SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2018, Vol. 61 052204:1–052204:18
doi: 10.1007/s11432-016-9115-2

Path planning for mobile robot using self-adaptive learning particle swarm optimization

Guangsheng LI[1*] & Wusheng CHOU[1,2]

[1]*School of Mechanical Engineering and Automation, Beihang University, Beijing* 100191*, China;*
[2]*State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing* 100191*, China*

Received 20 December 2016/Revised 19 March 2017/Accepted 16 May 2017/Published online 15 November 2017

Abstract As a challenging optimization problem, path planning for mobile robot refers to searching an optimal or near-optimal path under different types of constrains in complex environments. In this paper, a self-adaptive learning particle swarm optimization (SLPSO) with different learning strategies is proposed to address this problem. First, we transform the path planning problem into a minimisation multi-objective optimization problem and formulate the objective function by considering three objectives: path length, collision risk degree and smoothness. Then, a novel self-adaptive learning mechanism is developed to adaptively select the most suitable search strategies at different stages of the optimization process, which can improve the search ability of particle swarm optimization (PSO). Moreover, in order to enhance the feasibility of the generated paths, we further apply the new bound violation handling schemes to restrict the velocity and position of each particle. Finally, experiments respectively with a simulated robot and a real robot are conducted and the results demonstrate the feasibility and effectiveness of SLPSO in solving mobile robot path planning problem.

Keywords path planning, self-adaptive learning particle swarm optimization, learning strategy, learning mechanism, boundary violations handling

Citation Li G S, Chou W S. Path planning for mobile robot using self-adaptive learning particle swarm optimization. Sci China Inf Sci, 2018, 61(5): 052204, doi: 10.1007/s11432-016-9115-2

1 Introduction

As mobile robot becomes an integral part in human life, we charge them with increasingly varied and difficult tasks including search and rescue, medical assistance, reconnaissance and planetary exploration [1]. To these ends, path planning becomes one of the most important technologies required in the design of mobile robot. Path planning for mobile robot focuses on generating an optimal or near-optimal path from initial location to desired destination without collision with obstacles [2].

In the past decades, numerous researches have been devoted to this area and series of algorithms have been put forward to solve this optimization problem, such as A∗ algorithm [3, 4], potential field method [5], simulated annealing algorithm [6], genetic algorithm (GA) [7], particle swarm optimization (PSO) [8, 9], ant colony optimization [10], gravitational search algorithm (GSA) [11, 12] and Pigeon-inspired optimization [13]. Therein, as an effective tool for solving optimization problems, PSO has been actively studied and widely used in mobile robot path planning due to its concise mechanism and few control parameters. In most situations, PSO can provide similar or even better results than other

* Corresponding author (email: liguangsheng10@163.com)

© Science China Press and Springer-Verlag Berlin Heidelberg 2017

methods, such as genetic algorithm [14]. However, the standard PSO has the inherent shortcomings of premature convergence (trapped by local optimum) and lacking of guarantee in global convergence, which greatly limited its application in many practical systems.

To deal with the problems mentioned above, many improved versions of PSO have been proposed to generate feasible paths for mobile robot. The intuitionistic way is to improve the standard PSO by parameters adjustment or new operators for obtaining more reliable and accurate optimization results. For instance, Qin et al. [15] presented a modified PSO with a mutation operator for mobile robot to escape from local optima. In [16], Zhang et al. introduced a binary PSO with double-structure particle encoding and mutation operator for mobile robot path planning in an environment with danger sources and obstacles. Similar mutation-based method has also been applied in [17], where the self-adaptive mutation operation based on the degree of the path blocked by obstacles was used to repair the invalid paths. Also, Juang and Chang [18] designed a new fuzzy controller based on the evolutionary-group-based PSO for mobile robot navigation in unknown environments. Being new evolutionary operations, the group-based crossover operation and adaptive velocity-mutated operation are incorporated into PSO for performance improvement. In [19], to improve the exploration ability of finding a safety path, Chen and Li put forward a stochastic PSO for mobile robot path planning with obstacle avoidance. What's more, the new mechanism of updating velocities and positions proposed in [20] was also applied to plan the optimized path for multi-survivor rescue. Another recent study on this topic is by Zhang et al. [21], in which the multi-objective PSO based on random sampling and uniform mutation operations is developed for mobile robot navigation in the environment with uncertain danger sources. However, these methods can hardly solve some complex optimization problems with with multiple local optima duo to their common drawback: the monotonic learning mechanism for all particles, which reduces the adaptability of PSO to complex environments.

The other way is to combine some other optimization methods with the standard PSO algorithm. For example, Masehian and Sedighizadeh [22] proposed the PSO-based motion planning method to generate a short and smooth path for mobile robot. In this method, PSO is employed for global path planning, while the probabilistic roadmap method is used for local planning. Combining the basic model of PSO with membrane-inspired algorithm, Wang et al. [23] proposed an novel collision-free path planning approach for mobile robot in a grid-based environment with dynamic obstacles or dangerous objects. In [24], the new PSO method combined with Biogeography-based optimization (BBO) and approximate voronoi boundary network (AVBN) was proposed for global path planning of mobile robot. In addition, due to the excellent performance of Gravitational search algorithm (GSA) on the constrained optimization problems, Purcaru et al. [25] introduced a new optimal path planning algorithm for mobile robot based on the hybridization between GSA and PSO. By combining the local search ability of GSA with the social thinking ability of PSO, this method can ensure that the generated trajectories without collision with danger zones. However, although these hybrid algorithms can result in good effects, they always cause some new problems, such as heavy computation requirements and high operational complexity.

In this paper, we present a self-adaptive learning particle swarm optimization (SLPSO) algorithm for mobile robot global path planning, which aim to seek an optimal collision-free path in the environment with static obstacles. Our research work presents three important contributions. First, the global path planning problem for mobile robot is formulated as a minimization multi-objective optimization problem, which comprehensively takes into account three objectives of path length, path collision risk and path smoothness, instead of a single objective (path length) or bi-objectives (path length and risk degree) as previously considered in the literatures. Second, a self-adaptive learning particle swarm optimization is proposed to improve the search ability of PSO by introducing four different learning strategies with different advantages for mobile robot path planning problem. Instead of directly using the monotonic learning strategy for updating particles, SLPSO adopts a novel type of self-adaptive learning mechanism based on a given learning rates to adaptively select the most suitable strategy from the four different learning strategies. The purpose of using the self-adaptive learning mechanism is to adaptively adjust the selection ratios of the four learning strategies at different stages of the optimization procedure based on the feedback of previous optimization procedure. Third, the novel boundary violation handling schemes

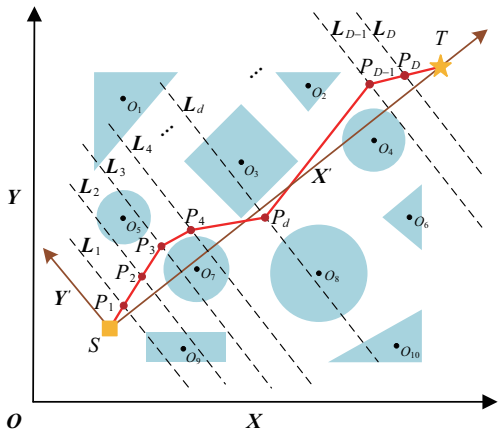

Figure 1 (Color online) The environment model for mobile robot path planning.

are incorporated into SLPSO for restricting the velocity and limiting the search pace of each particle, which contribute to improving the feasibility of the generated paths.

The remainder of this paper is organized as follows. In the next section, the mobile robot path planning problem is mathematically modeled. In Section 3, the standard PSO is introduced and then, the self-adaptive learning PSO is proposed. Subsequently, experiments are conducted and the results are presented and discussed in Section 4. Finally, our concluding remarks are contained in Section 5.

2 Mobile robot path planning problem

2.1 Mathematical modeling of mobile robot environment

In this paper, we model the environment based on the existing model presented in [26]. For convenience of modeling simplification, it is assumed that mobile robot and target are all considered as point mass in the two-dimensional workplace. To avoid collision with mobile robot, all the obstacles are inflated by the inscribed radius of mobile robot. Additionally, we adopt the discrete modeling method to simplify the modeling of arbitrary obstacles, which is similar to the one used in [27].

