Available online at www.sciencedirect.com

SCIENCE DIRECT®

J. Vis. Commun. Image R. 15 (2004) 241–260

www.elsevier.com/locate/jvci

A discrete particle swarm algorithm for optimal polygonal approximation of digital curves

Peng-Yeng Yin[*]

Department of Information Management, National Chi Nan University, Nantou 545, Taiwan, ROC

Received 17 September 2003; accepted 2 December 2003

Abstract

Polygonal approximation of digital curves is one of the crucial steps prior to many image analysis tasks. This paper presents a new polygonal approximation approach based on the particle swarm optimization (PSO) algorithm. Each particle represented as a binary vector corresponds to a candidate solution to the polygonal approximation problem. A swarm of particles are initiated and fly through the solution space for targeting the optimal solution. We also propose to use a hybrid version of PSO embedding a local optimizer to enhance the performance. The experimental results manifest that the proposed discrete PSO is comparable to the genetic algorithm, and it outperforms another discrete implementation of PSO in the literature. The proposed hybrid version of PSO can significantly improve the approximation results in terms of the compression ratio, and the results obtained in different runs are more consistent.
© 2003 Elsevier Inc. All rights reserved.

Keywords: Polygonal approximation; Particle swarm optimization; Genetic algorithm; Local optimal solution; Global optimal solution

1. Introduction

Shape feature representation is one of the crucial steps prior to many pattern analysis tasks such as object recognition, image matching, target tracking, just to name a

[*] Fax: +886-49-2915205.
E-mail address: pyyin@ncnu.edu.tw.

1047-3203/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2003.12.001

few. It is not only because the accuracy of the extracted features is the most relevant to the precision performance of pattern analysis but also the volume of the features directly impacts on the storage complexity and computational time of the subsequent tasks. Since the information of a shape is mainly preserved at the corners which have the maximal measure of significance, e.g., the curvature, in their neighborhood, it is desirable to represent a shape by a polygon. The resulting polygon should have two characteristics. *First*, it closely approximates the curve along the shape contours within a given tolerance error to preserve as much information as possible. *Second*, the *degree* of the polygon, as defined as the number of vertices at the corners of the polygon, is as small as possible to reduce the storage demand and to accelerate the processing speed of following tasks. In this context, we can define the problem in question as "approximate a given curve by a polygon with the minimal number of vertices such that the approximation error between the original curve and the corresponding polygonal line segments is no more than a pre-specified tolerance."

Many polygonal approximation techniques have been proposed in the literature. They can be classified into three categories: (1) sequential approaches, (2) split-and-merge approaches, and (3) dominant point-detection approaches. For sequential approaches, Sklansky and Gonzalez (1980) proposed a scan-along procedure which starts from a point chosen at random and tries to find the longest line segments sequentially. Kurozumi and Davis (1982) proposed a minimax method which derives the approximating segments by minimizing the maximum distance between a given set of points and the corresponding segment. Wall and Danielsson (1984) proposed a sequential method which scans along the given points and outputs a new segment when the area deviation per length unit of the current segment exceeds a pre-specified error. Ray and Ray (1993) proposed a method which determines the longest possible line segments with the minimum possible error. Most of the sequential approaches are simple and fast, but the quality of their approximating results is dependent upon the location of the point at which they start the scan-along process. For split-and-merge approaches, Ramer (1972) presented an iterative method which starts with an initial segmentation and splits the segment at the point which has the farthest distance from the corresponding segment unless the approximation error is no more than the pre-specified tolerance. Leu and Chen (1988) presented a hierarchical merging method which looks for consecutive arcs with their arc-to-chord deviation less than a tolerance error. Those arcs are then replaced by their chord. The process is iterated until no arcs can be replaced for the given tolerance. Ansari and Delp (1991) proposed another technique which first uses Guassian smoothing to smooth the boundary and then takes the maximal curvature points as break points to which the split-and-merge process is applied. Ray and Ray (1995) proposed an orientation-invariant and scale-invariant method by introducing the use of ranks of points and the normalized distances. The approximation results of the split-and-merge approaches may be far from the optimal one if a poor initial segmentation is used. For dominant point-detection approaches, Teh and Chin (1989) determine the region of support for each point based on its local property to evaluate the curvature. The dominant points are then detected by a nonmaxima suppression process. Wu and Wang (1993) combined the corner detection and the iterative partition methods to detect the dominant points. Held et al. (1994) first apply a coarse-to-fine smoothing scheme to detect dominant points, a hierarchical approximation is then constructed based on perceptual significance. Zhu and Chirlian (1995) determines the importance of the data points by critical levels, the dominant points are then detected using mathematical criteria. The performance of most dominant point-detection methods heavily depends on the accuracy of the curvature evaluation.

Most of the existing polygonal approximation approaches are local search methods, i.e., the number of vertices of the derived polygon is a local minimum which can be significantly larger than the global optimal one. Dunham (1986) and Sato (1992) used dynamic programming technique to derive the global optimal solution to the problem. Both of their methods require a worst-case complexity of $O(N^4)$ where N is the number of data points on the curve. However, their methods become impractical when N is large, which is the case of real-world applications. Fortunately, there exist a bunch of natural algorithms which borrow the intelligence from the nature organisms and inherit a global search mechanism, some typical examples are genetic algorithms (GA), ant colony optimization (ACO), particle swarm optimization (PSO), among others. Yin (1999) proposed a GA-based approach for solving polygonal approximation problem and empirically showed that his method outperforms many existing local heuristic methods. Huang and Sun (1999) proposed a GA method which had been shown to be independent of the start point and the initial segmentation. Ho and Chen (2001) developed an efficient evolutionary algorithm with an orthogonal array crossover for obtaining better results than the existing genetic-based approaches. More recently, Yin (2003) applies the ACO by simulating the ant foraging behavior for constructing the approximating polygon with the minimal degree. In this paper, we further investigate the feasibility of applying the PSO to polygonal approximation problems. Most existing PSO applications are resorting to continuous function value optimization (Eberhart and Shi, 1998a; Kennedy and Eberhart, 1995; Shigenori et al., 2003; Tandon, 2000; Yoshida et al., 1999), only a few researches have been conducted for discrete combinatorial optimization (Kennedy and Eberhart, 1997; Salman et al., 2002). The contribution of this paper is twofold. *First*, we further extend the research on polygonal approximation using natural algorithms which had exhibited superiority in many other applications. *Second*, the original PSO is tailored to continuous function value optimization, we adopt the discrete version and propose a hybrid strategy for enhancing its efficiency in solving combinatorial optimization problems.

The remainder of this paper is organized as follows. Section 2 reviews the particle swarm optimization. Section 3 describes the problem and renders the details of the proposed method. In Section 4, we present the experimental results and the discussions. Finally, a summary is given in Section 5.

