Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 5126808, 14 pages
http://dx.doi.org/10.1155/2016/5126808

Research Article

A Novel Chaotic Particle Swarm Optimization Algorithm for Parking Space Guidance

Na Dong,[1] Xing Fang,[2] and Ai-guo Wu[1]

[1]*School of Electrical Engineering and Automation, Tianjin University, Tianjin 300072, China*
[2]*Key Laboratory of Advanced Process Control for Light Industry of the Ministry of Education, School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China*

Correspondence should be addressed to Na Dong; dongna@tju.edu.cn

Received 5 March 2016; Revised 21 July 2016; Accepted 26 July 2016

Academic Editor: George S. Dulikravich

An evolutionary approach of parking space guidance based upon a novel Chaotic Particle Swarm Optimization (CPSO) algorithm is proposed. In the newly proposed CPSO algorithm, the chaotic dynamics is combined into the position updating rules of Particle Swarm Optimization to improve the diversity of solutions and to avoid being trapped in the local optima. This novel approach, that combines the strengths of Particle Swarm Optimization and chaotic dynamics, is then applied into the route optimization (RO) problem of parking lots, which is an important issue in the management systems of large-scale parking lots. It is used to find out the optimized paths between any source and destination nodes in the route network. Route optimization problems based on real parking lots are introduced for analyzing and the effectiveness and practicability of this novel optimization algorithm for parking space guidance have been verified through the application results.

1. Introduction

Recently, there are a large number of research works in the area of parking management systems [1–3]. Mei and Tian proposed a new parking guidance and information (PGI) configuration model based on the optimized combination method by analyzing of parking choice behavior [1]. Thompson et al. [2] described a behavioral model of parking choice incorporating drivers' perceptions of waiting times at car parks based on PGI signs and formulated a mathematical program to determine the optimal display PGI sign configuration and finally applied it to an existing PGI system operating in Tama New Town near Tokyo. Lee et al. [3] designed and implemented a hybrid artificial intelligent control scheme for a car-like vehicle to perform the task of optimal parking. In the literature, there have been several algorithms to solve different path optimization problems, but they all have their own shortcomings [4]. These traditional algorithms have some major shortcomings, as they only search for the shortest route but cannot determine any other similar/nonsimilar short routes; also, they exhibit high computational complexity when solving real-time problems [5]. As the mostly used A^* and Dijkstra algorithms, A^* is a quite efficient and widely used path optimization algorithm, but the users need to balance its $h(n)$ and $g(n)$ well in order to get a most satisfying result; Dijkstra algorithm is also widely used, but it is relatively slow in searching [6–8]. Artificial neural networks (ANN) are also successfully used to solve the path optimization problems [9, 10]. However, the ANN approaches also suffer from some limitations. The complexity of the hardware increases considerably with increasing number of network nodes, while decreasing the reliability of the solution. Also, suboptimal paths cannot be obtained by the ANN approaches [5]. In the pursuit for more efficient algorithms, evolutionary programming techniques have been widely investigated among other methodologies for this problem and successful applications have been reported [11–14]. The success of these evolutionary programming approaches inspires us to investigate the use of other more powerful evolutionary algorithms for this kind of problems. And in this paper, we explore a more efficient and accurate way, based on heuristic optimizations,

to solve the RO problem of the parking lot management system. This study of the parking inducing problem aimed at specific objectives, namely, searching the parking spaces with the shortest distance by a newly proposed Chaotic Particle Swarm Optimization algorithm.

Particle Swarm Optimization (PSO) technique is considered as one of the modern stochastic search algorithms for optimization introduced by Kennedy and Eberhart [15]. Due to its simple concept, easy implementation, and quick convergence, nowadays, PSO has gained much attention and wide applications in different fields. For examples, Sun [16] applied PSO to roundness measurement under a machine vision system and Omran et al. [17] proposed a PSO-based image clustering method. However, the standard PSO greatly depends on its parameters and exists as a premature phenomenon, especially in solving complex problems [18]. Also, PSO has an excessive decrease of particle variety as the number of generations increases and the swarm becomes stagnated after a certain number of iterations, which means it lacks the ability to achieve sustainable development. In order to keep the particles from the stagnation state and improve the PSO algorithm, the chaos optimization mechanism is introduced in this paper and a novel chaotic optimization algorithm is proposed.

Chaos is a kind of characteristic of nonlinear systems and chaotic motion can traverse every state in a certain region by its own regularity and, nowadays, has been applied in different fields [19, 20]. Due to the unique ergodicity and special ability in avoiding being trapped in local optima, chaos search has been proved to be much more efficient than some other stochastic algorithms [21], and chaotic systems have been exploited in some metaheuristic methods to solve global optimization problems with a large number of local minima. By making use of chaotic behaviors, those optimization methods search extensively for solutions and are able to find a desirable solution within a practical time. It has been indicated that a chaotic variable has three basic traits, namely, pseudorandomness, ergodicity, and irregularity, which makes it generally exhibit better numerical performance than random operators in searching [22]. A lot of existing optimization results demonstrated that chaos optimization algorithms (COAs) and hybrid COA can more easily escape from local minima than classical stochastic optimization algorithms and they have been proven to be more effective than stochastic algorithms [23–25]. Recently, several attempts for evolutionary optimization algorithms using chaos methods were made [26–28] and obtained rich harvests. Xie et al. [26] introduced chaos into the system by randomly reinitializing the particle positions with a small constant probability. Liu et al. [27] incorporated chaos into PSO with adaptive inertia weight factor to construct a Chaotic PSO. Chauhan et al. [28] investigated a Chaotic PSO, namely, Totally Disturbed Particle Swarm Optimization (TDPSO), and employed this enhanced variant of PSO for obtaining the optimal machining conditions during multipass turning operations subject to various constraints. In this paper, the notion of chaos is introduced into the PSO's position updating rules and the novel chaotic optimization algorithm is then applied to solve the parking space guidance problems. Simulation tests are conducted based on a real parking lot and the effectiveness of the proposed method has been revealed through the simulation results.

2. Chaotic Particle Swarm Optimization

PSO is an optimization technique which maintains a population of individuals, namely, particles, where each particle is guided by the social interaction in order to reach the most promising area of the search space. The particles start at a random initial position in a multidimensional search space and search for the minimum or maximum of a given objective function by flying through the search space. Each particle flies in D-dimensional problem space with a velocity, which is adjusted at each time step. The movement of ith particle, x_i, depends on its velocity, v_i, and the location where the personal best position so far, $p_i = (p_{i1}, p_{i2}, \ldots, p_{id})$, or the neighborhood best position, $lbest_i = (lbest_{i1}, lbest_{i2}, \ldots, lbest_{id})$, has already been found. Equation (1) updates the velocity of each particle, whereas (2) updates each particle's position in the search space, and c_1 and c_2 are cognitive coefficients and φ_1 and φ_2 are two uniform random numbers from $U(0, 1)$:

$$v_{id} = w \cdot v_{id} + c_1 \cdot \varphi_1 \cdot \left(p_{id} - x_{id}\right) + c_2 \cdot \varphi_2 \cdot \left(lbest_{id} - x_{id}\right), \tag{1}$$

$$x_{id} = x_{id} + v_{id}. \tag{2}$$

Here, in order to enrich the search behavior, the chaotic dynamics is incorporated into the optimization algorithm.

