Cluster Computing (2019) 22:S4745–S4766
https://doi.org/10.1007/s10586-018-2360-3

# Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm

Alaa Tharwat[1,4] · Mohamed Elhoseny[2,4] · Aboul Ella Hassanien[3,4] · Thomas Gabel[1] · Arun Kumar[5]

Received: 14 January 2018 / Revised: 13 February 2018 / Accepted: 1 March 2018 / Published online: 12 March 2018


## Abstract

Path planning algorithms have been used in different applications with the aim of finding a suitable collision-free path which satisfies some certain criteria such as the shortest path length and smoothness; thus, defining a suitable curve to describe path is essential. The main goal of these algorithms is to find the shortest and smooth path between the starting and target points. This paper makes use of a Bézier curve-based model for path planning. The control points of the Bézier curve significantly influence the length and smoothness of the path. In this paper, a novel Chaotic Particle Swarm Optimization (CPSO) algorithm has been proposed to optimize the control points of Bézier curve, and the proposed algorithm comes in two variants: CPSO-I and CPSO-II. Using the chosen control points, the optimum smooth path that minimizes the total distance between the starting and ending points is selected. To evaluate the CPSO algorithm, the results of the CPSO-I and CPSO-II algorithms are compared with the standard PSO algorithm. The experimental results proved that the proposed algorithm is capable of finding the optimal path. Moreover, the CPSO algorithm was tested against different numbers of control points and obstacles, and the CPSO algorithm achieved competitive results.

**Keywords** Particle Swarm Optimization (PSO) · Bézier curve · Path planning · Chaos

## 1 Introduction

Path planning technique is widely used in many applications such as mobile robotics and track trajectories [1]. The goal of path planning is to find the optimal path which (1) minimizes the distance between the starting and target points, (2) avoids collisions with obstacles, and (3) increases the smoothness of the curve [2, 3]. Path planning can be formulated as an optimization problem on a set of indices, e.g., shortest distance, and under some constraints, e.g., collision-free path.

Path planning is also known to be an NP-hard optimization problem that can be solved using heuristic algorithms such as evolutionary algorithms [4–6]. There are many metaheuristic optimization algorithm that were employed to optimize the path planning problem such as Genetic Algorithms (GA) [7, 8], Particle Swarm Optimization (PSO) [9], Artificial Bee Colony Evolutionary Programming (ABC-EB) [10], PSO + Gravitational Search Algorithm (GSA) [11], Tabu Search (TS) [12], and Firefly Algorithm (FA) [13].

✉ Mohamed Elhoseny
mohamed_elhoseny@mans.edu.eg

Alaa Tharwat
aothman@fb2.fra-uas.de

Aboul Ella Hassanien
aboitcairo@gmail.com
http://www.egyptscience.net

Thomas Gabel
tgabel@fb2.fra-uas.de

Arun Kumar
arun.nura@gmail.com

[1] Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany

[2] Faculty of Computers and Information, Mansoura University, Mansoura, Egypt

[3] Faculty of Computers and Information, Cairo University, Cairo, Egypt

[4] Scientific Research Group in Egypt (SRGE), Cairo, Egypt

[5] School of EEE, SASTRA University, Thanjavur, India

Traditional path planning algorithms assume that the environment is perfectly known and try to search for the optimal path that contains sharp turns and some polygonal lines. However, such algorithms are inflexible and have a negative influence on the smoothness of the generated curve which may add additional efforts on the mobile robot by switching between different modes such as stop, rotate, and restart to move along the polygonal lines. Hence, these algorithms are time and energy consuming, and the smooth movement is a requirement for some tasks [8].

Bézier curve has been applied in the smooth path planning problem, and it generates free curves perfectly [14]. Bézier curve can be drawn as a series of line segments that join the control points. Thus, choosing the control points controls the performance of the generated curve; hence, these control points are used to optimize the Bézier curve. Several studies have focused on optimizing these control points using metaheuristic optimization algorithms such as GA [15], TS [12], and FA [13]. In this paper, we target the use of Particle Swarm Optimization to optimize Bézier curve.

PSO has been used in many applications due to its simplicity [16]. For example, in image processing, the PSO algorithm was used to search for the optimal thresholding value in image segmentation [17, 18]. In machine learning, the PSO is widely used for parameter tuning of learning algorithms to improve the classification performance [19, 20], for feature selection [21], and to search for the optimal centroids in clustering [22, 23]. PSO algorithm is also applied to solve mathematical problems [24]. In electrical and power applications, the PSO was used for different purposes such as searching for the maximum power point tracking of multiple photovoltaic [25], and for optimizing the size and location of distributed generation for loss reduction [26].

In general, PSO lends itself strongly to exploitation which may lead to local optima problem. Hence, in some cases, PSO fails to find the global optimal solution(s). The search strategy used in standard PSO is mainly based on random walks; thus, it cannot always deal with the optimization problems successfully. Different strategies have been added to the optimization algorithms with the goal of improving its performance [27]. Chaos concept is one of these strategies that has been widely used in various applications, and it was already combined with some optimization algorithms [27]. This combination may generate solutions which may have higher mobility and diversity than standard optimization algorithms. Recently, the chaos theory has been combined with several metaheuristic optimization methods to solve the local optima problem [27, 28]. For example, chaos concept was used to tune the parameters of GA [29], Ant bee colony (ABC) optimization [30], FA [31], and such combinations achieved promising results.

In this paper, the main contribution is to use Chaotic PSO (CPSO) algorithm to present a novel algorithm which is used to optimize Bézier curve to get the shortest, smooth, and collision-free path, which can be applied in different applications. The optimal Bézier curve can be obtained by searching for the optimal positions of the control points to draw shortest, smooth, and collision free path. As a consequence, the dimension of the search space depends mainly on the number of control points. As different chaotic maps may lead to different behavior of the proposed algorithm, we then have two chaos-based PSO algorithms, namely, CPSO-I and CPSO-II. In these algorithms, we used different chaotic maps to replace the parameters of the PSO. Hence, different methods that use chaotic maps as efficient alternatives to pseudo-random sequences have been proposed. Many experiments have been conducted to (1) test the influence of different chaotic maps on the performance of the algorithm, (2) compare the proposed algorithms with the standard PSO, (3) examine the influence of the number of control points on the results of the CPSO algorithm, and (4) show the impact of the number of obstacles on the performance of the CPSO algorithm.

The rest of the paper is organized as follows: Sect. 2 summarizes the related work of the path planning and Bézier curve. Section 3 gives overviews of the techniques and methods used for the proposed model. Section 4 explains in details the proposed CPSO-based method to get the optimal path. Experimental results and discussion are introduced in Sect. 5. Finally, Sect. 6 concludes the paper.

## 2 Related work

In the path planning problem, classical methods required a long time and huge storage memory [8]. Thus, metaheuristic optimization algorithms are used frequently to optimize the path planning problem [32]. For example, ABC-EB algorithm was employed to solve the path planning problem, and the results of ABC-EB were compared with the classical Probabilistic Road Map method (PRM) [10]. The results proved that the ABC-EB algorithm outperformed the PRM algorithm. GA was fused with PSO to optimize the path of a mobile robot in a 3D environment which contained static obstacles [9]. In another research, the PSO algorithm was hybridized with the Gravitational Search Algorithm (GSA) to minimize the path length; and hence, reduces the arrival time of all robots to their destination [11].

Several studies used metaheuristic algorithms with Bézier curve for optimal path planning. FA and Bézier

curve were combined to find the shortest feasible (collision-free) path [13]. The results of FA were compared with GA and adaptive inertia weight PSO (PSO-w), and the FA achieved the highest success rates. Galvez et al. performed different experiments in 2D and 3D environments using Bézier curve and TS, and their proposed algorithm achieved competitive results [12]. PSO was used to optimize three different curves, namely: Bézier curve, Ferguson curve, and $\eta^3$ curve [33], and the PSO with Bézier curve achieved the best results. In another research, an improved dynamic multi-swarm PSO algorithm with crossover operator was proposed to optimize the Bézier curve [13].

Bézier curve also was utilized in many applications. For example, Ziolkowski et al. combined the GA and Bézier curve to design a shape of a solenoid which produces a uniform magnetic field on its axis [15]. Moreover, the parallel GA was employed with Bézier curve to generate trajectories for multi-unmanned aerial vehicle (UAV) systems, where the Bézier curve was used to enhance the smoothness of the obtained path [34]. Bézier + GA algorithm was also used to design a cambered airfoil and the proposed design achieved competitive results compared with some state-of-the-art methods [35]. Jolly et al. employed a Bézier-curve-based model for path planning in a multi-agent robot soccer system, and the velocity of the proposed approach was updated within its allowable limits by keeping its acceleration within the predefined safe limits [36]. Bézier curve with GA model was used to design 2-D turbine blades using target pressure considerations [37], and the proposed model was applied to design a steam turbine blade. Bézier curve was also used for generating the optimal trajectories [38]. Choi et al. used the Bézier curve to generate the optimal trajectories for vehicles to satisfy the path constraints [39]. In another study, the Bézier curve for path planning generated the reference trajectories, which are used by the intelligent wheelchair to pass a doorway [38].

## 3 Background

### 3.1 Bézier curve

The Bézier curve was proposed by Pierre Bézier, who used the Bézier curve to design the body of a car in 1962 [40]. Bézier curve has been used in computer graphics applications [33]. The Bézier curve is convex if the control points form a convex polygon, and it can express free curves and surfaces perfectly [33]. Mathematically, the Bézier curve can be drawn as a series of line segments joining the control points as in Fig. 1.

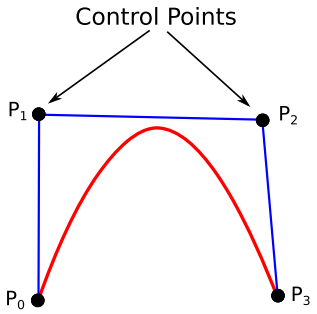



**Fig. 1** An example of a Bézier curve

Assume we have $n+1$ points, i.e., Bézier curve of degree $n$, $P_i, i = 0, 1, \ldots, n$, the Bézier curve is defined as follow:

$$R(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), t \in [0, 1], \tag{1}$$

where $t$ indicates the normalized time variable, $P_i = [x_i, y_i]^T$ is the coordinate vector of the $i$th control point with $x_i$ and $y_i$ being the components corresponding to the $X$ and $Y$ coordinates, respectively, $B_{i,n}$ is the Bernstein basis polynomials, which represents the base function in the expression of a Bézier curve, and it is given by:

$$\begin{aligned} B_{i,n}(t) &= C_n^i t^i (1-t)^{n-i} \\ &= \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, i = 0, 1, \ldots, n \end{aligned} \tag{2}$$

From Eqs. (1) and (2), the parameter equation (for the example shown in Fig. 1) for each control point can be generated as follows:

$$R(t) = P_0(1-t)^3 + 3P_1 t(1-t)^2 + 3P_2 t^2(1-t) + P_3 t^3, \tag{3}$$

where $t$ is in the range of [0, 1]. The Bézier curve is invariance under translation and rotation and this is called Geometry invariance property. Moreover, the Bézier curve starts at the starting point, $t = 0$, and stops at the ending/target, $t = 1$. In other words, $P_0 = R(0)$ and $P_n = R(1)$. Bézier curve has some control points which form a control polygon as shown in Fig. 1. As shown, the curve is tangent to the control polygon at the endpoints as in Eq. (4). Moreover, the first derivatives of the starting and ending points of the curve are only related to the two nearest control points, and in the same direction of the line of the two points. The Bézier curve can be formed by connecting several line segments of low-order Bézier curves.

$$R(\dot{t}) = \frac{dR(t)}{dt} = n \sum_{i=0}^{n-1} \Delta b_i B_{i,n-1}(t), t \in [0, 1], \tag{4}$$

where $\Delta b_i = b_{i+1} - b_i$.

Given two segments $L_1 = \{P_{10}, P_{11}, P_{12}, P_{13}\}$ and $L_2 = \{P_{20}, P_{21}, P_{22}, P_{23}\}$. To form a continuous curve from these two points the following equation should be satisfied:

$$P_{13} - P_{12} = P_{21} - P_{20}, P_{13} = P_{20}. \tag{5}$$

Hence, to satisfy the property of first-order continuous condition, $l$ segments of Bézier curve need $2l$ points and $4l$ parameters and the path can be generated as follows:

$$R(t) = \begin{cases} P_0(1-t)^3 + 3P_1^i t(1-t)^2 \\ +3P_2^i t^2(1-t) + P_3^i t^3, & i = 1 \\ P_3^{(i-1)}(1-t)^3 \\ +3\,(2P_3^{(i-1)} - P_2^{(i-1)})t(1-t)^2 \\ +3P_2^i t^2(1-t) + P_3^i t^3, & 1 \le i \le n \\ P_3^{(i-1)}(1-t)^3 \\ +3(2P_3^{(i-1)} - P_2^{(i-1)})t(1-t)^2 \\ +3P_2^i t^2(1-t) + P_1 t^3, & i = n, \end{cases} \tag{6}$$

where $P_0$ is the starting point and $P_1$ is the ending point, and when the value of $t$ changes in the interval (0, 1), we can get a cubic Bézier curve of the $i$th segment. These $n$ line segments of cubic Bézier curve form the entire path of the curve.

