in Fig. 5c, the bud is dead. The dead bud will never carry out any further calculation.

The obstacle could be the branch that appears first. The branch that appears first will grow first. The other branches which are obstructed by the branch will decide if continue to grow or stop, depending on where is the obstacle.

4 Implementation results

We take several two-dimensional path planning examples to validate PGPP. The types of obstacles in two-dimensional

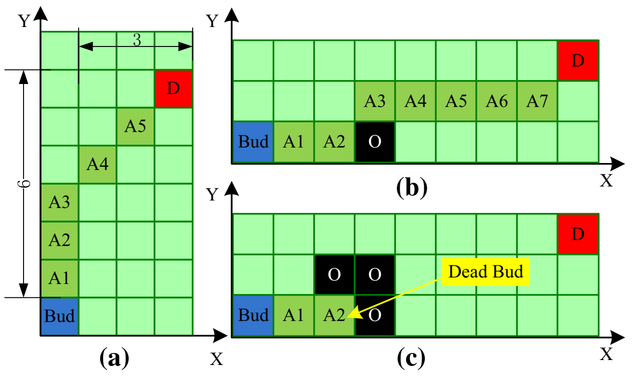

Fig. 5 Typical situations of cell division

Table 1 Parameters initialization

Parameter	Value	Parameter	Value
R	10	K_{G1}	1
K_{L1}	0.05	K_{G2}	1
K_{L2}	0.3	K_{G3}	−1.5
$\mathrm{Age}_{\mathrm{Br}}$	8	$\mathrm{AC}_{\mathrm{BestForBud}}$	2
P_{Br}	0.2	K_{n1}	2
K_{a1}	1	K_{n2}	0.1
K_{a2}	0.1	K_{n3}	0.1

maps include the circular obstacles, the rectangular obstacles and the combination of them. The calculating way of threat between circular obstacles and rectangular obstacles is different which helps to test the adaptive capacities of different algorithms to the obstacles. PGPP is compared with DE, ABC and A* algorithm which has been used widely in path planning as a heuristic algorithm.

The parameters of PGPP adopted in these examples are shown in Table 1. They are determined based on plant growth mechanism. They are the recommended values and can be optimized in further research to fit different applications.

DE algorithm selects the start point as the original point and sets the line segment between the starting point and the target point as the X-axis. We choose N waypoints whose X-coordinates divide the line segment into $N+1$ equal ones and optimize the Y-coordinates of these waypoints through the DE algorithm. Meanwhile, the length of path and the threat value of obstacles make up the objective function of DE with an appropriate weight. The population size of DE adopted in these examples is 20. The dimension N needing to be optimized is 24. The mutagenic factor is 0.5. The cross factor is 0.9. Adjust the weight between the length and threat value in the objective function to obtain the optimum solution.

Similar to DE algorithm, we try to use ABC algorithm to optimize the Y-coordinates of the waypoints. The population size of the bee colony adopted in these examples is 20. Half of the bees act as the employed sources, and the other half act as the unemployed foragers.

A* algorithm uses the distance between the extended node and the target point as the heuristic information. The step is set to 4.

The simulation in this paper is carried out on visual studio 2013.

4.1 Circular obstacle

This section takes the map including the circular obstacles as an example to carry out path planning. The center of circular obstacle 1 is (40, 40), with a radius of 15. The center of

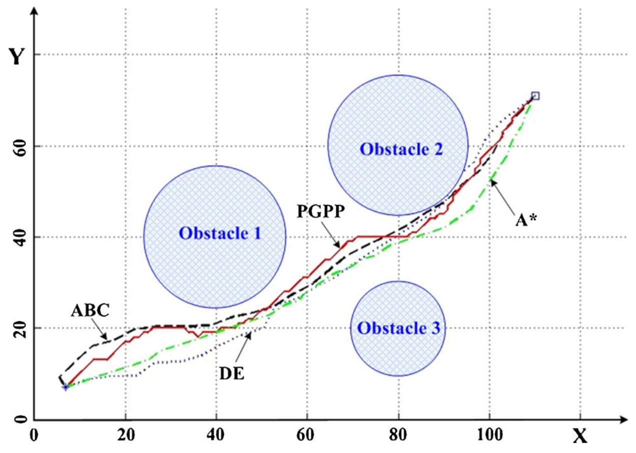

Fig. 6 The 1st path planning result about circular obstacles

Table 2 The 1st test results about circular obstacles

Algorithm	Iteration times	Time-consuming (s)	Route length
ABC	721	0.79	129
DE	86	0.27	127
A*	1244	0.95	127
PGPP	65	0.05	131

circular obstacle 2 is (80, 60), with a radius of 15. The circular obstacle 3 is (80, 20), with a radius of 10.