2.1. Incorporating Chaotic Dynamics into PSO. Since it gives the uniform distribution function in the interval $[0, 1]$, the tent map shows outstanding advantages and higher iterative speed than the logistic map [29]. In this paper, the tent map is used to generate chaos variables. The tent map is defined by

$$z_{n+1} = \mu\left(1 - 2\left|z_n - 0.5\right|\right), \quad 0 \le z_0 \le 1,\ n = 0, 1, 2, \ldots, \tag{3}$$

where $u \in (0, 1)$ is the bifurcation parameter. Specifically, when $\mu = 1$, the tent map exhibits entirely chaotic dynamics and ergodicity in the interval $[0, 1]$. Figure 1 shows the distribution of two chaos sequences after 1500 iterations with the initial point ($x_{10} = 0.231$, $x_{20} = 0.356$) in 2D space. Each point in Figure 1 can be described by (x_{1j}, x_{2j}), $j = 1, 2, \ldots, 1500$.

The chaotic dynamic is used for initialization, which can be described as follows.

Using the tent map ($\mu = 1$) to generate the chaos variables and rewriting (3) give

$$z_j^{(i+1)} = \mu\left(1 - 2\left|z_j^{(i)} - 0.5\right|\right), \quad j = 1, 2, \ldots, D, \tag{4}$$

where z_j denotes the jth chaos variable and i denotes the chaos iteration number. Set $i = 0$ and generate D chaos variables by (4). After that, let $i = 1, 2, \ldots, N$ in turn and

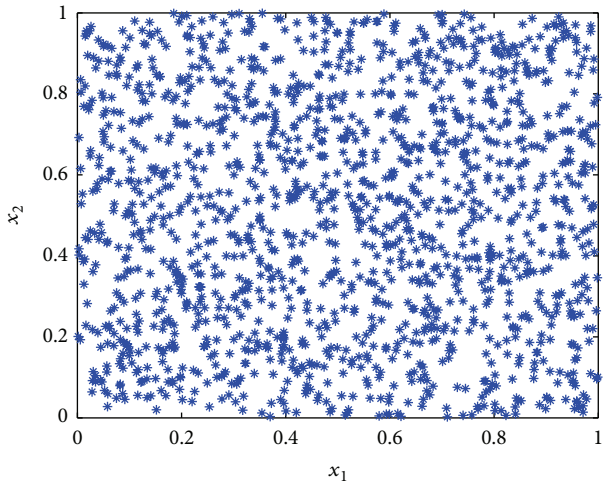


Figure 1: Distribution of the chaos variables.

generate the initial swarm. Then, the chaos variable above $z_j^{(i)}$, $i = 1, 2, \ldots, N$, will be mapped into the search range of the decision variable:

$$x_{ij} = x_{\min,j} + z_j^{(i)} \left(x_{\max,j} - x_{\min,j} \right), \quad j = 1, 2, \ldots, D. \tag{5}$$

Defining

$$x_i = \left(x_{i1}, x_{i2}, \ldots, x_{iD} \right), \quad i = 1, 2, \ldots, N, \tag{6}$$

the chaotic initialized swarm can be obtained.

According to the chaotic search, a chaotic disturbance can be added as follows:

$$z' = (1 - \gamma)\, \psi^* + \gamma z, \tag{7}$$

where γ is a parameter that falls in $[0, 1]$, $z' = (z_1', z_2', \ldots, z_D')$ is the chaos vector where the disturbance has been added, and ψ^* is the optimal chaos vector where the current optima $x^* = (x_1^*, x_2^*, \ldots, x_D^*)$ are mapped into the interval $[0, 1]$:

$$\psi^* = \frac{x^* - x_{\min}}{x_{\max} - x_{\min}}. \tag{8}$$

The Chaotic PSO algorithm is described as follows.

Step 1. Initialize the swarm ($k = 0$) by the abovementioned chaotic initialization method according to (4)–(6).

Step 2. Evaluate the fitness f_i and update p_i and $lbest_i$ if needed.

Step 3. Update the velocity v_i and position x_i of the particle by (1) and (2), respectively.

Step 4. Rank the swarm by fitness in an ascending order. If the current iteration $k \le (1/2)k_{\max}$ and the current optimum $f(p_i)$ stays the same for several iterations, go to Step 5; otherwise, turn to Step 6.

Step 5. Introduce the chaotic disturbance to update the best particle. Add a chaotic disturbance to the bottom 30% of particles in the swarm by (7), and, then, map them into the decision variables by (5). Rerank the swarm by fitness and find out the new best particle.

Step 6. If a stopping criterion is satisfied, stop; otherwise, let $k = k + 1$ and go back to Step 2.

2.2. Benchmark Function Tests. In order to test the performance of the proposed Chaotic PSO algorithm, minimization problems of four multimodal functions [30, 31], the Ellipsoidal function (f_1), the Rosenbrock function (f_2), the generalized Griewank function (f_3), and Rastrigin's function (f_4), are introduced for simulation tests:

$$\begin{aligned}
f_1(x) &= \sum_{i=1}^{D} i x_i^2, \quad -100 \le x_i \le 100, \\
f_2(x) &= \sum_{i=1}^{D-1} 100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2, \\
&\qquad -30 \le x_i \le 30, \\
f_3(x) &= \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \\
&\qquad -600 \le x_i \le 600, \\
f_4(x) &= \sum_{i=1}^{D} \left(x_i^2 - 10 \cos(2\pi x_i) + 10 \right), \\
&\qquad -5.12 \le x_i \le 5.12.
\end{aligned} \tag{9}$$

For both functions, the parameters are fixed as $m = 20$, $k_{\max} = 1500$. The fitness value is set as the function value. The standard PSO, PSO with mutation (MPSO) [32], and GA are introduced here for comparison tests. For each function, different dimensions (10, 20, and 30) are tested. $c_1 = c_2 = 2$, and ω is fixed as 0.5 for all PSO algorithms. The GA is real code, crossover probability $p_c = 0.8$, and mutation probability $p_m = 0.1$. Table 1 lists the mean minimum of 50 independent runs.

From Table 1, it can be seen that the proposed Chaotic PSO has better accuracy and is superior to other optimization algorithms in terms of the searching solutions.

3. Application in Parking Space Guidance

In this research, there are several assumptions about the workspace of the parking lots:

(1) Vehicles move in a limited two-dimensional space.

(2) The paths in parking lots are equivalent to "straight line" and a vehicle is equivalent to a "particle."

(3) It is assumed that there is one entry and one exit, with several free parking stalls. The weight of side corresponds to the length between two continuous

Table 1: Comparison results of mean fitness for f_1 and f_2.

f	D	Algorithms			
		PSO	MPSO	Chaotic PSO	GA
f_1	10	$5.49E-28$	$3.67E-29$	**2.63E − 29**	$6.81E-03$
	20	$3.28E-28$	$5.64E-30$	**2.152E − 33**	$3.12E-03$
	30	$8.26E-27$	**6.28E − 33**	$8.20E-33$	0.0582
f_2	10	24.3651	11.5834	**4.3021**	35.6902
	20	66.2781	17.6590	**15.0153**	84.6521
	30	268.1459	96.4923	**80.2643**	329.5826
f_3	10	0.0812	0.0538	**0.0237**	0.0953
	20	0.0350	0.0031	**1.1506E − 6**	0.0767
	30	0.0218	$5.2311E-5$	**1.3451E − 7**	0.0388
f_4	10	6.432	3.6253	2.463	**2.329**
	20	12.325	8.7524	**7.6038**	23.3028
	30	21.632	17.849	**12.168**	48.943

```
Input: Particles: P1(x0, x1, ..., x_{m-1}) and P2(y0, y1, ..., y_{m-1})
Output: P1 − P2 = d ∈ Integers
(1)  cnt = 0, n = 0
(2)  for i = 0; i < m − 1; i++
(3)      for j = 0; j < m − 1; j++
(4)          if x_i == y_j then
(5)              cnt = cnt + 1
(6)          end if
(7)      end for
(8)      if cnt == 0 then
(9)          n++
(10)     end if
(11) end for
(12) return n
```

Algorithm 1: Pseudocode of the proposed method to find the difference between any two particles.

points. All the points are numbered from 1 to n. The issue about parking guidance is to find a collection of points $P = \{S, p_1, p_2, \ldots, p_n, G\}$, where S stands for the starting point and G for the ending point; $P = \{p_1, p_2, \ldots, p_n\}$ is a sequence of points, named planning target. There are some requests to p_i. Firstly, p_i is a nonbarrier point. Secondly, there are no barrier points on the line connecting p_i and its adjacent point.