2. Particle swarm optimization

The particle swarm optimization (PSO) algorithm was proposed by Kennedy and Eberhart (1995), and had been applied to evolving weights and structure for artificial neural networks (Eberhart and Shi, 1998a), manufacture end milling (Tandon, 2000), reactive power and voltage control (Yoshida et al., 1999), state estimation for electric power distribution systems (Shigenori et al., 2003), to name a few. The convergence and parameterization aspects of the PSO have also been discussed thoroughly (Clerc and Kennedy, 2002; Parsopoulos and Vrahatis, 2002; Trelea, 2003). The PSO is inspired by the behavior of bird flocking and fish schooling. A large number of birds/fishes flock synchronously, change direction suddenly, and scatter and regroup together. Each individual, called a particle, benefits from the experience of its own and that of the other members of the swarm during the search for food. The PSO models the social dynamics of flocks of birds and serves as an optimizer for nonlinear functions.

The general principles for the PSO algorithm are stated as follows. Given an optimization function $f(P)$ where P is a vector of n real-valued random variables, the PSO initializes a swarm of particles, each of which is represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$, $i = 1, 2, \ldots, K$, where K is the swarm size. Thus, each particle is randomly positioned in the n-dimensional real number space and is a candidate solution to the optimization function. In PSO, particle i remembers the best position it visited so far, referred to as $pbest_i$, and the best position by its neighbors. There are two versions for keeping the neighbors' best position, namely *lbest* and *gbest*. In the local version, each particle keeps track of the best position *lbest* attained by its local neighboring particles. For the global version, the best position *gbest* is determined by any particles in the entire swarm. Hence, the *gbest* model is a special case of the *lbest* model. The PSO is an evolutionary computation algorithm and in each generation, particle i adjusts its velocity v_{ij} and position p_{ij} through each dimension j by referring to, with random multipliers, the personal best position ($pbest_{ij}$) and the swarm's best position ($gbest_j$, if the global version is adopted) using Eqs. (1) and (2) as follows:

$$v_{ij} = v_{ij} + c_1 r_1 (pbest_{ij} - p_{ij}) + c_2 r_2 (gbest_j - p_{ij}) \tag{1}$$

and

$$p_{ij} = p_{ij} + v_{ij}, \tag{2}$$

where c_1 and c_2 are the acceleration constants and r_1 and r_2 are random real numbers drawn from $U(0, 1)$. Thus the particle flies through potential solutions toward $pbest_i$ and *gbest* in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optima. The particle's velocity on each dimension is set restricted by a maximum velocity $v_{\max}$, which controls the maximum travel distance at each iteration to avoid this particle flying past good solutions. The PSO algorithm is terminated with a maximal number of generations or the best particle position of the entire swarm cannot be improved further after a sufficiently large number of generations.

The PSO algorithm has shown its robustness and efficacy in solving function value optimization problems in real number space (Eberhart and Shi, 1998a; Shigenori et al., 2003; Tandon, 2000; Yoshida et al., 1999), only a few researches have been conducted for extending PSO to discrete combinatorial optimization problems. Salman et al. (2002) apply the continuous version of PSO to the task assignment problem which is in nature a discrete combinatorial optimization problem. To facilitate the conversion from real numbers to positive integer numbers, they simply drop the sign and the fraction part of the real numbers. In this context, Kennedy and Eberhart (1997) have presented a generic discrete binary version of PSO where the particles take the values of binary vectors of length n and the velocity represents the probability that a bit will take the value 1. The velocity updating formula (Eq. (1)) remains unchanged, but it is constrained to the interval [0.0, 1,0] by a limiting transformation $S(v)$. Then the particle changes its bit value by

$$p_{ij} = \begin{cases} 1 & \text{if } r_3 \leqslant S(v_{ij}), \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where r_3 is a random number drawn from $U(0,1)$. In essence, p_{ij} will very likely be changed to 1 if both $pbest_{ij}$ and $gbest_j$ are 1's and p_{ij} is 0, and vice versa. When $pbest_{ij}$, $gbest_j$ and p_{ij} have the same value, the probability of being one or zero remains unchanged to preserve the good solutions. Thus, the particle moves toward $pbest_j$ and $gbest$ in the discrete binary space. The discrete binary PSO is a general-purpose optimizer to combinatorial optimization problems.

Although Kennedy and Eberhart (1997) have tested the robustness of the discrete binary version through De Jong's reputable function optimization testbed (Goldberg, 1989), few applications for combinatorial optimization have ever been developed based on their work. Additionally, the research about the feasibility of applying the discrete PSO to combinatorial optimization problems which often consider the ordering or selection of elements is scarcely explored. In this paper, a typical combinatorial optimization problem, namely the polygonal approximation of a digital curve, is solved based on the discrete PSO. A hybrid strategy embedding a local optimizer within the discrete PSO has been shown to outperform the original version.

3. Discrete PSO for polygonal approximation

In this section, we first define the polygonal approximation problem and then a discrete PSO for solving the problem is presented.

3.1. Problem definition

Given a digital curve with N points, the clockwise ordered sequence of these points can be represented by $S = \{x_1, x_2, \ldots, x_N\}$, which is circular, i.e., x_1 is considered as the succeeding point of x_N. We define arc $\widehat{x_ix_j}$ as the consecutive points $x_i, x_{i+1}, \ldots, x_j$ from the circular sequence S, and chord $\overline{x_ix_j}$ as the line segment connecting points x_i and x_j. The approximation error between $\widehat{x_ix_j}$ and $\overline{x_ix_j}$, denoted by $e(\widehat{x_ix_j}, \overline{x_ix_j})$, is usually estimated by the L_2 norm, i.e., the sum of squared perpendicular distance from every point on $\widehat{x_ix_j}$ to $\overline{x_ix_j}$. A polygon with a sequence of M vertices represented by $T = \{x_{p_1}, x_{p_2}, \ldots, x_{p_M}\}$, where $T \subset S$ and $3 \leqslant M \leqslant N$, approximates the curve with a total error $E = \sum_{i=1}^{M} e(\widehat{x_{p_i}x_{p_{i+1}}}, \overline{x_{p_i}x_{p_{i+1}}})$. Formally, the polygonal approximation problem is formulated as follows:

$$\arg\min_{T \subset S} |T| \quad \text{subject to } 3 \leqslant |T| \leqslant N \text{ and } E \leqslant \varepsilon, \tag{4}$$

where $|T|$ denotes the cardinality of T and ε is the specified error tolerance.

3.2. Particle representation and fitness evaluation

The solution space of a polygonal approximation problem consists of those polygons which can be produced by choosing vertices from the given point set S. Since particles of the PSO are candidate solutions of the underlying problem, we represent each particle by a binary vector as follows.

$$P_i = (p_{i1}, p_{i2}, \ldots, p_{iN}) \quad \text{subject to } \sum_{j=1}^{N} p_{ij} \geqslant 3 \text{ and } p_{ij} \in \{0, 1\}, \tag{5}$$

where $p_{ij} = 1$ if x_{ij} is one of the vertices chosen to produce the polygon, and $p_{ij} = 0$ otherwise. Thus, the particle representation indicates which points are chosen from the point set to construct the polygon.