(7, 7) and (110, 71) are, respectively, the starting point and the target point of the first test. The path planning results are shown in Fig. 6. Here the red solid line represents the route planned by PGPP. The black imaginary line represents the route planned by ABC. The blue dotted line represents the route planned by DE. The green dot dash line represents the route planned by A*. It can be found that ABC, A* and PGPP can plan a valid path. However, the path planned by DE is invalid because some part of the path is located in obstacle 2. The reason why this phenomenon appears in DE is that several waypoints being tangent to or slightly intersected with the obstacles contribute little to the integral objective function, which can hardly be eliminated effectively. It can also be found that partial path of ABC around the starting point is not good, but it still be kept. It is because that the proportion of the partial path in the whole path is very small, which means that it contributes little to the integral objective function. Similar to DE above, it cannot be improved in the principle of finding the smallest objective function. The path planned by A* is smooth while the path planned by PGPP appears like a step shape. This is due to the detection of obstacles by the minimum path unit in every optimization. It generates path in this way.

The test results are listed in Table 2. The time-consuming of PGPP is an order of magnitude lower than DE, ABC and A* algorithm. This is because that PGPP plans the path

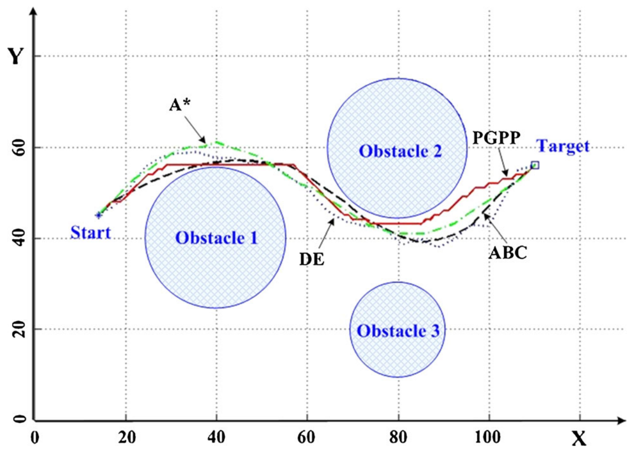

Fig. 7 The 2nd path planning result about circular obstacles

Table 3 The 2nd test results about circular obstacles

Algorithm	Iteration times	Time-consuming (s)	Route length
ABC	283	0.22	110
DE	468	0.6	118
A*	1043	0.8	111
PGPP	65	0.03	111

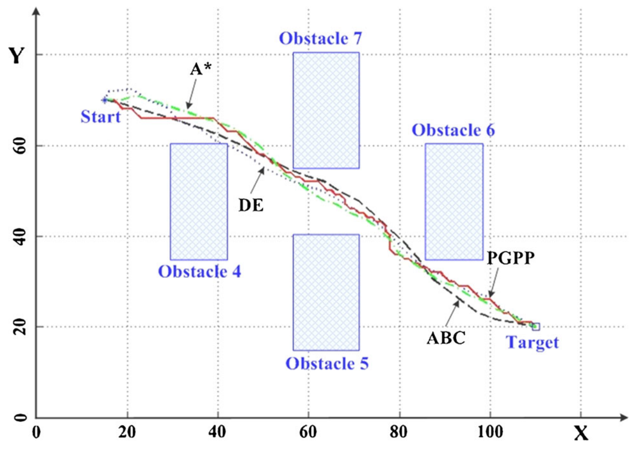

Fig. 8 The 1st path planning result about rectangular obstacles

Table 4 The 1st test results about rectangular obstacle

Algorithm	Iteration times	Time-consuming (s)	Route length
ABC	2046	1.26	110
DE	74	0.12	112
A*	446	0.34	111
PGPP	55	0.03	115

step by step. It calculates just the growing direction and the growing amount of the next step in each iteration with less computation amount. However, all the paths would be optimized in each iteration by DE or ABC, which may result in a relatively large amount of computation. All the nodes which have been traversed during the optimization process need to be stored. It has an impact on the computational efficiency. Besides these, it can be found that the length of PGPP is approximately equal to the length of the other three algorithms.

