Information Sciences 278 (2014) 311–326

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Polygonal approximation using integer particle swarm optimization

Bin Wang [a,b,*], Douglas Brown [b], Xiaozheng Zhang [b], Hanxi Li [b], Yongsheng Gao [b], Jie Cao [a]

[a] Key Laboratory of Electronic Business, Nanjing University of Finance and Economics, Nanjing 210046, China
[b] School of Engineering, Griffith University, Nathan, QLD 4111, Australia

ARTICLE INFO

Article history:
Received 26 April 2013
Received in revised form 6 March 2014
Accepted 11 March 2014
Available online 31 March 2014

Keywords:
Polygonal approximation
Particle swarm optimization
Genetic algorithms
Binary version of PSO
Integer coding

ABSTRACT

Polygonal approximation is an effective yet challenging digital curve representation for image analysis, pattern recognition and computer vision. This paper proposes a novel approach, integer particle swarm optimization (iPSO), for polygonal approximation. When compared to the traditional binary version of particle swarm optimization (bPSO), the new iPSO directly uses an integer vector to represent the candidate solution and provides a more efficient and convenient means for solution processing. The velocity and position updating mechanisms in iPSO not only have clear physical meaning, but also guarantee the optimality of the solutions. The method is suitable for polygonal approximation which could otherwise be an intractable optimization problem. The proposed method has been tested on commonly used synthesized shapes and lake contours extracted from the maps of four famous lakes in the world. The experimental results show that the proposed iPSO has better solution quality and computational efficiency than the bPSO-based methods and better solution quality than the other state-of-the-art methods.

1. Introduction

Digital curve representation is an important topic in image analysis, pattern recognition and computer vision. After a contour is extracted from an object in the scene, a sequence of coordinate points can be obtained by traversing along the contour. This sequence of points can be viewed as a digital curve. It contains a large amount of shape information of the object [1], especially around the corners. Effective representation of a digital curve facilitates the subsequent image analysis tasks, such as shape matching, object recognition and image retrieval.

Polygonal approximation is an effective digital curve representation method, providing a piecewise-linear representation of a curve by dividing the curve into a number of connected straight line segments. It removes a large number of redundant points from the digital curve while preserving major characteristics. As an example, take a square-shaped digital curve with 1000 points. Its optimal polygonal approximation consists of just the four corner points, which represents the same 1000-point curve without loss of shape information. Seeking the optimal approximation of a general digital curve is a challenging and unsolved problem however, because it is a combinatorial optimization problem without an

* Corresponding author at: Key Laboratory of Electronic Business, Nanjing University of Finance and Economics, Nanjing 210046, China. Tel.: +86 25 84028865.
E-mail address: wangbin@njue.edu.cn (B. Wang).

http://dx.doi.org/10.1016/j.ins.2014.03.055
0020-0255/© 2014 Elsevier Inc. All rights reserved.

analytical solution. In terms of different objectives, polygonal approximation is usually categorized into the following two types of optimizations [9,14].

- $\min -\varepsilon$: For a given fixed value M, select the optimal set of M vertices from the digital curve to construct a polygon with minimum approximation error.
- $\min -\#$: For a given error tolerance ε, select a minimum number of vertices from the curve to construct a polygon whose approximation error does not exceed ε.

These two types of polygonal approximation problems operate towards different goals under different constraints: $\min -\varepsilon$ aims to minimize the error while maintaining a compact polygonal approximation, whereas $\min -\#$ aims to minimize the number of vertices while maintaining the specified approximation accuracy. Both optimizations have large search spaces and high computational costs to obtain the exact optimal solutions. The $\min -\varepsilon$ problem has $\binom{N}{M}$ different ways of choosing M vertices out of N curve points in its search space [4]. In the $\min -\#$ problem, the number of vertices is variable and the size of the search space may be larger than that of $\min -\varepsilon$ problem. In real image analysis applications, the extracted digital curves usually have a large number of points and result in a huge search space, making finding the global optimal solution impractical. Classical optimization methods, such as dynamic programming (DP) [8,13] can only handle small scale problems (curves around 100 points) [28], and are not suitable for the larger-scale problems encountered in real applications. For this reason, many local heuristic search based methods such as dominant point deletion (DPD) [10], break point suppression (BPS) [15] and the betweenness method [2] have been proposed for solving polygonal approximation problems.

In recent years, researchers have attempted to apply nature-inspired algorithms to solving polygonal approximation problems. These methods include genetic algorithms (GA), particle swarm optimization (PSO) and its digital binary version (bPSO), ant colony search (ACS) and others, detailed in the following section. We propose extending the bPSO method into the integer space, to reduce the dimensionality of the polygon description vectors, improve the processing efficiency and increase the accuracy of the algorithm.

The remainder of this paper is organized as follows. Section 2 details works related to the problem of polygonal approximation. Section 3 formulates the $\min -\#$ polygonal approximation problem. Section 4 provides a review of the original version of PSO and the binary version of PSO in the context of polygonal approximation. Section 5 describes the details of the proposed iPSO. In Section 6, we present the experimental results and performance comparisons. Finally, we draw our conclusions in Section 7.

2. Related work

Yin [25] proposed a method based on genetic algorithms (GA) for solving the $\min -\varepsilon$ problem, which was empirically shown to have a higher performance than many existing local heuristic methods [16,17]. Huang and Sun [5] added a preprocessing step to a GA-based method, removing the collinear points and reducing the computation time. Ho and Chen [4] developed a GA-based method with a novel orthogonal array crossover for the $\min -\varepsilon$ problem. Their experimental results showed that it outperformed the methods in [25,5]. Sarkar et al. [20] incorporated chromosome differentiation into GA-based methods for solving $\min -\varepsilon$ problems, thus improving the search performance. Sun and Huang [21] developed a crossover operator with constraints to maintain feasibility of the candidate solutions, for their proposed GA-based method for solving $\min -\#$ problems. To overcome the drawbacks of existing GA-based methods handling the constraints, Wang et al. [24] proposed a genetic algorithm with chromosome-repairing for both types of polygonal approximation problems, empirically showing that their method outperformed the existing GA-based methods and the local search based method proposed by Masood and Haq [11]. Apart from the genetic algorithms, ant colony search (ACS) and particle swarm optimization (PSO) have also been applied to polygonal approximation. Yin proposed an ACS-based method simulating ant foraging behavior [26] and later went onto develop a PSO-based method simulating the behavior of bird flocking [27], for solving min–# problems. Wang et al. [23] proposed a novel method incorporating a mutation operator into PSO for min–# problems.

Compared to GA, PSO has a simpler concept, fewer parameters and easier implementation, although the number of works applying PSO to polygonal approximation is limited. Originally, PSO was designed for real-valued problems [6], but because polygonal approximation is a discrete optimization problem, Kennedy and Eberhart [7] proposed a binary version of PSO (bPSO), which became the basis of other methods [23,27].

The existing PSO-based methods for polygonal approximation have the following limitations: (1) a digital curve usually has many points and the subsequent binary position vector is very long, significantly affecting the efficiency of the solution representation, and more importantly, (2) bPSO cannot guarantee the optimality of the solution. According to the position updates of bPSO as in [23,27], if the value of the velocity is greater than 0, at least half of the curve points will be selected. In practical applications, however, the solution vertex count is much smaller than the number of curve points; therefore the bias of bPSO towards equal probability of being vertex and non-vertex is not appropriate.

In order to effectively apply PSO to the min–# problem, this paper proposes a novel integer PSO (iPSO) to represent the particles directly and greatly reduce the size of the position vector. Moreover, the velocity and position update in integer space guarantee the optimality of the obtained solutions. iPSO is therefore more suitable for solving polygonal approximation problems than bPSO.

3. Problem formulation

A closed digital curve C with N points can be represented by a clockwise ordered sequence of pixel points $C = \{z_i(x_i, y_i) | i = 1, 2, \ldots, N\}$, where z_i is the ith point having coordinate (x_i, y_i) and $z_{N+i} = z_i$. Let $A_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$ denotes an arc from z_i to z_j. Let L_{ij} denote the straight line segment connecting z_i and z_j, i.e., the chord of A_{ij}. The error in approximating A_{ij} using its chord L_{ij} is calculated by

$$e(L_{ij}, A_{ij}) = \sum_{z_k \in A_{ij}} d^2(z_k, L_{ij}), \tag{1}$$

where $d(z_k, L_{ij})$ is the perpendicular distance from the chord L_{ij} to the point z_k. A polygon V which approximates C can be represented by an ordered set of line segments (or chords) $V = \{L_{t_1 t_2}, L_{t_2 t_3}, \ldots, L_{t_{M-1} t_M}, L_{t_M t_{M+1}}\}$, subject to $t_1 < t_2 < \cdots < t_M$, $t_{M+1} = t_1$ and $\{z_{t_1}, z_{t_2}, \ldots, z_{t_M}\} \subseteq C$, where M is the number of vertices in V. The min–# polygonal approximation problem is: given an allowable error ε, find a polygon V with the minimum number of vertices among all the polygons approximating C whose integral square error (ISE) is less than ε. The integral square error (ISE) between curve C and polygon V is calculated by

$$ISE(V, C) = \sum_{i=1}^{M} e(L_{t_i t_{i+1}}, A_{t_i t_{i+1}}). \tag{2}$$

4. Particle swarm optimization (PSO)

The particle swarm optimization (PSO) algorithm is a population-based evolutionary computation technique originally proposed by Kennedy and Eberhart [6]. PSO was developed based on observations of animal social behaviors, such as bird flocking, fish schooling, etc. In this section, we present a brief review of PSO and its binary version, bPSO.