Figure 1 shows the environment model for mobile robot path planning. In this model, obstacles are presented in the form of entities with different shapes and sizes, inside of which will be blocked with a certain collision risk degree, whereas out of it will be safe for mobile robot. There are three main steps in the path planning. First, we equally divide the distance from the starting point S to the target point T into D+1 segments by D points in the local coordinate system S-$X'Y'$, where D represents the desired number of waypoints for mobile robot path. Second, draw the vertical dashed lines at each segment point, which are denoted as $\boldsymbol{L}_1$, $\boldsymbol{L}_2$, …, $\boldsymbol{L}_D$. Third, randomly select one node in the perpendicular line of $\boldsymbol{L}_i$, for $i = 1, 2, \ldots, D$. If the selected node intersects with any obstacles, the node is infeasible; otherwise, it is feasible. If the node is infeasible, we should to stochastically reselect another node until no collision occurs. In this way, a complete path can be constructed by connecting these discrete waypoints as well as the starting and target points in sequence. Accordingly, the generated path can be given as follows:

$$\boldsymbol{Path} = \{P_1 - S, P_2 - P_1, \ldots, P_D - P_{D-1}, T - P_D\}, \tag{1}$$

where each line segment among the path set does not intersect with obstacles.

In addition, to get arbitrary points on the path in global coordinate system O-XY, it requires the transformation matrix between the local coordinate system and the global coordinate system. The point $x'(d), y'(d)$ in the local coordinate system can be transformed into the point $x(d), y(d)$ in the global coordinate system by the following transformation equation:

$$\begin{bmatrix} x(d) \\ y(d) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x'(d) - x_s \\ y'(d) - y_s \end{bmatrix}, \tag{2}$$

where the (x_s, y_s) represents the coordinate of the starting point S in the global coordinate system, and the x coordinate of each waypoint can be obtained by the formula $x(d) = \frac{|\mathrm{ST}|}{D+1} \cdot d$ in the local coordinate system. The θ is the angle that the global x-axis contrarotates to parallel segment ST.

However, the generated path is usually hard for mobile robot to follow because of some turning points on the generated path, which can result in difficulty with control execution due to mobile robot with kinematic and dynamic constraints. Thus, it is necessary to smooth the path. In this paper, we adopt a classic trajectory smooth strategy called Ferguson Splines described in [8].

2.2 Performance criterions

The objective of path planning is to generate an optimal collision-free path while considering some performance criteria, i.e., path length, time consumption, energy cost. In this study, we pay attention to the path length, collision risk degree and smoothness for quantifying the quality of the generated paths.

For the first performance criterion, path length, the aim of which is to get path as short as possible. Assuming that the start position and the target position are P_0 and P_{D+1}, the path length is defined as

$$L(P) = \sum_{i=0}^{D} \|P_{i+1} - P_i\|, \tag{3}$$

where $\|P_{i+1} - P_i\|$ indicates the Euclidean distance between P_i and P_{i+1}.

For the second performance criterion, path collision risk degree, the objective of which is to evaluate the degree of a path blocked by obstacles. For evaluating the collision risk degree, the two-dimensional Gaussian-like model [28] is adopted to establish the collision risk function, which can be defined as

$$R(\boldsymbol{X}_{\mathrm{rob}}, \boldsymbol{X}_{\mathrm{obs}}) = \sum_{j=1}^{n} R_{\mathrm{rob,obsj}}(\boldsymbol{X}_{\mathrm{rob}}, \boldsymbol{X}_{\mathrm{obsj}}), \tag{4}$$

where $R_{\mathrm{rob,obsj}}(\boldsymbol{X}_{\mathrm{rob}}, \boldsymbol{X}_{\mathrm{obsj}})$ denotes collision risk degree generated by robot and obstacle j, and n is the total number of obstacles. $\boldsymbol{X}_{\mathrm{rob}}$ and $\boldsymbol{X}_{\mathrm{obsj}}$ are the position vector of mobile robot and obstacle j, respectively. The collision risk degree between robot and obstacle j is mathematically expressed as

$$R(\boldsymbol{X}_{\mathrm{rob}}, \boldsymbol{X}_{\mathrm{obsj}}) = \begin{cases} \mathrm{e}^{-\frac{1}{2}\left(\frac{\left\|\boldsymbol{x}_{\mathrm{rob}} - \boldsymbol{x}_{\mathrm{obsj}}\right\|^2}{\rho^2}\right)^C}, & \|\boldsymbol{X}_{\mathrm{rob}} - \boldsymbol{X}_{\mathrm{obsj}}\| \leqslant R_j, \\ 0, & \|\boldsymbol{X}_{\mathrm{rob}} - \boldsymbol{X}_{\mathrm{obsj}}\| > R_j, \end{cases} \tag{5}$$

where R_j is the maximum influence range of obstacle j. To ensure the repulsive force of obstacle j is regional, R_j is limited to $1.5R_{\mathrm{rob}}$, where R_{rob} is the inscribed radius of mobile robot. The variance ρ controls the affecting area of obstacle correspond to its physical size, and the positive integer C determines the effective rang of obstacle. Generally, the parameters of ρ and C are usually set to the values of 3 and 1 respectively.

For the third performance criterion, path smoothness, the purpose of which is to measure how much snaky is the obtained path. To evaluate the path smoothness, we introduce the deflection angle formed by any three neighboring waypoints. The path smoothness can be calculated by using the following formula:

$$S(P) = \sum_{i=1}^{D} \alpha_i = \sum_{i=1}^{D} \arccos\left(\frac{(P_i - P_{i-1}) \cdot (P_{i+1} - P_i)}{|P_i - P_{i-1}| \times |P_{i+1} - P_i| \times 180}\right), \tag{6}$$

where α_i refers to the value of the i-th deflection angle of the generated path (measured in radians in the range from 0 to π). $(P_i - P_{i-1}) \cdot (P_{i+1} - P_i)$ indicates the inner product between vectors of $P_i - P_{i-1}$ and $P_{i+1} - P_i$, while $|P_i - P_{i-1}|$ denotes the vector norm.

2.3 Problem formulation

In practical applications, some performance merits for mobile robot path are in inconsistency. For example, minimizing path length will result in increasing collision risk degree, whereas minimizing collision risk degree will lead to an increase in path length. Thus, the performance merits should be well balanced to achieve better overall performance.

In this paper, we use a common way called simple additive weighted method to address this problem. The overall objective function for optimizing mobile robot path is defined as the weighted combination of path length, path collision risk degree and path smoothness, which can be described as

$$J(P, \boldsymbol{X}_{\rm rob}, \boldsymbol{X}_{\rm obs}) = w_1 \times L(P) + w_2 \times R(\boldsymbol{X}_{\rm rob}, \boldsymbol{X}_{\rm obs}) + w_3 \times S(P), \tag{7}$$

where w_1, w_2 and w_3 are weight coefficients between 0 and 1, which give the designer certain flexibility to dispose relations among path length, path collision risk degree and path smoothness. Generally, the parameters set for w_1, w_2 and w_3 are based on mobile robot working environment and practical experience to take the appropriate values. In our experiments, these parameters are set to $w_1 = 0.6$, $w_2 = 0.3$ and $w_3 = 0.1$.

Accordingly, the mobile robot path planning problem can be formulated as the following optimization problem of minimizing the overall objective function with constraints, which is expressed as

$$\min\ J(P, \boldsymbol{X}_{\rm rob}, \boldsymbol{X}_{\rm obs}) \quad \text{s.t.} \quad P, \boldsymbol{X}_{\rm rob} \notin S, \tag{8}$$

where S is the set of obstacles.

3 Improvements on PSO for mobile robot path planning

3.1 Principle of the standard PSO

PSO is generally known as a population-based stochastic optimization algorithm, which is motivated from the social behavior of bird flocking and fish schooling in nature [29]. In PSO, each particle has memory allowing it to keep track of the previous best positions: personal best position $\boldsymbol{X}_{\rm pbest}$, and global best position $\boldsymbol{X}_{\rm gbest}$ [30]. It is assumed that there are N particles in the D-dimensional search space, the velocity and position of the i-th particle at iteration k are $\boldsymbol{X}_i^k = (x_{i,1}^k, \ldots, x_{i,d}^k, \ldots, x_{i,D}^k)$ and $\boldsymbol{V}_i^k = (v_{i,1}^k, \ldots, v_{i,d}^k, \ldots, v_{i,D}^k)$ respectively, for $i = 1, 2, \ldots, N$. The position and velocity of i-th particle at iteration k+1 will be updated as follows:

$$\boldsymbol{V}_i^{k+1} = w_4^k \times \boldsymbol{V}_i^k + \eta_1 \times {\rm rand}_1 \times (\boldsymbol{X}_{\rm ipbest}^k - \boldsymbol{X}_i^k) + \eta_2 \times {\rm rand}_2 \times (\boldsymbol{X}_{\rm gbest}^k - \boldsymbol{X}_i^k), \tag{9}$$

$$\boldsymbol{X}_i^{k+1} = \boldsymbol{X}_i^k + \boldsymbol{V}_i^{k+1}, \tag{10}$$

where ${\rm rand}_1$ and ${\rm rand}_2$ are random variables generated uniformly among $[0, 1]$, and η_1 and η_2 are the acceleration coefficients. w_4^k is the inertia weight at k iteration, which decreases linearly from $w_{\max}$ to $w_{\min}$ during iterations. The dynamic inertia weight w_4^k can be calculated as

$$w_4^k = w_{\max} - \frac{k}{{\rm Iter}_{\max}} (w_{\max} - w_{\min}), \tag{11}$$

where ${\rm Iter}_{\max}$ is the predefined maximum number of iterations, $w_{\max}$ and $w_{\min}$ are the maximal and minimal inertia weights, which are usually set to 0.9 and 0.4 respectively.