(14, 45) and (110, 56) are, respectively, the starting point and the target point of the second test. The path planning results are shown in Fig. 7. It can be found that ABC, DE, A* and PGPP all can plan a valid path. The paths planned by ABC, A* and PGPP are smooth while the path planned by DE appears large fluctuation. This is because the weight of the threat value of obstacles in the objective function is increased to avoid the path go across the obstacles. The increasing weight leads to the dramatic changes in path.

From the test results listed in Table 3, a conclusion can be drawn that the time-consuming of PGPP is an order of magnitude lower than DE, ABC and A*. The length of PGPP is approximately equal to the length of ABC and A*, which is shorter than DE.

4.2 Rectangular obstacle

This section takes the map including rectangular obstacles as an example to carry out path planning.

There are four rectangular obstacles in the first test, including rectangular obstacle 4, rectangular obstacle 5, rectangular obstacle 6 and rectangular obstacle 7. The center coordinates of them are (36, 48), (64, 28), (92, 48) and (64, 68) respectively. The lengths of them are all 25. The widths of them are, respectively, 12, 14, 12 and 14. The starting point locates in (15, 70). The target point locates in (110, 20).

The path planning results are shown in Fig. 8. It can be found that ABC, A* and PGPP can plan a valid path although the path planned by ABC goes along the edge of obstacle 7. However, the path planned by DE is invalid because some part of the path is located in obstacle 4.

From the test results listed in Table 4, it can be seen that the time-consuming of PGPP is an order of magnitude lower than DE, ABC and A*. The time-consuming of ABC is far beyond the other three algorithms. This is because the path goes through obstacles frequently during optimization. It means that the threat value of obstacles should be calculated which adds the amount of calculation. Therefore, the time-consuming becomes very long. The length of PGPP is approximately equal to the length of the other three algorithms.

The center coordinates of the four obstacles modeled in the second test including rectangular obstacle 8, rectangular obstacle 9, rectangular obstacle 10, rectangular obstacle 11 are (30, 48), (65, 24), (100, 48) and (65, 70), respectively. The lengths of them are all 20. The widths of them are all 8. The starting point locates in (5, 55). The target point locates in (125, 55).

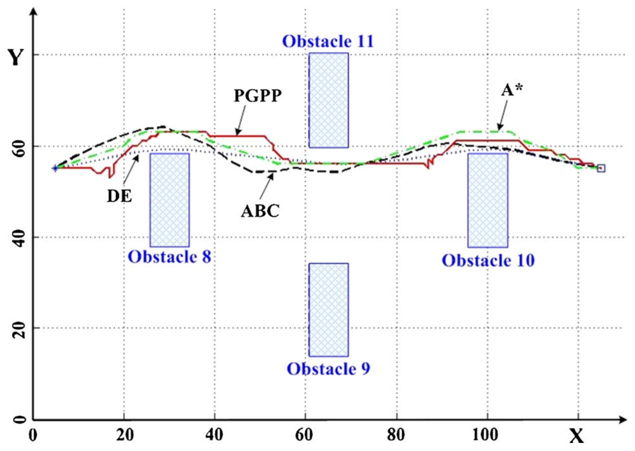

Fig. 9 The 2nd path planning result about rectangular obstacles

Table 5 The 2nd test results about rectangular obstacles

Algorithm	Iteration times	Time-consuming (s)	Route length
ABC	933	0.62	126
DE	39	0.07	121
A*	1422	1.09	127
PGPP	66	0.05	133

The path planning results are shown in Fig. 9. It can be found that ABC, DE, A* and PGPP all can plan a valid path. The path of PGPP appears upper and lower volatility because the searching areas are all affected by the shadow of obstacle. It leads the optimization value in upper and lower areas mainly depending on light intensity almost the same. Therefore, the initial path appears stochastic upper and lower volatility.

From the test results listed in Table 5, it can be seen that the time-consuming of PGPP is an order of magnitude lower than ABC and A*, while approximately equal to DE. The time-consuming of A* is very long here, which means the layout of obstacles is difficult to A*. It has to store a large number of traversal points, and constantly back to find the right path. Of course this will take a large amount of calculation. The length of PGPP is approximately equal to the length of the other three algorithms here.