Assume a swarm consisting of N particles in a D-dimensional search space, $S \subset R^D$. The position of the ith particle is a D-dimensional vector denoted by $X_i = [x_{i1}, x_{i2}, \ldots, x_{iD}] \in S$, and its velocity by $V_i = [v_{i1}, v_{i2}, \ldots, v_{iD}] \in S$. Let $P_i = [p_{i1}, p_{i2}, \ldots, p_{iD}]$ denote the best previous position encountered by the ith particle in the search space and g denote the index of the particle that found the best previous position among the entire swarm. The velocity and the position of the ith particle in the next time step are then updated using the following formulas.

$$v_{ij}(t+1) = v_{ij}(t) + c_1 * r_1 * (p_{ij}(t) - x_{ij}(t)) + c_2 * r_2 * (p_{gj}(t) - x_{ij}(t)), \tag{3}$$

and

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{4}$$

where $t = 1, 2, \ldots$, indicates the iterations, $i = 1, 2, \ldots, N$ is the particle's index, and $j = 1, 2, \ldots, D$ is the index of the vector dimension. c_1 and c_2 are positive constants, referred to as cognitive and social parameters, respectively, and r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$. To avoid rapid movements of particles in the search space, the velocity updates are usually clamped to within the range $[-v_{max}, v_{max}]$.

PSO was originally developed for problems with continuous-valued variables, e.g. $x_{ij} \in R$. To adapt the PSO to problems with binary-valued variables, Kennedy and Eberhart [7] proposed a binary version, bPSO. In bPSO, the velocity is treated as a probability vector, where each element is the probability that an element of the position vector takes the value of 1. For bPSO, the velocity update remains unchanged and the position update becomes

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r < S(v_{ij}(t+1)), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where

$$S(v_{ij}(t+1)) = \frac{1}{1 + e^{-v_{ij}(t+1)}} \tag{6}$$

and r is a random number uniformly distributed in $[0, 1]$. From Eq. (5), we can see that a limiting transformation function (Eq. (6)) should be designed to map the real valued numbers of velocity to the range $[0, 1]$. In [27], Eq. (6) was replaced by the following function for solving the min–# problem.

$$S(v_{ij}(t+1)) = \frac{v_{ij}(t+1)}{2 v_{max}} + 0.5, \tag{7}$$

where v_{max} is the maximum velocity. From Eqs. (5) and (6), we will have $S(v_{ij}(t+1)) \geqslant 0.5$, if $v_{ij}(t+1) \geqslant 0$. It is also easy to conclude that if the convergence of bPSO holds, about 50% of the elements in the position vector $X_i(t)$ generated by Eq. (5) will take the value 1.

5. The proposed integer PSO (iPSO)

In this section, a novel integer PSO (iPSO), which operates in integer space, is proposed for solving the polygonal approximation problem.

5.1. Particle representation and fitness evaluation

Given a digital curve $C = \{z_i(x_i, y_i) | i = 1, 2, \ldots, N\}$ with N points, let $V = \{L_{t_1t_2}, L_{t_2t_3}, \ldots, L_{t_{M-1}t_M}, L_{t_Mt_1}\}$ be an approximating polygon of C with M vertices, where $t_1 < t_2 < \cdots < t_M$ and $t_i \in [1, N]$ is the serial number of the ith vertex of V. The existing bPSO-based methods adopt binary coding to represent the particle [23,27]. In these schemes, the candidate polygon V is represented by a binary vector $[b_1, b_2, \ldots, b_N]$, in which b_i corresponds to curve point z_i. b_i takes 1 if z_i is selected as the polygon's vertex, and 0 otherwise. Since a digital curve usually has a large number of points, the length of the corresponding binary vector will be equally long. This coding scheme also requires a decoding operation, i.e. the serial number of each vertex must be computed from the binary vector. These two issues greatly affect the efficiency of bPSO in polygonal approximation.

To address these issues in bPSO, this paper proposes a novel PSO integer coding scheme. This new scheme directly uses the serial number of the vertex of the approximating polygon to code the particle. Particle X_i is represented by an integer vector which can be expressed as

$$X_i = [x_{i1}, x_{i2}, \ldots, x_{iM}] \text{ subject to } x_{i1} < x_{i2} < \cdots < x_{iM} \text{ and } x_{ij} \in \{1, 2, \ldots, N\}, \tag{8}$$

where M is the dimension of X_i and also denotes the number of vertices in V. The particle denotes a candidate solution $V = \{L_{x_{i1}x_{i2}}, L_{x_{i2}x_{i3}}, \ldots, L_{x_{M-1}x_M}, L_{x_{iM}x_{i1}}\}$ to approximate C. In most polygonal approximation cases, the number of selected polygon vertices is much smaller than the number of curve points (i.e. $M \ll N$). Compared to bPSO's coding on N curve points, the new representation on M polygon candidate points in iPSO is more efficient. The vertex indices of V can be directly retrieved from X_i, so that it does not require a decoding step as in bPSO.

Fig. 1 shows a circular curve, digitized to 16 points, approximated to a triangle and square, and the comparison of the binary coding of bPSO and the integer coding of the proposed iPSO. The length of the binary code is fixed, whereas the length of the integer code is variable and much shorter than that of a binary code. In the min–# problem, the objective is to minimize the number of vertices of the approximating polygon. In the proposed integer coding, the length of the position vector can be used to directly represent the fitness value of the particle; the shorter the vector, the better the fitness value.

5.2. Velocity update and position update

The original velocity update formula (Eq. (3)) and position update formula (Eq. (4)) cannot produce integer values. To allow PSO to perform in integer space, these two equations are modified as shown in Eqs. (9) and (11), respectively. The new velocity update formula is expressed as

$$v_{ij}(t+1) = \begin{cases} \lfloor \tilde{v}_{ij}(t+1) \rfloor \textit{ mod } N & (\tilde{v}_{ij}(t+1) \geqslant 0) \\ -(\lfloor -\tilde{v}_{ij}(t+1) \rfloor \textit{ mod } N) & (\tilde{v}_{ij}(t+1) < 0), \end{cases} \tag{9}$$

where

$$\tilde{v}_{ij}(t+1) = v_{ij}(t) + c_1 * r_1 * (p_{ij}(t) - x_{ij}(t)) + c_2 * r_2 * (p_{gj}(t) - x_{ij}(t)). \tag{10}$$

The new position update formula is expressed as

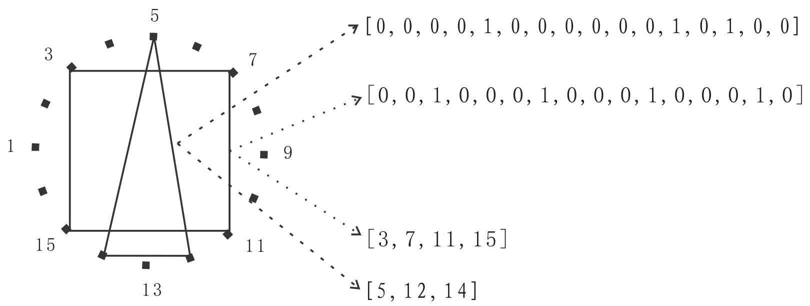


Fig. 1. A comparison between binary coding and the proposed integer coding of a square and a triangle approximating a 16-point digital curve.

$$x_{ij}(t+1) = \begin{cases} \tilde{x}_{ij}(t+1) - N & (\tilde{x}_{ij}(t+1) > N) \\ \tilde{x}_{ij}(t+1) & (0 < \tilde{x}_{ij}(t+1) \leqslant N), \\ \tilde{x}_{ij}(t+1) + N & (\tilde{x}_{ij}(t+1) \leqslant 0) \end{cases} \tag{11}$$

where

$$\tilde{x}_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{12}$$

In Eq. (9), '*mod*' denotes the modulo operation and $\lfloor \cdot \rfloor$ denotes the floor function, used to convert a real expression $\tilde{v}_{ij}(t+1)$ to an integer expression $v_{ij}(t+1)$, restricted to within $[-(N-1), N-1]$. The value of the updated position $\tilde{x}_{ij}(t+1)$ is restricted to $[1, N]$. The velocity v_{ij} denotes the displacement of the polygon's vertex along the curve, where $v_{ij} > 0$ indicates a clockwise displacement, and $v_{ij} < 0$ denotes a displacement in a counter-clockwise direction. Fig. 2 illustrates the process and mechanism of the proposed position update using Eq. (11).