3.1. Problem Description. Let us consider a route network $L(Q, E)$, where Q is the set of parking stalls and E is the set of edges. Each edge is represented by (i, j), where $i, j \in Q$ and i is the starting node and j is its adjacent node. The paths should be found from the source s to the destination node d. Any path is represented by $P_t = \{(s, i), (i, j), \ldots, (q, d)\}$, $(s, i), (i, j), \ldots, (q, d) \in E$ and $s, i, j, \ldots, q, d \in Q$. In the proposed work here, each edge (i, j) is associated with one parameter $l(i, j)$, which represents the average length for the vehicle to travel on the edge (i, j).

The route optimization problem in the parking lots can be then defined as

$$\text{minimize} \quad \left(\sum_{(i,j) \in P_t} l(i, j) \right). \tag{10}$$

3.2. Proposed Chaotic Optimization Algorithm for Parking Space Guidance. Let us consider a swarm of N particles which is represented by

$$S^m = \{P_1, P_2, P_3, \ldots, P_N\}. \tag{11}$$

Any particle $P_i \in S^m$, for $i = 1$ to N, is similar to P_t which was described in the previous subsection.

The proposed work assumes an m-dimensional search space represented by R^m, where m is the total number of the parking points. The objective function value of any particle (P_i) can be determined by using the following formula:

$$\text{Obj}(P_i) = \sum_{j=0}^{m-2} l\left(p_j, p_{j+1}\right). \tag{12}$$

(p_j, p_{j+1}) represents an edge $e_x \in E$, whose starting node is p_j and ending node is p_{j+1}. In PSO, the positions of particles change iteratively during the optimization process by adding the velocities into their current positions. The velocity of any particle (P_i) at time or iteration t is represented as $v_{P_i}(t)$. The particles' velocities are also updated iteratively. In PSO, the best position of any particle can be decided by

$$\arg\min_k \left(\text{Obj}\left(P_i(k)\right)\right), \tag{13}$$

where k represents the iteration count. The global best position (k) can be obtained as

$$gbest(k) = \arg\min_k \left(pbest\left(P_i(k)\right)\right), \quad \text{for } i = 1, 2, \ldots, N. \tag{14}$$

In the kth iteration, the velocity of each particle for the next iteration $k + 1$ can be calculated as

$$v_{P_i}(k+1) = w v_{P_i}(k) + c_1 r_1 \left[pbest\left(P_i(k)\right) - P_i(k)\right] + c_2 r_2 \left[gbest(k) - P_i(k)\right], \quad \text{for } i = 1, 2, \ldots, N. \tag{15}$$

c_1 and c_2 are real numbers between $[0, 2]$. r_1 and r_2 are randomly generated real numbers between $[0, 1]$.

The calculation of the velocities here is a little different from the traditional PSO, and it involves the notion of the difference between two particles. The method proposed to find the difference between any two particles, $P_1(x_0, x_1, \ldots, x_{m-1})$ and $P_2(y_0, y_1, \ldots, y_{m-1})$, is shown in Algorithm 1. The idea behind the proposed method is that it returns the number of nodes which exist in the particle P_1 but do not exist in the particle P_2.

Now we know that, in the proposed approach, the velocity of the particle for the next iteration which is represented by $v_{P_i}(k + 1)$ is a positive real number. And after the velocities are determined, they will be used to update the particles' positions.

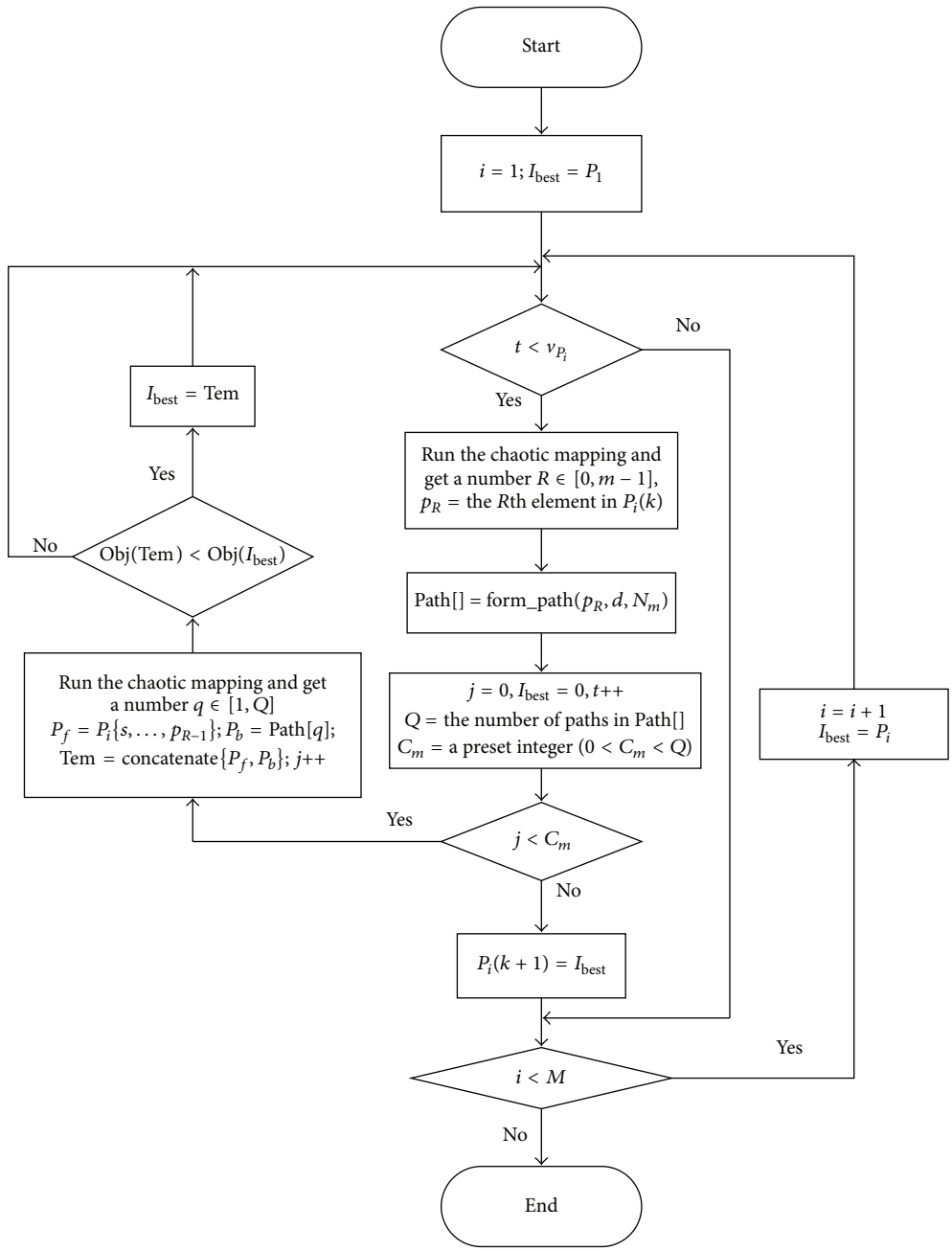


Figure 2: Flowchart of the proposed chaotic optimization algorithm for parking space guidance.