4.3 The combination of circular and rectangular obstacles

This section takes the map including the combination of the circular obstacles and rectangular obstacles as an example to carry out path planning.

There are three circular obstacles including circular obstacle 12, circular obstacle 13 and circular obstacle 14 in this map. The center coordinates of them are (40, 40), (80, 20) and (95, 60), respectively. The radius of them is, respectively,

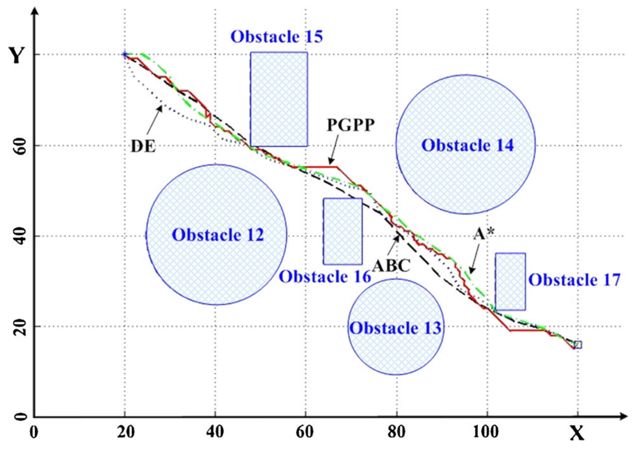

Fig. 10 Path planning result about combination obstacles

Table 6 The test results about combination obstacles

Algorithm	Iteration times	Time-consuming (s)	Route length
ABC	963	1.15	120
DE	98	0.22	124
A*	815	0.62	123
PGPP	65	0.03	129

15, 10 and 15. There are three rectangular obstacles including rectangular obstacle 15, rectangular obstacle 16 and rectangular obstacle 17. The center coordinates of them are (54, 70), (68, 41) and (105, 30), respectively. The lengths of them are, respectively, 20, 14 and 12. The widths of them are, respectively, 12, 8 and 6. The starting point locates in (20, 80). The target point locates in (120, 16).

The path planning results are shown in Fig. 10. It can be found that DE, A* and PGPP can plan a valid path. However, the path planned by ABC is invalid because some part of the path is located in obstacle 15 and 16. Similar to DE, several waypoints being tangent to or slightly intersected with the obstacles contribute little to the integral objective function, which can hardly be eliminated effectively.

From the test results listed in Table 6, it can be seen that it can be seen that the time-consuming of PGPP is an order of magnitude lower than ABC, DE and A*. The time-consuming of ABC is far beyond the other three algorithms. This is because the path goes through obstacles frequently during optimization. It means that the threat value of obstacles should be calculated which adds the amount of calculation. Therefore, the time-consuming becomes very long. The length of PGPP is approximately equal to the length of the other three algorithms here.

4.4 Discussion

The effectiveness of path planning in different tests is listed in Table 7. It can be seen that PGPP can improve the success

Table 7 The test results about combination obstacles

Obstacle algorithm		ABC	DE	A*	PGPP
Circular obstacle	The 1st test	Valid	Invalid	Valid	Valid
	The 2nd test	Valid	Valid	Valid	Valid
Rectangular obstacle	The 1st test	Valid	Invalid	Valid	Valid
	The 2nd test	Valid	Valid	Valid	Valid
Combination obstacle		Invalid	Valid	Valid	Valid
Success (valid) rate		80 %	60 %	100 %	100 %

Table 8 The test results about combination obstacles

Obstacle algorithm		ABC	DE	A*	PGPP
Circular obstacle	The 1st test	0.79	0.27	0.95	0.05
	The 2nd test	0.22	0.6	0.8	0.03
Rectangular obstacle	The 1st test	1.26	0.12	0.34	0.03
	The 2nd test	0.62	0.07	1.09	0.05
Combination obstacle		1.15	0.22	0.62	0.03

rate of path planning when compared with ABC and DE. It has the same success rate as A* which has been widely used in traditional path planning.

The time-consuming of path planning in different tests is listed in Table 8. It can be found that the time-consuming of PGPP is almost the same while the time-consuming of ABC, DE and A* has significant changes. This means the adaptability of PGPP is high. It is more suitable to apply in mission planning system, which has very strict time limit, such as path planning on line.