Unlike the fixed position vector length in bPSO, the proposed iPSO has position vectors with variable lengths meaning that the position vector X_i may have a different length to the position vectors P_i and P_g. Accordingly, the term "$p_{ij}(t) - x_{ij}(t)$" and the term "$p_{gj}(t) - x_{ij}(t)$" in Eq. (10) are modified to

$$p_{ij}(t) - x_{ij}(t) = \begin{cases} p_{ij}(t) - x_{ij}(t) & \text{if } j \leqslant L, \\ p_{iL}(t) - x_{ij}(t) & \text{otherwise}, \end{cases} \tag{13}$$

and

$$p_{gj}(t) - x_{ij}(t) = \begin{cases} p_{gj}(t) - x_{ij}(t) & \text{if } j \leqslant H, \\ p_{gH}(t) - x_{ij}(t) & \text{otherwise}, \end{cases} \tag{14}$$

where L and H are the lengths of P_i and P_g, respectively. Fig. 3 illustrates the application of Eq. (13).

Ideally, the position vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$ satisfies the sequence rule $x_{i1} < x_{i2} < \cdots < x_{iM}$. In reality, however, the position update result may cause violation of this condition. For example, if $X_i(t) = [1, 4, 7, 9]$ and $V_i(t+1) = [4, 0, 2, 0]$, the update result following Eq. (12) becomes $X_i(t+1) = X_i(t) + V_i(t+1) = [5, 4, 9, 9]$ which violates the sequence rule. This violation is rectified through two steps: (1) sorting the elements of $X_i(t+1)$ in ascending order, and (2) removing all the repetitive elements in $X_i(t+1)$. In the above example $X_i(t+1) = [5, 4, 9, 9]$ becomes $X_i(t+1) = [4, 5, 9]$.

5.3. Position adjustment

Polygonal approximation is a constraint optimization problem, but PSO may generate solutions which violate the constraint conditions. In these situations, the particles fly out of the feasible solution space and therefore a proper constraint-handling method is required. Yin [27] adopted a penalty method to enforce the constraints, whereby particles leaving the feasible solution space are penalized by assigning a negative value reflecting the violation degree. This penalty aims to drive the particles back into the feasible solution region, however it is generally difficult to determine the penalty function, as a weak penalty may leave many particles continuously searching outside the feasible solution space. To overcome the disadvantage of the penalty method, Wang et al. [24] proposed a chromosome-repairing scheme which utilized the traditional split and merge techniques to mend the 'invalid' chromosomes. This scheme not only aims to preserve the feasibility of the solution, but also attempts to maintain its optimality. Following on from the idea of the constraint handling method

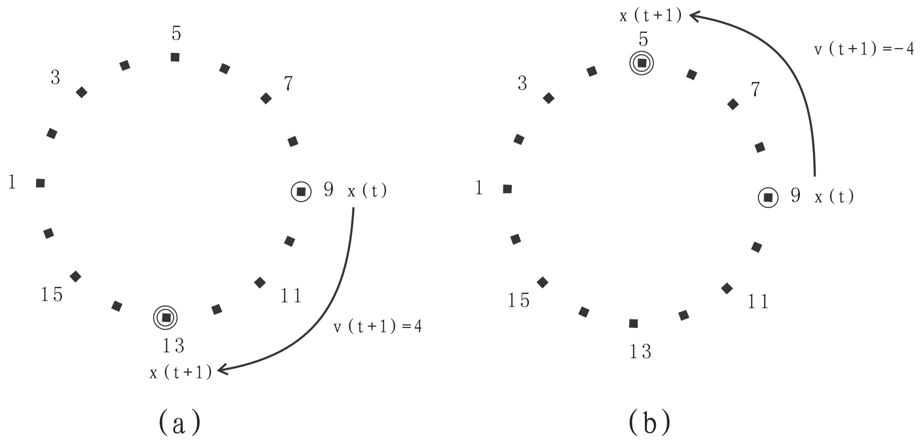


Fig. 2. Position updating of the proposed method; (a) when $v(t+1) > 0$, the position $x(t)$ moves clockwise to $x(t+1)$, and (b) when $v(t+1) < 0$, the position moves counterclockwise.

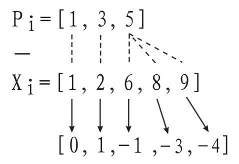


Fig. 3. An example illustrating the calculation process of Eq. (13).

proposed by Wang et al. [24], we use the chromosome-repairing scheme to cope with the problem of the particles leaving the feasible solution space.

When a particle leaves the feasible solution space, the approximation error of its corresponding solution exceeds the given error tolerance ε. The following steps can be taken to move it back into the feasible region: (1) travel along the polygon in a clockwise direction, examine and adjust each vertex to a new curve point on the arc between its two adjacent vertices to reduce the approximation error as much as possible; and (2) if the resulting approximation error still exceeds the prescribed tolerance, the solution is repaired using the scheme proposed by Wang et al. [24].

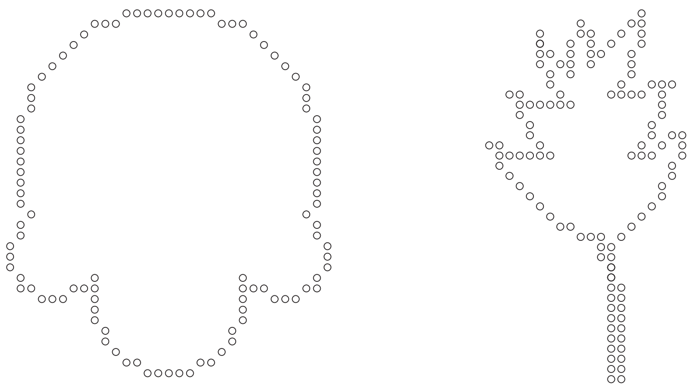
Similarly to GA, PSO has a strong global, but poor local search ability. To improve the local search ability of PSO, many researchers have suggested embedding a local optimizer such as hill-climbing into PSO. For example, Yin [27] embedded an adjusting-and-merging optimizer. This optimizer performs the following two steps: (1) each vertex of the polygon is adjusted, if possible, to a new curve point on the arc between its two adjacent vertices such that the approximation error is reduced as much as possible; and (2) a merging process is repeated until no adjacent segments can be merged further. The merging process is defined as merging two adjacent segments if the resulting polygon still satisfies the error constraint. Although the local optimizer can further improve the quality of the solutions, it is time-consuming. To balance the computation time and solution quality, this paper only uses this optimizer on the best solution found by all the particles so far at each iteration. This is different from Yin's method in which the local optimization is performed against every particle.

5.4. Algorithm flow

This section elaborates on the algorithm flow of the proposed iPSO. Given a digital curve C with N points, let K be the size of the particle swarm, and T be the pre-specified maximum number of iterations. The pseudocode of the proposed iPSO is shown in Algorithm 1.

Algorithm 1. Integer PSO (iPSO) for polygon approximation

```
Input:
        · Digital curve $C$
        · Error tolerance $\varepsilon$
        · PSO parameters $T$, $c_1$, $c_2$, $r_1$ and $r_2$.
Output: Optimal polygon approximation $P$
begin
    Initialize
    · Position: Random $\{X_1(0), X_2(0), \ldots, X_K(0)\}$ (Eq. 8)
    · Velocity: Random $V_i(0) = \{v_{i1}(0), \ldots, v_{iM}(0)\}$ where
      $v_{ij}(0) \in \{1-N, \ldots, N-1\}$, $i \in [1, N]$;
    · Best positions $P_1(0) = P_2(0) = \ldots = P_K(0) = P_g(0) = +\infty$;
    for $t \leftarrow 1$ to $T$ do
        for $i \leftarrow 1$ to $K$ do
            · If particle $X_i(t)$ falls in the infeasible solution space,
                  $X_i(t)$ =Chromosome-Repair($X_i(t)$) (Section 5.3);
            · Evaluate the fitness value $F_i(t)$ for $X_i(t)$;
            · Update best previous position $P_i(t)$:
                  If $F_i(t) < F(P_i(t-1))$, $P_i(t) = X_i(t)$, otherwise $P_i(t) = P_i(t-1)$
            · Update best position $P_g(t)$:
                  If $F(P_i(t)) < F(P_g(t))$, $P_g(t) = P_i(t)$
        · Local optimize $P_g(t)$:
              $P_g(t)$ =Adjust-and-Merge($P_g(t)$) (Section 5.3);
        for $i \leftarrow 1$ to $K$ do
            · Update velocity $V_i(t+1)$ (Eq. 9) and position $X_i(t+1)$ (Eq. 11)
    · Return $P = P_g(T)$;
end
```



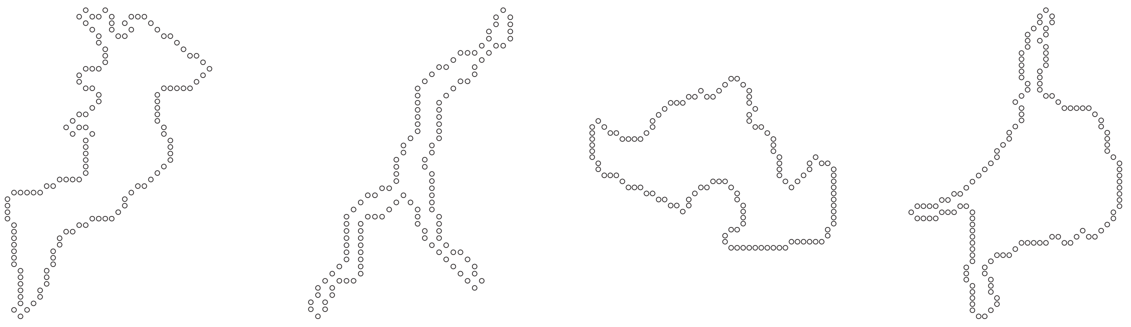
```
00007  00777  77766  76666  66665  76766
56454  43436  66656  55454  44434  33232
22254  54434  23221  21322  22222  21221
11111  00100  00
```