### 3.2 Particle Swarm Optimization (PSO)

PSO is one of the well-known optimization algorithms, and it was implemented by Reynolds and Heppner, and then simulated by Kennedy and Eberhart [41–43]. The aim of the PSO is to search for optimal or near optimal solutions in the search space. Each particle in the PSO algorithm has its own (1) position ($x^i \in \mathcal{R}^n$) which represents a point in the search space $\mathcal{R}^n$, where $n$ is the dimension of the search space, (2) velocity, i.e., rate of position change, ($v^i$), and (3) the previous best positions ($p^i$). These positions of the particles are initialized randomly, and the velocity values are initialized with zero. The current positions of all particles are evaluated using the fitness function. The fitness value of the current position for each particle is then compared with its best position, and best value is stored in ($p^i$), i.e.,, the previous best positions store the positions of the particles that have better values. Moreover, in PSO, the global best position ($G$) represents the best fitness value [44].

Each particle is moved by adding the velocity to the current position as follows:

$$x^i_{(t+1)} = x^i_{(t)} + v^i_{(t+1)}, \tag{7}$$

where $x^i_{(t)}$ is the current position, $x^i_{(t+1)}$ is the new position, and $v^i_{(t+1)}$ indicates the new velocity. The velocity of each particle is adjusted as follows:

$$v^i_{(t+1)} = wv^i_{(t)} + C_1 r_1(p^i_{(t)} - x^i_{(t)}) + C_2 r_2(G - x^i_{(t)}), \tag{8}$$

where $w$ represents the inertia weight, $C_1$ is the cognition learning factor, $C_2$ represents the social learning factor, and $r_1$, $r_2$ are the uniformly generated random numbers in the range of [0, 1]. According to Eq. (8), the new velocity for each particle is determined by:

1. The original velocity of the particle ($wv^i_{(t)}$).
2. The position of the previous best position of that particle, this is so-called particle memory or cognitive component. This term as in Eq. 8 is used to adjust the velocity towards the best position visited by that particle ($C_1 r_1(p^i_{(t)} - x^i_{(t)})$).
3. The position of the global best fitness value, this is the so-called social component [$C_2 r_2(G - x^i_{(t)})$], and it is used to adjust the velocity towards the global best position of all particles [44].

Figure 2 illustrates an example of the movement of two particles in a one-dimensional search space. As shown, the two particles have three different velocities which are used to adjust the velocity.

High values of velocity make the particles very fast, which may prevent the particles from converging to the optimal solution; hence, the velocity of the particles could be limited to a range $[-V_{max}, V_{max}]$. This is much similar the learning rate in learning algorithms. On the contrary, small values of velocity cause the particles to search within a small area; hence, it may lead to slow convergence [44].

The positions and velocities of all particles are updated iteratively until it reaches a predefined stopping criterion

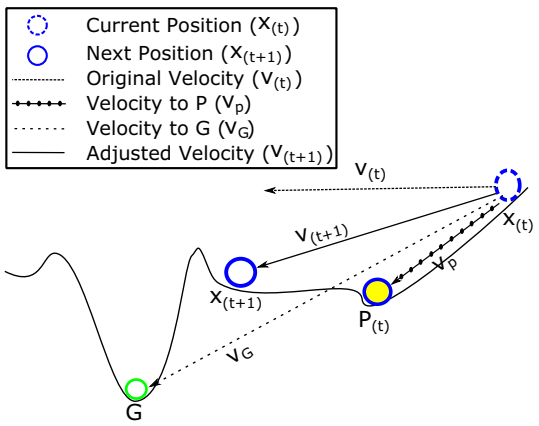



**Fig. 2** Illustration of the movement of one particle using PSO algorithm in one-dimensional space

[43]. The details of the PSO algorithm are summarized in Algorithm (1).

**Algorithm 1** : Particle Swarm Optimization (PSO)

1: Initialize the particles' positions ($x^i$), velocity ($v^i$), previous best positions ($p^i$), the number of particles ($N$), and maximum number of iterations ($Max_{iter}$).
2: **while** ($t < Max_{iter}$) **do**
3: **for all** Particles ($i$) **do**
4: Evaluate the fitness value for the current particle, $x^i$, ($\mathcal{F}(x^i)$).
5: **if** ($\mathcal{F}(x^i) < \mathcal{F}(p^i)$) **then**
6: $p^i = x^i$
7: **end if**
8: **if** ($\mathcal{F}(x^i) < \mathcal{F}(G)$) **then**
9: $G = x^i$
10: **end if**
11: Adjust the velocity and position of the current particle according to Eqs. (7 and 8).
12: **end for**
13: Stop the algorithm if a sufficiently good fitness function is met.
14: **end while**

### 3.3 Chaotic maps

In many metaheuristic optimization algorithms, the randomness can be achieved using normal or Gaussian distributions, i.e., some parameters of these algorithms are drawn randomly from uniform or Gaussian distribution. Chaos acts similar to randomness, but with better dynamical and statistical properties which are vital to ensure that the chaotic variables can go through all states in certain ranges without repetition [45]. Such dynamical properties are important to ensure that the generated solutions by the algorithm can be diverse enough to reach every mode in the multimodal objective search space potentially. Hence, chaos search can escape more easily from a local optimal solution than the standard stochastic search; and hence, the random parameters in optimization algorithms can be replaced with a one-dimensional chaotic map, this is so-called *chaotic optimization*. Due to the mixing property of chaos, chaotic optimization algorithms can converge faster than standard stochastic search [46].

In the rest of this section, eight well-known one-dimensional chaotic maps which are used in our experiments are outlined.

– *Gauss map* The Gauss or Mouse map generates sequence in (0, 1), and it can be defined as follows [46]:

$$x_{k+1} = \begin{cases} 0, & x_k = 0 \\ \dfrac{1}{x_k mod(1)}, & \text{otherwise,} \end{cases} \tag{9}$$

where $\frac{1}{x_k mod(1)} = \frac{1}{x_k} - \left\lfloor \frac{1}{x_k} \right\rfloor$, $x_k \in (0, 1)$ represents the *kth* chaotic number, $k$ is the iteration number, and the generated sequence is in (0, 1).

– *Singer map* This map is defined as follows [27]:

$$x_{k+1} = \mu(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4), \tag{10}$$

where $\mu$ is a parameter between 0.9 and 1.08.

– *Tent map* This map is similar to Logistic map and it can be defined as follows [27]:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{0.7}, & x_k < 0.7 \\ \dfrac{10}{3}(1 - x_k), & \text{otherwise} \end{cases} \tag{11}$$

– *Circle map* The circle map is defined as follows [27]:

$$x_{k+1} = x_k + b - \left(\frac{a}{2\pi}\right) sin(2\pi x_k)\ mod(1) \tag{12}$$

where $a = 0.5$, $b = 0.2$, and the generated sequence is in (0, 1).

– *Logistic map* This map can be written as follows [27]:

$$x_{k+1} = ax_k(1 - x_k) \tag{13}$$

In our experiments $a = 4$ was used, and also under the condition that $x_0 \notin (0.0, 0.25, 0.5, 0.75, 1.0)$.

– *Sine map* Sine map is given by [45]:

$$x_{k+1} = \frac{a}{4} sin(\pi x_k), \tag{14}$$

where $0 < a \leq 4$.

– *Piecewise map* The Piecewise map is given by [45]:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{P}, & P \geq x_k \geq 0 \\ \dfrac{x_k - P}{0.5 - P}, & 0.5 \geq x_k \geq P \\ \dfrac{1 - P - x_k}{0.5 - P}, & 1 - P \geq x_k \geq 0.5 \\ \dfrac{1 - x_k}{P}, & 1 \geq x_k \geq 1 - P, \end{cases} \tag{15}$$

where $P$ represents the control parameter between 0 and 0.5 and $P \neq 0$.

– *Sinusoidal map* This map is defined as in Eq. (16) [27]:

$$x_{k+1} = ax_k^2 sin(\pi x_k) \tag{16}$$

Figure 3 visualizes the chaotic value distributions of 500 iterations for all eight maps with random initial values. As shown in the figure the behaviors of all maps are different,

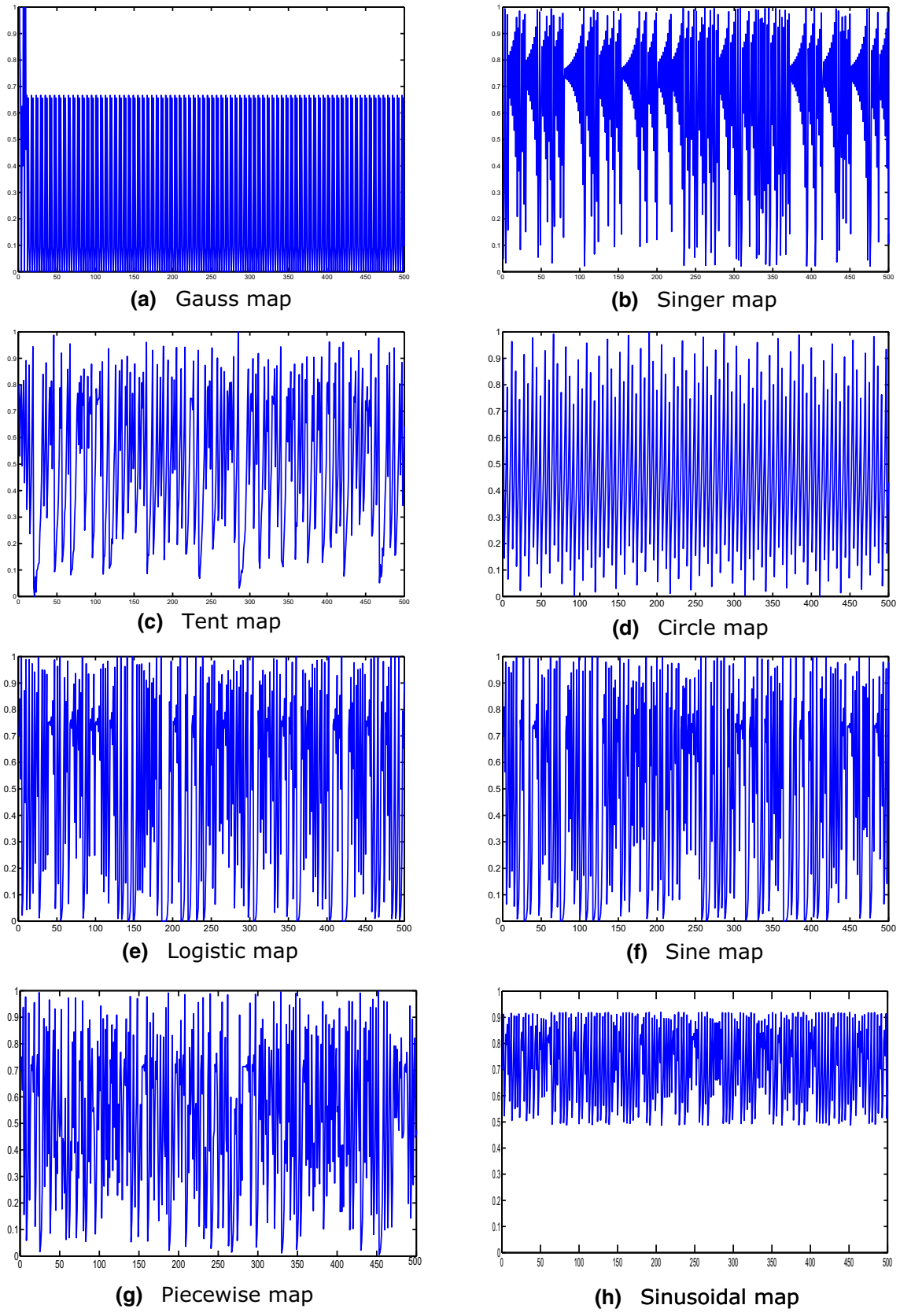



**Fig. 3** Chaotic value distributions during 500 iterations.

and each map can reach different modes in the search space. For example, the Gaussian map in Fig. 3a initially reached to a high or maximum value, but it drops dramatically after that and it fluctuates between 0 and 0.7. This behavior can be employed in our optimization algorithm by maximizing the range of exploration capability at the beginning and then reduce that range to improve the current solutions. It is worth mentioning that the behaviors of chaotic maps are highly affected by the initial conditions.