In the application of parking guidance here, the chaotic dynamics is mainly used to do the selection, and, thus, the scheme of chaotic selection is proposed, which can be described in the following process:

(i) Firstly, suppose the searching range is $[I, J]$ $(I < J)$ and the aim is to generate one random round number in that range.

(ii) Then, run the chaotic map once and get one chaos variable z by (4).

(iii) Map the chaos variable into the range of the decision variable, which is $[I, J]$ in this case, and get the variable x by (5).

(iv) At last, get the round value of x, and complete the chaotic selection for once.

This work combines the chaotic selection scheme into the position updating rules and the proposed method of updating the particles is shown in Figure 2. The inputs are $P_i(k)$, which is the position of the particle in the iteration k, $v_{P_i}(k+1)$ or just v_{P_i}, which is the velocity of the particle in the $(k+1)$th iteration, and $N_m \in Z^+$, which is usually

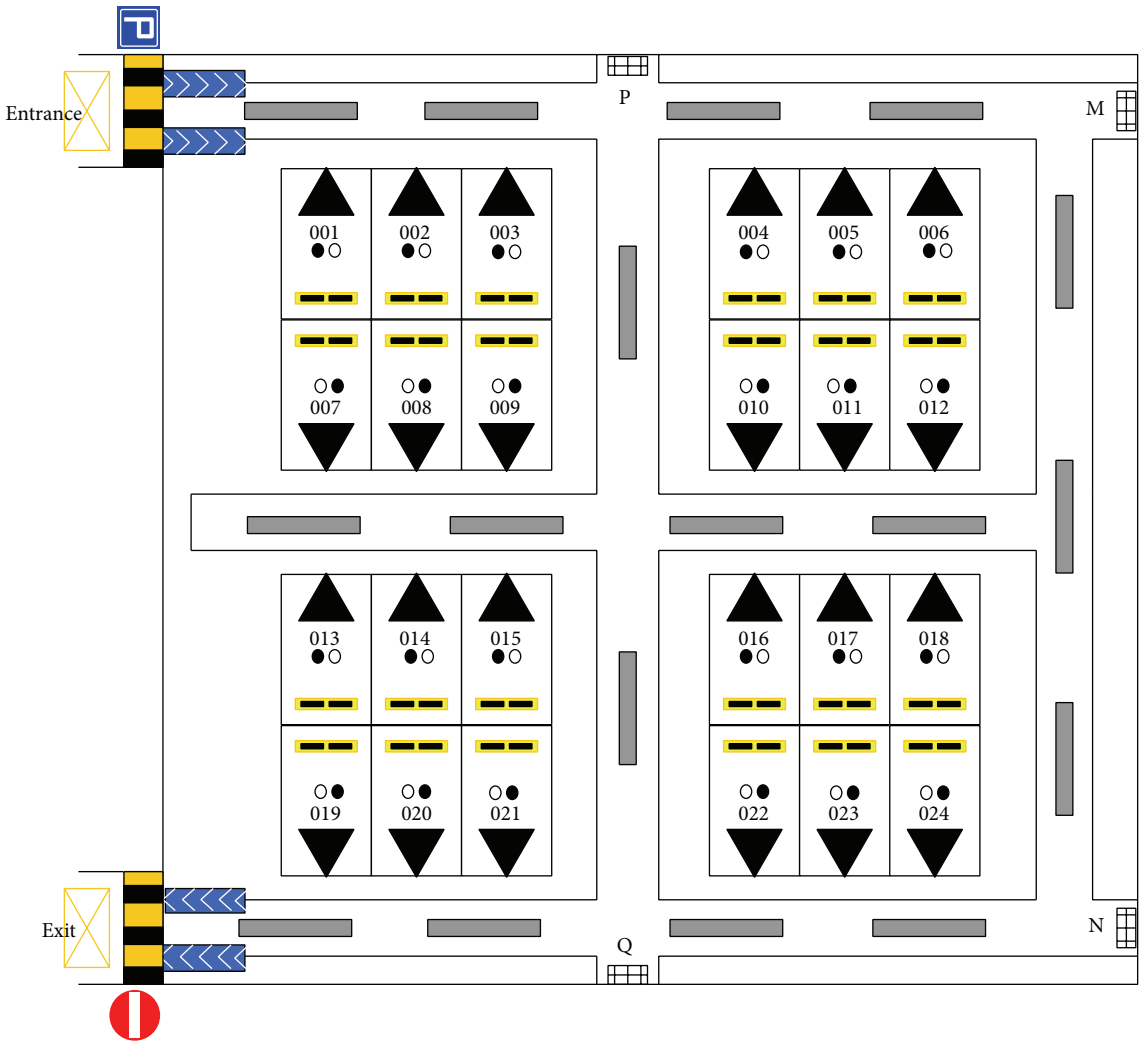


Figure 3: A 24-stall parking area.

predetermined. The flowchart assumes that m is the number of nonnull elements in any particle. The objective function is determined by calling the function Obj( ) and Obj(P) = ∞ when P = null.

In each iteration, an element is selected by using the chaotic map from $P_i(k)$ and storing it in p_R. Then, the function *form_path*( ) is called, to build at-most N_m number of paths from p_R to the destination node. The paths are stored in the array *Path*[ ]. The inner loop is executed for each path in the array *Path*[ ]. The paths in *Path*[ ] exist from the node p_R to the destination node. C_m is a preset vector that controls how many newly formed paths should be evaluated for P_i. P_f contains the portion of $P_i(k)$ from its first element to the Rth position. P_b contains the path which is at the qth position in *Path*[ ], while q is a positive integer that is generated through chaotic mapping. The path *Tem* is the concatenation of the P_f and P_b paths. The variable I_{best} stores the position (or path) which has minimum value of the objective function among all the positions built in the inner loop. The last portion of the flowchart returns the position of P_i for the $(k + 1)$th iteration which is represented by $P_i(k + 1)$.

4. Simulation Analysis

In order to test the performance of the proposed methodology, a route optimization problem based on a real parking lot (shown in Figure 3) is introduced for analyzing. Each parking stall has a pilot light to indicate whether this parking stall is available. As shown in Figure 3, there is one light for each stall, and when the light is on (white), it means this stall is still available, and when it is off (black), it means it is occupied. In order to facilitate the simulation test, a relatively simple parking structure is chosen here. There are one entrance and one exit for this parking lot and four blocks connected by five two-lane roads. Each block has six parking stalls. To make it more flexible, four extensible nodes are designed at the ends of the roads, denoted as P, Q, M, and N, which means they can lead to other parking areas. The simulation tests are done based on the 24-stall parking area as shown in Figure 3, and P, Q, M, and N are treated as four return points. To make it clear and easier for analysis, a sketch map is given in Figure 4.

In the sketch, all the parking stalls and the crosses are marked by numbers and the objective is to find the shorted

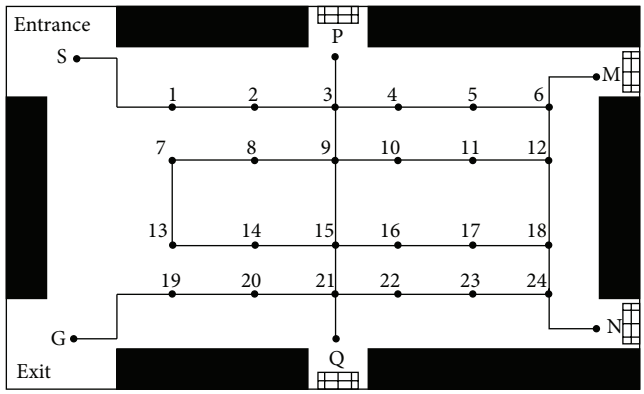


Figure 4: Sketch map of the 24-stall parking area.

path from the entrance to the assigned parking stall and then to the exit. There are several assumptions:

(i) All the information of the parking lot (number and the exact locations of the parking stalls that are available) is saved on a server and real-time data can be obtained when a car comes.