(a) Semicircle

```
33332  30700  00003  32307  00003  32322
26777  22212  76661  11116  66566  55000
10056  65655  00110  66565  65555  56667
66666  66664  22222  22222  23224  43433
```

(b) Leaf

Fig. 4. Two synthesized shapes and their chain codes generated from [22].



```
12310  01235  32211  12111  21177  76666
70000  10100  07777  67707  76667  67670
70070  07077  75344  44454  44444  54443
22223  23222  34444  43567  53323  34565
43223  44545  553
```

(a) Arlington

```
10007  67770  76777  00000  70701  00000
10001  12110  17555  55554  54455  77766
70000  00076  67070  75434  33334  44444
33323  23444  34334  44444  33323  32233
4343
```

(b) Como

```
21100  17012  11001  00011  33343  12100
00000  00707  66666  76666  66666  54323
44454  55667  67707  56565  65665  56654
54444  43112  12223  34333  22323  12333
```

(c) Managua

```
10707  10007  00012  11222  21101  00011
00000  00707  77657  76567  66667  76677
01001  07765  44445  44344  44444  45676
67666  53222  32323  33333  43343  34453
34544  44344  3343
```

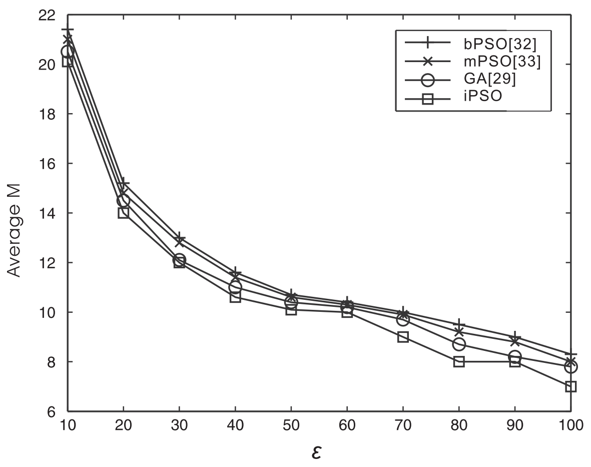
(d) Simcoe

Fig. 5. Four lake shapes extracted from the maps of four famous lakes, namely Arlington (in US), Como (in Italy), Managua (in Nicaragua) and Simcoe (in Canada), with their chain codes generated from [24].

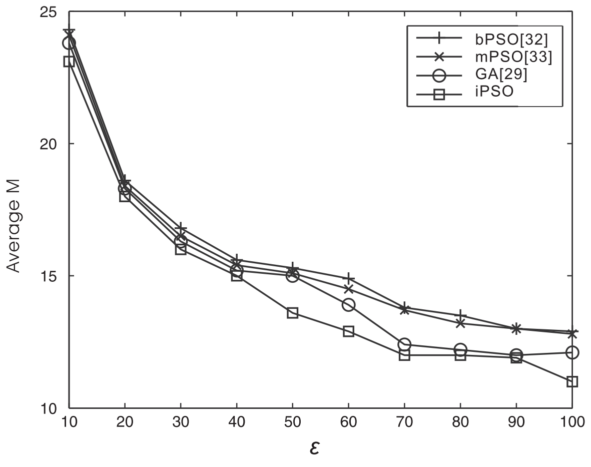
5.5. Comparison between bPSO and iPSO

Compared to bPSO, the proposed iPSO has the following advantages.

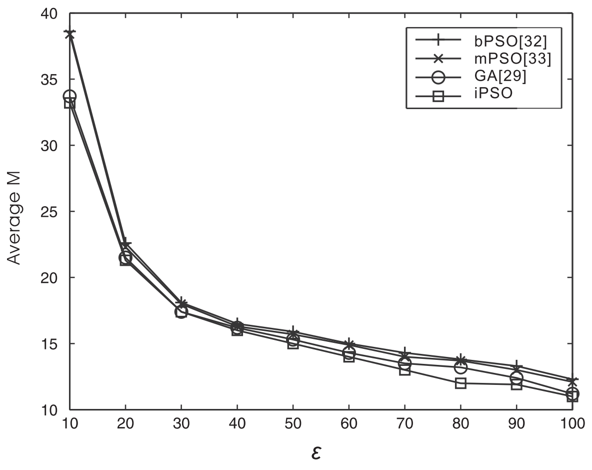
- With regards to particle representation, iPSO provides a direct solution representation which benefits the processing of the solution (e.g. calculation of approximation error, fitness evaluation and solution local optimization). In bPSO, the binary code requires decoding and transformation into an integer code before processing the solution. In addition, the solution representation of iPSO is much more compact and efficient than bPSO.
- With respect to the position update, iPSO has a clearer physical meaning and a more sound mechanism than bPSO. In the design of iPSO, both position and velocity have clear physical meanings. Its particle position is the polygon vertex position on the curve, and its velocity is the distance and direction that a particle needs to travel in the position update. These intuitive designations naturally follow their equivalent kinetic meanings. Conversely, the update mechanism of bPSO is based on probability. The velocity is transformed to a probability that a point is chosen as a polygon vertex where zero velocity denotes an equal chance of being selected or not being selected. This is contrary to the sense of kinetics in which a zero velocity particle would remain in the same position.
- With regards to the solution quality, iPSO generates higher quality solutions than bPSO due to being unbiased and having an increased efficiency. As detailed in Sections 2 and 4, bPSO is biased towards the state that half of the curve points are selected as the polygon vertices. The solutions consequently lack optimality. In iPSO, the length of the posi-



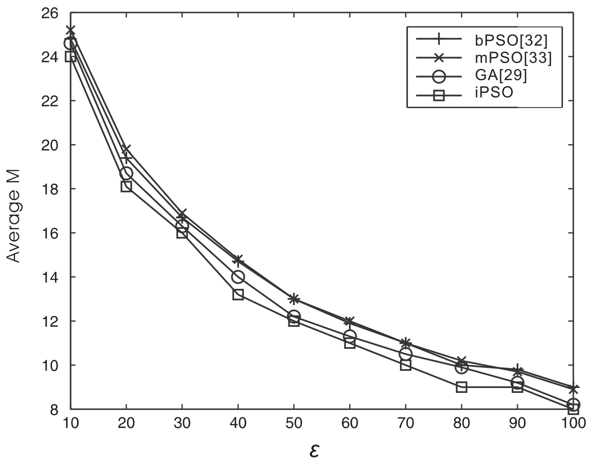
(a) Semicircle



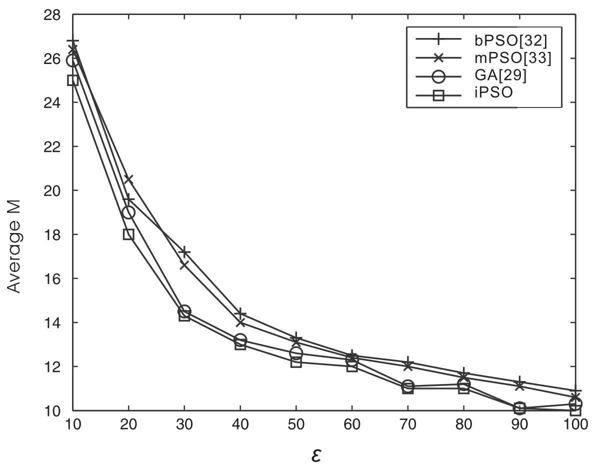
(b) Leaf



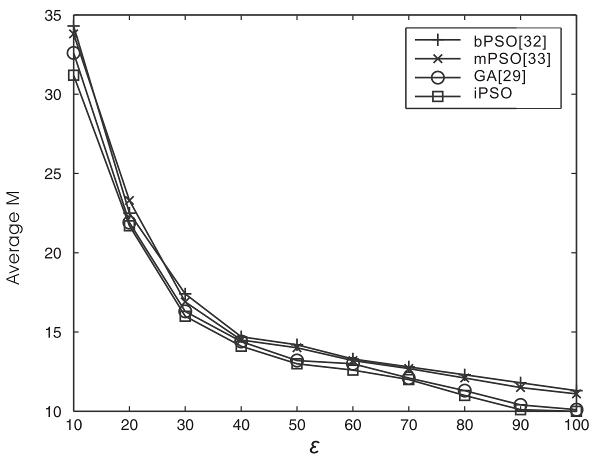
(c) Arlington



(d) Como



(e) Managua



(f) Simcoe

Fig. 6. The average results of bPSO [27], mPSO [23], GA [24], and the proposed iPSO on all testing instances.

tion vector is the number of vertices in the approximating polygon and the position update does not increase. Moreover, the position update tends to reduce the length of the position vector. We find that iPSO has a higher probability of generating more-optimal solutions than bPSO.