## 4 Chaotic Particle Swarm Optimization (CPSO)

This section describes the proposed algorithms in detail. Generally speaking, the proposed algorithms depend on optimizing the Bézier curve for path planning through searching for the optimal positions of the control points. In this study, CPSO algorithm was employed to optimize Bézier curve to find smooth and shortest Bézier curve.

### 4.1 The CPSO algorithms

The optimization algorithms have two main phases: *exploration* and *exploitation*. In the exploration phase, the goal is to explore the search space for finding the optimal solutions, or simply this phase may lead to new search regions that may contain better solutions. In the exploitation phase, the aim is to search locally around the current good solution. The optimization algorithms should be balanced between random selection and greedy selection to bias the search towards better solutions, i.e., exploitation, while exploring the search space to find different solutions, i.e., exploration.

In the PSO algorithm, the parameters $r_1$ and $r_2$ control the trade-off between exploration and exploitation phases. These two parameters control the movement of particles towards the previous best and global best positions. In other words, high values of $r_1$ and $r_2$ drive the particles to enhance the current solution by moving towards the best and global best positions, respectively. On the other hand, small values of $r_1$ and $r_2$ increase the exploration capabilities. However, although the PSO algorithm proved efficient for solving different optimization problems, it still has the following drawbacks:

– *Sub-optimal selection* at the beginning of the optimization process, the random walk of the particles are fast in the search space, which allows the particles to walk randomly in almost the entire search space. This may cause the algorithm to select sub-optimal solutions.
– *Stagnation* this problem is one of the main problems in almost all optimization algorithms. This problem occurs when the algorithm falls into a trap of some local optima, and it cannot find better solutions because its exploration capability is very limited. This makes the algorithm continue enhancing the existing solutions that have already been found, even if they are sub-optimal.

These two problems motivate our work on adapting the parameters $r_1$ and $r_2$ to obtain successive periods of exploration and exploitation. In the PSO algorithm, when reaching a solution, exploitation phase will be employed to enhance the current solutions found, followed by another exploration, which may jump to another searching region, followed by using exploitation again to further enhance the current solution found, and so on. Chaotic maps with their properties (see Sect. 3.3), can be used to adapt these parameters, i.e., $r_1$ and $r_2$, allowing for the required mix between exploitation and exploration. The proposed CPSO algorithm has two variants as in Fig. 4.

#### 4.1.1 CPSO-I algorithm

In this algorithm, the parameter $r_1$ in Eq. (8) is modified by chaotic maps ($\zeta$) during iterations, and the modified equation is given by:

$$v^i_{(t+1)} = wv^i_{(t)} + C_1\zeta_t(p^i_{(t)} - x^i_{(t)}) + C_2r_2(G - x^i_{(t)}) \qquad (17)$$

In the standard PSO, $r_1$ is a random number between 0 and 1 and in the CPSO-I algorithm, it is a chaotic number between 0 and 1.

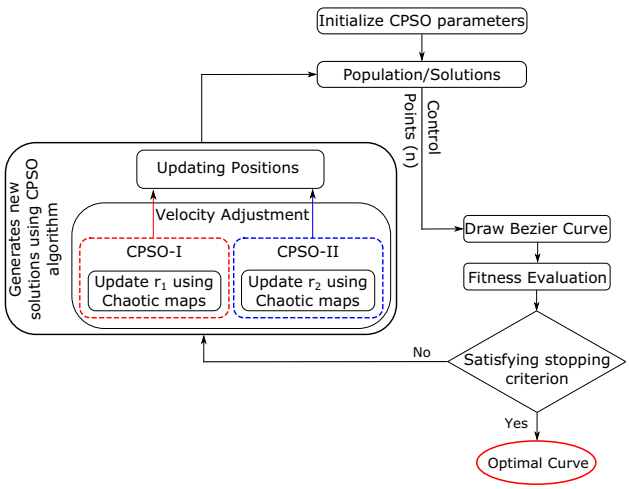



**Fig. 4** Flowchart of the proposed path planning curve using CPSO algorithm

#### 4.1.2 CPSO-II algorithm

In this algorithm, the parameter $r_2$ in Eq. (8) is modified by chaotic maps ($\zeta$) during iterations, and the velocity will be modified as follows:

$$v^i_{(t+1)} = wv^i_{(t)} + C_1 r_1 (p^i_{(t)} - x^i_{(t)}) + C_2 \zeta_t (G - x^i_{(t)}) \quad (18)$$

In the standard PSO, $r_2$ is a random number between 0 and 1 and in the CPSO-II algorithm, it is a chaotic number between 0 and 1.

### 4.2 Optimizing Bézier curve using CPSO algorithm

In this section, more details about how the CPSO algorithm was employed to optimize the Bézier curve.

#### 4.2.1 Parameters settings

In the proposed algorithm, the CPSO algorithm provides the Bézier curve with the positions of the control points to draw a path. Hence, the dimension of the search space depends mainly on the number of control points. The positions of particles are initialized randomly, and the searching range of control points was bounded by the dimensions of our environment. Increasing these limits enlarges the search space; thus, more particles are needed to search for the optimal solution, which results in more computation and slow convergence rate. The solutions are then evaluated using a fitness function that will be explained in the next section. The positions of particles are then updated as mentioned in Sect. 3.2.

When the termination criteria are satisfied, the process ends; otherwise, we proceed with the next iteration. In the proposed algorithm, the CPSO algorithm is terminated when a maximum number of iterations are reached or when the best solution is not modified for a given number of iterations.

#### 4.2.2 Fitness function

In our proposed algorithm, the fitness function should comply with the following goals.

– *Shortest distance* The first goal is to minimize the distance between the starting point, $s$, and the target point, $t$. This can be achieved by selecting control points that minimize the total distance, $\|P(t)\|$, between $s$ and $t$, and this can be formulated as follows:

$$min\ \|P(t)\|,\ 0 \le t \le 1 \quad (19)$$

– *Collision-free* The second goal is to obtain a collision-free path. Assume the obstacle represents a rigid body and it is denoted by $\alpha_k$. For simplicity, the obstacles are represented by circles, with the center $\rho_k$, where $k$ is the number of obstacles in our problem. To obtain a collision-free path, the safe distance, $D_{safe}$, between the path and the obstacles should be larger than a threshold $d_{min}$, which represents the minimum distance between the path and obstacles.

$$D_{safe} = \sqrt{(P_x(t) - \alpha^k_x)^2 + (P_y(t) - \alpha^k_y)^2} - r_{\alpha^k} \quad (20)$$

where $P_x(t)$ and $P_y(t)$ are the coordinates of the control points of Bézier curve, $\alpha^k_x$ and $\alpha^k_y$ are position of the obstacle $k$, and $r_{\alpha^k}$ represents the radius of the obstacle $k$. The path is feasible only if $D_{safe} < d_{min}$; otherwise, the path is not feasible. This goal can be formulated mathematically as follows:

$$P(t) = \begin{cases} \text{infeasible path}, & D_{safe} < d_{min} \\ \text{feasible path}, & D_{safe} \ge d_{min} \end{cases} \quad (21)$$

– *Smoothness* The third aim was to get a smooth movement of the path. This objective is satisfied by the second-order continuity of the path which makes the transition from one line segment to the next smooth.

The objective function of the proposed algorithm aims to achieve all the three goals, and the objective function was formulated as follows:

$$fitness = min\ \|P(t)\|,$$
$$\text{s.t.} \begin{cases} P(t) \in C^2 \\ P(t) \in P_{free}, \end{cases} \quad (22)$$

where $t \in [0, t_f]$ and the path moves from the $s$ at time $t = 0$ to the target $t$ at time $t_f$, $C^2$ indicates a set of second-order differentiable function, $P_{free}$ represents a set of collision-free paths which satisfying the constraint in Eq. (21). As indicated in Eq. (22), the goal in the first line is to find a path with the shortest distance ($min\ \|P(t)\|$). Moreover, there are two constraints; the first one is to find a smooth curve and this can be obtained by making the objective function satisfy the second-order continuity which makes the transition from line segment to the next one smooth. The second constraint is responsible for obtaining a collision free path.

## 5 Experimental results and discussion

The experiments were carried out to evaluate the performance of the proposed CPSO algorithms to find the optimal path planning using Bézier curve. In this section, the results and discussions of our experimental scenarios were presented. In the first experiment (Sect. 5.3), a comparison

between the two proposed algorithms, i.e.,, CPSO-I and CPSO-II, was presented. In the second experiment (Sect. 5.4), a comparison between the standard PSO and the two proposed algorithms was introduced. The third experiment (Sect. 5.5) was conducted to test the influence of the number of control points of the Bézier curve on the performance of the proposed algorithm. In the fourth experiment (Sect. 5.6), the influence of different numbers of obstacles on the proposed algorithm was tested.

### 5.1 Experimental setup

In all experiments, the environment or working field was two-dimensional ($x$ and $y$), and the domains of $x$ and $y$ were ranged from $-10$ to 10. In the first three experiments, there were three obstacles with the same sizes. Figure 5 shows the environment of the first three experiments. In the fourth experiment, the proposed model was tested against different numbers of obstacles. Moreover, in the fourth experiment, the positions of the obstacles were initialized randomly. In all experiments, the position of the starting and ending points were (0, 0) and (4, 6), respectively; hence, the straight distance from the starting to the ending point is 7.21. In the first, second, and fourth experiments, the number of control points of the Bézier curve was three. The number of the control points in the third experiment has been varied from one to ten.

The CPSO algorithm is terminated when a maximum number of iterations is reached or when the best solution is not modified for ten iterations.

In order to get an unbiased comparison of CPU times, all the experiments are performed using the same PC with the detailed settings as shown in Table 1.

For a comparison between different algorithms, the average ranks were used [47]. For each run, the methods

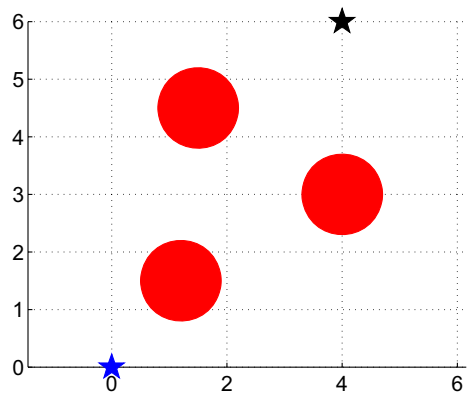



**Fig. 5** The environment, i.e., working field, of our problem with three obstacles (in red circles), the starting point (blue star), and the target point (black star) (Color figure online)

**Table 1** The detailed settings

| Name | Detailed settings |
|---|---|
| Hardware | |
| CPU | Core (TM) i5-2400 |
| Frequency | 3.10 GHz |
| RAM | 4 GB |
| Hard drive | 160 GB |
| Software | |
| Operating system | Windows 7 |
| Language | MATLAB R2012a (7.14) |

are sorted from best to worst, and the best method receives rank 1, i.e.,, the highest or best rank, the second best method receives rank 2, and so on. The average ranks are assigned in case of a tie, e.g., if two methods tie for the top rank, they both receive rank 1.5. Average ranks of all algorithms are then calculated. Moreover, the average number of collisions was also used for further comparisons between different algorithms. For each algorithm, the average number of collisions that were occurred in all runs was calculated, and a small number of collisions indicates a good result. In addition, the average of the best fitness values for each algorithm was also calculated.

### 5.2 Parameters setting for PSO algorithm

Tuning the parameters for any optimization algorithm is important as designing the algorithm itself. In this section, the effect of the number of particles, i.e., population size, and the number of iterations on the results and computational time of the proposed model (CPSO-II) with Singer map were investigated. In this experiment, the number of the control points was three, and the number of obstacles was also three.

#### 5.2.1 Population size

The number of particles needs to be sufficient for exploring the search space. In this experiment, the effect of the population size on the results and computational time of the proposed model (CPSO-II with Singer chaotic map) was investigated when the number of particles ranged from 5 to 50 particles. The fitness value and computational time of three runs are shown in Fig. 6a and b, respectively. From the figure, it is clear that increasing the number of particles improves the results, but requires more computational time. Moreover, the minimum fitness value was achieved when the number of particles was more than 20.