For each iteration, PSO evaluates the fitness of all particles to find both the personal best positions and the global best position, and then update them according to (9) and (10).

3.2 Self-adaptive learning PSO

There are two models in PSO, called Gbest and Pbest, which the Gbest model focuses more on exploration while the Pbest model biases more toward exploitation [31]. In the Gbest model, each particle learns information from $\boldsymbol{X}_{\text{gbest}}$. On the contrary, in the Pbest model, each particle learns information from $\boldsymbol{X}_{\text{pbest}}$. However, learning information from $\boldsymbol{X}_{\text{gbest}}$ can make particles more vulnerable to the attraction of local optima than that from $\boldsymbol{X}_{\text{pbest}}$, whereas learning information from $\boldsymbol{X}_{\text{pbest}}$ can encourage slower convergence speed than that from $\boldsymbol{X}_{\text{gbest}}$ [32]. Especially when tackling the complex multi-objective optimization problems characterized with multimodality, the most PSO algorithms fail to effectively approach the optimal solutions. The reason for this is that each particle only adopts a single search strategy that simultaneously learning information from both $\boldsymbol{X}_{\text{pbest}}$ and $\boldsymbol{X}_{\text{gbest}}$ to update its velocity and position, which may make the algorithms suffer from the demerits of both models [33]. To overcome this problem, it is essential to balance the performance of the Pbest and Gbest models.

In this subsection, a self-adaptive learning PSO with alterative learning strategies, which can achieve high convergence speed and avoid being trapped by local optima, is developed to serve the purpose of exploration and exploitation. In SLPSO, instead of using the monotonic learning strategy for update the current learning information directly, each particle adaptively chooses the most suitable search behavior from a set of different learning strategies based on its selection ratio of corresponding operator. The fundamental idea of calculating the selection ratio of each operator is analogous to that of [31], but the implementation is different. Besides, to reduce the complexity of computation, we use two different key parameters: update frequency (U_{f}) and learning information for $\boldsymbol{X}_{\text{gbest}}$. In our proposed method, the update frequency is replaced by a fixed frequency, and $\boldsymbol{X}_{\text{gbest}}$ learns useful information from the population rather than that from some dimensions of one particle with a certain learning probability.

3.2.1 *Self-adaptive learning strategies*

Different from the standard PSO, SLPSO has four types of learning strategies, namely, exploitation, convergence, jumping out of local optima, and exploration, which can enable particles to independently deal with various situations during the search phase. Corresponding to the learning strategies, we also define four operators, each of which is assigned to a selection ratio.

The four operators of i-th particle at k iteration are described as follows:

- Operator a: $\boldsymbol{V}_i^k = \omega \times \boldsymbol{V}_i^k + \eta_3 \times \text{rand}_i^k \times (\boldsymbol{X}_{\text{ipbest}}^k - \boldsymbol{X}_i^k)$,
- Operator b: $\boldsymbol{V}_i^k = \omega \times \boldsymbol{V}_i^k + \eta_3 \times \text{rand}_i^k \times (\boldsymbol{X}_{\text{ipbest\_nearest}}^k - \boldsymbol{X}_i^k)$,
- Operator c: $\boldsymbol{X}_i^k = \boldsymbol{X}_i^k + \boldsymbol{V}_{\text{average}}^k \times N(0,1)$, and
- Operator d: $\boldsymbol{V}_i^k = \omega \times \boldsymbol{V}_i^k + \eta_3 \times \text{rand}_i^k \times (\boldsymbol{X}_{\text{gbest}}^k - \boldsymbol{X}_i^k)$,

where $\boldsymbol{X}_{\text{ipbest\_nearest}}^k$ and $\boldsymbol{V}_{\text{average}}^k$ are the personal best position of the nearest particle to particle i and the average velocity of all particles at iteration k, respectively. $N(0,1)$ represents a random number drawn from uniform distribution between 0 and 1. ω and η_3 are weight coefficient and acceleration coefficient respectively, which can balance the effect of exploration and exploitation during the search process. Generally, the values of ω and η_3 are set based on the designer's practical experience.

In SLPSO, besides learning from the personal best position (operator a) and learning from the global best position (operator d), each particle possesses the other two different choices to adjust its search behavior according to the surrounding environment where it is. Learning from the closest neighbor (operator b) gives a particle the ability to explore the region of local minimum around itself. With this learning strategy, the particles that located at the local minimum will come closer to that region, and gradually generate a local cluster around that local minimum. This strategy can help swarm find more local minimum rather than one optimum as the standard PSO does [34]. Once there is a more promising region nearby without particles converge it or particles converge on a local optimum, learning from a random position (operator c) can help particles jump to that promising region or jump out of the local optimum.

3.2.2 *Self-adaptive learning mechanism*

In SLPSO, the ultimate purpose of utilizing the self-adaptive learning strategies is to well guide its search behavior of exploration and exploitation. To this end, a self-adaptive learning framework using the aforementioned four learning operators is introduced to enable each particle to automatically select the most suitable learning strategy during the optimization process. For each learning strategy, we assign a selection ratio to determine the probability that this learning strategy be adopted to update the current particle. The adjustment of a learning strategy is carried out progressively by specifying the maximum decrement or increment between two successive iterations according to the selection ratio of corresponding operator, which is determined by its previous performances. In other words, the operator that results in higher fitness value will make its selection ratio increased, whereas the operator that leads to lower fitness value will have its selection ratio decreased. Gradually, the most suitable operator will be selected automatically to adjust the learning strategy for each particle in different situations.

For each particle, the selection ratios of all operators are equally initialized to 0.25. Within the framework of self-adaptive learning mechanism, the selection ratio of the operator t of particle i at iteration k+1 will be updated by means of the following modified equation:

$$s_i^{k+1}(t)=\frac{r_i^k(t)}{\sum_{j=1}^{4} r_i^k(j)}(1-4\times s_{\min})+s_{\min}, \tag{12}$$

where $s_{\min}$ is the minimum selection ratio of operator t for particle i, which is set to 0.01. $r_i^k(t)$ is the reward value of operator t for particle i at iteration k, which can be expressed as

$$r_i^k(t)=\frac{\mathrm{pr}_i^k(t)}{\sum_{j=1}^{4}\mathrm{pr}_i^k(j)}\times\beta+\frac{l_i^k(t)}{L_i^k(t)}(1-\beta)+c_i^k(t)\times s_i^k(t), \tag{13}$$

where $\mathrm{pr}_i^k(t)$ is the progress value of operator t for particle i at iteration k. $l_i^k(t)$ is the successful learning times of operator t for particle i, whose objective function value at iteration k is better than that after applying operator t at iteration $k-1$. $L_i^k(t)$ is the total times of operator t is selected by particle i. β is a random weight within the range $[0,1]$. $c_i^k(t)$ is the penalty factor for operator t of particle i at iteration k, which is defined as

$$c_i^k(t)=\begin{cases}0.9, & \text{if } l_i^k(t)=0 \text{ and } s_i^k(t)=\max_{j=1}^{4}(s_i^k(j)),\\ 1, & \text{otherwise.}\end{cases} \tag{14}$$

The progress value $\mathrm{pr}_i^k(t)$ of operator t for particle i at iteration k is calculated as

$$\mathrm{pr}_i^k(t)=\begin{cases}\left|f\left(\boldsymbol{X}_i^k(t)\right)-f\left(\boldsymbol{X}_i^{k-1}(t)\right)\right|, & \text{if operator } t \text{ chosen by } \boldsymbol{X}_i^k(t) \text{ and } f(\boldsymbol{X}_i^k(t))\\ & \text{is better than } f(\boldsymbol{X}_i^{k-1}(t)),\\ 0, & \text{otherwise.}\end{cases} \tag{15}$$

According to the aforementioned definitions, we can see that each particle always has one operator with the highest selection ratio to be chosen. However, when the environment changes, the current operator of a particle may no longer be the most suitable choice for it. If this case occurs, the selection ratio of the current operator will decrease according to (12), whereas the selection ratios of other operators will increase. Gradually, a new most suitable operator will be automatically selected based on its relatively better performance during the updating period for each particle. In addition, it should be noted that the selection ratios of four operators are updated at fixed frequency U_{f} rather than updated at each iteration. Based on the analysis of the adaptive learning mechanism, we can see that SLPSO is able to choose the optimal learning strategy for each particle in accordance with the environmental changes.