The fitness of the particle is evaluated in two ways. If the approximation error produced by a polygon exceeds the specified error tolerance, i.e., $E > \varepsilon$, the fitness of the corresponding particle will be assigned a negative value to express the infeasibility degree of this candidate solution, else the particle fitness is set to the inverse of the sum of bit values to assess the solution quality in terms of the number of vertices. The swarm of particles evolve to maximize the fitness function as described by

$$\max \text{ fitness}(P_i) = \begin{cases} -E/\varepsilon N & \text{if } E > \varepsilon, \\ 1/\sum_{j=1}^{N} p_{ij} & \text{otherwise.} \end{cases} \tag{6}$$

Here, we use $1/\sum_{j=1}^{N} p_{ij}$ instead of $N - \sum_{j=1}^{N} p_{ij}$ because the former formula would assign approximate equal fitness values to particles when their bit-sum (i.e., $\sum_{j=1}^{N} p_{ij}$) is high and give differential fitness values when the bit-sum is relative small. This is a desired property in modern search heuristics such that exploration search is emphasized at the beginning (i.e., when most particles have high values of bit-sum) to give all candidate solutions equal probabilities to be explored, while exploitation search is intensified as the number of iterations increase (i.e., when particles have low values of bit-sum) to further plunge the search into the neighborhood of high-quality solutions.

It is worth noting that there are two optimization goals in our objective function. The first one is to move the particle from infeasible solution space to feasible regions, and the second one is to fly the particle to a new position which may result in a polygon with fewer vertices. The two optimization goals are pursued simultaneously since the PSO evolves with a swarm of particles and each of which may invoke different fitness evaluation depending on the incurred approximation error.

3.3. Hybrid strategy

Modern search heuristics combine two elements: *exploration* and *exploitation*. The exploration strategy searches for new regions, and once it finds a good region the exploitation strategy kicks in. However, since the two strategies are usually interwound, the search is conducted to other regions before it finds the local optima. Many researchers suggest to employ a hybrid strategy which embeds a local optimizer such as hill-climbing in between the iterations of the global search heuristics (Goldberg, 1989). In the light of this, we also embed a local search heuristic in between the generations of the discrete PSO.

Not every existing local search heuristics to polygonal approximation can be directly adopted as a local optimizer within the PSO. For example, the sequential approaches which iteratively search for the next longest possible line segment emitting from previous vertex do not guarantee to locally improve the solutions delivered by particles if the sequential scan-along techniques are embedded to the PSO. Also, the dominant point-detection methods are not suited to be embedded because the approximating polygon having vertices with local maximum curvature does not correspond to a local optimum of particles. Fortunately, some of the split-and-merge heuristics to polygonal approximation can be adopted here. We propose to use the hierarchical segment-merging method (Leu and Chen, 1988) which iteratively looks for consecutive arcs and replaces them with their chord if the resulting error is still less than the tolerance amount. In addition, we propose another local optimizer, called segment-adjusting-and-merging technique, for comparison with the segment-merging method. At the adjusting phase, each vertex of the polygon is examined in turn and is, if possible, adjusted to a new data point on the arc between the two adjacent vertices such that the approximation error is reduced as much as possible. At the merging phase, two adjacent segments are merged if the resulting polygon still satisfies the error bound constraint, and the merging process is applied until no adjacent segments can be merged further. The adjusting-and-merging optimizer is more sophisticated and is able to bring the particles to local optima in some complex situations due to the error reduction process conducted in the adjusting phase while the segment-merging method cannot. However, the extra computational time incurred by the adjusting phase is negligible because only one scan of all data points on the curve is needed.

3.4. The algorithm

The precise algorithm of the proposed discrete hybrid PSO for solving the polygonal approximation problem is presented in Fig. 1. Initially, a swarm of particles are created and each of which is a binary vector corresponding to a candidate approximating polygon to the underlying problem. Moreover, the velocities of those particles are initialized at random by drawing real values from the range $[0.0, 1.0]$. Then, the evolutionary process of particle movement is iterated until a maximal number of iterations has been passed. At each iteration, the fitness values of particles are evaluated, and the particle individual best and swarm's best positions are thus

```
1. Initialize.
    1.1 Generate $K$ particles, $P_1, P_2, \ldots, P_K$, according to Eq. 5.
    1.2 Generate velocities $v_{ij}$, $1 \le i \le K$ and $1 \le j \le N$, where $v_{ij}$ is randomly drawn from [0.0, 1.0].
2. Repeat until a given maximal number of iterations is achieved.
    2.1 Evaluate the fitness of each particle using Eq. 6.
    2.2 Determine the best position $pbest_i$ visited so far by each particle.
    2.3 Determine the best position $gbest$ visited so far by the whole swarm.
    2.4 Update velocities $v_{ij}$ using Eq. 1 restricted by a maximum threshold $v_{max}$.
    2.5 Set $S(v_{ij}) = \frac{v_{ij}}{2v_{\max}} + 0.5$
    2.6 Update particles' positions using Eq. 3.
    2.7 Improve the solution quality of each particle using the embedded local optimizer.
```

Fig. 1. The discrete hybrid PSO algorithm for the polygonal approximation.

determined. The particle position adjustment follows the guidelines of discrete PSO, i.e., the velocity serves as a probability that the corresponding bit takes the value one, and it should be scaled into [0.0, 1.0] using any transformation function. In this implementation, we use a linear interpolation function (see Step 2.5). The particles' bit vectors are updated according to their velocities which measure the trajectory to the target solution based on the local experience (*pbest*$_i$) and the global intelligence (*gbest*). Before getting into the next iteration, each particle's position is improved using the embedded local optimizer.

In all of our experiments presented in the next section, we use a swarm of 20 particles, acceleration constants $c_1 = c_2 = 2$, and the maximal velocity $v_{max} = 6$. These parameter values are determined empirically and conform to most settings in existing applications.

4. Experimental results and discussions

In this section, we present our computational experiments and analyze the numerical results. The platform of the experiments is a PC with a 1.8 GHz CPU and 192 MB RAM. The algorithms are coded in C++. Three synthesized benchmark curves (see Fig. 2) which are broadly used in the literature (Held et al., 1994; Ho and Chen, 2001; Huang and Sun, 1999; Ray and Ray, 1993, 1995; Teh and Chin, 1989; Wall and Danielsson, 1984; Yin, 1999, 2003; Zhu and Chirlian, 1995) and two real image curves (see Figs. 7 and 8) are used for the experiments. Fig. 2A is a leaf curve with 120 points, Fig. 2B is a chromosome curve with 60 points, Fig. 2C is a semi-circle curve with 102 points, Fig. 7A is a fish contour image with 700 edge points, and Fig. 8A is a plane contour image with 682 edge points.