5 Conclusion

A path planning algorithm based on plant growth mechanism is proposed in this paper. The basic principles, assumptions and processes of PGPP are described in detail. PGPP is validated using several two-dimensional path planning problems. The results compared with ABC, DE and A* show that PGPP has good path planning ability with proper parameter configuration. The operation efficiency of PGPP is high. The operation process is stable. It can apply to the path planning system that has strict requirements of operation time and result. The application of PGPP in other optimal problems will be a necessary and interesting research in the future.

Acknowledgments The authors gratefully acknowledge the support of Aerospace Science and Industry Fund.

Compliance with ethical standards

Conflict of interest The authors have no conflict of interests.

References

Alejandro HP, Miguel AVR, Joaquin F (2015) MOSFLA-MRPP: multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning. Eng Appl Artif Intell 44:123–136

Atsushi T, Ryo K, Toshiyuki N (2006) Physarum solver: a biologically inspired method of road-network navigation. Phys A Stat Mech Appl 363(1):115–119

Atsushi T, Ryo K, Toshiyuki N (2007) A mathematical model for adaptive transport network in path finding by true slime mold. J Theor Biol 244(4):533–564

Aydin S, Temeltas H (2004) Fuzzy-differential evolution algorithm for planning time-optimal trajectories of a unicycle mobile robot on a predefined path. Adv Robot 18(7):725–748

Bayat FM (2014) A numerical optimization algorithm inspired by the strawberry plant. Eprint Arxiv

Bhattacharjee P, Rakshit P, Goswami I, Konar A, Nagar AK (2011) Multi-robot path-planning using artificial bee colony optimization algorithm. In: Proceedings of third world congress on nature and biologically inspired computing, pp 219–224, 2011

Chen M, Wu QX, Jiang CS (2008) A modified ant optimization algorithm for path planning of UCAV. Appl Soft Comput 8:1712–1718

Coello CA (1999) A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl Inform Syst 1(3):269–308

Das PK, Pradhan SK, Patro SN, Balabantaray BK (2012) Artificial immune system based path planning of mobile robot. Stud Comput Intell 395:195–207

Deepak BBVL, Parhi DR, Kundu S (2012) Innate immune based path planner of an autonomous mobile robot. Cent Eur J Comput Sci 2(2):2663–2671

Duan HB, Yu YX, Zhou R (2008) UCAV path planning based on ant colony optimization and satisficing decision algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 957–962, 2008

Duan HB, Yu YX, Zhang XY (2010) Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. Simul Model Pract Theory 18(8):1104–1115

Foo JL, Knutzon J, Kalivarapu V, Oliver J, Winer E (2009) Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization. J Aerosp Comput Inform Commun 6(4):271–290

Fu YG, Ding MY, Zhou CP (2012) Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV. IEEE Trans Syst Man Cybern Part A Syst Hum 42(2):511–526

Garcia MAP, Montiel O, Castillo O, Sepulveda R, Melin P (2009) Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. Appl Soft Comput 1(2):1102–1110

Gong DW, Zhang JH, Zhang Y (2011) Multi-objective particle swarm optimization for robot path planning in environment with danger sources. J Comput 6(8):1554–1561

Hao JJ, Kang ZL (2005) Plant physiology, Chaps. 7, 8. Chemical Industry Press, Beijing **(in Chinese)**

Hassanzadeh I, Madani K, Badamchizadeh MA (2010) Mobile robot path planning based on shuffled frog leaping optimization algorithm. In: Proceedings of 6th annual IEEE conference on automation science and engineering, pp 680–685, 2010

Hossain MA, Ferdous I (2015) Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot Auton Syst 64:137–141

Jati A, Singh G, Rakshit P, Konar A, Kim E, Nagar AK (2012) A hybridisation of improved harmony search and bacterial foraging for multi-robot motion planning. In: Proceedings of WCCI 2012 IEEE world congress on computational intelligence, 2012

Karc A (2007) Natural inspired computational intelligence method: saplings growing up algorithm. In: Proc of IEEE Int Conf Computational Cybernetics, Gammarth, Tunisia

Li BL, Liu LJ, Zhang QH, Lv DJ, Zhang YF, Zhang JH, Shi XL (2014) Path planning based on firefly algorithm and Bezier curve. In: Proceeding of the IEEE international conference on information and automation, pp 630–633, 2014

Li B, Gong LG, Yang WL (2014) An improved artificial bee colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning. Sci World J 2014:1–10