(ii) An available parking stall is assigned randomly at each time.

(iii) The distance from one node to its adjacent node is fixed previously and is irrelevant to the nodes' locations.

The focus here is the shortest distance for a car to drive in the parking lot, which includes two parts: one is the distance from the entrance to the parking stall, and the other is the distance from the parking stall to the exit.

The proposed Chaotic PSO is used for the path optimization and each particle stands for a path. Assume the parking stall number 10 is chosen at this moment, examples of the drive-in and drive-out particles can be illustrated in Figure 5. Thus, the fitness function can be written as follows:

$$J = \min\left(\sum_{j} l\left(p_j, p_{j+1}\right) + \sum_{h} l\left(p_h, p_{h+1}\right)\right), \tag{16}$$

where $l(a, b)$ represents the average length for the vehicle to travel on the edge (a, b) and $\sum_j$ stands for the total distance from the entrance to the chosen parking stall, while $\sum_h$ stands for the total distance from the chosen stall to the exit. For each car entering this parking lot, once an available parking stall is assigned, run the chaotic optimization algorithm proposed above twice, to get the optimized paths to drive in from the entrance to the assigned parking stall and to drive out from the parking stall to the exit. And the complete optimized path from the entrance to the exit can be obtained by joining the two optimized paths together.

In order to test the performance of the proposed method, standard PSO and GA are introduced for comparison. They are used to search for the shortest path in the space of all the possible paths passing through a given parking stall. All the trial tests are coded in MATLAB and executed independently on a Genuine Intel 3.0 GHz CPU with a 4 G RAM desktop computer. A performance index function is introduced for comparison of different route optimization methods in terms of the effectiveness:

$$E = \frac{l_{\min\_in} + l_{\min\_out}}{\sum_j l\left(p_j, p_{j+1}\right) + \sum_h l\left(p_h, p_{h+1}\right)}, \tag{17}$$

where $l_{\min\_in}$ and $l_{\min\_out}$ stand for the actual shortest distance from the entrance to the given parking stall and from the parking stall to the exit, while $\sum_j$ and $\sum_h$ stand for the corresponding distances of the optimized paths calculated by the optimization methods. Thus, $0 < E \leqslant 1$, and when the route optimization algorithm gives the shortest path, $E = 1$; otherwise, E will be a number between 0 and 1. The bigger E is, the better the optimized route is, which also means the corresponding route optimization method is more effective. All the simulation tests have been conducted based on the sketch map shown in Figure 4. Most of the distances between two adjacent parking stalls (noted as $l(p_j, p_{j+1})$) are fixed as 1 in the simulation tests, except for five special ones, which are fixed as $l(S, 1) = 3$, $l(19, G) = 3$, $l(7, 13) = 2$, $l(9, 15) = 2$, and $l(12, 18) = 2$. The distances can also be decided in other ways based on different situations when applying the path optimization method to other parking lots.

Different parking stalls are assigned each time for simulation tests. GA, PSO, and our proposed method are applied independently to solve the same problem. Each method ran 30 independent trials to get the mean value of E and its corresponding standard deviation. All data of different methods has been recorded and analyzed, and the comparison results of each test are reported in Tables 2–5. The optimized paths for different given parking stalls, which are obtained by the

Table 2: Comparison results of different methods for the given stall number 8 within 5 seconds out of 30 independent runs.

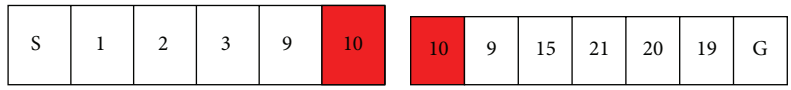
Methods	Performances		
	The mean E	Standard deviation	Successful rate* (%)
GA	0.942	0.103	66.67
PSO	0.958	$9.44E-02$	76.67
Proposed method	**1.00**	**0.00**	**100**

*Success rate means how many times that the actual shortest path can be found successfully during 30 runs.

Table 3: Comparison results of different methods for the given stall number 17 within 5 seconds out of 30 independent runs.

Methods	Performances		
	The mean E	Standard deviation	Successful rate* (%)
GA	0.957	$8.91E-02$	80.00
PSO	0.953	$8.79E-02$	76.67
Proposed method	**1.00**	**0.00**	**100**

*Success rate means how many times that the actual shortest path can be found successfully during 30 runs.



(a) The drive-in particle which ends at 10 (b) The drive-out particle which starts at 10

Figure 5: Format of the particles.

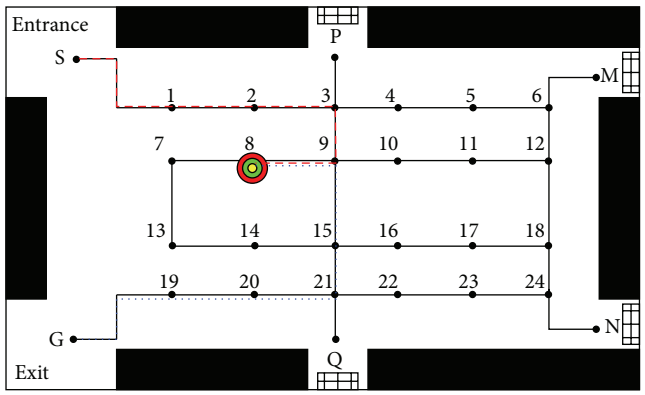


Figure 6: The optimized route obtained by the proposed method for the given parking stall number 8.

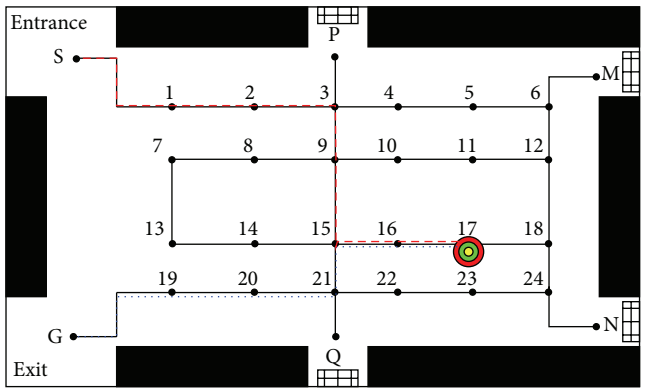


Figure 7: The optimized route obtained by the proposed method for the given parking stall number 17.

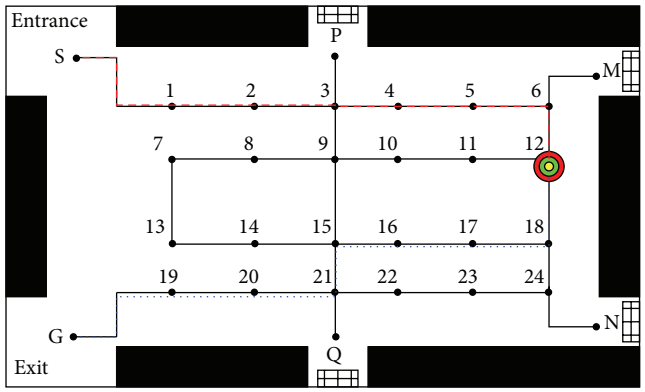


Figure 8: The optimized route obtained by the proposed method for the given parking stall number 12.