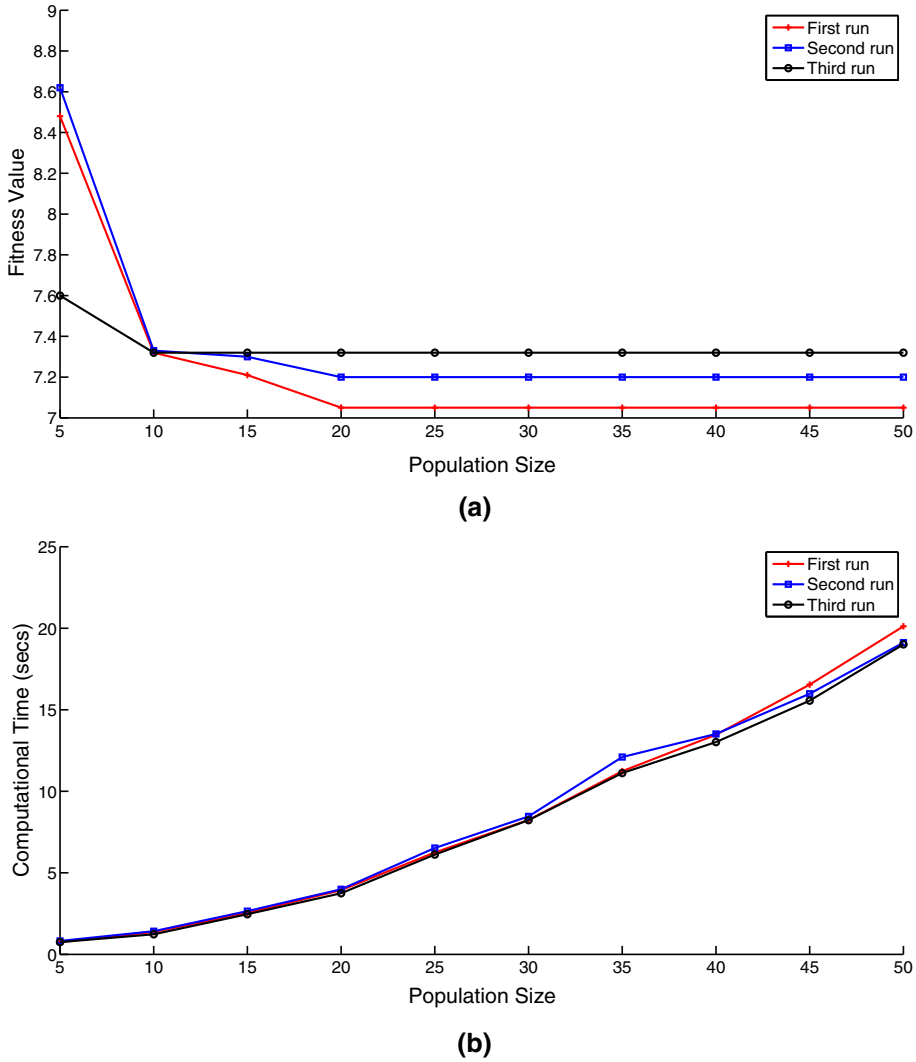



**Fig. 6** Effect of the number of particles on the performance of CPSO-II model with Singer map: (**a**) fitness value of the proposed model with different numbers of particles; (**b**) computational time of the proposed model using different numbers of particles

#### 5.2.2 Number of iterations

The number of iterations also has a great impact on the performance of the proposed model. In this experiment, the effect of the number of iterations on the fitness value and computational time of the proposed model was tested when the number of iterations was ranged from 10 to 100. The fitness value and computational time of three runs are shown in Fig. 7a and b, respectively. From the figure, it can be noticed that, when the number iterations was increased, the fitness value was decreased until it reached an extent at which increasing the number of iterations did not affect the results. Moreover, the computational time increased when the number of iterations was increased.

On the basis of the above parameter analysis and research results, Table 2 lists the detailed settings for the PSO algorithm that were used in our model.

### 5.3 First experimental scenario: CPSO-I versus CPSO-II

In this experiment, a comparison between the two proposed algorithms, i.e., CPSO-I and CPSO-II, was performed. In this experiment, the chaotic maps that are mentioned in Sect. (3.3) were used. For fair comparisons, the CPSO-I and CPSO-II algorithms with all chaotic maps were started from the same positions. In other words, the initial solutions of the CPSO-I and CPSO-II algorithms were identical with all chaotic maps. Figures 8 and 9 compare the means of the results over 60 iterations for the CPSO-I and CPSO-II algorithms, respectively, for the eight different chaotic maps. Moreover, Tables 3 and 4 list the total number of collisions, average ranks, and the average of the best results, i.e., best fitness values, of the CPSO-I and CPSO-II algorithms, respectively.

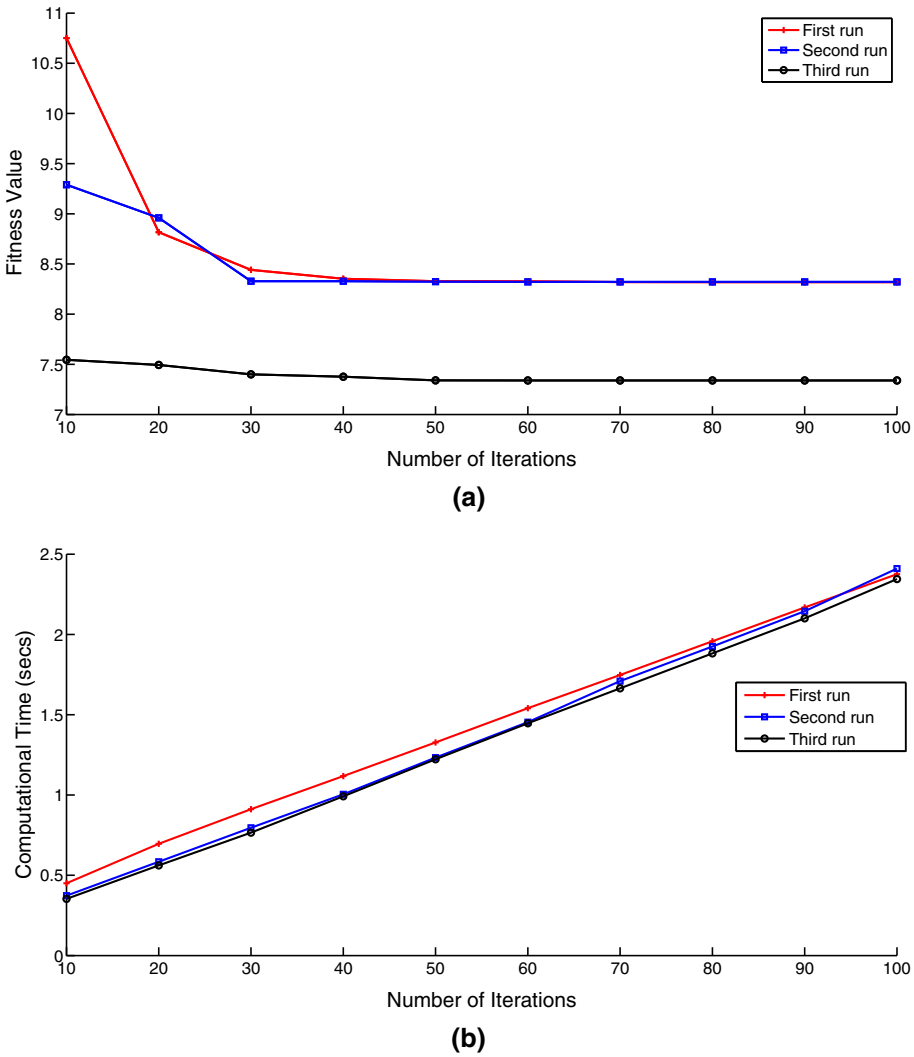



**Fig. 7** Effect of the number of iterations on the performance of CPSO-II model with Singer map: (**a**) fitness value of the proposed model with different numbers of iterations; (**b**) computational time of the proposed model using different numbers of iterations

**Table 2** The initial parameters of the PSO algorithm

| Parameter | Value |
|---|---|
| $C_1$ | 1.5 |
| $C_2$ | 1.5 |
| Population size | 20 |
| Maximum number of iterations | 60 |
| Problem dimension | $n$ |

As shown in Fig. 8, the CPSO-I model using Singer chaotic map achieved results better than the other chaotic maps. Further, the Logistic and Sine maps achieved the second and third best results, respectively. Additionally, the Tent and Piecewise maps yielded the worst results. Also, there are further conclusions can be drawn from Table 3.

1. In terms of the number of collisions, Piecewise and Sine maps achieved the minimum number of collisions, while the Gauss map yielded the maximum number of collisions.
2. Regarding the average ranks, Singer and Sine maps achieved the best, i.e., minimum, average ranks, and Piecewise map revealed the worst, i.e., highest, average rank.
3. Regarding the average best fitness values, Singer and Sine maps achieved the best and second best values, respectively, and the Tent map achieved the worst results.

Figure 9 presents the results of the CPSO-II algorithm. As shown, the CPSO-II algorithm using Singer chaotic map achieved the best results, and the Sine map achieved the second best results. Also, the Gauss map yielded the worst results. Also, from Table 4, it can be noticed that the

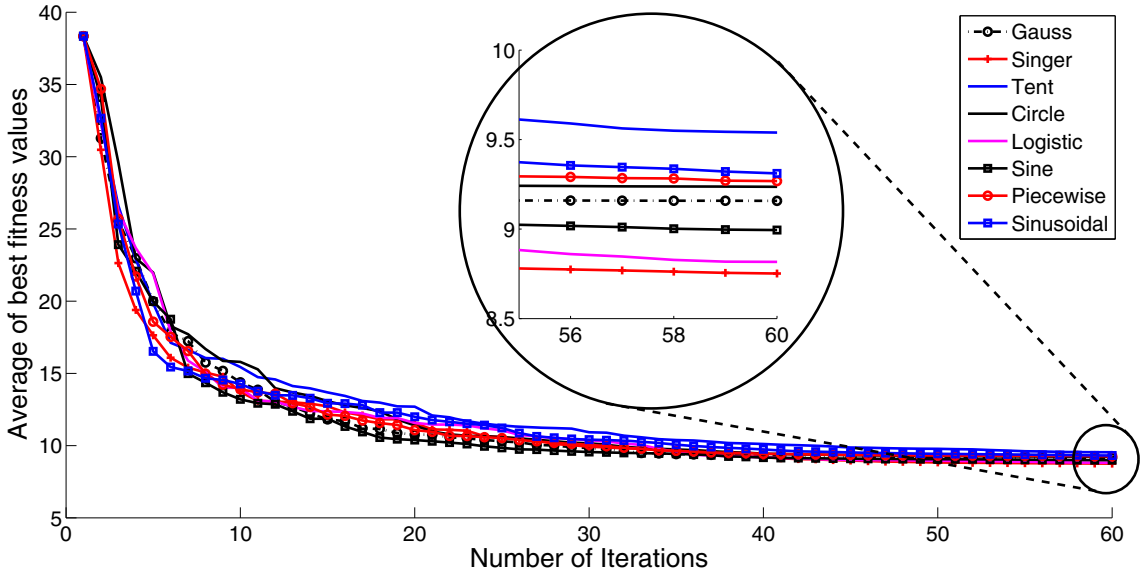



**Fig. 8** Comparison of the mean best results for CPSO-I algorithm

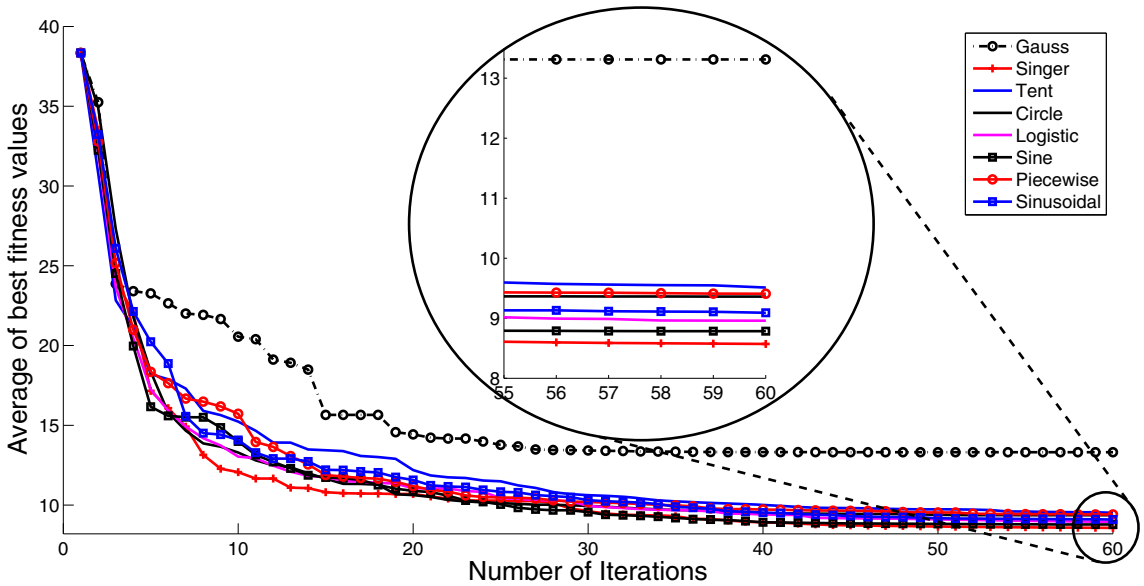



**Fig. 9** Comparison of the mean best results for CPSO-II algorithm

**Table 3** Comparison of the average number of collisions, average ranks, and average of best results of the CPSO-I algorithm using different chaotic maps

| Map name | Average number of collisions | Average ranks | Average of the best results |
|---|---|---|---|
| Gauss | 23.2 | 4.1 | 10.883 |
| Singer | 22.6 | 3.7 | 10.51 |
| Tent | 21.1 | 5.2 | 11.49 |
| Circle | 19.4 | 4.6 | 11.18 |
| Logistic | 14.6 | 4.9 | 10.77 |
| Sine | 10.4 | 3.7 | 10.60 |
| Piecewise | 10.3 | 5.9 | 10.95 |
| Sinusoidal | 16.2 | 3.9 | 11.02 |