Liang XD, Li LY, Wu JG, Chen HN (2013) Mobile robot path planning based on adaptive bacterial foraging algorithm. J Cent South Univ 20:3391–3400

Li T, Su WL (2007) Research on plant growth simulation algorithm based on finite element method. In: Proceedings of second international conference on innovative computing, information and control, p 419, 2007

Li T, Su WL, Wang CF (2004) A global optimization bionics algorithm for solving integer programming - Plant growth simulation algorithm. In: Proceedings of international conference on management science and engineering, pp 531–535, 2004

Liu W, Niu B, Chen HN, Zhu YL (2013) Robot path planning using bacterial foraging algorithm. J Comput Theor Nanosci 10:2890–2896

Liu C, Zhao YX, Gao F, Liu LQ (2015) Three-dimensional path planning method for autonomous underwater vehicle based on modified firefly algorithm. Math Probl Eng 2015:1–10

Liu C, Gao ZQ, Zhao WH (2012) A new path planning method based on firefly algorithm. In: Proceedings of 2012 fifth international joint conference on computational sciences and optimization, pp 775–778, 2012

Luh GC, Liu WW (2008) An immunological approach to mobile robot reactive navigation. Appl Soft Comput 8(1):30–45

Ma QZ, Lei XJ (2010) Application of artificial fish school algorithm in UCAV path planning. In: IEEE fifth international conference on bio-inspired computing: theories and applications, pp 555–559, 2010

Mernik M, Liu S-H, Karaboga MD, Crepinšek M (2015) On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. Inform Sci 291:115–127

Miyaji T, Onishi I (2007) Mathematical analysis to an adaptive network of the *Plasmodium* system. Hokkaido Math J 36(2):445–465

Mohanty PK, Parhi DR (2014) A new real time path planning for mobile robot navigation using invasive weed optimization algorithm. In: Proceedings of ASME 2014 gas turbine india conference, p V001T07A002, 2014

Mohanty PK, Kumar S, Parhi DR (2014) A new ecologically inspired algorithm for mobile robot navigation. Adv Intell Syst Comput 327:755–762

Mohanty PK, Parhi DR (2014) A new efficient optimal path planner for mobile robot based on invasive weed optimization algorithm. Front Mech Eng 9(4):317–330

Mo HW, Meng LL (2012) Robot path planning based on differential evolution in static environment. Int J Digital Content Technol Appl 6(20):122–129

Ni JJ, Yin XH, Chen JF, Li XY (2014) An improved shuffled frog leaping algorithm for robot path planning. In: Proceedings of 2014 10th international conference on natural computation, pp 545–549, 2014

Peng JS, Li X, Qin ZQ, Luo G (2013) Robot global path planning based on improved artificial fish-swarm algorithm. Res J Appl Sci Eng Technol 5(6):2042–2047

Rao RS, Narasimham SVL (2008) Optimal capacitor placement in a radial distribution system using plant growth simulation algorithm. Int J Electr Power Energy Energy Syst Eng 1(2):123–130

Salhi A, Fraga ES (2011) Nature-inspired optimization approaches and the new plant propagation algorithm. In: Proceedings of the the international conference on numerical analysis and optimization, Yogyakarta, Indonesia, pp K2-1–K2-8, 2011

Sulaiman M, Salhi A, Selamoglu BI, Kirikchi OB (2014) A plant propagation algorithm for constrained engineering optimization problems. Math Probl Eng 2014:1–10

Tan GZ, He H, Sloman A (2007) Ant colony system algorithm for real-time globally optimal path planning of mobile robots. Acta Automatica Sinica 33(3):279–285

Wang YN, Lee TS, Tsao TF (2007) Plan on obstacle-avoiding path for mobile robots based on artificial immune algorithm. Lect Notes Comput Sci 4491:694–703

Wang GG, Guo LH, Duan H, Liu L, Wang HQ (2012) A modified firefly algorithm for UCAV path planning. Int J Hybrid Inform Technol 5(3):123–144

Xu CF, Duan HB, Liu F (2010) Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning. Aerosp Sci Technol 14(8):535–541

Zhang XG, Zhang YJ, Zhang ZL, Mahadevan S (2014) Rapid Physarum algorithm for shortest path problem. Appl Soft Comput 23:19–26

Zhang ZR, Yin JY (2012) The study on mobile robot path planning based on frog leaping algorithm. Adv Mater Res 3:490–495