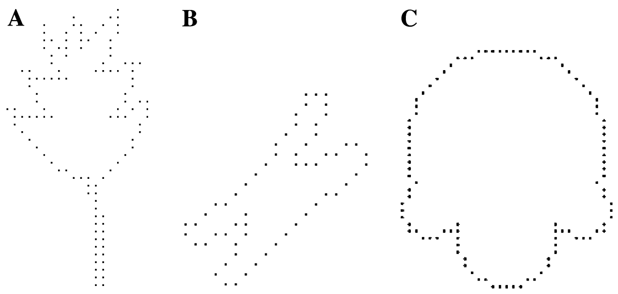

Fig. 2. Three benchmark curves: (A) leaf curve, (B) chromosome curve, and (C) semicircle curve.

4.1. Validation of the algorithm

To validate the feasibility of the proposed algorithm, we first conduct our experiments in the following two aspects. The first issue is about the appropriateness of the particle representation and fitness evaluation proposed in Section 3.2 which is crucial to the evolutionary improvement of the quality of the best solution. Fig. 3 shows a typical run of the discrete binary PSO to the Leaf curve for the swarm's best solution in terms of the number of the polygon vertices obtained by *gbest* as the number of evolutionary iterations increases. We observe that during the first 60 iterations, the quality of the swarm's best solution is improved dramatically. At this stage where the number of vertices delivered by the particle is still large and the candidate solution is very likely to pass the error bound constraint, the solution improvement is mainly due to the part of fitness evaluation that intends to minimize the number of vertices (see the second part of Eq. (6)). For the period between the 60th and the 150th iterations the quality of the best solution is further improved

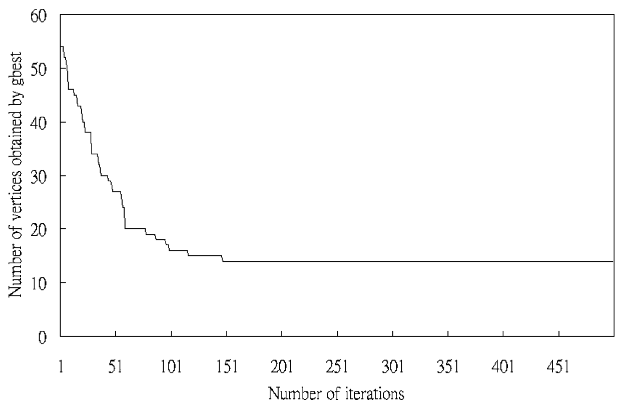

Fig. 3. The swarm's best solution in terms of the number of vertices obtained by *gbest* versus the number of iterations.

but at a slower speed, it is because when the number of vertices is getting small the candidate solution may incur an error that exceeds the given tolerance, the particle position needs to be readjusted to the feasible region again by costing some evolutionary iterations due to the error-reducing part of the fitness evaluation (see the first part of Eq. (6)) while the solution quality in terms of the number of vertices may not be improved under this purpose. However, this is an inevitable element to escape from a local optimum when the algorithm gets trapped after a significant improvement for the reduction of the number of vertices. After the 150th iteration the quality of the best solution is hardly improved, the swarm is about to converge and the particles deliver highly similar vectors and hardly exchange new information between one another.

The second issue is about the convergence behavior of the particles to testify whether the swarm evolves to the same optimization goal and the best solution with high quality is not obtained by chance due to a lucky particle. We propose the information *entropy* for measuring the similarity convergence among the particle vectors as follows. Let p_{ij} be the binary value of the jth bit for the ith particle, $i = 1, 2, \dots, K$. We can calculate prob_j as the conditional probability that value one happens at the jth bit given the total number of bits that take value one in the entire swarm as follows:

$$\text{prob}_j = \frac{\sum_{i=1}^{K} p_{ij}}{\sum_{i=1}^{K} \sum_{h=1}^{N} p_{ih}}. \tag{7}$$

The *particle vector entropy* can be then defined as

$$\text{Entropy} = -\sum_{j=1}^{N} \text{prob}_j \log_2 \left(\text{prob}_j\right). \tag{8}$$

The particle vector entropy analyzes the convergence of the swarm in two aspects. *First*, it assesses the convergence about the average fitness quality. Since the value of particle vector entropy decreases as the number of non-zero prob_j is lesser, and the latter expresses the number of distinct vertices currently considered by the whole swarm and if this number is small, the average quality of all swarm particles is higher. *Second*, the particle vector entropy also measures the convergence about the similarity among all particles. If the particle vectors are highly similar to one another, the values of those non-zero prob_j would be high, resulting in less entropy value. This also means the swarm particles are consistent in the decisions about which bits should take value one. Both of the two convergence assessments require decrements in the particle vector entropy as the number of evolutionary iterations increases.

Fig. 4A shows a typical run of the Leaf curve for the particle vector entropy as the number of evolutionary iterations increases. It is observed that the entropy value drops drastically during the first 150 iterations since the particles exchange information by referring to the swarm's best solution. After this period, the entropy value is relatively fixed due to the good quality solutions found and the high similarity among the particle vectors. It is interesting to note that the particle vector entropy does not exhibit two clear phases of reduction as the above-mentioned swarm's best solution does (see

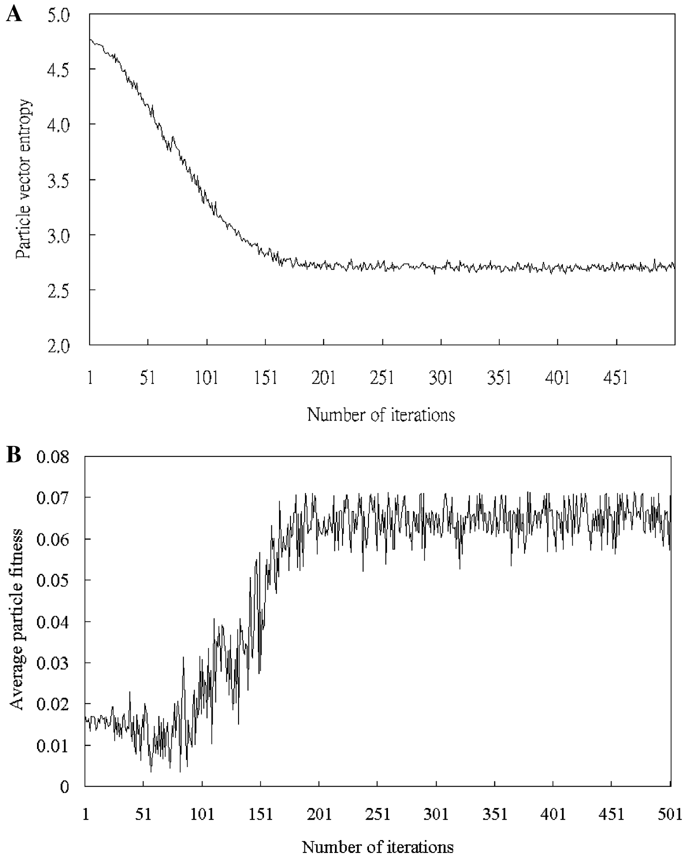

Fig. 4. Convergence of the algorithm. (A) The particle vector entropy versus the number of iterations. (B) The average particle fitness versus the number of iterations.