3.2.3 *Boundary violations handling*

In SLPSO, each particle searches a new position according to its current position and velocity. However, if the current position of a particle is far away from $\boldsymbol{X}_{\mathrm{pbest}}$ and $\boldsymbol{X}_{\mathrm{gbest}}$, its velocity may become too high,

and this will further lead to larger displacement. As a result, the particle will move beyond the search boundary. Consequently, it is essential to restrict the values of velocity and position for each particle.

To solve the uncontrolled increase of magnitude of velocity, we constrain the velocity for i-th particle at k iteration by using the following operation:

$$\left\|\boldsymbol{V}_i^k\right\| \leqslant \boldsymbol{V}_{\max}, \tag{16}$$

where $\boldsymbol{V}_{\max}$ is the predefined maximum velocity, which is set to half of the search domain in this paper.

Furthermore, we also present the reflecting method to solve the position violation problem (out of the search space) for each particle. This method uses the reflecting position with a clamping at desirable position to prevent the particle taking extremely large step from its current position to the next position. For i-th particle, the position is adjusted according to the reflecting operator by following equation:

$$\boldsymbol{X}_i'^k = \begin{cases} 2 \times u - \boldsymbol{X}_i^k, & \text{if } \boldsymbol{X}_i^k > u, \\ 2 \times l - \boldsymbol{X}_i^k, & \text{if } \boldsymbol{X}_i^k < l, \\ \boldsymbol{X}_i^k, & \text{else}, \end{cases} \tag{17}$$

where u and l are the upper and lower boundaries of the search space, respectively.

According to the operation (16) and (17), SLPSO will take restricted velocity values for all the particles in the interval $[-\boldsymbol{V}_{\max}, \boldsymbol{V}_{\max}]$, and bring the infeasible particles to re-enter the search space from an end which is opposite to where they left the search space.

The pseudo-code of the SLPSO algorithm is given in Algorithm 1.

Algorithm 1 The SLPSO algorithm

```
Initialize the parameters of SLPSO. Set the population size N, generate the initial particles with position and velocity, initialize the update frequency (U_f), and set the generation counter k = 1.
while k < Iter_max do
    Calculate the inertia weight value by using (11).
    for i = 1 to N do
        if k%U_f == 0 then
            Update the selection ratio of each learning operator for particle i.
        end if
        Select the most suitable operator according to its selection ratio for particle i.
        Hand the boundary violation of velocity and position for particle i by using (16) and (17).
        Evaluate the objective function of particle i according to (7).
        Calculate the reward value, progress value and selection ratio of each operator of particle i for the iteration k+1 according to (12)–(15).
        if the updated particle i is better than its X_pbest^k then
            Update X_pbest^k.
        end if
        if the updated particle i is better than X_gbest^k then
            Update X_gbest^k.
        end if
    end for
    k=k+1.
end while
```

3.3 Complexity of self-adaptive learning PSO

According to the aforementioned description, the computational complexity of the overall SLPSO algorithm is $O(DTN)$, where D represents the dimensions of the path planning problem to be solved, T denotes the iterations required to obtain the optimized solution, and N is the number of the particles population. Compared with PSO, SLPSO needs to perform some extra computation on calculating and updating the selection ratios. However, the computational complexity is $O(N)$ at each iteration. Hence, the overall computational complexity of SLPSO is not very high in comparison with PSO.

3.4 Implementation of self-adaptive learning PSO for mobile robot path planning

One key issue of applying SLPSO to mobile robot path planning is how to initialize the position for each particle. In the SLPSO-based path planning method, the dimensions of the created particles are determined by the number of waypoints (excluding starting point S and target point T). In this subsection, we assumed that there are N particles and each particle is D-dimensional. To realize the path planning algorithm, the d-th dimension of each particle is treated as the d-th waypoint (for $d = 1, 2, \ldots, D.$), i.e., each dimension of a particle maps one waypoint and each particle represents a candidate path. Consequently, the larger number of waypoints, the larger would be the number of dimensions of particle, and therefore the better accuracy of the optimal solution would be for path planning problem.

Since the X'-coordinate of each dimension of every particle has been already known in the given local coordinate system S-$X'Y'$, the path planning algorithm only seeks the Y'-coordinate of each dimension of every particle, which are adjusted to change position upon vertical dashed lines $\boldsymbol{L}_i$, for $i = 1, 2, \ldots, D$, as shown in Figure 1. In each iteration, SLPSO proceeds with fitness evaluation of all particles to minimize the overall objective values, and then update the position and velocity of each particle according to the four types of learning strategies, whose selection ratios are adjusted according to (12)–(15). Accordingly, each particle constantly undergo path refinement so that the length, collision risk and smoothness in the candidate paths are rectified and the quality of the paths is improved.

The detailed implementation procedure of SLPSO for mobile robot path planning are shown in Figure 2. It should note that the stopping criterias contain the specified maximum number of iterations and the minimum improvement threshold (1% during 10 iterations).

4 Experimental results

In this section, a series of simulation and real-world experiments have been performed to verify the feasibility and effectiveness of SLPSO for solving mobile robot path planning problem.

4.1 Experimental setup for mobile robot path planning

In this paper, all the algorithms are implemented in C++, and experiments are performed on an PC with Intel (R) Core (TM) i7 2.80 GHz processor, 8.0 GB RAM memory and 64-bit GNU/Linux operating system.

In mobile robot path planning, we use the standard PSO [16] and GA [35] as the baseline, due to the fact that GA is similar to PSO in the sense that they are both population-based stochastic optimization algorithm and they both depend on information sharing among their population members to improve their searching ability using a combination of probabilistic and deterministic rules. To realize the PSO-based path planning algorithm, we use the same way to initialize the position for each particle as the SLPSO-based path planning method. For PSO, the parameters are set as follows: $\eta_1 = 2$ and $\eta_2 = 2$. For SLPSO, the parameters are set as follows: $\omega = 0.73$, $\eta_3 = 1.496$ and $U_{\mathrm{f}} = 3$. For a fair comparison, the population size of PSO and SLPSO are both set to 30 ($P_p = 30$), as recommended in [36].

In the GA-based path planning, we adopt the decimal coding method proposed in [35] to encode waypoints into a chromosome. Each chromosome consists of D genes, which are decimal numbers. In our case, each gene of a chromosome is viewed as a waypoint and each chromosome denotes a candidate path, i.e., a chromosome represents a candidate path as a sequence of genes. Therefore, the number of the genes of a chromosome equals the number of waypoints (excluding starting point S and target point T). For a fair comparison, the population size that will be used for GA runs is same as the population size in PSO and SLPSO runs, and is fixed at 30 chromosomes in GA population. Furthermore, the objective function in PSO and SLPSO is also applied for GA. For GA, the parameters are set as follows: population size $P_{\mathrm{s}} = 30$, crossover rate $P_{\mathrm{c}} = 0.9$, mutational rate $P_{\mathrm{m}} = 0.08$.