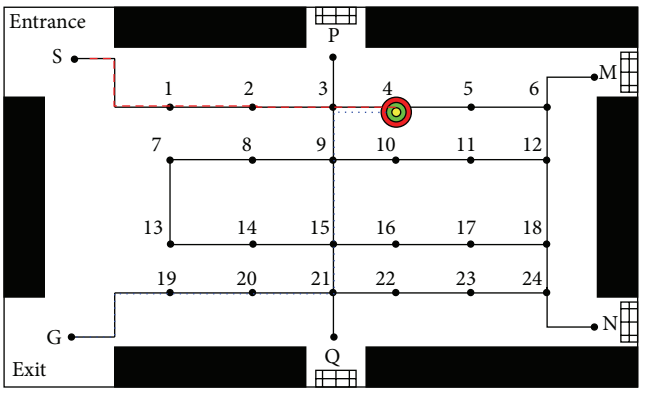


Figure 9: The optimized route obtained by the proposed method for the given parking stall number 4.

Table 4: Comparison results of different methods for the given stall number 12 within 5 seconds out of 30 independent runs.

Methods	Performances		
	The mean E	Standard deviation	Successful rate* (%)
GA	0.964	$8.33E-02$	83.33
PSO	0.978	$5.67E-02$	86.67
Proposed method	**1.00**	**0.00**	**100**

*Success rate means how many times that the actual shortest path can be found successfully during 30 runs.

Table 5: Comparison results of different methods for the given stall number 4 within 5 seconds out of 30 independent runs.

Methods	Performances		
	The mean E	Standard deviation	Successful rate* (%)
GA	0.893	1.196	56.67
PSO	0.916	1.083	60.00
Proposed method	**0.984**	**5.97E − 02**	**93.33**

*Success rate means how many times that the actual shortest path can be found successfully during 30 runs.

proposed Chaotic Particle Swarm Optimization method, are shown in Figures 6–9.

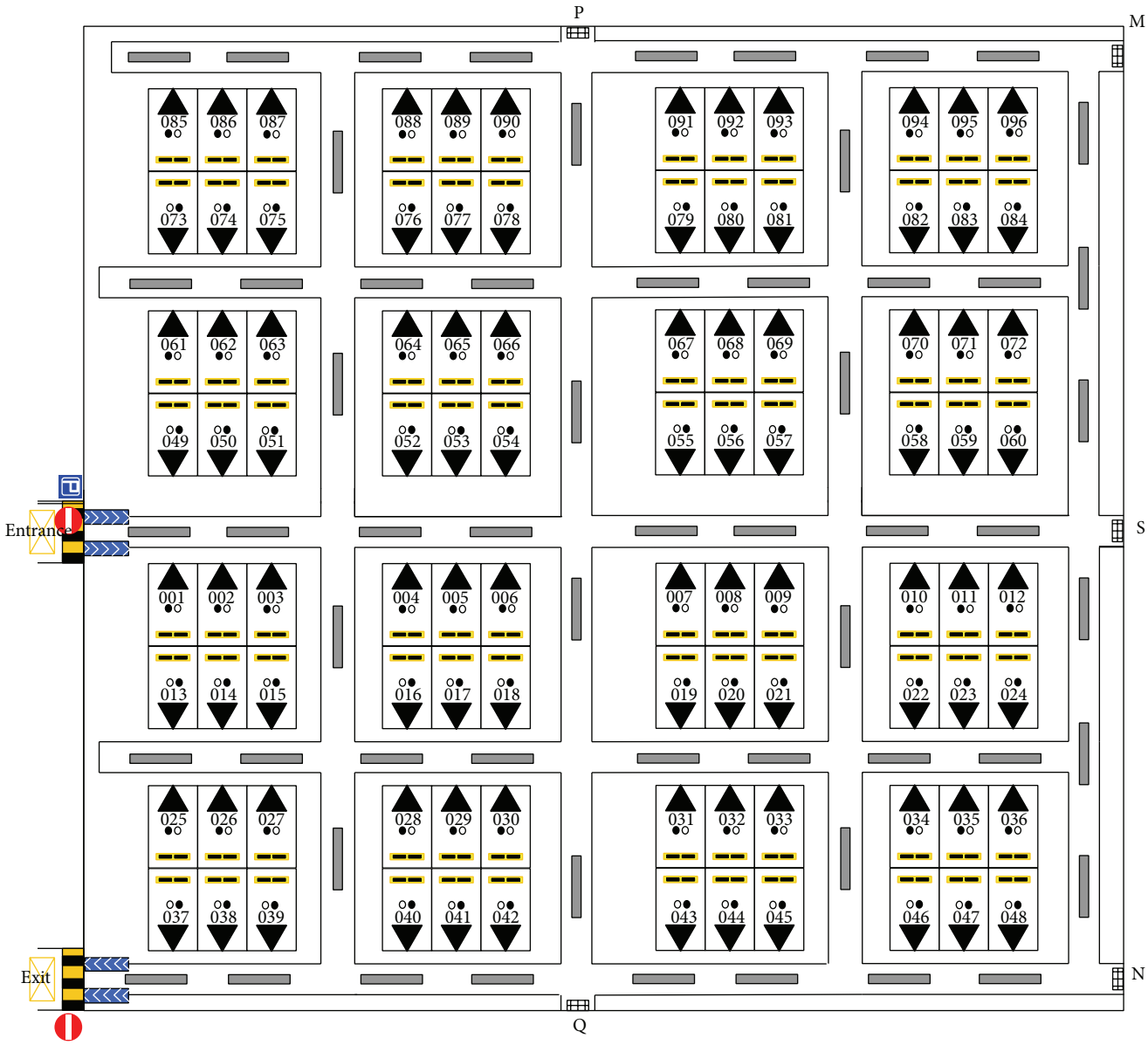


Figure 10: A 96-stall parking area.

From the comparison results shown in Tables 2–5, it can be seen obviously that the proposed Chaotic PSO method gets much higher successful rate (mostly 100%) and can get the most optimized route within 5 seconds, which is quite satisfying and eligible for practical use. It can be concluded that the proposed method is more effective than the other two optimization methods and is a fast and reliable way to solve the route optimization problem of parking lots.

In order to further test the proposed optimization method's capability of handling more complex RO problems of parking lots, a larger size instance with 96 stalls (Figure 10) is introduced for analyzing. Its sketch map is given in Figure 11. Parking stalls numbers 28, 82, and 56 are assigned for simulation tests. The optimized paths for different given parking stalls, which are obtained by the proposed Chaotic Particle Swarm Optimization method, are given in Figures 12–14. 30 independent trails have been run for each given stall and the average time to find the shortest paths has been recorded. For stall number 28, the average time of 30 trails is 3.276 seconds, and the optimized path is shown in Figure 12. For stalls numbers 82 and 56, the shortest paths are not unique, and here we set the algorithm to find the first three different shortest paths for each given stall. The average run time for stall number 82 is 6.732 seconds and the three shortest paths found by a single trail are given in Figure 13. For stall number 56, the average run time is 6.150 seconds and the corresponding results are shown in Figure 14.