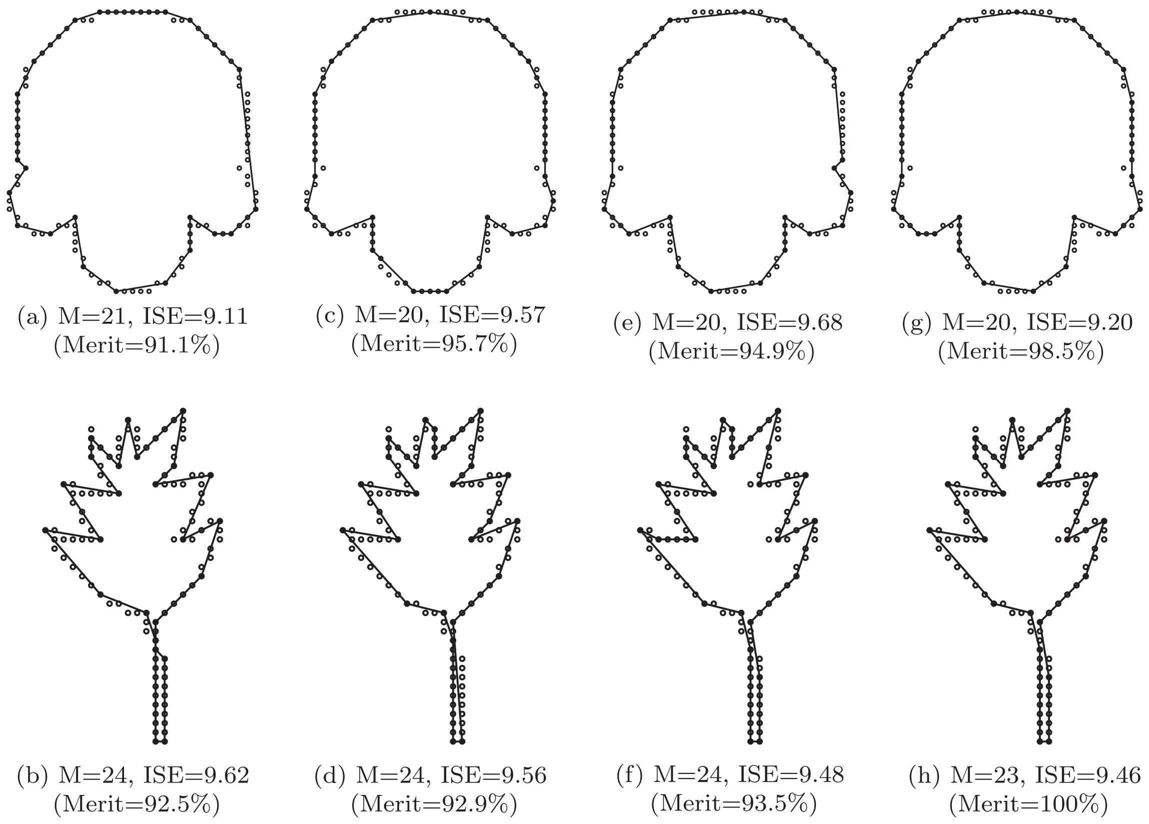


Fig. 7. A visual comparison of the best solutions generated by the various methods on two synthesized shapes: (a)–(b) for bPSO, (c)–(d) for mPSO, (e)–(f) for GA, and (g)–(h) for the proposed iPSO.

6. Experimental results

For our experiments, two groups of shapes were used to evaluate the performance of the proposed iPSO. Fig. 4 shows the first group of two synthesized shapes designed by Teh and Chin [22]. Fig. 4(a) is a shape with 102 points, and Fig. 4(b) is a leaf shape with 120 points. This collection of shapes has been widely used as a testing benchmark for existing polygonal approximation methods as in [4,11,20–27]. The second shape group consists of four lakes as shown in Fig. 5, which were extracted from the maps of four famous lakes of America, Italy, Nicaragua and Canada, respectively, by Wang et al. [24]. Fig. 5 shows Lake Arlington with 133 points (a), Lake Como with 124 points (b), Lake Managua with 120 points (c), and Lake Simcoe with 134 points (d). Since a shape is usually represented by a chain code, the chain codes of these test instances are included in these figures. An algorithm for translating a chain code into a sequence of points in Cartesian coordinates is provided in Algorithm 2, in the Appendix A.

In the experiments, the proposed method was compared to three baseline algorithms: bPSO [27], mPSO [23] and GA [24]. Among them, bPSO [27] is the first work to apply the binary version of PSO to polygonal approximation problems. mPSO [23] also adopted the binary version of PSO and incorporated a mutation operator. GA [24] is a recently published GA-based method which has the best performance over the other nature inspired algorithms.

Due to the probabilistic nature of the searches, each method was run ten times with the best results and average results reported. All the methods were implemented in Borland Delphi 7.0 and were tested on a PC with Pentium Dual-core 2.80 GHz CPU and 2 GB RAM. For the proposed iPSO, parameters were set to $c_1 = 2$ and $c_2 = 2$. The population size was set at 30 and the number of iterations was 100 for all four methods. The other parameters for bPSO [27], mPSO [23] and GA [24] were set as detailed in their respective papers.

To evaluate the quality of an approximating polygon, the integral square error (ISE) and the number of the vertices, M, can be used to indicate the precision and compactness, respectively. These two measurements are interrelated and two polygons with different ISEs and vertex numbers cannot be directly compared. To overcome this problem, Rosin [18] proposed a unified performance measure for assessing the relative merits of the various methods. Rosin's method is based on measuring the difference between the obtained solution and the global optimal solution. An optimal algorithm, such as the dynamic programming method [13], is used to obtain the ground truth. Let E_{appr} and M_{appr} denote the ISE and the vertex count, respectively, of the approximating polygon produced by the algorithm to be tested. Let E_{opt} denote the ISE of the approximating