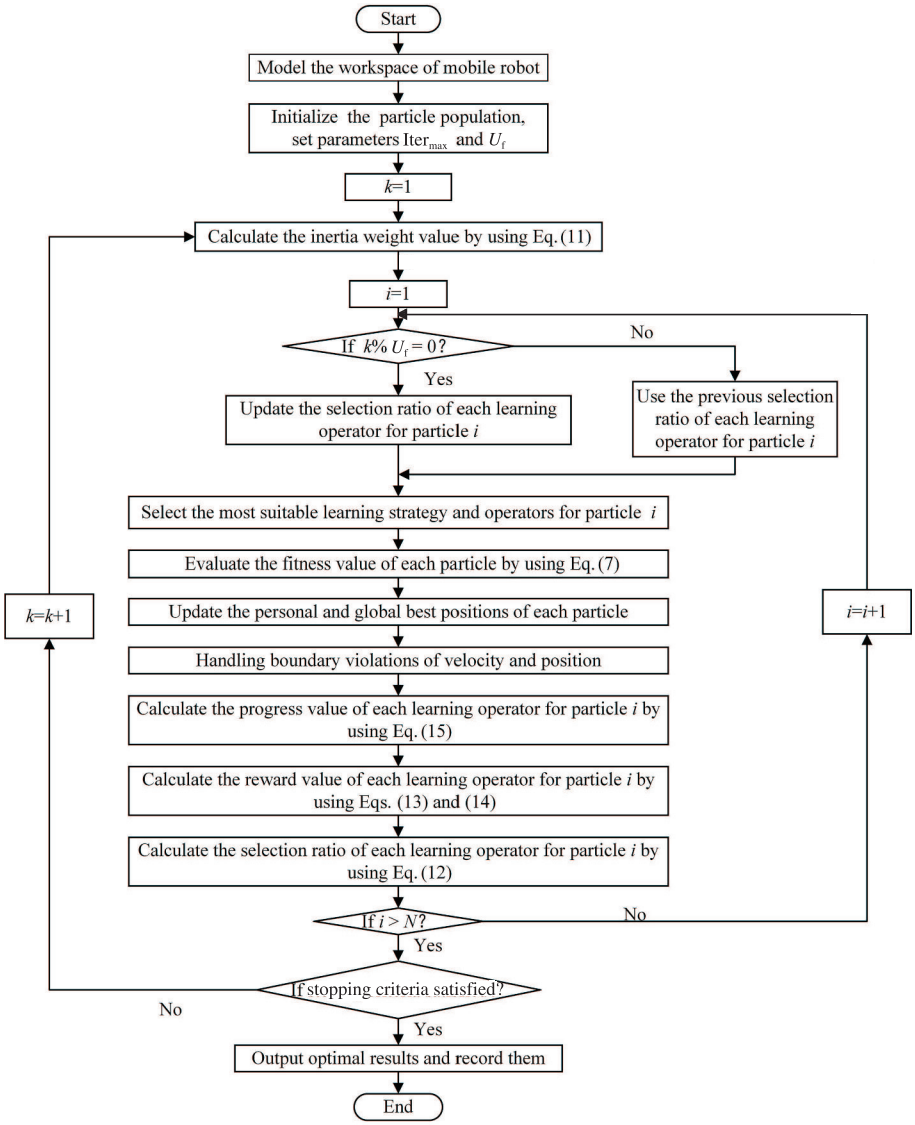

Figure 2 The flowchart of SLPSO for mobile robot path planning.

4.2 Benchmark test and comparison

In order to comprehensively evaluate the performance of SLPSO, we conduct function optimization experiments on the CEC-2013 benchmark minimization problems [37]. The CEC-2013 testbed comprises 28 benchmark functions with different characteristics, which can be grouped into three categories: unimodal functions (f_1–f_5), multimodal functions (f_6–f_{20}) and composition functions (f_{21}–f_{28}). The search space for all the benchmark functions is $[-100, 100]$, and the maximum iteration number is $10000 \times D$, where D is dimension.

In this subsection, we compare the performance of SLPSO to the classical PSO [16], GA [35], DE [38], ACO [39] and APSO [40]. For all the algorithms, we use the same population size of 100, and other related parameters use the default values. For all the test functions, each algorithm carries out 101 independent runs with dimension D=10. The experiment results of all the algorithms with respect to their mean value and standard deviation on the 28 test functions are shown in Table 1, in which the best results are typed in bold.

As can be seen in Table 1, for the unimodal benchmark functions, all the algorithms can find the global optimum solution on f_1 and f_5. For the rest of unimodal functions, SLPSO obtains the best solutions while GA and DE provide the worst solutions. For the multi-modal functions, it is difficult to find the

Table 1 Performance comparison of GA, DE, ACO, PSO, APSO and SLPSO on the 28 benchmark functions (D=10)[a)]

Test function		GA	DE	ACO	PSO	APSO	SLPSO
$f_1(x)$	Mean	**−1.400E+03**	**−1.400E+03**	**−1.400E+03**	**−1.400E+03**	**−1.400E+03**	**−1.400E+03**
	Std. Dev.	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
$f_2(x)$	Mean	1.058E+05	1.870E+05	6.546E+04	7.534E+04	3.504E+04	**2.021E+04**
	Std. Dev.	7.250E+04	6.670E+04	1.091E+04	3.549E+04	1.003E+04	**1.073E+04**
$f_3(x)$	Mean	9.445E+04	1.933E+05	2.358E+04	2.438E+03	−9.579E+02	**−1.088E+03**
	Std. Dev.	4.794E+04	1.137E+05	2.293E+04	2.320E+03	1.460E+02	**1.070E+02**
$f_4(x)$	Mean	−1.088E+03	−1.089E+03	−1.096E+03	−1.097E+03	−1.092E+03	**−1.099E+03**
	Std. Dev.	6.435E+00	4.985E+00	3.040E+00	2.050E+00	3.931E+00	**9.893E−01**
$f_5(x)$	Mean	**−1.000E+03**	**−1.000E+03**	**−1.000E+03**	**−1.000E+03**	**−1.000E+03**	**−1.000E+03**
	Std. Dev.	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
$f_6(x)$	Mean	−8.902E+02	−8.902E+02	−8.986E+02	−8.996E+02	−8.986E+02	**−8.999E+02**
	Std. Dev.	6.174E+00	7.224E+00	3.224E−01	2.431E−01	1.207E+00	**7.224E−02**
$f_7(x)$	Mean	−7.890E+02	−7.925E+02	−7.979E+02	−7.978E+02	−7.983E+02	**−7.984E+02**
	Std. Dev.	5.124E+00	6.404E+00	1.548E+00	1.127E+00	4.230E−01	**7.652E−01**
$f_8(x)$	Mean	−6.797E+02	−6.797E+02	−6.797E+02	−6.797E+02	**−6.799E+02**	−6.799E+02
	Std. Dev.	6.645E−01	7.908E−01	2.192E−01	6.772E−01	**7.953E−02**	9.750E−02
$f_9(x)$	Mean	−5.954E+02	−5.957E+02	−5.967E+02	−5.986E+02	−5.988E+02	**−5.993E+02**
	Std. Dev.	2.896E+00	3.742E+00	2.354E+00	1.112E+00	1.099E+00	**6.444E−01**
$f_{10}(x)$	Mean	−4.988E+02	−4.986E+02	−4.995E+02	−4.994E+02	−4.997E+02	**−4.998E+02**
	Std. Dev.	1.082E+00	1.148E+00	4.528E−01	5.713E−01	1.365E−01	**1.076E−01**
$f_{11}(x)$	Mean	−3.990E+02	−3.990E+02	**−4.000E+02**	**−4.000E+02**	**−4.000E+02**	**−4.000E+02**
	Std. Dev.	5.673E−01	4.893E−01	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
$f_{12}(x)$	Mean	−2.791E+02	−2.741E+02	−2.881E+02	−2.901E+02	−2.930E+02	**−2.970E+02**
	Std. Dev.	1.287E+01	1.648E+01	7.256E+00	6.565E+00	4.876E+00	**2.781E+00**
$f_{13}(x)$	Mean	−1.799E+02	−1.718E+02	−1.823E+02	−1.814E+02	−1.922E+02	**−1.980E+02**
	Std. Dev.	1.237E+01	1.653E+01	6.783E+00	5.981E+00	4.325E+00	**1.911E+00**
$f_{14}(x)$	Mean	1.476E+02	1.127E+02	−2.673E+01	−5.660E+01	−8.482E+01	**−9.646E+01**
	Std. Dev.	1.743E+02	1.013E+02	9.994E+01	3.657E+01	1.164E+01	**3.206E+00**
$f_{15}(x)$	Mean	8.286E+02	8.817E+02	5.737E+02	6.063E+02	4.089E+02	**3.933E+02**
	Std. Dev.	3.928E+02	1.968E+02	1.525E+02	1.872E+02	1.068E+02	**2.368E+02**
$f_{16}(x)$	Mean	2.010E+02	2.010E+02	2.008E+02	**2.004E+02**	2.007E+02	2.006E+02
	Std. Dev.	3.453E−01	4.196E−01	2.241E−01	**1.044E−01**	3.650E−01	2.005E−01
$f_{17}(x)$	Mean	3.118E+02	3.118E+02	3.115E+02	3.103E+02	3.104E+02	**3.089E+02**
	Std. Dev.	1.054E+01	1.024E+01	1.057E+01	6.562E−01	1.627E−01	**5.873E+00**
$f_{18}(x)$	Mean	4.301E+02	4.353E+02	4.299E+02	4.178E+02	4.085E+02	**4.065E+02**
	Std. Dev.	2.014E+01	1.497E+01	1.893E+01	4.534E+00	8.786E+00	**5.287E+00**
$f_{19}(x)$	Mean	5.007E+02	5.007E+02	5.006E+02	5.004E+02	**5.001E+02**	5.002E+02
	Std. Dev.	3.092E−01	2.670E−01	2.143E−01	2.886E−01	**9.650E−02**	6.490E−02
$f_{20}(x)$	Mean	6.035E+02	6.035E+02	6.032E+02	6.034E+02	6.010E+02	**6.007E+02**
	Std. Dev.	1.598E+00	1.201E+00	1.881E+00	4.194E−01	1.596E−01	**1.895E−01**
$f_{21}(x)$	Mean	1.100E+03	1.100E+03	1.100E+03	1.100E+03	9.000E+02	**8.000E+02**
	Std. Dev.	1.414E+01	1.732E+01	0.000E+00	0.000E+00	0.000E+00	**0.000E+00**
$f_{22}(x)$	Mean	1.143E+03	1.156E+03	1.027E+03	1.129E+03	8.206E+02	**8.186E+02**
	Std. Dev.	1.751E+03	1.842E+03	1.630E+03	1.515E+03	1.142E+03	**1.072E+03**
$f_{23}(x)$	Mean	1.751E+03	1.842E+03	1.630E+03	1.515E+03	1.142E+03	**1.072E+03**
	Std. Dev.	5.929E+02	4.958E+02	3.631E+02	3.596E+02	8.774E+01	**5.451E+01**
$f_{24}(x)$	Mean	1.215E+03	1.211E+03	1.210E+03	1.214E+03	1.206E+03	**1.201E+03**
	Std. Dev.	1.413E+01	2.822E+01	1.456E+01	1.370E+01	4.654E+00	**9.401E+00**
$f_{25}(x)$	Mean	1.318E+03	1.315E+03	1.316E+03	1.312E+03	1.306E+03	**1.301E+03**
	Std. Dev.	7.124E+00	9.752E+00	9.832E+00	5.943E+01	4.831E+00	**5.239E+002**
$f_{26}(x)$	Mean	1.510E+03	1.511E+03	1.400E+03	1.400E+03	1.314E+03	**1.307E+03**
	Std. Dev.	2.011E+02	1.096E+02	1.973E+02	9.513E+01	9.300E+01	**8.615E+01**
$f_{27}(x)$	Mean	1.846E+03	1.850E+03	1.841E+03	1.636E+03	1.610E+03	**1.601E+03**
	Std. Dev.	2.115E+02	2.941E+02	1.836E+03	1.985E+02	8.800E+00	**1.384E+01**
$f_{28}(x)$	Mean	**1.700E+03**	**1.700E+03**	**1.700E+03**	**1.700E+03**	**1.700E+03**	**1.700E+03**
	Std. Dev.	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**