**Table 4** Comparison of the average number of collisions, average ranks, and average of best results of the CPSO-II algorithm using different chaotic maps

| Map name | Average number of collisions | Average ranks | Average of the best results |
|---|---|---|---|
| Gauss | 14.1 | 5.1 | 14.90 |
| Singer | 11.2 | 3.9 | 10.21 |
| Tent | 19.1 | 5.2 | 11.31 |
| Circle | 18.9 | 4.1 | 10.88 |
| Logistic | 22.9 | 4.3 | 10.58 |
| Sine | 12.2 | 3.8 | 10.43 |
| Piecewise | 20.9 | 5.8 | 11.1 |
| Sinusoidal | 15.0 | 3.9 | 10.91 |

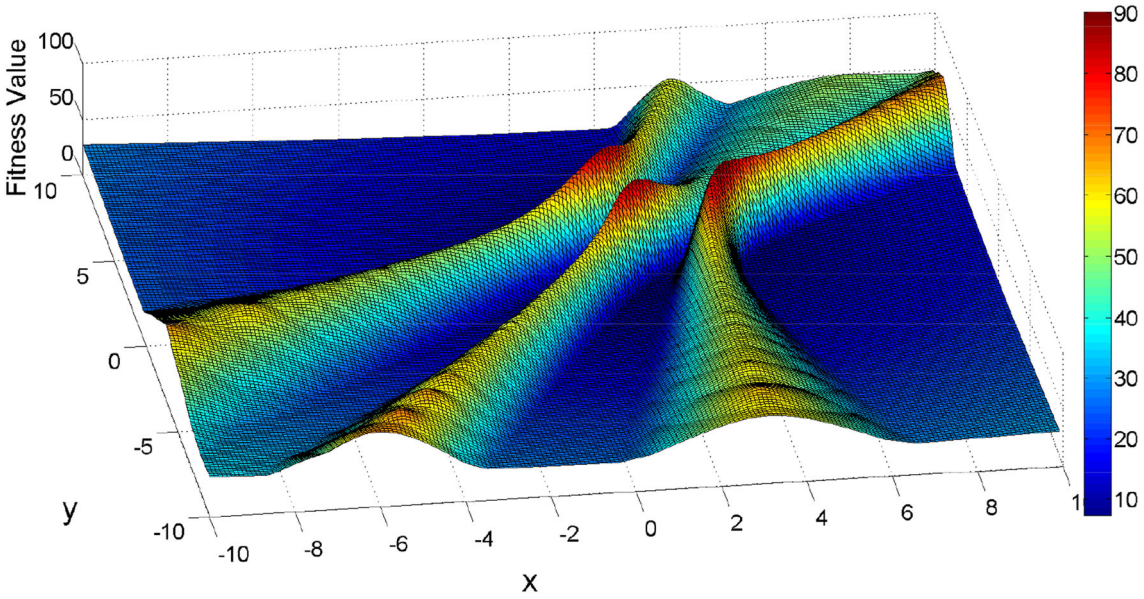



**Fig. 10** Surface plot for the CPSO-II algorithm with Singer map

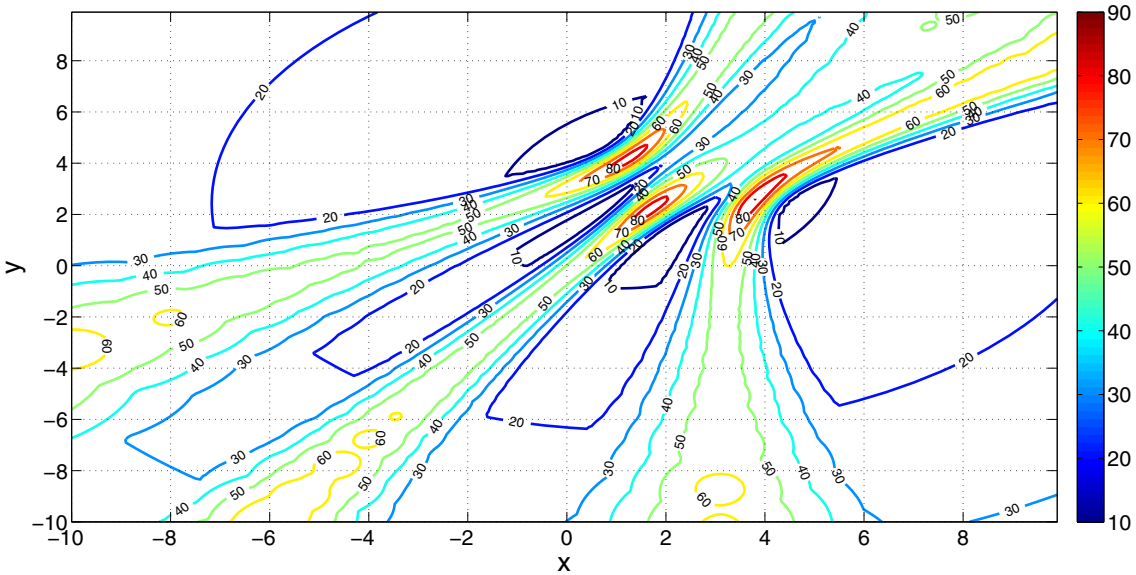



**Fig. 11** Contour plot for the CPSO-II algorithm with Singer map

CPSO-II algorithm with Singer map achieved the minimum number of collisions, while the Logistic map yielded the maximum number of collisions. Moreover, in terms of the average ranks, Singer, Sine, and Sinusoidal maps achieved the best average ranks, and Piecewise map achieved the worst average ranks. In addition, concerning to the average best fitness values, Singer map achieved the best result, and the Gauss map achieved the worst results.

Figures 10 and 11 shows the surface and contour plot of the fitness value of the CPSO-II algorithm with Singer map, where $n = 3$ and the environment has three obstacles. Each node or point in the contour plot or surface represents the position of the control point. In the surfaces figure, the z-axis denotes the fitness value with the control points' positions in $(x, y)$-plane. As shown, the fitness values are approximately like a bowl shape; hence, the global solution(s) can be reached. On the other hand, there are many local optimal solutions; thus, it is not easy to find an optimal path planning. This means that the optimized procedure to search for the optimized control curve is feasible and achieved high results. Figure 12 shows how the solutions are converged to the optimal solution(s) with different runs. As shown, despite the different initial positions of the two solutions, which are indicated by black and red lines, both were converged to two different optimal solutions.

Figures 13 and 14 show the boxplot for the results of the CPSO-I and CPSO-II algorithms. The results in Figs. 13 and 14 are in agreement with our findings. As shown, the Singer map achieved the lowest median. In addition, the median of the results using Gauss map in Fig. 14 was higher than the other maps, this also in agreement with Fig. 9 and reflects that the Gauss map achieved the worst results. Moreover, the interquartile range and overall range which represent the spread of the results, of Singer map was lower than the results of the other maps. This reflects that the stability of the proposed algorithm using Singer map compared with the other maps. As a consequence, the results of Singer map has no outliers as in both Figs. 13 and 14, while results of some of the other maps have outliers. This reflects how the results using Singer map have a high level of agreement with each other and have no outliers, while some of the other results are quite different or unstable.

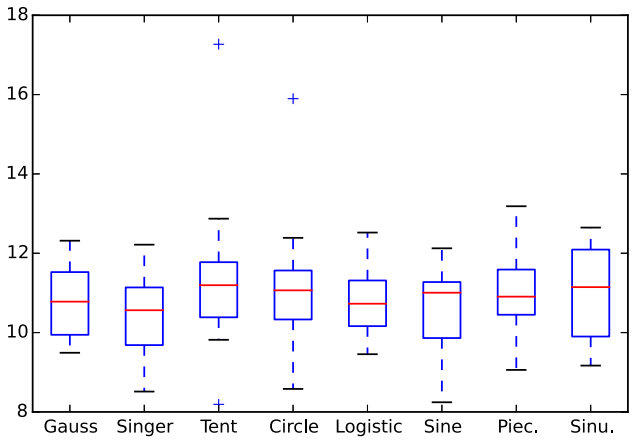



**Fig. 13** Boxplot for the results of CPSO-I algorithm using eight chaotic maps (Piec. for Piecewise map and Sinu. for Sinusoidal map)

To conclude, the Singer map achieved the best results in both CPSO-I and CPSO-II algorithms, and the Sine map

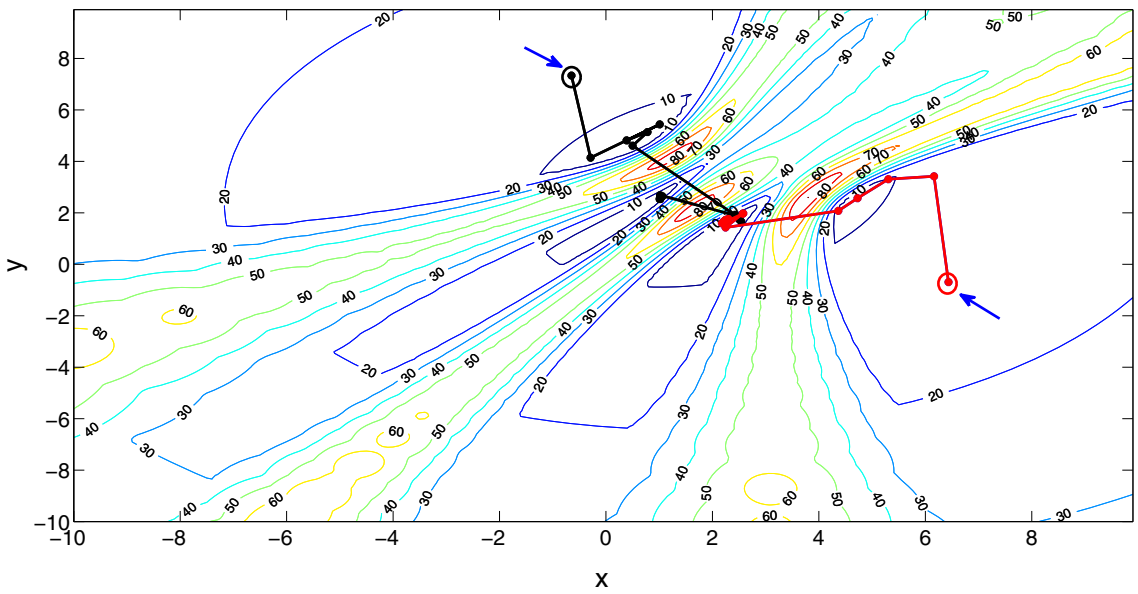



**Fig. 12** Illustration of how the CPSO-II algorithm with Singer map converged to two different optimal solutions using. Blue arrow with a small circle indicates the initial position of both solutions (Color figure online)

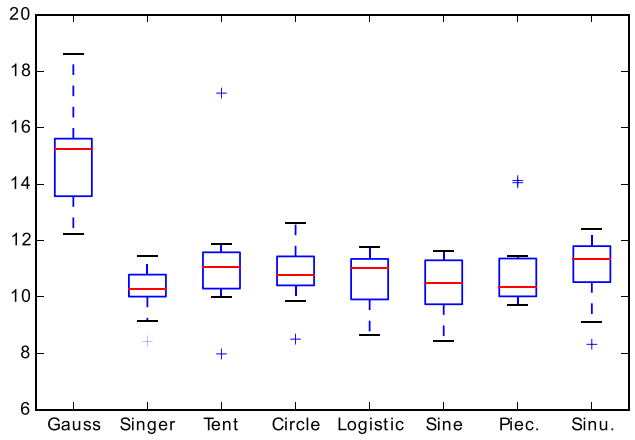



**Fig. 14** Boxplot for the results of CPSO-II algorithm using eight chaotic maps (Piec. for Piecewise map and Sinu. for Sinusoidal map)

achieved the second best results. Moreover, the Tent and Gauss maps achieved the worst results. Based on the results of this experiment, the Singer and Sine maps will be used in the next experiment to compare between the CPSO-I, CPSO-II, and the standard PSO algorithms.