Fig. 3) due to the two parts of fitness evaluation. The reason is, although the solution quality in terms of the number of vertices is not always improved during the error reduction phase, the particles exchange information and resort to those bits that cause less approximation error and still gain a great reduction on the particle vector entropy. It further validates our previous observation that the error reduction is an essential part of our algorithm. Fig. 4B shows the average particle fitness as the number of evolutionary iterations increases. We observe that the average solution quality in terms of the fitness value is significantly improved with the experienced iterations, meaning that the particles are resorting to high quality solutions as the swarm converges.

4.2. Comparison with existing algorithms

A great number of algorithms have been developed for solving the polygonal approximation problems, it is laborious to compare our algorithm to each of them but

Table 1
The comparative performances on three benchmark curves using the GA-based, Salman's PSO, DPSO, HDPSO1, and HDPSO2 methods

Curves	ε	GA		Salman's PSO		DPSO		HDPSO1		HDPSO2	
		$M(\sigma_M)$	Merit	$M(\sigma_M)$	Merit	$M(\sigma_M)$	Merit	$M(\sigma_M)$	Merit	$M(\sigma_M)$	Merit
Leaf	150	15.6 (0.6)	34.5	11.6 (0.8)	66.2	11.9 (0.8)	63.8	10.0 (0.0)	79.0	10.0 (0.0)	79.0
($N = 120$)	100	16.3 (0.5)	42.5	14.4 (0.7)	56.1	14.6 (1.0)	55.0	12.5 (0.5)	78.2	12.0 (0.0)	87.6
	90	17.3 (0.5)	38.5	16.1 (0.8)	47.9	15.2 (1.1)	55.6	13.0 (0.30)	74.5	12.5 (0.5)	85.1
	30	20.5 (0.6)	57.2	26.6 (2.0)	39.0	20.1 (1.4)	59.1	16.7 (0.5)	86.9	16.0 (0.0)	98.9
	15	23.8 (0.6)	68.5	31.7 (2.5)	44.5	24.5 (0.9)	65.6	20.0 (0.0)	92.1	20.0 (0.0)	92.1
Chromosome	30	7.3 (0.4)	70.0	6.6 (0.5)	81.6	6.7 (0.6)	79.9	6.0 (0.0)	92.1	6.0 (0.0)	92.1
($N = 60$)	20	9.0 (0.6)	68.5	8.6 (0.5)	72.1	8.6 (0.7)	72.1	7.0 (0.0)	99.7	7.0 (0.0)	99.7
	10	10.2 (0.4)	85.7	11.5 (0.7)	73.0	11.6 (1.2)	72.0	10.0 (0.0)	87.6	10.0 (0.0)	87.6
	8	12.2 (0.5)	76.6	12.0 (0.9)	78.1	12.4 (0.9)	75.2	11.0 (0.0)	89.9	11.0 (0.0)	89.9
	6	15.2 (0.6)	72.9	14.6 (1.1)	76.5	14.4 (1.0)	77.7	12.7 (0.5)	91.5	12.0 (0.0)	97.9
Semicircle	60	13.2 (0.4)	48.0	11.6 (1.0)	61.1	12.0 (0.6)	57.4	10.0 (0.0)	76.9	10.1 (0.3)	75.8
($N = 102$)	30	13.9 (0.7)	69.5	14.9 (0.8)	64.0	14.8 (0.6)	64.5	12.4 (0.5)	85.4	12.0 (0.0)	90.6
	25	16.8 (0.7)	61.3	20.1 (1.6)	47.4	16.4 (1.2)	63.0	13.3 (0.5)	85.0	13.0 (0.0)	88.2
	20	19.2 (0.6)	59.5	22.8 (1.1)	44.0	18.4 (1.6)	63.2	14.8 (0.4)	85.1	14.2 (0.4)	89.2
	15	23.0 (0.9)	54.0	26.0 (1.9)	40.0	20.0 (1.2)	69.4	15.8 (0.6)	96.0	15.2 (0.4)	100
Average			60.5		59.4		66.2		86.7		90.2

to compare our method to some typical samples. As we previously mentioned, the quality of the solution obtained using natural algorithms is better than those using local heuristic methods, we therefore focus our comparison with natural algorithms. Also, the performances of various PSO versions are analyzed.

We have implemented the GA-based approach (Yin, 1999) for detailed comparisons with our algorithms since they both belong to the class of natural algorithms. The relationships between GA and PSO can be found in Eberhart and Shi (1998b). The GA-based approach (Yin, 1999) uses the same solution representation for a chromosome as in Eq. (5) and applies a specific crossover operator to the polygonal approximation. The adopted fitness function is defined as $L - \sum_{j=1}^{N} p_{ij} - E_i$ where L is a constant, p_{ij} is the jth bit of chromosome i, and E_i is equal to the approximation error if it exceeds the error tolerance, and $E_i = 0$ otherwise. The GA method also embeds a learning process within its evolutionary generations where the best chromosome is improved by replacing its substrings with those from other chromosomes. To better control the search, adaptive crossover and mutation probabilities are employed. The Salman's PSO (Salman et al., 2002) is also implemented for comparison since it can deal with discrete combinatorial optimization problems by dropping the sign and fraction parts of the real-valued particle representation. The Salman's PSO is coded with the same fitness function as we proposed in Eq. (6) for a fair comparison with our algorithms.

Table 1 shows the comparative performances of the GA, Salman's PSO, the discrete PSO proposed by Kennedy and Eberhart (referred to as DPSO), and the discrete PSO hybridized with the segment-merging optimizer (referred to as HDPSO1) and the segment-adjusting-and-merging optimizer (referred to as HDPSO2). To provide a fair comparison, we allocate to various algorithms approximately the same computational resource in terms of the number of evaluations for their respective fitness function. In particular, GA is executed for 100 generations with a population size of 100 chromosomes, and the average number of fitness evaluations is 11,970 (the number of evaluations varies with different runs due to the embedded learning process). The Salman's PSO and the DPSO are executed for 500 iterations, both of their numbers of evaluations are 10,020. As for HDPSO1 and HDPSO2, they are executed for 125 and 62 iterations, respectively, and their average numbers of fitness evaluations are 10,128 and 9941. The numerical results presented in Table 1 are estimated by averaging over 10 independent runs of each algorithm on the three benchmark curves. Various levels of approximation error tolerances (ε) are specified and the competing algorithms are conducted to minimize the number of polygon vertices (M), the standard deviation (σ_M) of M over those independent runs is also reported.