a) "Std. Dev." stands for the standard deviation.

true global optima due to the existence of many local minima. As for function f_{11}, ACO, PSO, APSO and SLPSO perform the best and can find the global optimum. For solving function f_8 and f_{19}, only

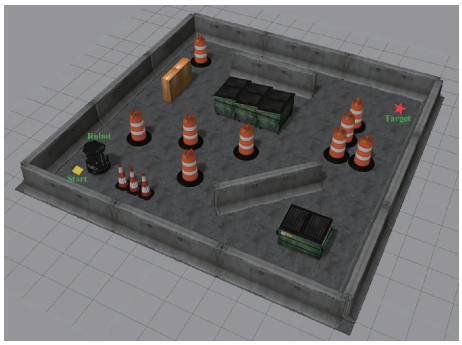

Figure 3 (Color online) The simulation environment for mobile robot path planning in Gazebo.

APSO perform the best, but it does not show any distinct advantages compared to SLPSO. For the function f_{16}, PSO achieved the highest accuracy solution and SLPSO obtains the second best solutions. For the rest of multi-modal functions, SLPSO performs significantly better than other algorithms. The satisfactory solutions and the low standard deviations achieved by SLPSO present competitive results compared with other algorithms, which exposes the powerful search ability and reliability of SLPSO. For the composition functions, all the algorithms can find the global optimum solution on f_{28}, and SLPSO achieves the best solutions on rest of composition functions.

Overall, according to the above experiment results on the 28 benchmark problems, it can be concluded that applying adaptive learning mechanism with the four learning strategies is useful for SLPSO to effectively handle the unimodal, multimodal and composition problems.

4.3 Performance comparison in simulation environment

In this subsection, the comparative experiments were carried out under a simulation scenario. We based our path planning implementation within the ROS navigation framework [41], and simulated the path planning in Player/Stage (Gazebo) [42]. Figure 3 shows the simulation environment, where the obstacles with different sizes and shapes are generated to represent a complex environment. The workspace is 12 m×12 m and the minimum gap between obstacles is 0.5 m. In the simulation experiments, mobile robot starts from the point of (1, 1) and is required to arrive the target at (11, 11). In Figures 4–6 and 9, the "PSO" represents the standard PSO based planner, the "SLPSO" denotes the proposed PSO with adaptive learning mechanism based planner, the "GA" indicates GA based planner.

In SLPSO, the number of waypoints of the optimal path depends crucially on the dimension D of the created particles. Therefore, this subsection investigates the effect of different dimension D on the performance of mobile robot path planning. Furthermore, to prove the performance of SLPSO against PSO and GA, all the methods are carried out with 50 independent experiments with maximum iteration number of 150 for each parameter D, the value of which are varied from 5 to 30.

Figure 4 graphically compares the statistical results, path length, path collision risk degree, path smoothness and overall path cost, under different dimension D settings. As shown in Figure 4(a), the path length result of SLPSO is the best for each dimension D as the resulting path length is shorter than that of GA and PSO, and PSO performs better than GA. As indicated in Figure 4(b), SLPSO behaves much better than the other methods in providing the smallest path collision risk degree, which means that SLPSO can find the safest path that avoids obstacle areas. As displayed in Figure 4(c), with the increase of dimension D, the path smoothness of all the algorithms increases. The reason is that large D can easily lead to the great sum of the deflection angles. For each dimension D, SLPSO outperforms PSO and GA because of the generated path with the least sum of deflection angles, and GA gives the snaky path with the largest sum of deflection angles. As reported in Figure 4(d), SLPSO performs best due to it consumes much less overall path cost than the other methods in all the cases, which means that SLPSO can find a more optimal path than that of PSO and GA. With the dimension D increasing, the overall path cost of the three methods decrease, and all the methods obtained the best results using

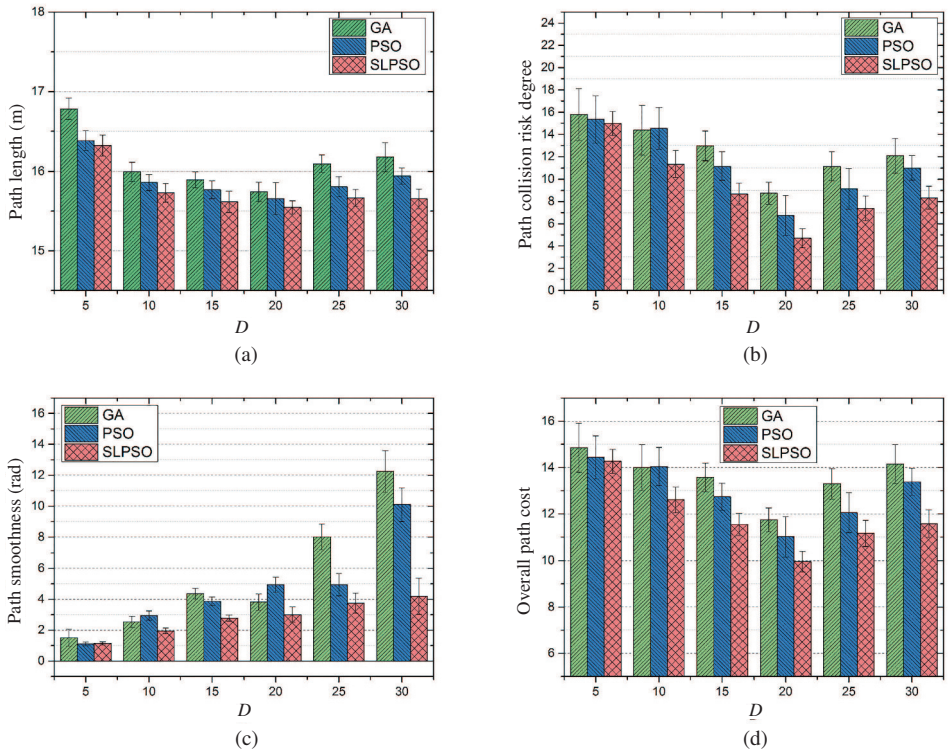

Figure 4 (Color online) The path planning results of mobile robot using GA, PSO and SLPSO for different D. (a) Path length; (b) path collision risk degree; (c) path smoothness; (d) overall path cost.

a D-value of around 20. However, with the value of D increases continuously, the performances of all the methods deteriorate rapidly. This can be explained by the fact that a larger D makes increase of the computation complexity of all the approaches. Furthermore, the results also illustrate that with the increase of computation complexity the improvement of SLPSO over PSO and GA becomes more evident.