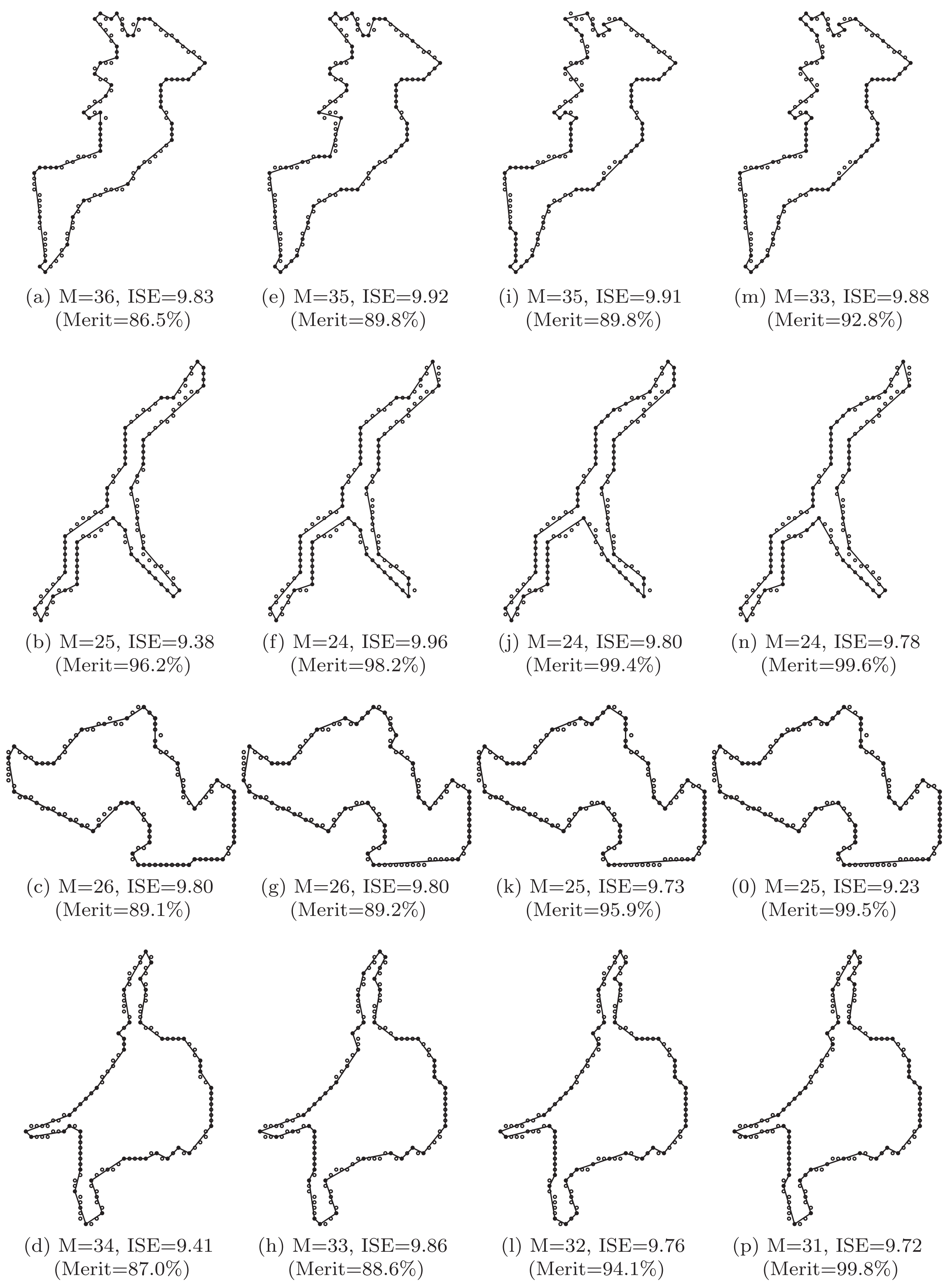

(a) M=36, ISE=9.83 (Merit=86.5%)
(e) M=35, ISE=9.92 (Merit=89.8%)
(i) M=35, ISE=9.91 (Merit=89.8%)
(m) M=33, ISE=9.88 (Merit=92.8%)
(b) M=25, ISE=9.38 (Merit=96.2%)
(f) M=24, ISE=9.96 (Merit=98.2%)
(j) M=24, ISE=9.80 (Merit=99.4%)
(n) M=24, ISE=9.78 (Merit=99.6%)
(c) M=26, ISE=9.80 (Merit=89.1%)
(g) M=26, ISE=9.80 (Merit=89.2%)
(k) M=25, ISE=9.73 (Merit=95.9%)
(0) M=25, ISE=9.23 (Merit=99.5%)
(d) M=34, ISE=9.41 (Merit=87.0%)
(h) M=33, ISE=9.86 (Merit=88.6%)
(l) M=32, ISE=9.76 (Merit=94.1%)
(p) M=31, ISE=9.72 (Merit=99.8%)

Fig. 8. A visual comparison of the best solutions generated by the various methods on four lake shapes, where (a)–(d) for bPSO, (e)–(h) for mPSO, (i)–(l) for GA, and (m)–(p) for the proposed iPSO.

polygon with M_{appr} vertices, generated by the optimal algorithm, and M_{opt} denote the vertex count that the optimal algorithm would require to produce the same ISE as the tested algorithm. The measurements proposed by Rosin [18] are calculated as

Table 1
The best results of GA [24], bPSO [27], mPSO [23] and the proposed iPSO on the synthesized shapes.

Curves	Method	ε	M	ISE	Fidelity (%)	Efficiency (%)	Merit (%)
Semicircle	bPSO [27]	10	21	9.11	87.6	94.8	91.1
	mPSO [23]		20	9.57	94.1	97.3	95.7
	GA [24]		20	9.68	93.1	96.7	94.9
	iPSO		20	9.20	97.9	99.1	98.5
	bPSO [27]	30	13	27.12	76.4	77.2	83.4
	mPSO [23]		12	28.78	90.3	96.3	93.3
	GA [24]		12	27.87	93.3	97.5	95.4
	iPSO		12	26.00	100	100	100
	bPSO [27]	60	10	50.45	77.1	95.1	85.7
	mPSO [23]		10	47.51	81.9	96.4	88.9
	GA [24]		10	45.39	87.8	97.3	91.3
	iPSO		10	38.92	100	100	100
	bPSO [27]	90	8	88.47	89.4	93.6	91.5
	mPSO [23]		8	81.30	92.9	95.9	94.4
	GA [24]		8	80.44	93.9	96.5	95.2
	iPSO		8	76.84	98.3	99.1	99.7
	bPSO [27]	120	7	116.22	80.1	93.0	86.3
	mPSO [23]		7	103.92	89.50	96.7	93.1
	GA [24]		7	102.24	90.0	97.2	94.1
	iPSO		7	93.05	100	100	100
	bPSO [27]	150	7	112.25	82.9	94.2	88.4
	mPSO [23]		7	104.23	89.3	96.6	92.9
	GA [24]		6	145.52	96.5	99.4	97.9
	iPSO		6	141.29	99.4	99.9	99.7
Leaf	bPSO [27]	10	24	9.62	89.8	95.3	92.5
	mPSO [23]		24	9.56	90.3	95.5	92.9
	GA [24]		24	9.48	91.2	95.8	93.5
	iPSO		23	9.46	100	100	100
	bPSO [27]	30	16	27.60	96.5	98.7	97.6
	mPSO [23]		16	27.56	95.5	98.3	96.9
	GA [24]		16	27.56	96.5	98.7	97.6
	iPSO		16	26.60	100	100	100
	bPSO [27]	60	13	57.65	83.4	93.8	88.4
	mPSO [23]		13	59.70	80.5	92.5	86.3
	GA [24]		13	54.80	87.8	95.7	91.6
	iPSO		12	59.94	100	100	100
	bPSO [27]	90	13	59.39	80.9	92.7	86.6
	mPSO [23]		12	76.90	77.9	95.1	86.1
	GA [24]		12	69.58	86.1	97.2	91.5
	iPSO		11	88.81	100	100	100
	bPSO [27]	120	12	79.27	75.6	94.4	84.5
	mPSO [23]		11	118.53	74.9	86.5	80.5
	GA [24]		10	119.04	87.8	95.0	91.3
	iPSO		10	104.57	100	100	100
	bPSO [27]	150	10	142.08	73.6	85.2	79.2
	mPSO [23]		10	147.22	71.0	82.3	76.5
	GA [24]		9	149.09	89.5	90.3	89.9
	iPSO		9	135.47	98.5	98.7	98.6

$$Fidelity = \frac{E_{opt}}{E_{appr}} \times 100\%, \tag{15}$$

$$Efficiency = \frac{M_{opt}}{M_{appr}} \times 100\%, \tag{16}$$

$$Merit = \sqrt{Fidelity \times Efficiency}. \tag{17}$$

Fidelity measures how well the test polygon fits the curve relative to the optimal polygon in terms of the approximation error. Efficiency measures how compact the testing polygon is, relative to the optimal polygon with the same error bound. Merit provides a combined measure of the precision and compactness. The advantage of Rosin's measure over the other measures [19] is that it can provide a relative measure and can also be used to compare approximations having different numbers of vertices. For these reasons, we use Rosin's method to measure and compare the results of the proposed method and the other benchmark methods, in this paper.

Table 2
The best results of GA [24], bPSO [27], mPSO [23] and the proposed iPSO on the lake shapes.

Curves	Method	ε	M	ISE	Fidelity (%)	Efficiency (%)	Merit (%)
Arlington	bPSO [27]	10	36	9.83	82.6	90.5	86.5
	mPSO [23]		35	9.92	87.0	92.6	89.8
	GA [24]		35	9.91	87.0	92.7	89.8
	iPSO		33	9.88	97.4	98.5	92.8
	bPSO [27]	30	18	28.36	86.7	95.9	92.8
	mPSO [23]		17	29.75	98.8	99.5	99.1
	GA [24]		17	29.56	99.5	99.8	99.6
	iPSO		17	29.45	99.8	99.9	99.9
	bPSO [27]	60	15	53.44	79.3	92.3	85.6
	mPSO [23]		15	45.89	92.4	97.5	94.9
	GA [24]		14	54.62	94.9	98.1	96.4
	iPSO		14	51.81	100	100	100
	bPSO [27]	90	12	86.62	82.2	89.0	85.5
	mPSO [23]		12	82.59	86.2	91.9	89.0
	GA [24]		11	86.74	95.5	97.0	96.3
	iPSO		11	84.44	98.2	98.8	98.5
	bPSO [27]	120	10	111.51	84.7	89.4	87.0
	mPSO [23]		10	117.13	80.6	87.5	84.0
	GA [24]		9	117.63	93.3	97.1	95.2
	iPSO		9	109.73	100	100	100
	bPSO [27]	150	9	135.74	80.8	90.3	85.5
	mPSO [23]		9	128.58	85.3	93.0	89.1
	GA [24]		8	148.50	94.0	96.9	95.5
	iPSO		8	145.70	95.9	97.9	96.9
Como	bPSO [27]	10	25	9.38	94.8	97.7	96.2
	mPSO [23]		24	9.96	97.7	98.8	98.2
	GA [24]		24	9.80	99.2	99.6	99.4
	iPSO		24	9.78	99.4	99.7	99.6
	bPSO [27]	30	17	29.52	72.2	88.1	79.7
	mPSO [23]		16	28.34	88.2	95.2	91.6
	GA [24]		15	29.53	99.5	99.8	99.7
	iPSO		15	29.40	100	100	100
	bPSO [27]	60	13	55.22	70.9	84.5	77.4
	mPSO [23]		12	53.85	83.3	92.7	87.9
	GA [24]		11	57.62	95.7	97.8	96.7
	iPSO		11	55.11	100	100	100
	bPSO [27]	90	10	86.20	76.9	84.0	79.9
	mPSO [23]		9	88.74	88.0	91.2	89.6
	GA [24]		9	86.67	90.1	92.9	91.5
	iPSO		9	80.68	96.7	97.8	97.3
	bPSO [27]	120	8	119.63	76.5	76.4	76.5
	mPSO [23]		8	100.39	91.2	93.1	92.2
	GA [24]		7	112.16	96.0	95.2	95.6
	iPSO		7	108.11	99.6	99.5	99.5
	bPSO [27]	150	7	136.43	78.9	82.6	80.7
	mPSO [23]		6	140.32	86.3	95.4	90.8
	GA [24]		6	131.20	92.3	97.6	94.9
	iPSO		6	121.13	100	100	100
Managua	bPSO [27]	10	26	9.80	85.0	93.5	89.1
	mPSO [23]		26	9.80	85.0	93.5	89.2
	GA [24]		25	9.73	94.2	97.5	95.9
	iPSO		25	9.23	99.3	99.7	99.5
	bPSO [27]	30	16	28.35	77.8	88.5	83.0
	mPSO [23]		16	26.09	84.6	92.0	89.2
	GA [24]		14	29.43	98.4	99.5	99.0
	iPSO		14	29.00	100	100	100
	bPSO [27]	60	12	57.00	81.6	94.0	87.6
	mPSO [23]		12	55.38	84.0	94.9	89.3
	GA [24]		12	50.62	91.7	97.6	94.7
	iPSO		12	47.58	97.8	99.4	98.6
	bPSO [27]	90	11	83.56	73.0	90.6	81.3
	mPSO [23]		11	81.32	75.0	91.4	82.8
	GA [24]		10	84.82	97.1	99.2	98.2
	iPSO		10	82.36	100	100	100
	bPSO [27]	120	10	109.56	75.2	91.7	83.0
	mPSO [23]		10	108.53	75.9	92.0	83.5
	GA [24]		9	117.81	97.6	99.1	98.3