### 5.4 Second experiment: CPSO-I and CPSO-II versus PSO

In this experiment, a comparison between the two proposed algorithms, i.e., CPSO-I and CPSO-II, and the standard PSO algorithm was performed. In this experiment, only the Singer and Sine chaotic maps were used with the CPSO-I and CPSO-II algorithms. As in the first experiment, the initial solutions of the CPSO-I, CPSO-II, and PSO algorithms were identical. Figure 15 shows the means of the results over 60 iterations for the CPSO-I, CPSO-II, and PSO algorithms. Moreover, Table 5 illustrates the average number of collisions, average ranks, and the average of the best results of the CPSO-I, CPSO-II, and PSO algorithms.

As shown from Fig. 15, both CPSO algorithms achieved results better than the standard PSO, which reflects how the chaotic maps improved the standard PSO algorithm. Moreover, the Singer map in both CPSO-I and CPSO-I algorithms achieved results better than the Sine map. In addition, the CPSO-II algorithm with Singer map achieved the best results. Hence, replacing the $r_2$ parameter in CPSO-II algorithm is more efficient than the other algorithms. From Table 5 many conclusions can be drawn.

1. CPSO-I-Sine, PSO, and CPSO-II-Singer achieved a small number of collisions, while the CPSO-I-Singer achieved the maximum number of collisions.
2. The CPSO-II-Sine obtained the minimum average ranks, and also the PSO yielded the highest average ranks.
3. In terms of the average of best fitness values, the CPSO-II-Singer achieved the best results, and the PSO achieved the worst results.

The results of this experiment indicate that the proposed CPSO-I and CPSO-II algorithms improved the performance of the standard PSO algorithm. Thus, replacing the $r_1$ and $r_2$ parameters in the CPSO-I and CPSO-II algorithms, respectively, is more powerful than the standard PSO algorithm. Moreover, the CPSO-II algorithm achieved results better than the CPSO-I algorithm, and the Singer map was the most suitable for CPSO-I and CPSO-II algorithms. It is worth mentioning that, although some

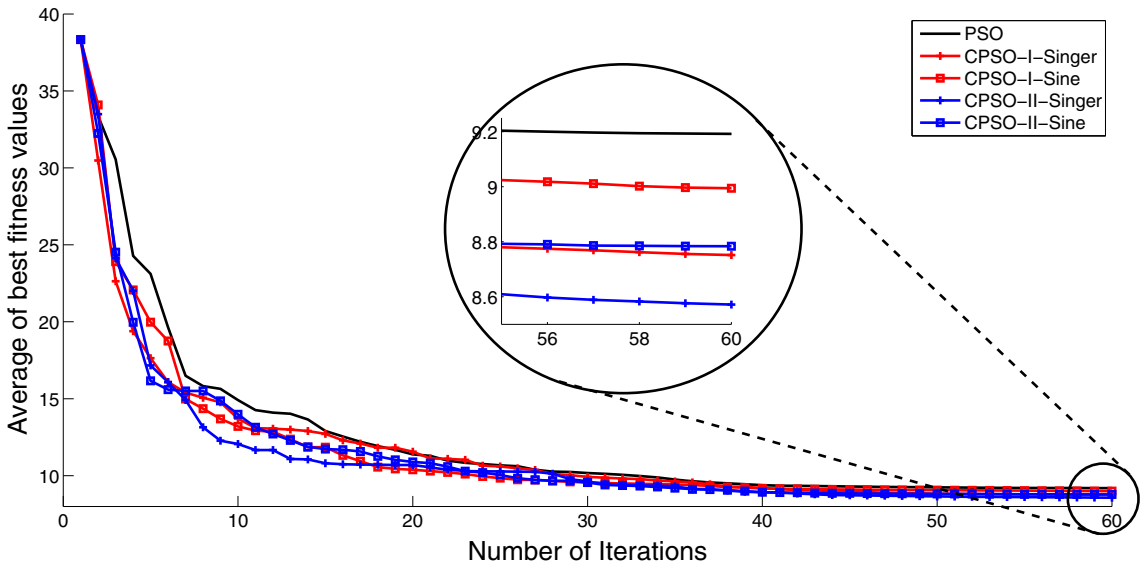



**Fig. 15** Comparison of the mean best results for CPSO-I, CPSO-II, and PSO algorithms

**Table 5** Comparison of the average number of collisions, average ranks, and average of best results of the CPSO-I, CPSO-II, and PSO algorithm

| Algorithm | Average number of collisions | Average ranks | Average of best results |
|---|---|---|---|
| PSO | 11.1 | 3.6 | 11.15 |
| CPSO-I-Singer | 22.6 | 2.6 | 10.51 |
| CPSO-I-Sine | 10.4 | 3.1 | 10.60 |
| CPSO-II-Singer | 11.2 | 2.4 | 10.21 |
| CPSO-II-Sine | 12.2 | 2.1 | 10.43 |

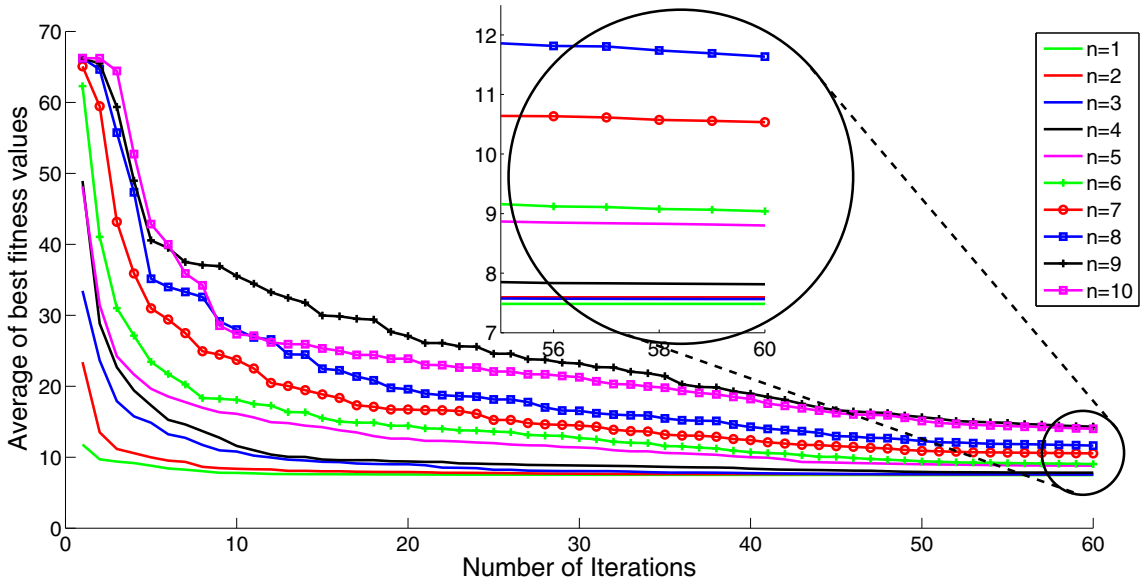
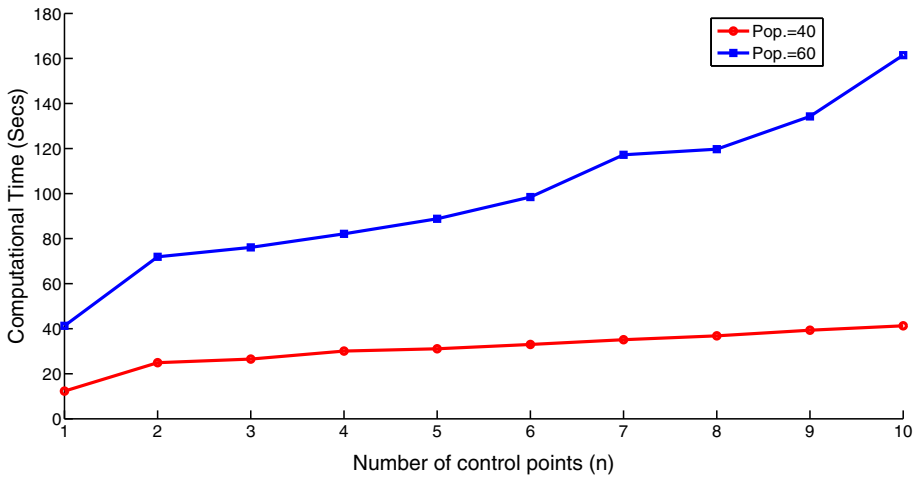



**Fig. 16** The performance of the CPSO-II algorithm with Singer chaotic map using different numbers of control points ($n$) (population size = 20)

**Fig. 17** Computational time of the proposed algorithm CPSO-II with Singer chaotic map using different numbers of control points ($n$) and with population size (pop.) = 20 and 40




chaotic maps improved the performance of the PSO algorithm, some of them are not suitable at all. Based on the results of this experiment, the CPSO-II algorithm with Singer map will be used in the next two experiments.

### 5.5 Third experiment: different numbers of control points

The aim of this experiment is to test the influence of the number of control points on the proposed algorithm CPSO-

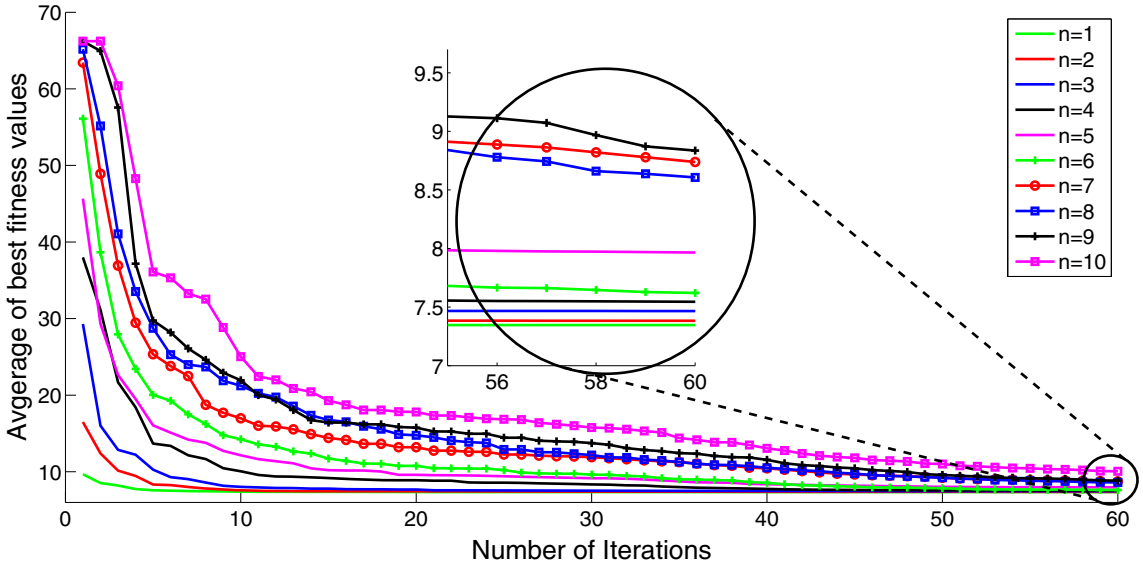



**Fig. 18** The performance of the proposed algorithm CPSO-II with Singer chaotic map using different numbers of control points ($n$) (population size $= 40$)

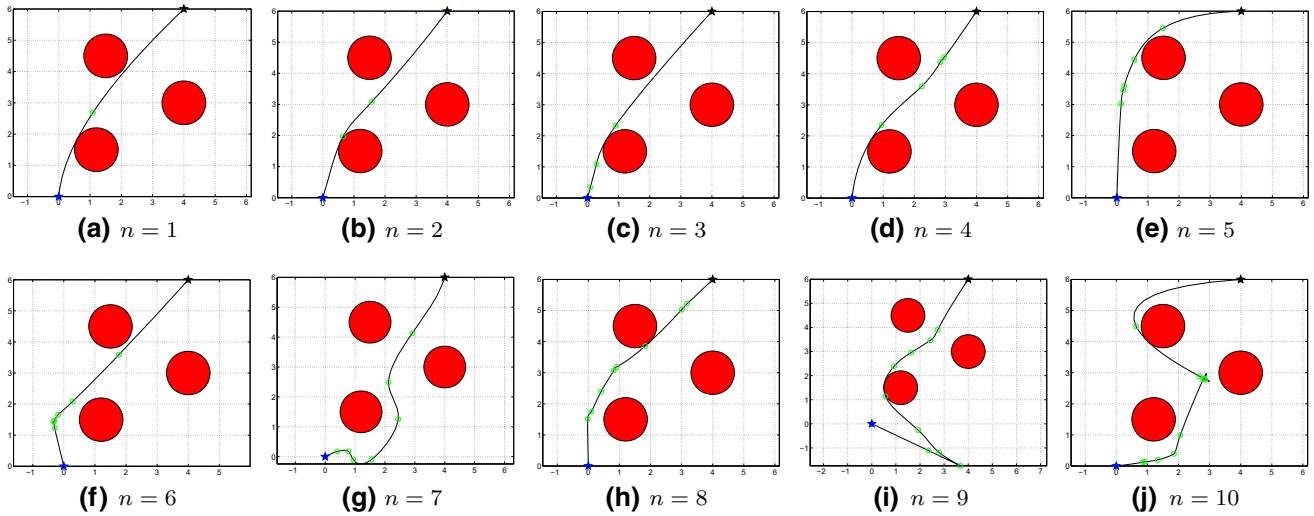



**Fig. 19** Simulation of one run for the proposed CPSO-II algorithm using Singer map with different numbers of control points

II with Singer chaotic map. In this experiment, the number of control points was ranged from one to ten control points. This experiment has two sub-experiments. In the first sub-experiment, the population size was 20. Figures 16 and 17 show the results of this sub-experiment. The number of population size increased to 60 in the second sub-experiment. Figures 17 and 18 present the results of the second sub-experiment.