The simulation results above showed that the newly proposed optimization method can successfully find out the

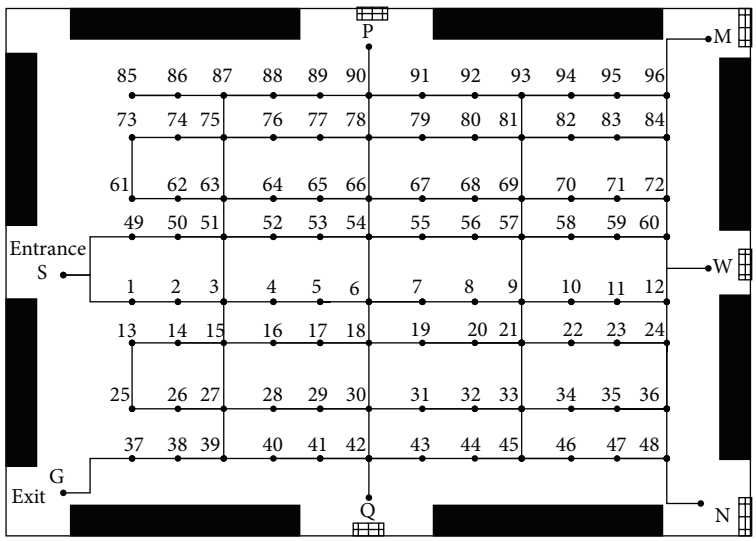


Figure 11: Sketch map of the 96-stall parking area.

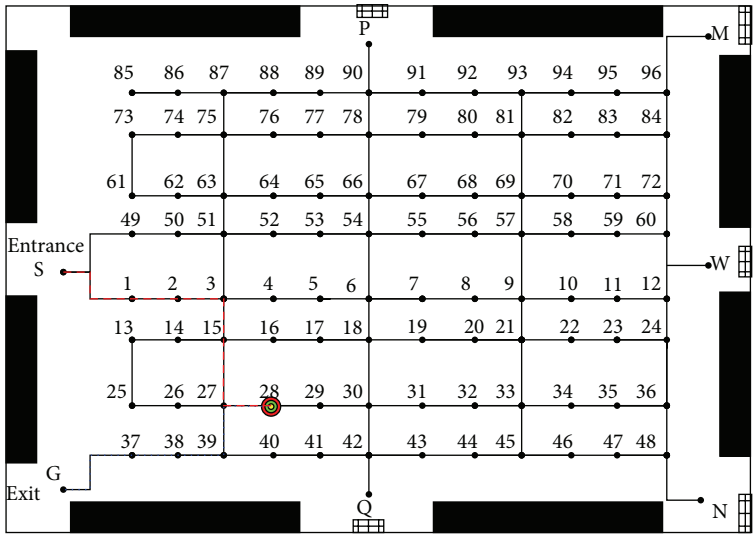


Figure 12: The optimized route obtained by the proposed method for the given parking stall number 28.

shortest paths within an acceptable time, and, thus, the capability and effectiveness of our proposed methodology handling RO problems of parking lots under more complex circumstances have been fully illustrated.

5. Conclusions

This paper presents a Chaotic PSO-based route optimization method for parking guidance. The proposed method has successfully solved the route optimization problems based upon real parking lots. The first experimental results show that the proposed algorithm can successfully find the optimized route for a randomly given parking stall within 5 seconds, and with a much higher successful rate than using other approaches, which is more than 90% in all runs. Most importantly, better stability has been achieved using the proposed Chaotic PSO-based route optimization method as the standard errors of E were 0 in most cases and are much smaller comparing with the other two approaches. The second simulation test shows the capability and effectiveness of our proposed chaotic methodology handling more complex RO problems of parking lots. The capability and applicability of the proposed method have been fully illustrated through all the simulation results. The Chaotic PSO-based approach in the area of parking guidance is still untapped; hence, our future research will aim at investigating more possible applications in some more complicated and large-scale parking lots.

Competing Interests

The authors declare that they have no competing interests.

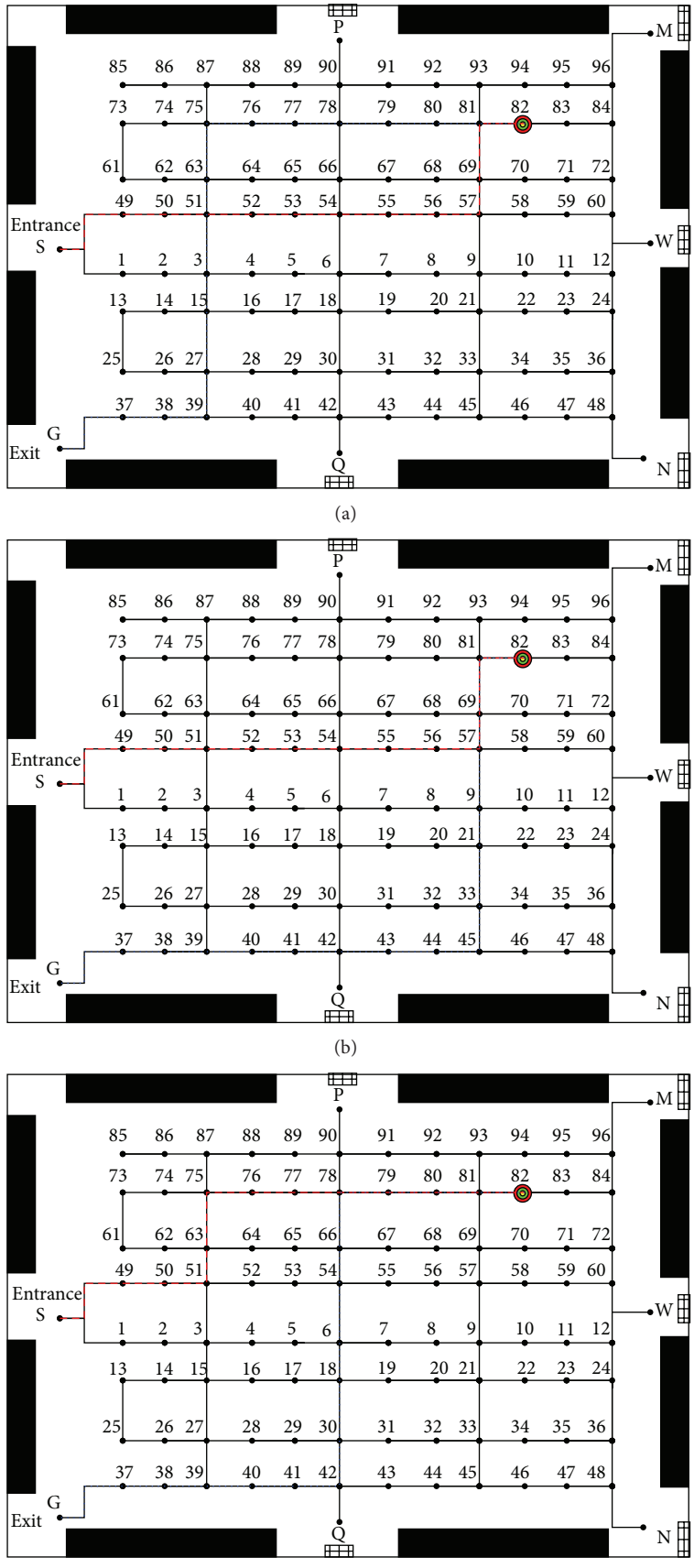


Figure 13: The optimized route obtained by the proposed method for the given parking stall number 82.