Table 2 (*continued*)

Curves	Method	ε	M	ISE	Fidelity (%)	Efficiency (%)	Merit (%)
	iPSO		9	115.59	99.4	99.8	99.6
	bPSO [27]	150	9	149.33	77.0	88.8	82.7
	mPSO [23]		9	142.12	80.9	91.0	85.8
	GA [24]		8	149.31	99.6	99.9	99.8
	iPSO		8	148.65	100	100	100
Simcoe	bPSO [27]	10	34	9.41	82.0	92.3	87.0
	mPSO [23]		33	9.86	84.4	93.1	88.6
	GA [24]		32	9.76	91.8	96.5	94.1
	iPSO		31	9.72	99.7	99.9	99.8
	bPSO [27]	30	17	28.61	89.9	93.6	91.7
	mPSO [23]		17	27.89	92.2	94.9	93.6
	GA [24]		16	28.90	97.7	99.0	98.4
	iPSO		16	28.24	100	100	100
	bPSO [27]	60	13	55.35	78.9	93.8	86.0
	mPSO [23]		13	52.76	82.8	95.1	88.7
	GA [24]		12	59.63	97.4	98.9	98.1
	iPSO		12	58.07	100	100	100
	bPSO [27]	90	11	87.93	79.3	88.8	83.9
	mPSO [23]		11	86.68	80.4	89.2	84.7
	GA [24]		10	87.53	93.1	97.8	95.4
	iPSO		10	81.73	99.8	99.9	99.8
	bPSO [27]	120	10	111.89	72.9	89.5	80.8
	mPSO [23]		10	109.51	74.5	89.9	81.8
	GA [24]		9	114.08	95.1	99.2	97.1
	iPSO		9	108.46	100	100	100
	bPSO [27]	150	9	144.58	75.0	94.5	84.2
	mPSO [23]		9	139.36	77.8	95.3	86.1
	GA [24]		9	130.29	83.2	96.7	89.7
	iPSO		9	119.77	90.6	98.3	94.3

Fig. 6 shows the average numbers of polygon vertices (M) of ten independent runs of the four methods on all the test instances. The $\varepsilon - M$ graphs show the average performance versus M of bPSO [27], mPSO [23], GA [24], and the proposed iPSO, respectively, for $\min - \varepsilon$ optimization. The best performances of the four methods on the two synthesized shapes are listed in Table 3. Table 2 lists performances for the four lake shapes.

For an error tolerance of $\varepsilon = 10$, Fig. 7 shows a visual comparison of polygonal approximations using the four methods on the two synthesized shapes, while Fig. 8 shows a visual comparison when applied to the lake shapes. The small circles denote the curve points, and the dot denotes the selected polygon vertices. Each pair of adjacent vertices is connected by a line segment to form the approximating polygon. The calculation time for all cases shown in Fig. 7 are as follows: bPSO requires 0.134 s, mPSO requires 0.156 s, GA requires 0.031 s and iPSO requires 0.076 s. Likewise, the calculation times for all cases shown in Fig. 8 are as follows: bPSO requires 0.293 s, mPSO requires 0.317 s, GA requires 0.063 s and iPSO requires 0.169 s.

On the two groups of testing shapes, the proposed iPSO can be seen to outperform the benchmark methods (i.e. bPSO [27], mPSO [23] and GA [24]), in terms of the best and average solutions. With a fixed error tolerance ε, iPSO approximated polygons with the smallest numbers of vertices. Where the number of vertices from all four methods are equal (e.g. $\varepsilon = 60$ for semicircle in Tables 1 and 2), the proposed iPSO still generates the best approximation in terms of error. When using Rosin's method to measure the performances, the proposed iPSO systematically outperformed the benchmark methods in terms of Fidelity, Efficiency and Merit for all instances, as shown in Tables 1 and 2. In some cases (e.g. $\varepsilon = 30$ for Semicircle, $\varepsilon = 10$ for Leaf, $\varepsilon = 120$ for Arlington, $\varepsilon = 60$ for Como, $\varepsilon = 90$ for Managua and $\varepsilon = 30$ for Simcoe), iPSO achieved Fidelity, Efficiency and Merit values of 100, indicating that iPSO found the global optimal solutions in the search space. With regards to computational speed, iPSO is faster than the other two PSO-based methods due to the efficient coding and updating mechanism, and while being slower than GA, iPSO offers better accuracy.

At this point, the proposed iPSO has been compared with three benchmark methods, bPSO [27], mPSO [23], and GA [24] and the experimental results indicates its higher performance over the other approaches. To further demonstrate the superiority of the proposed method, other recent algorithms developed for solving the polygonal approximation problem are included for comparison. They are the Ant Colony Search (ACS) [26], Dominant Point Deletion (DPD) [10] and Betweenness [2] methods. Table 3 shows the average results over 10 independent runs on the synthesized shapes for all the competing methods, where the values of M and ISE of polygonal approximations generated by ACS [26], DPD [10] and Betweenness [2] are from the reported results [2] and the corresponding values of Fidelity, Efficiency and Merit are calculated by Eqs. (15)–(17) respectively.

It can be seen in Table 3 that in the seven test cases, from the absolute measure values and relative measure values, the proposed iPSO outperforms the ACS [26], DPD [10] and Betweenness [2] methods. Only in three cases ($\varepsilon = 25$ and $\varepsilon = 60$ for Semicircle shape and $\varepsilon = 150$ for leaf shape), is iPSO not the best among the four competing methods, however, our method

Table 3
Comparison of the proposed iPSO and the other methods including ACS [26], DPD [10] and Betweenness [2] on the synthesized shapes.

Curves	Method	ε	M	ISE	Fidelity (%)	Efficiency (%)	Merit (%)
Semicicle	Betweenness [2]	15	15.0	14.3	100	100	100.0
	ACS [26]		18.0	14.1	79.4	85.1	82.2
	DPD [10]		15.0	14.4	100	100	100
	iPSO		15.0	14.3	100	100	100
	Betweenness [2]	20	14.0	19.8	87.8	94.8	91.3
	ACS [26]		16.4	19.9	65.1	80.8	72.5
	DPD [10]		14.0	18.0	96.6	98.7	97.6
	iPSO		14.0	17.9	97.1	98.9	98.0
	Betweenness [2]	25	13.0	24.8	83.6	94.1	88.7
	ACS [26]		13.4	23.6	82.2	93.0	87.4
	DPD [10]		13.0	21.4	96.8	99.0	97.9
	iPSO		13.0	21.8	95.1	98.4	96.7
	Betweenness [2]	30	12.0	28.9	89.9	96.2	93.0
	ACS [26]		12.6	27.9	81.9	92.9	87.2
	DPD [10]		12.0	27.3	95.3	98.3	96.8
	iPSO		12.0	26.9	96.7	98.8	97.7
	Betweenness [2]	60	10.0	38.9	100	100	100
	ACS [26]		10.0	58.6	66.4	91.7	78.0
	DPD [10]		10.0	38.9	100	100	100.0
	iPSO		10.0	39.8	97.8	99.6	98.7
Leaf	Betweenness [2]	15	20.0	14.1	94.9	97.9	96.4
	ACS [26]		22.2	15.0	69.5	85.7	77.2
	DPD [10]		20.0	14.1	94.9	97.9	96.4
	iPSO		20.0	13.9	96.3	98.5	97.4
	Betweenness [2]	30	16.0	27.5	96.7	98.8	97.8
	ACS [26]		17.2	21.3	95.6	98.9	97.2
	DPD [10]		16.0	27.0	98.5	99.5	99.0
	iPSO		16.0	26.8	99.3	99.7	99.5
	Betweenness [2]	90	12.0	70.5	85.0	97.0	90.8
	ACS [26]		13.2	81.8	57.6	85.2	70.1
	DPD [10]		12.0	85.9	69.8	92.5	80.3
	Betweenness [2]	100	11.0	99.7	89.1	93.7	91.4
	ACS [26]		13.0	88.6	54.3	84.7	67.8
	DPD [10]		12.0	85.9	69.8	92.5	80.3
	iPSO		11.0	90.9	97.7	98.8	98.3
	Betweenness [2]	150	9.0	136.2	98.0	98.3	98.1
	ACS [26]		11.2	149.5	55.5	72.3	63.4
	DPD [10]		10.0	127.2	82.2	92.2	87.0
	iPSO		9.0	136.6	97.7	98.0	97.8

still achieves results comparable with the best method. In these cases, our method creates polygonal approximations having the same number of vertices as the ones generated by the best method, and the *ISE* approximation error of the best method is only marginally better than our method (less than one pixel difference of *ISE* between our method and the best method). It is worth noting that among the four competing methods, the proposed iPSO and Ant Colony Search (ACS) [26] are both nature inspired algorithms, though our method is far better than ACS [26] in terms of absolute and relative measures.