Figure 16 shows the convergence curve of the CPSO-II algorithm when the number of control points was ranged from one to ten, and when the population size was 20. As shown, the best results achieved when the number of control points was small, i.e., $n = 1, 2, 3,$ or 4 (see Fig. 19a–d, and the results dramatically decreased when the value of $n$ was more than five (see Fig. 19e), and the worst results achieved when $n = 9$ and $n = 10$ (see Fig. 19i and j). A possible explanation for these results might be that increasing the number of control points increased the search space; hence, larger population is required to obtain good results. For this reason, the population size was increased to 40 in the second sub-experiment as shown in Fig. 18. Comparing the results in Figs. 16 and 18, it can be seen that the results when the population size increased to 40, as in Fig. 18, were much better than the results in the

first sub-experiment, as in Fig. 16, when the population size was 20. Moreover, the results when the value of $n$ was five and six improved. However, large search spaces may require a high number of particles which may need more computational time.

Figure 17 provides the computational time of the CPSO-II algorithm using different values of $n$. A shown, the computational time is proportional with the number of control points. Moreover, the computational time rapidly increased when the population size increased from 20 to 40.

In summary, increasing the number of control points increases the search space by adding extra dimensions. Hence, a high number of the population size is required to perfectly scan the search space to find the optimal or near optimal solution(s). However, increasing the search space and the population size requires high computational time.

### 5.6 Fourth experiment: different numbers of obstacles

The goal of this experiment is to test the influence of the number of obstacles on the CPSO-II algorithm with Singer map. In this experiment, the number of obstacles was ranged from one to ten, and all obstacles were in the same size. Moreover, the obstacles were positioned randomly in the search space. Figure 20 shows the convergence curve of the CPSO-II algorithm using different numbers of obstacles. Moreover, Fig. 21 depicts the simulation of the proposed algorithm using different numbers of obstacles. In addition, Table 6 shows a comparison between the proposed algorithm when the number of obstacles was changed.

Many remarks can be drawn from Fig. 20.

1. Increasing the number of obstacles has a negative impact on the obtained results. As shown, the worst results achieved when the number of obstacles was nine and ten (see Fig. 21i and j), and the best results were achieved when small numbers of obstacles were used. This is because increasing the number of obstacles may increase the collisions (see Table 6, Fig. 21), which may increase the distance from the starting to the ending points.
2. As shown in Fig. 21, the CPSO-II algorithm reached to the near optimal solution faster when the number of obstacles was small. In other words, small number of obstacles may result in a straight line (see Fig. 21a and b), i.e., optimal solution, between the starting and ending points; on the other hand, increasing the number of obstacles increases the distance of the optimal curve (see Fig. 21g and i).

As illustrated in Table 6, the number of collisions is proportional with the number of obstacles, and the maximum number of collisions is achieved when the number of obstacles was ten. Moreover, from the table, the best average ranks and average best fitness value were achieved when the number of obstacles was small. It seems possible that these high numbers of collisions are due to the small environment in our experiments. Despite this high collision numbers, our proposed algorithm found the optimal path in most cases, and the number of collisions may be reduced

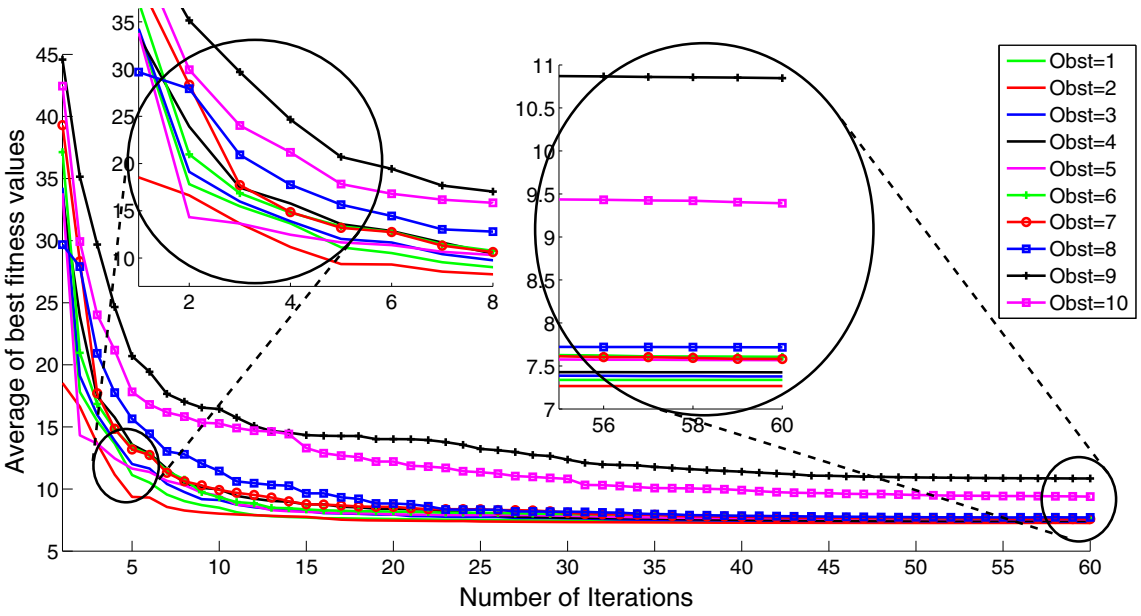



**Fig. 20** The performance of the CPSO-II algorithm with Singer chaotic map using different numbers of obstacles

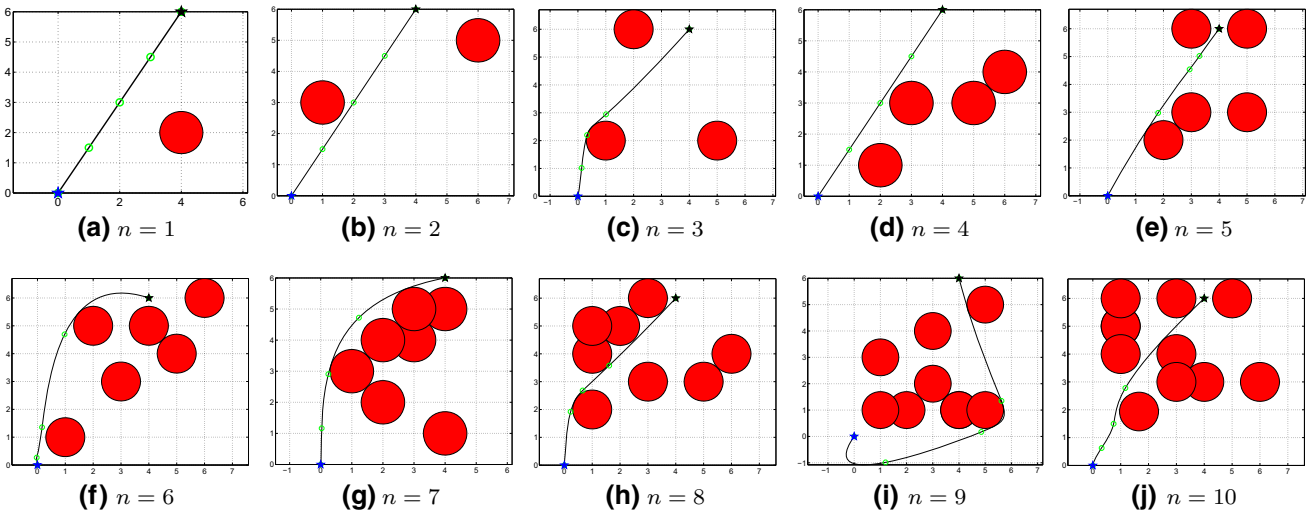



**Fig. 21** Simulation of one run for the proposed CPSO-II algorithm using Singer map with different numbers of obstacles

**Table 6** The results (average number of collisions, average ranks, and average of best fitness values) of the CPSO-II algorithm with Singer map using different numbers of obstacles

| No. of obstacles | Average number of collisions | Average ranks | Average of best results |
|---|---|---|---|
| 1 | 3.7 | 0.5 | 8.04 |
| 2 | 1.6 | 0.5 | 7.71 |
| 3 | 2.5 | 1.3 | 8.22 |
| 4 | 3.1 | 2.5 | 8.45 |
| 5 | 4.7 | 2.4 | 8.29 |
| 6 | 6.0 | 3.6 | 8.57 |
| 7 | 5.3 | 4 | 8.74 |
| 8 | 9.3 | 4.2 | 8.89 |
| 9 | 13.7 | 5.8 | 12.86 |
| 10 | 15.2 | 7.4 | 12.86 |

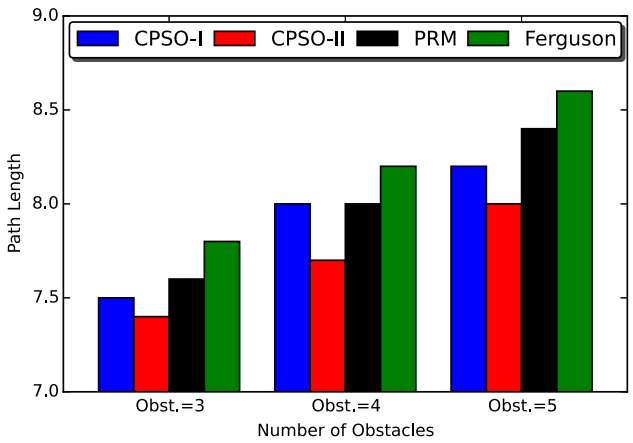



**Fig. 22** Comparison between the proposed algorithms and the PRM and Ferguson curves. The unit of Path length depends on the scale of your problem [in our case the unit is meter (m)]

by extending the space of the environment to be consistent with real applications.

To sum up, increasing the number of obstacles increases the difficulty of the problem and may lead to a high number of collisions which is not feasible for real-time applications.

To further prove that our approach is better than other related work, as illustrated in Fig. 22, a comparison with two of the state-of-the-art algorithms, namely, *Ferguson* curve [33], and *Probabilistic Road Map* (PRM) [48] was conducted. As shown, this experiment was conducted using three, four, and five obstacles. From this figure, it can be remarked that both CPSO-I and CPSO-II achieved path shorter than the PRM and Ferguson curves.

To conclude, the proposed algorithm, i.e., CPSO, achieved results better than the standard PSO algorithm. Moreover, the CPSO-II algorithm achieved results better than the CPSO-I, and Singer map was suitable for both algorithms. In addition, the CPSO-II algorithm revealed competitive results with a high number of obstacles. However, due to the stochastic nature of CPSO algorithms, CPSO algorithms are never guaranteed to find an optimal

solution for any problem, but they will often find a good solution if one exists.