Figure 5 shows the simulation results of the three approaches for mobile robot path planning. We can find that the path is composed of D equal parts, and all three methods can successfully produce feasible paths that avoid all obstacles, which implies that they can work well in the mobile robot path planning.

Figure 6 shows the evolutionary process of the overall path cost on different D. As can be seen, SLPSO has a higher convergence precision and faster convergence speed in all of these tests. Therefore, SLPSO has a strong ability to find the optimal solutions.

Meanwhile, Table 2 tabulates the statistical results of path planning time for each dimension D, where the best results are identified with bold font. It is seen from this table that SLPSO performs best for each dimension D as the resulting path planning time is less than that of GA and PSO, which means that SLPSO can find an optimal path faster. Besides, all the methods obtain the best results using a D-value of 20.

From the above simulation results, it can be concluded that on mobile robot path planning problem, SLPSO performs better than PSO and GA with the advantages of providing the short path, possessing the small path collision risk degree, achieving the smooth path and consuming the less path planning time. What is more, the results also illustrate that SLPSO is significantly superior to PSO and GA independent of the choice of parameter D.

4.4 Performance comparison in practical environment

Compared to the simulation experiments, path planning in practical environment produces more challenges. In this subsection, we present an experimental study through a real world implementation on mobile robot. To perform these experiments, we used the TurtleBot2 robot equipped with LiDAR,

Figure 5 (Color online) The path planning results of mobile robot using GA, PSO and SLPSO for different D. (a) $D = 5$; (b) $D = 10$; (c) $D = 15$; (d) $D = 20$; (e) $D = 25$; (f) $D = 30$.

odometer and gyroscope, as shown in Figure 7. It is noted that the path planning system has been implemented using ROS, and the robot has means for solving the underlying simultaneous localization and mapping (SLAM) problem given laser and odometer data. The mapping process is done by using the GMapping algorithm and the localization process is done by using the AMCL pack, which are open source algorithms. In addition, to visualizing the map published from Turtlebot, the RVIZ tool is also used.

Figure 8 shows the practical environment for mobile robot path planning, where the workspace is 12 m×12 m with obstacles and the minimum gap between obstacles is 0.6 m. For all the experiments, the dimension D is set to 20, and the robot starts at (1, 1) and find the path to Goal (11, 11).

Figure 9 illustrates the experimental results of the robot path planning using GA, PSO and SLPSO

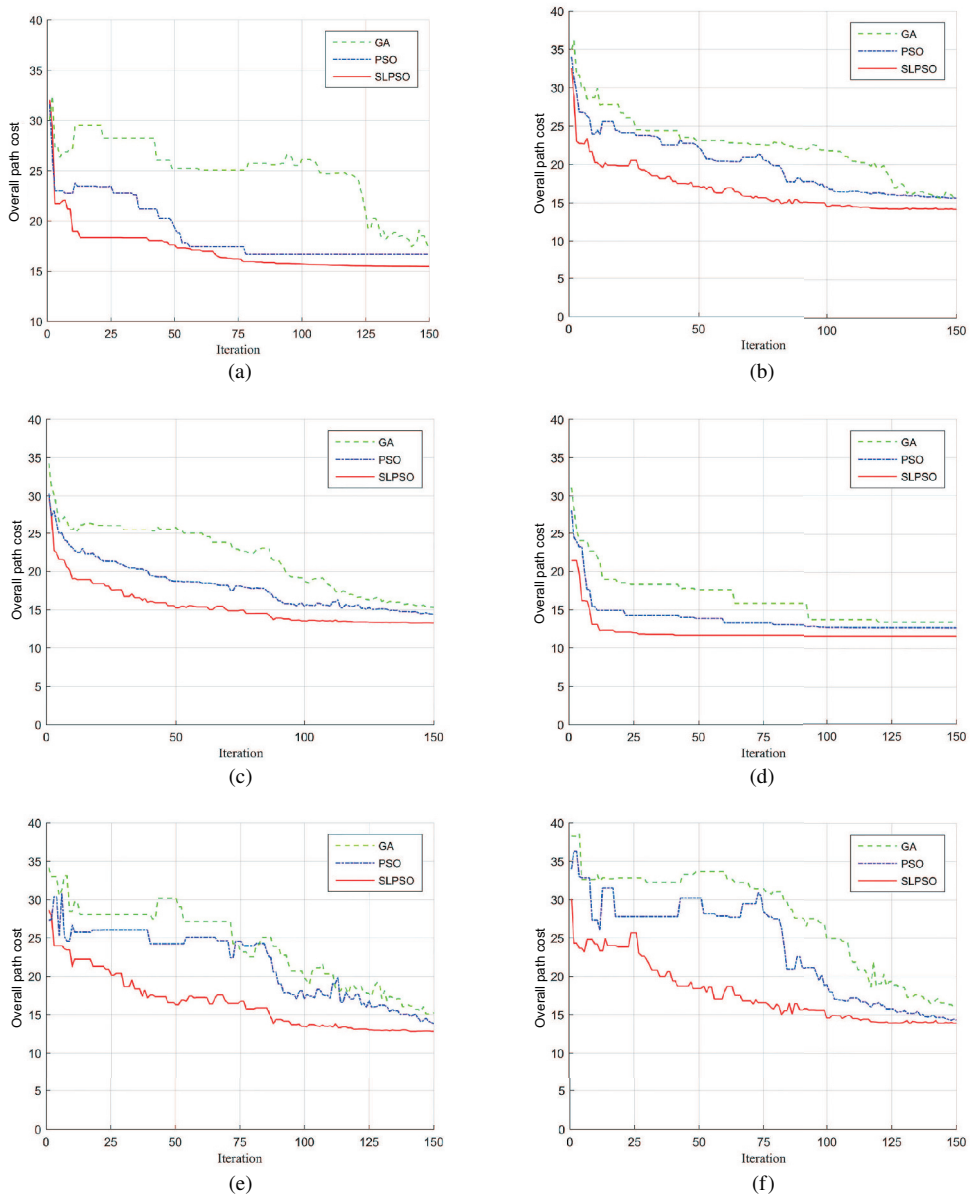

Figure 6 (Color online) The overall path cost results of GA, PSO and SLPSO for mobile robot path planning with different D. (a) $D = 5$; (b) $D = 10$; (c) $D = 15$; (d) $D = 20$; (e) $D = 25$; (f) $D = 30$.

Table 2 The path planning time (s) of GA, PSO, SLPSO for mobile robot path planning with different D

Algorithm		D=5	D=10	D=15	D=20	D=25	D=30
GA	Mean	2.097E+00	1.338E+00	1.087E+00	9.368E−01	1.512E+00	5.422E+00
	Deviation	6.689E−01	1.276E−01	1.260E−01	2.524E−02	1.339E−01	1.230E+00
PSO	Mean	1.203E+00	1.220E+00	9.741E−01	6.257E−01	1.276E+00	3.193E+00
	Deviation	1.574E−01	1.264E−01	1.393E−02	2.473E−02	1.605E−01	1.265E+00
SLPSO	Mean	**7.105E−01**	**7.771E−01**	**5.552E−01**	**3.226E−01**	**7.958E−01**	**1.957E+00**
	Deviation	**2.655E−02**	**1.479E−02**	**1.665E−02**	**1.629E−02**	**1.311E−02**	**1.554E−01**

in the 2D occupancy grid map with resolution of 2 cm×2 cm, which built from the physical environment by the SLAM system. As can be seen already visually, all three algorithms can guarantee the generation of the feasible paths for mobile robot.

Table 3 tabulates the results of path length, path collision risk degree, path smoothness, overall path

Figure 7 (Color online) Turtlebot2 robot.

Figure 8 (Color online) The practical environment for TurtleBot2 robot path planning.

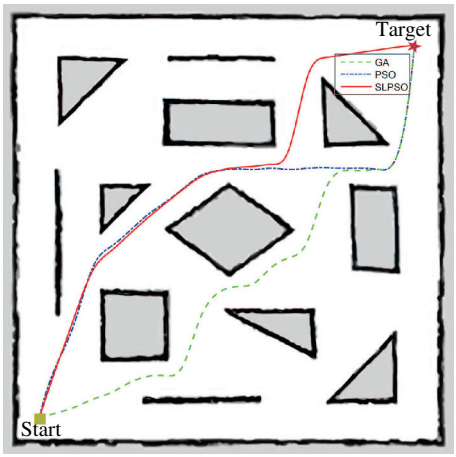

Figure 9 (Color online) The path planning results of TurtleBot2 robot using GA, PSO and SLPSO in the RVIZ.