7. Conclusion

In this paper, a novel integer PSO algorithm, iPSO, has been proposed for solving the polygonal approximation problem. The proposed iPSO algorithm utilizes integer coding to provide a direct solution representation. Compared to the binary version of PSO (bPSO), iPSO's coding is significantly shorter and its updating mechanism has clearer physical meaning. These characteristics offer iPSO a better performance in terms of solution quality and computational speed. In the experiments, the proposed iPSO has been tested on two commonly used synthesized shapes and four real lake shapes. The results show that the proposed iPSO has higher performance on all test shapes than the bPSO-based methods [23,27], in terms of the quality of the solutions and the computational efficiency. As a nature inspired algorithm, the proposed iPSO was also compared with two other types of nature inspired algorithms, that is, the GA-based method [24] and the Ant Colony Search method (ACS) [26]. The comparative results indicate its superiority to them. Two other recent methods, namely DPD [10] and Betweenness [2], are also included in our comparison. Our method achieves better or comparable performances than them in terms of solution merit.

It is worth noting that although the proposed iPSO was developed for solving min–# polygonal approximation problems in this paper, the framework of the proposed method can also be applied to solving min $-\varepsilon$ polygonal approximation problems. When applying iPSO for min $-\varepsilon$ problems, only the fitness evaluation and position adjustment need be updated. For min $-\varepsilon$ problems, the fitness evaluation is associated with the approximation error instead of the number of vertices, i.e. the polygons with smaller approximation error have better fitness values. For the position adjustment, since the constraint condition is changed and the infeasible solutions are those whose number of vertices is not equal to the pre-specified number of vertices, we can mend the infeasible solutions via vertex insertion and removal techniques. That is to say only a small change is required to be made against the proposed iPSO on applying it to min $-\varepsilon$ problems.

Polygonal approximation is a combinatorial optimization problem and this paper demonstrates that the proposed iPSO provides a better solution than bPSO can offer. Future research may be conducted to extend the idea of iPSO to other combinatorial optimization problems such as, e.g. the traveling salesman [3] and job shop scheduling [12] problems.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China (Grant No. 61372158), Mayor project of Natural Science Foundation of Jiangsu Higher Education Institutions of China (Grant No. 11KJA520004), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions and National Center for International Joint Research on E-Business Information Processing under Grant 2013B01035.

Appendix A

The pseudo code for translating the chain code into a sequence of Cartesian coordinate points is described as follows:

Algorithm 2. Translating the chain code into a sequence of points in cartesian coordinates

```
Input: a chain code $Z = \{z(1), z(2), \ldots, z(N)\}$,where N is the length of $Z$.
Output: a sequence of points $C = \{(x(i), y(i)) \mid i = 1, 2, \ldots, N\}$, where $(x(i), y(i))$
        is the coordinate of the ith point.
begin
    · Initialize $tx = (1, 1, 0, -1, -1, -1, 0, 1)$, $ty = (0, 1, 1, 1, 0, -1, -1, -1)$;
    · $x(1) = tx(z(1) + 1)$;
    · $y(1) = ty(z(1) + 1)$;
    for $i \leftarrow 1$ to $N - 1$ do
        · $x(i + 1) = x(i) + tx(z(i) + 1)$;
        · $y(i + 1) = y(i) + ty(z(i) + 1)$;
    · Return $\{(x(i), y(i)) \mid i = 1, 2, \ldots, N\}$;
end
```

References

[1] F. Attneave, Some informational aspects of visual perception, Psychol. Rev. 61 (4) (1954) 183–193.

[2] A. R Backes, O.M. Bruno, Polygonal approximation of digital planar curves through vertex betweenness, Inf. Sci. 222 (2013) 795–804.

[3] S.M. Chen, C.Y. Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Syst. Appl. 38 (2011) 14439–14450.

[4] S.Y. Ho, Y.C. Chen, An efficient evolutionary algorithm for accurate polygonal approximation, Pattern Recogn. 34 (12) (2001) 2305–2317.

[5] S.C. Huang, Y.N. Sun, Polygonal approximation using genetic algorithms, Pattern Recogn. 32 (8) (1999) 1409–1420.

[6] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. IEEE Int. Conf. Neural Networks-ICNN'95, NJ, Piscataway, vol. IV, 1995, pp. 1942–1948.

[7] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm optimization, in: Proc. Int. Conf. systems man cybernetics'97, NJ, Piscataway, 1997, pp. 4104–4108.

[8] A. Kolesnikov, P. Fränti, Min–# polygonal approximation of closed curves, in: Proc. Int. Conf. Image Processing-ICIP'05, Genova, Italy, vols 1–5, September, 2005, pp. 1553–1556.

[9] A. Kolesnikov, P. Fränti, Polygonal approximation of closed discrete curves, Pattern Recogn. 40 (4) (2007) 1282–1293.

[10] A. Masood, Optimized polygonal approximation by dominant point deletion, Pattern Recogn. 41 (1) (2008) 227–239.

[11] A. Masood, S.A. Haq, A novel approach to polygonal approximation of digital curves, J. Vis. Commun. Image Represent. 18 (3) (2007) 264–274.

[12] J.C.H. Pan, H.C. Huang, A hybrid genetic algorithm for no-wait job shop scheduling problems, Expert Syst. Appl. 36 (3) (2009) 5800–5806.

[13] J.C. Perez, E. Vidal, Optimum polygonal approximation of digitized curves, Pattern Recogn. Lett. 15 (8) (1994) 743–750.

[14] A. Pikaz, I. Dinstein, An algorithm for polygonal approximation based on iterative point elimination, Pattern Recogn. Lett. 16 (6) (1995) 557–563.

[15] A.C. Poyato, F.J.M. Cuevas, R.M. Carnicer, R.M. Salinas, Polygonal approximation of digital planar curves through break point suppression, Pattern Recogn. 43 (1) (2010) 14–25.

[16] B.K. Ray, K.S. Ray, An algorithm for detection of dominant points and polygonal approximation of digitized curves, Pattern Recogn. Lett. 13 (12) (1992) 849–856.

[17] B.K. Ray, K.S. Ray, Determination of optimal polygon from digital curve using L_1 norm, Pattern Recogn. 26 (4) (1993) 505–509.

[18] P.L. Rosin, Techniques for assessing polygonal approximations of curves, IEEE Trans. Pattern Anal. Mach. Intell. 19 (6) (1997) 659–666.

[19] D. Sarkar, A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves, Pattern Recogn. Lett. 14 (12) (1993) 959–964.

[20] B. Sarkar, L.K. Singh, D. Sarkar, A genetic algorithm-based approach for detection of significant vertices for polygonal approximation of digital curves, Int. J. Image Graph. 4 (2) (2004) 223–239.

[21] Y.N. Sun, S.C. Huang, Genetic algorithms for error-bounded polygonal approximation, Int. J. Pattern Recogn. Artif. Intell. 14 (3) (2000) 297–314.

[22] H.C. Teh, R.T. Chin, On detection of dominant points on digital curves, IEEE Trans. Pattern Anal. Mach. Intell. 11 (8) (1989) 859–872.

[23] B. Wang, H.Z. Shu, B.S. Li, Z.M. Niu, A mutation-particle swarm algorithm for error-bounded polygonal approximation of digital curves, Lect. Notes Comput. Sci. 5226 (2008) 1149–1155.
[24] B. Wang, H.Z. Shu, L.M. Luo, A genetic algorithm with chromosome-repairing for min–# and min–ε polygonal approximation of digital curves, J. Vis. Commun. Image Represent. 20 (1) (2009) 45–56.
[25] P.Y. Yin, A new method for polygonal approximation using genetic algorithms, Pattern Recogn. Lett. 19 (11) (1998) 1017–1026.
[26] P.Y. Yin, Ant colony search algorithms for optimal polygonal approximation of plane curves, Pattern Recogn. 36 (8) (2003) 1783–1997.
[27] P.Y. Yin, A discrete particle swarm algorithm for optimal approximation of digital curves, J. Vis. Commun. Image Represent. 15 (2) (2004) 241–260.
[28] H. Zhang, J. Guo, Optimal polygonal approximation of digital planar curves using meta heuristics, Pattern Recogn. Lett. 34 (7) (2001) 1429–1436.