We observe that, for the same error tolerance, the GA-based approach produces approximating polygon with relatively larger number of vertices than the other approaches. However, the performance of Salman's PSO deteriorates as the specified error tolerance decreases since the conversion from real-valued representations to binary vectors by dropping the sign and fractions is not an efficient way for searching in the discrete binary space especially when the error constraint is more strict. The performance of Salman's PSO may be improved if a multi-level representation scheme can be adopted as it exhibited for the task assignment problem. On the other hand, the DPSO is comparable to Salman's PSO in those cases with larger values of error tolerance, and the former becomes superior as the error tolerance decreases, manifesting the DPSO is more efficient in the binary search space.

Table 2
The cross-reference *t* test on the merits of various algorithms

	Salman's PSO	DPSO	HDPSO1	HDPSO2
GA	−0.18	1.43	6.02*	5.87*
Salman's PSO		1.58	4.65*	4.67*
DPSO			9.95*	7.98*
HDPSO1				2.13*

*Statistically significant.

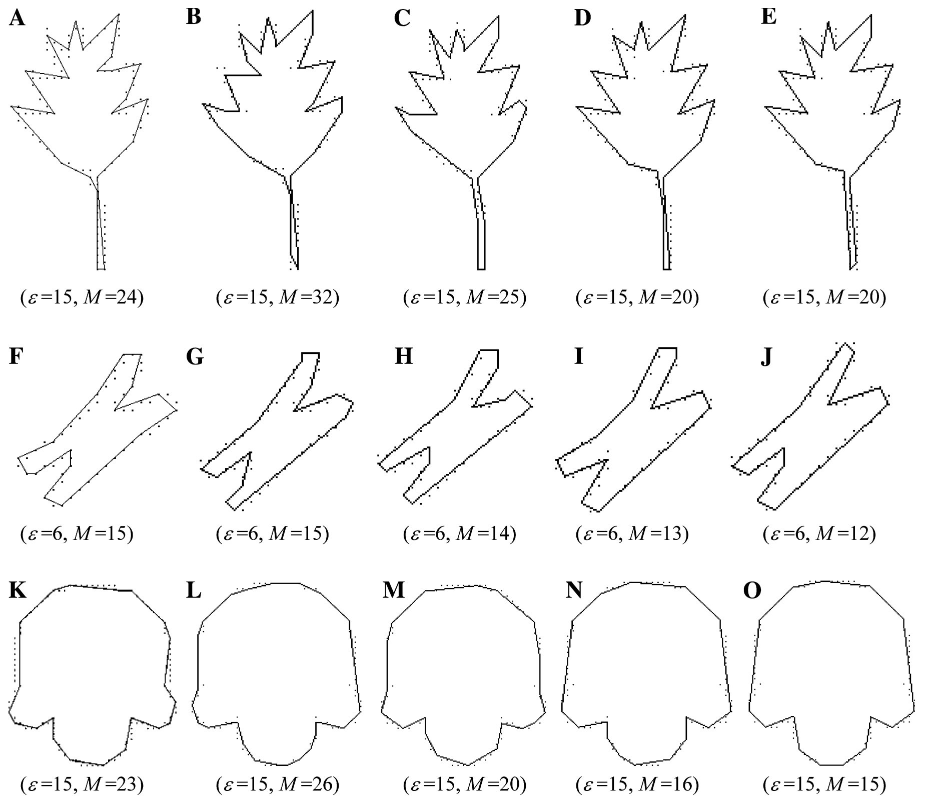

Fig. 5. The approximating polygon with its approximation error (ε) and the number of vertices (M) obtained by the GA-based, Salman's PSO, DPSO, HDPSO1, and HDPSO2 methods. (A) GA-based, (B) Salman's PSO, (C) DPSO (D) HDPSO1, (E) HDPSO2, (F) GA-based, (G) Salman's PSO, (H) DPSO, (I) HDPSO1, (J) HDPSO2, (K) GA-based, (L) Salman's PSO, (M) DPSO, (N) HDPSO1, and (O) HDPSO2.

As for the two hybrid PSO versions, namely the HDPSO1 and the HDPSO2, they have the best average performance among all competing methods in all of the testing cases, the improvement in terms of the number of polygon vertices is significant. The performance of HDPSO2 is better than that of HDPSO1 due to that the proposed adjusting-and-merging optimizer employed by the former approach is more sophisticated than the segment-merging optimizer (Leu and Chen, 1988) adopted by the latter. The superiority of both Hybrid PSO versions over the other methods is because when the particle is moved to a new region it is navigated to a local optimum of that region by the embedded local optimizer, so the information exchanged among particles is the local optimal solutions instead of the intermediate solutions hanging around the hill. This feature guarantees that the particle exploits the region well before it explores another region. The standard deviations of M yielded by the Hybrid PSO versions are also lesser, manifesting that the embedded local optimizers can help the PSO provide more consistent results in different runs.

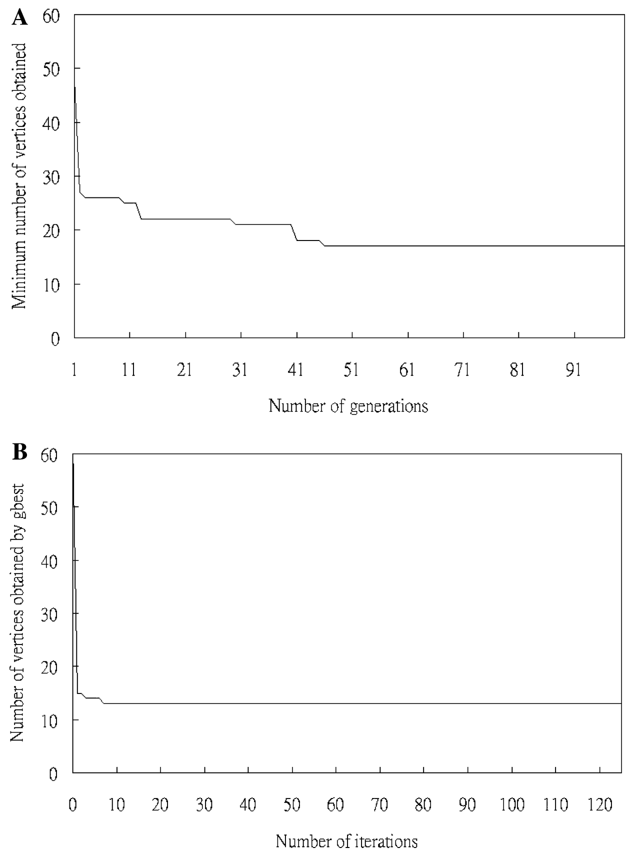

Fig. 6. The best solution in terms of number of vertices versus the number of iterations using: (A) GA and (B) HDPSO2.