## 6 Conclusions

Path planning is widely used for different applications such as mobile robotics and tracking for nonholonomic vehicles. There are many algorithms used to find the path planning such as Ferguson curve, PRM, and Bézier curve. In this paper, Bézier curve was used for path planning. However, the control points of the Bézier curve have a significant impact on finding the optimal smooth path that minimizes the total distance between the starting and ending points. This study proposes two variants of the Chaotic Particle Swarm Optimization (CPSO), namely, CPSO-I and CPSO-II, algorithms that can be employed to search for the optimal path. In this paper, the first experiment was conducted to test the results of the two proposed algorithms using different chaotic maps. The results of this experiment showed that the Singer and Sine maps were suitable for both algorithms. The second experiment was conducted to compare between the CPSO-I, CPSO-II, and PSO algorithms, and the results of this experiment revealed that (1) the CPSO-II algorithm achieved the best results, and (2) replacing random parameters with chaotic maps in the PSO algorithm improved the performance of the PSO. In the third experiment, the influence of the number of control points was tested. The results proved that increasing the number of control points extends the search space by adding extra dimensions; and hence, a high number of population size for the PSO are required to scan the search space perfectly. In the last experiment, the CPSO-II algorithm was tested against different numbers of obstacles, and the results proved that increasing the number of obstacles increases the difficulty of the problem. However, the proposed algorithm found the optimal path in most cases.

Several directions for future studies are suggested. First, the experiments in this paper were performed using only a static environment, but a dynamic environment, i.e., moving obstacles, should be tested in the future to verify and extend the proposed algorithm. Second, searching for the optimal path in a three-dimensional environment should be tested in the future to simulate real robots' movements.

## Compliance with ethical standards

**Conflict of interest** The authors declare that there is no conflict of interests regarding the publication of this paper.

## References


1. Elhoseny, M., Tharwat, A., Farouk, A., Hassanien, A.E.: K-coverage model based on genetic algorithm to extend wsn lifetime. IEEE sensors letters **1**(4), 1–4 (2017)
2. Li, R., Wu, W., Qiao, H.: The compliance of robotic hands-from functionality to mechanism. Assem. Autom. **35**(3), 281–286 (2015)
3. Robinson, D.C., Sanders, D.A., Mazharsolook, E.: Ambient intelligence for optimal manufacturing and energy efficiency. Assem. Autom. **35**(3), 234–248 (2015)
4. Manikas, T.W., Ashenayi, K., Wainwright, R.L.: Genetic algorithms for autonomous robot navigation. IEEE Instrum. Meas. Mag. **10**(6), 26–31 (2007)
5. Metawa, N., Hassan, M.K., Elhoseny, M.: Genetic algorithm based model for optimizing bank lending decisions. Expert Syst. Appl. **80**, 75–82 (2017)
6. Elhoseny, M., Shehab, A., Yuan, X.: Optimizing robot path in dynamic environments using genetic algorithm and bezier curve. J. Intell. Fuzzy Syst. **33**(4), 2305–2316 (2017)
7. Tharwat, A.: Linear vs. quadratic discriminant analysis classifier: a tutorial. Int. J. Appl. Pattern Recognit. **3**(2), 145–180 (2016)
8. Elhoseny, M., Tharwat, A., Hassanien, A.E.: Bezier curve based path planning in a dynamic field using modified genetic algorithm. J. Comput. Sci. (2017). https://doi.org/10.1016/j.jocs.2017.08.004
9. Roberge, V., Tarbouchi, M., Labonté, G.: Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. IEEE Trans. Ind. Inform. **9**(1), 132–141 (2013)
10. Contreras-Cruz, M.A., Ayala-Ramirez, V., Hernandez-Belmonte, U.H.: Mobile robot path planning using artificial bee colony and evolutionary programming. Appl. Soft Comput. **30**, 319–328 (2015)
11. Das, P., Behera, H., Panigrahi, B.: A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evol. Comput. **28**, 14–28 (2016)
12. Gálvez, A., Iglesias, A., Cabellos, L.: Tabu search-based method for Bézier curve parameterization. Int. J. Softw. Eng. Appl. **7**, 283–296 (2013)
13. Li, B., Liu, L., Zhang, Q., Lv, D., Zhang, Y., Zhang, J., Shi, X.: Path planning based on firefly algorithm and Bezier curve. In: IEEE International Conference on Information and Automation (ICIA), IEEE, pp. 630–633 (2014)
14. Arana-Daniel, N., Gallegos, A.A., López-Franco, C., Alanis, A.Y.: Smooth global and local path planning for mobile robot using particle swarm optimization, radial basis functions, splines and Bezier curves. In: IEEE Congress on Evolutionary Computation (CEC), IEEE, pp. 175–182 (2014)
15. Ziolkowski, M., Gratkowski, S.: Genetic algorithm coupled with Bézier curves applied to the magnetic field on a solenoid axis synthesis. Arch. Electr. Eng. **65**(2), 361–370 (2016)
16. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning. Springer, New York, pp. 760–766 (2010)
17. Maitra, M., Chatterjee, A.: A hybrid cooperative-comprehensive learning based pso algorithm for image segmentation using multilevel thresholding. Expert Syst. Appl. **34**(2), 1341–1350 (2008)
18. Ibrahim, A., Tharwat, A., Gaber, T., Hassanien, A.E.: Optimized superpixel and adaboost classifier for human thermal face recognition. Signal Image Video Process. (2017). https://doi.org/10.1007/s11760-017-1212-6

19. Tharwat, A., Hassanien, A.E., Elnaghi, B.E.: A ba-based algorithm for parameter optimization of support vector machine. Pattern Recogn. Lett. **93**, 13–22 (2017)
20. Tharwat, A., Gaber, T., Ibrahim, A., Hassanien, A.E.: Linear discriminant analysis: a detailed tutorial. AI Commun. **30**(2), 169–190 (2017)
21. Subasi, A.: Classification of emg signals using pso optimized svm for diagnosis of neuromuscular disorders. Comput. Biol. Med. **43**(5), 576–586 (2013)
22. Van der Merwe, D., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, CEC'03, vol. 1., IEEE, pp. 215–220 (2003)
23. Tharwat, A.: Principal component analysis-a tutorial. Int. J. Appl. Pattern Recogn. **3**(3), 197–240 (2016)
24. Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Congress on Evolutionary Computation, CEC2004, vol. 2, IEEE, pp. 1980–1987 (2004)
25. Miyatake, M., Veerachary, M., Toriumi, F., Fujii, N., Ko, H.: Maximum power point tracking of multiple photovoltaic arrays: a pso approach. IEEE Trans. Aerosp. Electron. Syst. **47**(1), 367–380 (2011)
26. Molazei, S., Ghazizadeh-Ahsaee, M.: Mopso algorithm for distributed generator allocation. In: Fourth International Conference on Power Engineering, Energy and Electrical Drives (POWERENG), IEEE, pp. 1340–1345 (2013)
27. Gandomi, A.H., Yang, X.S.: Chaotic bat algorithm. J. Comput. Sci. **5**(2), 224–232 (2014)
28. Wang, G.G., Guo, L., Gandomi, A.H., Hao, G.S., Wang, H.: Chaotic krill herd algorithm. Inf. Sci. **274**, 17–34 (2014)
29. Gharooni-fard, G., Moein-darbari, F., Deldari, H., Morvaridi, A.: Scheduling of scientific workflows using a chaos-genetic algorithm. Proc. Comput. Sci. **1**(1), 1445–1454 (2010)
30. Talatahari, S., Azar, B.F., Sheikholeslami, R., Gandomi, A.: Imperialist competitive algorithm combined with chaos for global optimization. Commun. Nonlinear Sci. Numer. Simul. **17**(3), 1312–1319 (2012)
31. Gandomi, A., Yang, X.S., Talatahari, S., Alavi, A.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
32. Ma, Y., Zamirian, M., Yang, Y., Xu, Y., Zhang, J.: Path planning for mobile objects in four-dimension based on particle swarm optimization method with penalty function. In: Mathematical Problems in Engineering (2013)
33. Liang, J., Song, H., Qu, B., Liu, Z.: Comparison of three different curves used in path planning problems based on particle swarm optimizer. In: Mathematical Problems in Engineering (2014)
34. Sahingoz, O.K.: Generation of Bezier curve-based flyable trajectories for multi-uav systems with parallel genetic algorithm. J. Intell. Robotic Syst. **74**(1–2), 499–511 (2014)
35. Gardner, B., Selig, M.: Airfoil design using a genetic algorithm and an inverse method. In: 41st Aerospace Sciences Meeting and Exhibit, pp. 1–12 (2003)
36. Jolly, K., Kumar, R.S., Vijayakumar, R.: A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. Robot. Auton. Syst. **57**(1), 23–33 (2009)
37. Giannakoglou, K.: A design method for turbine-blades using genetic algorithms on parallel computers. Comput. Fluid Dyn. **98**(1), 1–2 (1998)
38. Chen, L., Wang, S., Hu, H., McDonald-Maier, K.: Bézier curve based trajectory planning for an intelligent wheelchair to pass a doorway. In: International Conference on Control (CONTROL), IEEE, pp. 339–344 (2012)
39. Choi, J.w., Curry, R., Elkaim, G.: Path planning based on Bézier curve for autonomous ground vehicles. In: Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science, (WCECS'08), IEEE, pp. 158–166 (2008)
40. Wagner, R., Birbach, O., Frese, U.: Rapid development of manifold-based graph optimization systems for multi-sensor calibration and slam. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 3305–3312 (2011)
41. Heppner, F., Grenander, U.: A stochastic nonlinear model for coordinated bird flocks. Ubiquity Chaos **99**, 233–238 (1990)
42. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. ACM Siggraph Comput. Graph. **21**(4), 25–34 (1987)
43. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, vol. 1., New York, pp. 39–43 (1995)
44. Yang, X.S.: Nature-Inspired Optimization Algorithms, 1st edn. Elsevier, Amsterdam (2014)
45. Ren, B., Zhong, W.: Multi-objective optimization using chaos based pso. Inf. Technol. J. **10**(10), 1908–1916 (2011)
46. Vohra, R., Patel, B.: An efficient chaos-based optimization algorithm approach for cryptography. Commun. Netw. Secur. **1**(4), 75–79 (2012)
47. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
48. Masehian, E., Sedighizadeh, D.: A multi-objective pso-based algorithm for robot path planning. In: Proceedings of IEEE International Conference on Industrial Technology (ICIT), IEEE, pp. 465–470 (2010)


**Alaa Tharwat** is a lecturer at the Faculty of Engineering, Suez Canal University, Ismailia, Egypt. His research interest includes evolutionary computing, image processing, and quantum computing. He has several Publications in reputed and high impact journals like Nature Scientific Reports, QINP, and Security and Communications. He has publications in international conferences held by IEEE, Springer and ACM.

**Mohamed Elhoseny** received his Ph.D. in Computer and Information Sciences from Mansoura University Egypt (in a scientific research channel with Department of Computer Science and Engineering, University of North Texas, USA ). His Ph.D. thesis was awarded the best Ph.D. thesis prize (2016) at Mansoura University. Dr. Elhoseny is currently an Assistant Professor at the Faculty of Computers and Information, Mansoura University, Egypt Collectively, Dr. Elhoseny authored/co-authored over 50 International Journal articles, Conference Proceedings, Book Chapters, and 1

Springer brief book. His research interests include Network Security, Cryptography, Machine Learning Techniques, Internet of Things, and Quantum Computing. He has several publications in reputed and high impact journals published by IEEE, Elsevier, Springer, and others. Dr. Elhoseny is a TPC Member or Reviewer in 30+ International Conferences and Workshops. Furthermore, he has been reviewing papers for 20+ International Journals.

**Aboul Ella Hassanien** is a Professor at Cairo University, Faculty of Computers & Information. He is the founder and Chair of the Scientific Research Group in Egypt (SRGE). He is the Ex-Dean of faculty of computers and Information, Beni-Suef University, Egypt.

**Thomas Gabel** is a Professor for Computer Science and Mathematics with focus on machine learning and artificial intelligence at the Faculty of Computer Science and Engineering at Frankfurt University of Applied Sciences. Thomas' interests in research and development cover various aspects of learning and intelligent systems. He is fascinated by the idea of creating adaptive software that learns autonomously how to solve a task or how to support a human in a smart way. He has focused on both, theoretical foundations of machine learning, multi-agent systems and decentralized control as well as on the practical implementation and deployment of adaptive agent-based applications in domains like recommender systems, mobile applications, robotics, or computer games.

**Arun Kumar** has completed in his B.E., M.E. and Ph.D. in Electronics and Communication Engineering with specialization in Biomedical Engineering. He has a strong academic teaching and research experience of more than 10 years in SASTRA University, India. He is appreciated for his innovative research oriented teaching related practical life experiences to the principles of engineering. He is active in research and has been giving directions to active researchers across the globe.