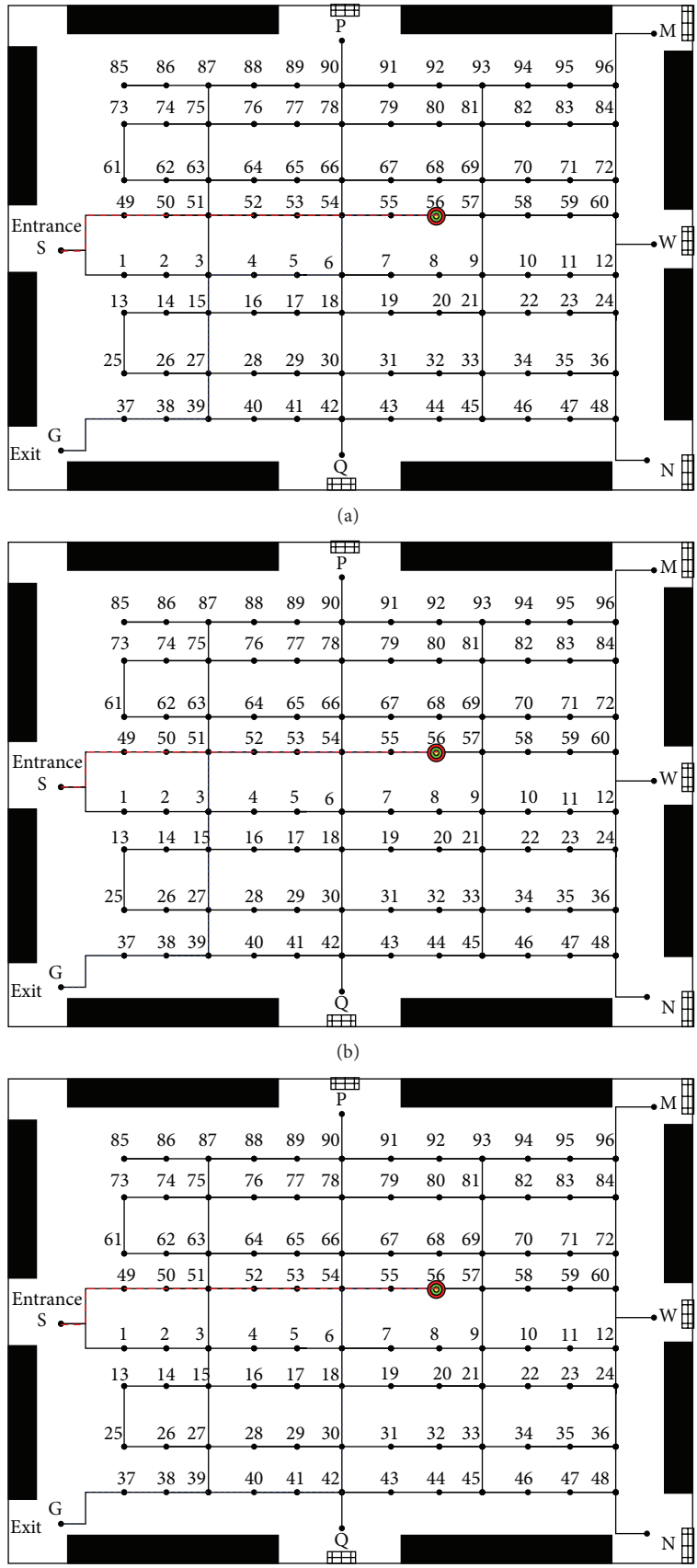


Figure 14: The optimized route obtained by the proposed method for the given parking stall number 56.

Acknowledgments

This work is supported by the Natural Science Foundation of China under Grant nos. 61403274 and 61473203 and the Application Base and Frontier Technology Research Project of Tianjin of China under 13JCQNJC03600.

References

[1] Z. Y. Mei and Y. Tian, "Optimized combination model and algorithm of parking guidance information configuration," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, article 104, 2011.

[2] R. G. Thompson, K. Takada, and S. Kobayakawa, "Optimisation of parking guidance and information systems display configurations," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 69–85, 2001.

[3] C.-K. Lee, C.-L. Lin, and B.-M. Shiu, "Autonomous vehicle parking using hybrid artificial intelligent approach," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 56, no. 3, pp. 319–343, 2009.

[4] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, NY, USA, 1976.

[5] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1643–1653, 2008.

[6] Q. Song and X. F. Wang, "Survey of speedup techniques for shortest path algorithms," *Journal of University of Electronic Science and Technology of China*, vol. 41, pp. 176–184, 2012.

[7] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: state of the art," *Computers & Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006.

[8] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: theory and experimental evaluation," *Mathematical Programming*, vol. 73, no. 2, pp. 129–174, 1996.

[9] M. K. Mehmet Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 941–954, 1993.

[10] F. Araújo, B. Ribeiro, and L. Rodrigues, "A neural network for shortest path computation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1067–1073, 2001.

[11] C.-W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.

[12] G. R. Raidl and B. A. Julstrom, "A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem," in *Proceedings of the ACM Symposium on Applied Computing (SAC '00)*, pp. 440–445, March 2000.

[13] Z. Fu, A. Kurnia, A. Lim, and B. Rodrigues, "Shortest path problem with cache dependent path lengths," in *Proceedings of the Congress on Evolutionary Computation*, pp. 2756–2761, Canberra, Australia, 2003.

[14] J. Kuri, N. Puech, M. Gagnaire, and E. Dotaro, "Routing foreseeable light path demands using a tabu search meta-heuristic," in *Proceedings of the IEEE Global Telecommunication Conference (GLOBECOM '02)*, pp. 2803–2807, Taipei, Taiwan, March 2003.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Network (ICNN '93)*, pp. 1942–1948, Perth, Australia, 1995.

[16] T.-H. Sun, "Applying particle swarm optimization algorithm to roundness measurement," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3428–3438, 2009.

[17] M. Omran, A. P. Engelbrecht, and A. Salman, "Particle swarm optimization method for image clustering," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 3, pp. 297–321, 2005.

[18] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Evolutionary Programming VII: 7th International Conference, EP98 San Diego, California, USA, March 25–27, 1998 Proceedings*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 601–610, Springer, Berlin, Germany, 1998.

[19] K.-W. Wong, B. S.-H. Kwok, and W.-S. Law, "A fast image encryption scheme based on chaotic standard map," *Physics Letters A*, vol. 372, no. 15, pp. 2645–2652, 2008.

[20] C.-M. Ou, "Design of block ciphers by simple chaotic functions," *IEEE Computational Intelligence Magazine*, vol. 3, no. 2, pp. 54–59, 2008.

[21] B. Li and W. Jiang, "Optimizing complex functions by chaos search," *Cybernetics and Systems*, vol. 29, no. 4, pp. 409–419, 1998.

[22] J. A. Koupaei, S. Hosseini, and F. M. Ghaini, "A new optimization algorithm based on chaotic maps and golden section search method," *Engineering Applications of Artificial Intelligence*, vol. 50, pp. 201–214, 2016.

[23] L. D. S. Coelho, "Reliability-redundancy optimization by means of a chaotic differential evolution approach," *Chaos, Solitons and Fractals*, vol. 41, no. 2, pp. 594–602, 2009.

[24] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons and Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[25] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, and A. Rizzo, "Does chaos work better than noise?" *IEEE Circuits and Systems Magazine*, vol. 2, no. 3, pp. 4–19, 2002.

[26] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang, "A dissipative particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1456–1461, May 2002.

[27] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[28] P. Chauhan, M. Pant, and K. Deep, "Parameter optimization of multi-pass turning using chaotic PSO," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 2, pp. 319–337, 2015.

[29] H. Zhang, J.-H. Shen, T.-N. Zhang, and L. Yang, "An improved chaotic particle swarm optimization and its application in investment," in *Proceedings of the International Symposium on Computational Intelligence and Design (ISCID '08)*, pp. 124–128, Wuhan, China, October 2008.

[30] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Western Australia, 1995.

[31] X. D. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization," Tech. Rep., Evolutionary Computation and Machine Learning Group, RMIT University, Melbourne, Australia, 2013.

[32] A. Stacey, M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, vol. 2, pp. 1425–1430, IEEE, December 2003.