Table 3 The experiment results of GA, PSO, SLPSO for $D = 20$

Algorithm	Length (m)	Collision risk degree	Smoothness (rad)	Overall cost	Running time (s)
GA	2.259E+01	1.310E+01	7.761E+00	1.731E+01	3.388E+02
PSO	2.180E+01	7.638E+00	4.298E+00	1.438E+01	3.269E+02
SLPSO	**2.051E+01**	**5.869E+00**	**3.516E+00**	**1.296E+01**	**3.077E+02**

cost and running time. Again, the results show the similar conclusion as the former simulation experiments. It is seen from this table that SLPSO behaves with the best performance in all the performance criterias, while GA gives the worst performance. With respect to the path length, SLPSO gives the best behavior as the resulting path length is shorter than that of PSO and GA, where SLPSO provides 5.89% less length to plan the path than PSO, and 9.18% less length than GA. In the performance of path collision risk degree, although all the approaches have similar collision risk degree except GA which gives the largest collision risk degree, SLPSO performs slightly better than PSO, which means that SLPSO can find the most safest path avoiding obstacles areas with little values of collision risk degree. With respect to the path smoothness, SLPSO provides the smoothest path as it consumes much less path smoothness than PSO and GA do. Consequently, the overall path cost generated by SLPSO is significantly smaller than that of PSO and GA, where SLPSO consumes 9.93% and 25.15% less overall path cost than PSO and GA respectively, which means that SLPSO is able to find the most optimal path. In the performance of the running time, SLPSO performs best due to it consumes much less time than the other methods, where SLPSO provides 5.87% less time than PSO and 10.11% less time than GA.

According to the above experimental results, it can be concluded that SLPSO exceeded PSO and GA in providing a more optimal path with comparable path length, path collision risk degree, path smoothness

and running time to that from GA and PSO.

In summary, the results from both simulations and experiments demonstrate that the SLPSO developed in this work is a effective and feasible method in solving mobile robot path planning problem.

5 Conclusion and future work

This paper presents a self-adaptive learning PSO with different learning strategies to addressed the multi-objective optimization problem for mobile robot path planning. In the proposed approach, a novel type of self-adaptive learning mechanism is developed to improve the search ability of PSO. Furthermore, to enhance the path feasibility, we further apply the new bound violation handling schemes to restrict the velocity and position for each particle. To verify the performance of SLPSO, a series of comparative experiments have been carried out in simulation as well as on a real robot. The results demonstrate that SLPSO is more feasible and effective than GA, PSO on solving the mobile robot path planning problem.

Following this work, we will further investigate how to apply SLPSO with non-equidistant distributed dimensions to enhance the adaptability to complex environments for mobile robot path planning. Besides, the applications of SLPSO for path planning in 3D environments or multi-robot systems will also be another key issue that needs to be further studied.

Acknowledgements This work was supported by National Basic Research Program of China (973 Program) (Grant No. 2013CB035503).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Volos C K, Kyprianidis I M, Stouboulos I N. A chaotic path planning generator for autonomous mobile robots. Robot Auton Syst, 2012, 60: 651–656
2 Chaari I, Koubaa A, Trigui S, et al. SmartPATH: an efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots. Int J Adv Robot Syst, 2014, 11: 399–412
3 Zuo L, Guo Q, Xu X, et al. A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots. Neurocomputing, 2015, 170: 257–266
4 Wu Y, Qu X J. Path planning for taxi of carrier aircraft launching. Sci China Technol Sci, 2013, 56: 1561–1570
5 Li G S, Chou W S. An improved potential field method for mobile robot navigation. High Technol Lett, 2016, 22: 16–23
6 Miao H, Tian Y C. Dynamic robot path planning using an enhanced simulated annealing approach. Appl Math Comput, 2013, 222: 420–437
7 Tsai C C, Huang H C, Chan C K. Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. IEEE Trans Ind Electron, 2011, 58: 4813–4821
8 Saska M, Macaš M, Přeučil L, et al. Robot path planning using particle swarm optimization of Ferguson splines. In: Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). New York: IEEE Press, 2006. 833–839
9 Raja P, Pugazhenthi S. On-line path planning for mobile robots in dynamic environments. Neural Netw World, 2012, 22: 67–83
10 Chen X, Kong Y, Fang X, et al. A fast two-stage ACO algorithm for robotic path planning. Neural Comput Appl, 2013, 22: 313–319
11 Purcaru C, Precup R E, Iercan D, et al. Optimal robot path planning using gravitational search algorithm. Int J Artif Intell, 2013, 10: 1–20
12 Li P, Duan H B. Path planning of unmanned aerial vehicle based on improved gravitational search algorithm. Sci China Technol Sci, 2012, 55: 2712–2719
13 Duan H B, Qiao P X. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. Int J Intell Comput Cybern, 2014, 7: 24–37
14 Rania H, Babak C, Olivier D. A comparison of particle swarm optimization and the genetic algorithm. In: Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Material Conference, Austin, 2005. 1–13
15 Qin Y Q, Sun D B, Li N, et al. Path planning for mobile robot using the particle swarm optimization with mutation operator. In: Proceedings of the 3rd IEEE International Conference on Machine Learning And Cybernetics, Shanghai, 2004. 2473–2478
16 Zhang Q, Guochang G, Zhang Q. Path planning based on improved binary particle swarm optimization algorithm. In: Proceedings of the 2008 IEEE Conference on Robotics, Automation and Mechatronics, Chengdu, 2008. 462–466

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

18 Juang C F, Chang Y C. Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments. IEEE Trans Fuzzy Syst, 2011, 19: 379–392

19 Chen X, Li Y. Smooth path planning of a mobile robot using stochastic particle swarm optimization. In: Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation, Luoyang, 2006. 1722–1727

20 Geng N, Gong D W, Zhang Y. PSO-based robot path planning for multisurvivor rescue in limited survival time. Math Probl Eng, 2014

21 Zhang Y, Gong D W, Zhang J H. Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing, 2013, 103: 172–185

22 Masehian E, Sedighizadeh D. A multi-objective PSO-based algorithm for robot path planning. In: Proceedings of the 2010 IEEE International Conference on Industrial Technology (ICIT), Valparaiso, 2010. 465–470

23 Wang X, Zhang G, Zhao J, et al. A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning. Int J Comput Commun Control, 2015, 10: 732–745

24 Mo H, Xu L. Research of biogeography particle swarm optimization for robot path planning. Neurocomputing, 2015, 148: 91–99

25 Purcaru C, Precup R E, Iercan D, et al. Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones. In: Proceedings of the 17th IEEE International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, 2013. 434–439

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

27 Min C L, Min G P. Artificial potential field based path planning for mobile robots using a virtual obstacle concept. In: Proceedings of the 17th IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Kobe, 2003. 735–740

28 Mcisaac K A, Ren J, Huang X. Modifed Newton's method applied to potential field navigation. In: Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, 2003. 5873–5878

29 Cleghorn C W, Engelbrecht A P. Particle swarm convergence: standardized analysis and topological influence. In: Proceedings of the 9th International Conference on Swarm Intelligence, Brussels, 2014. 134–145

30 Frans V D B, Engelbrecht A P. A convergence proof for the particle swarm optimiser. Fundam Inform, 2010, 105: 341–374

31 Li C, Yang S. An adaptive learning particle swarm optimizer for function optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Trondheim, 2009. 381–388

32 Poli R, Kennedy J, Blackwell T. Particle swarm optimization: an overview. Swarm Intell, 2007, 1: 33–57

33 Lin Q, Li J, Du Z, et al. A novel multi-objective particle swarm optimization with multiple search strategies. Eur J Oper Res, 2015, 247: 732–744

34 Li C, Yang S. An adaptive learning particle swarm optimizer for function optimization. In: Proceedings of the 2009 IEEE Congress on on Evolutionary Computation, Trondheim, 2009. 381–388

35 Elshamli A, Abdullah H A, Areibi S. Genetic algorithm for dynamic path planning. In: Proceedings of the IEEE Electrical and Computer Engineering, Niagara Falls, 2004. 677–680

36 Carlisle A, Dozier G. An off-the-shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, 2001. 1–6

37 Liang J J, Qu B Y, Suganthan P N, et al. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization. Technical Report 201212. 2013

38 Tvrdík J. Competitive differential evolution. In: Proceedings of the 12th International Conference on Soft Computing, Brno, 2006. 7–12

39 Montemanni R, Gambardella L M, Rizzoli A E, et al. Ant colony system for a dynamic vehicle routing problem. J Comb Optim, 2005, 10: 327–343

40 Zhan Z, Zhang J. Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern Part B-Cybern, 2009, 39: 1362–1381

41 Quigley M, Conley K, Gerkey B P, et al. ROS: an open-source robot operating system. In: Proceedings of the ICRA Workshop on Open-Source Software, Kobe, 2009

42 Gerkey B P, Vaughan R T, Howard A. The player/stage project: tools for multi-robot distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, Coimbra, 2003. 317–323