Since both of ε and M are impact factors, fixing either one and using the other alone to justify the relative performance is not completely fair. To provide a unified performance measure, Rosin (1997) defined two terms as follows:

$$\text{Fidelity} = \frac{E_{\text{opt}}}{E_{\text{approx}}} \times 100 \tag{9}$$

$$\text{Efficiency} = \frac{M_{\text{opt}}}{M_{\text{approx}}} \times 100 \tag{10}$$

where E_{approx} is the incurred error by the tested algorithm and E_{opt} is the approximation error incurred by the optimal algorithm that produces the same number of vertices as the tested algorithm, M_{approx} is the number of vertices produced by the tested algorithm and M_{opt} is the number of vertices yielded by the optimal algorithm that produces the same error as the tested algorithm. Since an exact value of M_{opt} is not generally available it is estimated by linear interpolation of the two closest values of M. Fidelity measures how well the approximating polygon fits the curve in terms of the approximation error relative to the optimal polygon with the same number of vertices, while Efficiency measures how compact the approximating polygon is, relative to the optimal polygon which incurs the same error. Then Rosin proposes a unified figure of merit as

$$\text{Merit} = \sqrt{\text{Fidelity} \times \text{Efficiency}}. \tag{11}$$

We have implemented the dynamic programming approach (Sato, 1992) to derive the optimal polygons and incurred errors by specifying various numbers of vertices for the benchmark curves. The merits of the approximating polygons produced by various algorithms are also presented in Table 1. It is seen that the comparative

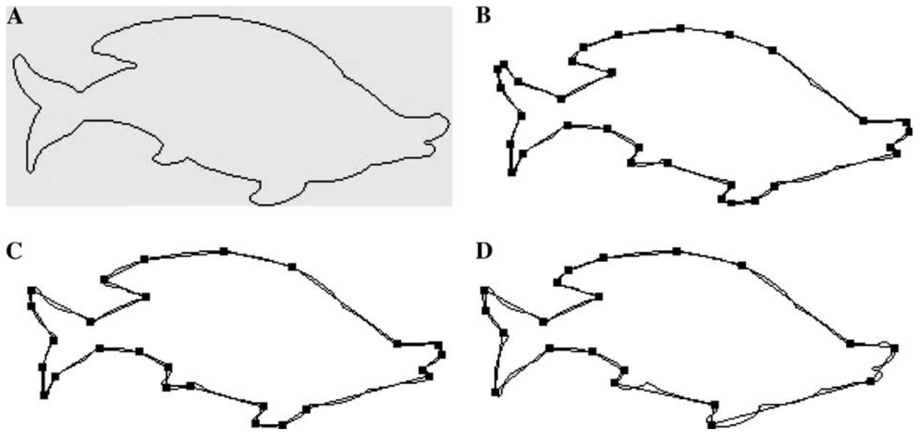

Fig. 7. The fish contour image and the approximating polygons with approximation error (ε) and the number of vertices (M) obtained by the proposed HDPSO2 algorithm. (A) Fish contour image; (B) $\varepsilon = 500$, $M = 31$; (C) $\varepsilon = 1000$, $M = 26$; (D) $\varepsilon = 2000$, $M = 20$.

performances in terms of merits conform to our previous analysis. According to the average merits shown at the bottom of Table 1, GA and Salman's PSO are comparable, DPSO is better than the former two algorithms, and both of HDPSO1 and HDPSO2 have significantly higher values of merits.

To further realize the relative performance of various algorithms, we also conduct the *t* test for the values of merits obtained by various algorithms in those samples. Table 2 shows the cross-reference *t* test on the merits of all algorithms for the 15 testing samples. Since the confidence coefficient is 1.76 for the 95% confidence interval of this case, we observe that the performance of Salman's PSO is slightly worse than that of GA, DPSO is better than Salman's PSO and GA but not significant, both of the HDPSO1 and HDPSO2 significantly outperform the other methods, and the superiority of HDPSO2 over HDPSO1 is also statistically significant.

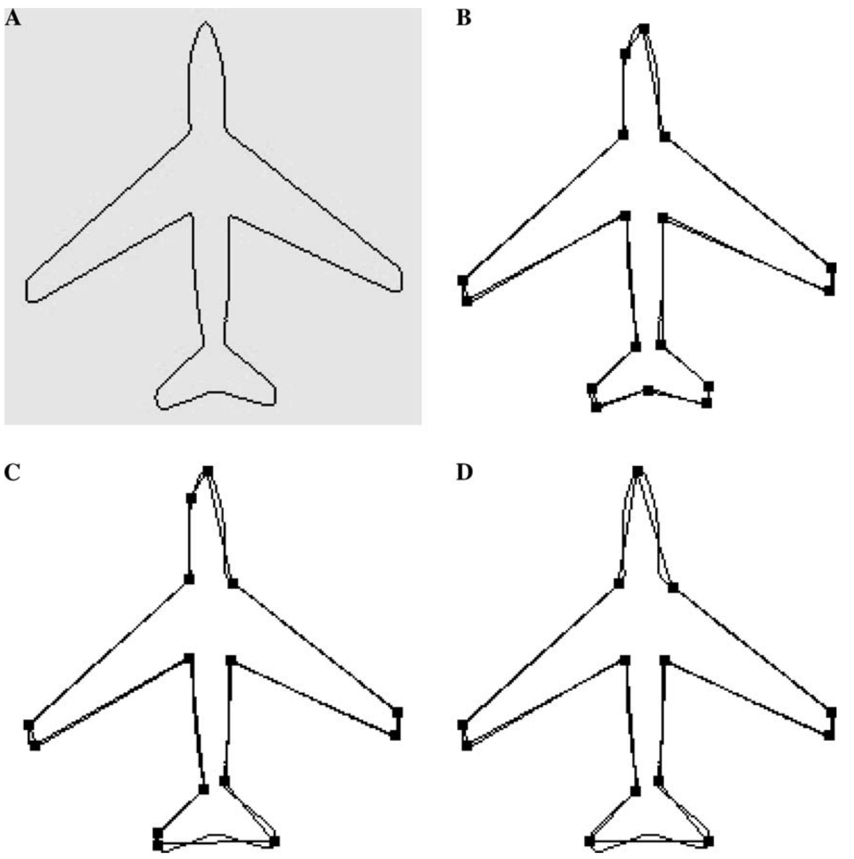

Fig. 8. The plane contour image and the approximating polygons with approximation error (ε) and the number of vertices (M) obtained by the proposed HDPSO2 algorithm. (A) Plane contour image; (B) $\varepsilon = 500$, $M = 17$; (C) $\varepsilon = 1000$, $M = 15$; (D) $\varepsilon = 2000$, $M = 13$.

Some experimental samples regarding to the obtained approximating polygon with its approximation error (ε) and the number of vertices (M) are shown in Fig. 5 for visual comparison of the approximation. It is seen that the DPSO produces fewer number of vertices than Salman's PSO for the same error tolerance. Also, the hybrid versions of DPSO significantly outperform the other methods in terms of M and they still preserves the main corner information of the curves by the same error bound.

Figs. 6A and B show the best solution on the leaf curve in terms of number of vertices versus the number of iterations using GA and HDPSO2, respectively. GA converges at the 46th generation with 100 chromosomes experiencing about 5000 times of fitness evaluations. On the other hand, the proposed HDPSO2 converges more quickly at the 7th iteration with 20 particles only costing about 1120 times of fitness evaluations. This is because the segment-adjusting-and-merging optimizer embedded in the HDPSO2 brings the particles to nearby local optima and provides more informative building blocks to be referred to, therefore accelerating the convergence speed.

To demonstrate the scalability of the proposed HDPSO2 method to real-world applications, two real images containing symbols of a fish and a plane are further experimented with. The two images are binarized and the edge points are extracted by detecting the black–white transitions (see Fig. 7A and Fig. 8A). The first image contains 700 edge points and the second image has 682 edge points, both numbers are significantly larger than those in the benchmark curves. By specifying various values of error tolerance, the proposed HDPSO2 yields approximating polygons of different degrees (see Figs. 7 and 8). The larger the specified error tolerance is, the lesser the number of vertices of the obtained polygon. It is seen that the profile information of the images is well captured by the approximating polygons, and thus facilitating the speed-up for the subsequent image analysis tasks without sacrificing the accuracy.

5. Summary

In this paper, we have proposed a new polygonal approximation method based on the particle swarm optimization (PSO) algorithm. The PSO belonging to the class of natural algorithms has exhibited successful applications for real-valued functional optimization. However, the number of researches for combinatorial optimization using the discrete version of PSO is limited. We have demonstrated the precise steps of using a discrete PSO, namely the particle representation, fitness evaluation, and particle position adjustment, for solving the polygonal approximation problem which is one of the instances in combinatorial optimization. The experimental results manifest that the devised discrete PSO is comparable to the genetic algorithm which is another example of natural algorithms, and it also outperforms another discrete implementation of PSO in the literature. Further, we have proposed to use a hybrid version of PSO embedding a local optimizer within its evolutionary iterations and empirically shown that the hybrid

PSO can enhance the performance significantly in terms of the number of polygon vertices required to produce the same error and the variations of results between different runs.

Acknowledgments

The author thanks the reviewers whose suggestions have greatly improved the readability of the paper. This research is partially supported by National Science Council of ROC, under Grant NSC-90-2213-E-130-006.

References

Ansari, N., Delp, E.J., 1991. On detection dominant points. Pattern Recognition 24, 441–450.

Clerc, M., Kennedy, J., 2002. The particle swarm explosion, stability, and convergence in a multidimensional complex space. IEEE Transaction on Evolutionary Computation 6, 58–73.

Dunham, J.G., 1986. Optimum uniform piecewise linear approximation of planar curves. IEEE Transaction on Pattern Analysis and Machine Intelligence 8, 67–75.

Eberhart, R.C., Shi, Y., 1998. Evolving artificial neural networks. In: Proceedings of the International Conference on Neural Networks and Brain, pp. PL5–PL13.

Eberhart, C.R., Shi, Y., 1998. Comparison between genetic algorithms and particle swarm optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (Eds.), Evolutionary Programming VII: Proceedings of the 7th Annual Conference on Evolutionary Programming.

Goldberg, D.E., 1989. Genetic Algorithms in Search. Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Held, A., Abe, K., Arcelli, C., 1994. Towards a hierarchical contour description via dominant point detection. IEEE Transaction on System, Man, and Cybernetics 24, 942–949.

Ho, Y.S., Chen, Y.C., 2001. An efficient evolutionary algorithm for accurate polygonal approximation. Pattern Recognition 34, 2305–2317.

Huang, C.S., Sun, Y.N., 1999. Polygonal approximation using genetic algorithms. Pattern Recognition 32, 1409–1420.

Kennedy, J., Eberhart, R.C, 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948.

Kennedy, J., Eberhart, R.C, A discrete binary version of the particle swarm algorithm. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1997, pp. 4104–4 108.

Kurozumi, Y., Davis, W.A., 1982. Polygonal approximation by the minimax method. Computer Graphics and Image Processing 19, 248–264.

Leu, J.G., Chen, L., 1988. Polygonal approximation of 2-D shapes through boundary merging. Pattern Recognition Letters 7, 231–238.

Parsopoulos, E.K., Vrahatis, M.N., 2002. Recent approaches to global optimization problems through particle swarm optimization. Natural Computing 1, 235–306.

Ramer, U., 1972. An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing 1, 244–256.

Ray, B.K., Ray, K.S., 1993. Determination of optimal polygon from digital curve using L_1 norm. Pattern Recognition 26, 505–509.

Ray, K.B., Ray, K.S., 1995. A new split-and-merge technique for polygonal approximation of chain coded curves. Pattern Recognition Letters 16, 161–169.

Rosin, P.L., 1997. Technique for accessing polygonal approximations of curves. IEEE Transaction on Pattern Analysis and Machine Intelligence 19, 659–666.

Salman, A., Ahmad, I., Al-Madani, S., 2002. Particle swarm optimization for task assignment problem. Microprocessors and Microsystems 26, 363–371.

Sato, Y., 1992. Piecewise linear approximation of plane curves by perimeter optimization. Pattern Recognition 25, 1535–1543.

Shigenori, N., Takamu, G., Toshiku, Y., Yoshikazu, F., 2003. A hybrid particle swarm optimization for distribution state estimation. IEEE Transaction on Power Systems 18, 60–68.

Sklansky, J., Gonzalez, V., 1980. Fast polygonal approximation of digitized curves. Pattern Recognition 12, 327–331.

Tandon, V., 2000. Closing the gap between CAD/CAM and optimized CNC end milling. Master thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis.

Teh, H.C., Chin, R.T., 1989. On the detection of dominant points on digital curves. IEEE Transaction on Pattern Analysis and Machine Intelligence 11, 859–872.

Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325.

Wall, K., Danielsson, P.E., 1984. A fast sequential method for polygonal approximation of digitized curves. Computer Vision, Graphics, and Image Processing 28, 220–227.

Wu, Y.W., Wang, M.J., 1993. Detecting the dominant points by the curvature-based polygonal approximation. CVGIP: Graphical Models and Image Processing 55, 79–88.

Yin, P.Y., 1999. Genetic algorithms for polygonal approximation of digital curves. International Journal of Pattern Recognition and Artificial Intelligence 13, 1–22.

Yin, P.Y., 2003. Ant colony search algorithms for optimal polygonal approximation of plane curves. Pattern Recognition 36, 1783–1797.

Yoshida, H., Kawata, K., Fukuyama, Y., Nakanishi, Y., 1999. A particle swarm optimization for reactive power and voltage control considering voltage stability. In: Proceedings of the International Conference on Intelligent System Application to Power Systems, pp. 117–121.

Zhu, P., Chirlian, P.M., 1995. On critical point detection of digital shapes. IEEE Transaction on Pattern Analysis and Machine Intelligence 17, 737–748.