Available online at www.sciencedirect.com
ScienceDirect
Information Processing Letters 103 (2007) 169–176
Information Processing Letters
www.elsevier.com/locate/ipl

Particle swarm optimization-based algorithms for TSP and generalized TSP

X.H. Shi [a], Y.C. Liang [a,b,*], H.P. Lee [b,c], C. Lu [b], Q.X. Wang [a]

[a] *College of Computer Science and Technology, Jilin University, Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China*

[b] *Institute of High Performance Computing, Singapore 117528, Singapore*

[c] *Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 119260, Singapore*

Received 31 July 2005; received in revised form 10 February 2007

Available online 31 March 2007

Communicated by Wen-Lian Hsu

Abstract

A novel particle swarm optimization (PSO)-based algorithm for the traveling salesman problem (TSP) is presented. An uncertain searching strategy and a crossover eliminated technique are used to accelerate the convergence speed. Compared with the existing algorithms for solving TSP using swarm intelligence, it has been shown that the size of the solved problems could be increased by using the proposed algorithm.

Another PSO-based algorithm is proposed and applied to solve the generalized traveling salesman problem by employing the generalized chromosome. Two local search techniques are used to speed up the convergence. Numerical results show the effectiveness of the proposed algorithms.

Keywords: Algorithms; Particle swarm optimization; Traveling salesman problem; Generalized traveling salesman problem; Swap operator

1. Introduction

The Particle swarm optimization (PSO) algorithm, originally developed by Kennedy and Eberhart [1], is a method for optimization on metaphor of social behavior of flocks of birds and/or schools of fish. Similar to genetic algorithms (GAs), the PSO is also an optimizer based on population. The system is initialized firstly in a set of randomly generated potential solutions, and then is performed to search for the optimum one iteratively. It finds the optimum solution by swarms following the best particle. Compared to GAs, the PSO has much better intelligent background and could be performed more easily. According to its advantages, the PSO is not only suitable for scientific research, but also engineering applications. Presently the PSO has attracted broad attention in the fields of evolutionary computing, optimization and many others [2–5]. Although the PSO is developed for continuous optimization problems initially, there have been some reported works focused on discrete problems recently [6,7].

The traveling salesman problem (TSP) is a well-known and extensively studied benchmark for many new developments in combinatorial optimization [8,9],

* Corresponding author.
E-mail address: ycliang@jlu.edu.cn (Y.C. Liang).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2007.03.010

including techniques in evolutionary computation, such as nearest neighborhood search (NNS), simulated annealing (SA), tabu search (TS), neural networks (NN), ant colony system (ACS), GAs and some others [10–14]. Furthermore, Clerc [15], Hendtlass [16] and Wang [17] proposed different PSO methods for solving TSP problems. Although the PSO method could be applied to the TSP, the size of solved problems reported in the references are all smaller than 17 cites. It seems that the size of the solved TSP problems using algorithms of PSO is limited.

A very simple and practical extension of the TSP is the generalized TSP (GTSP) in which the set of nodes is divided into clusters and the objective is to find a minimum-cost tour passing through one node from each cluster. The GTSP represents a kind of combinatorial optimization problem, which has been introduced by Henry-Labordere [18], Saskena [19] and Srivastava [20] in the context of computer record balancing and of visit sequencing through welfare agencies since 1960s. It has many extensive application fields. Just as mentioned in [21], "for many real-world problems that are inherently hierarchical, the GTSP offers a more accurate model than the TSP". Generally, the GTSP provides a more ideal modeling tool for many real problems. Furthermore, the GTSP can include the grouped and isolated vertices at the same time according to our present extension. Therefore, the GTSP includes the TSP theoretically and the application fields of the GTSP are wider than those of the TSP. Although since late 1960s the GTSP has been proposed [18–20], the related literatures are very limited compared with those on the TSP [8–14] and the existing algorithms for the GTSP are mainly based on dynamic programming techniques [18–20,22]. The main methodology of the dynamic programming algorithms is to transform the GTSP into the TSP and then to solve the TSP using existing algorithms. The shortcoming of these methods is that the transformation increases the problem dimension dramatically. Therefore, although theoretically the GTSP could be solved using the corresponding transformed TSP, the technological limitation ruins its practical feasibility. By designing a generalized chromosome (GC), Wu et al. [23] have developed a generalized chromosome-based genetic algorithm. The GC could unify the GTSP and TSP problems into one uniform mode by introducing a "super vertex". Therefore, in general, this algorithm does not increase the size of the solved problem.

In this paper, the "subtraction" operator between two particle positions is modified and a discrete PSO method is constructed for the TSP. The crossover eliminated technique is also executed in the proposed method. Numerical results show that the proposed method could improve the size of resolvable TSP problems. Based on the coding technique of the GTSP in [23], another PSO-based algorithm is proposed in this paper for solving the GTSP. Two local search techniques are also included in the proposed method. To test the effectiveness of the method, 19 GTSP problems are examined for benchmarking. The results show that the method is effective for solving the GTSP problem. To the best of our knowledge, there has been no attempt in proposing a PSO-based algorithm for the GTSP so far. The proposed PSO-based algorithm could provide a suitable approach for solving the GTSP.

2. Particle swarm optimization (PSO) algorithm

First, the standard PSO algorithm is introduced briefly. Suppose that the searching space is D-dimensional and m particles form the colony. The ith particle is represented by a D-dimensional X_i $(i = 1, 2, \ldots, m)$ vector which means that the particle locates at $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ $(i = 1, 2, \ldots, m)$ in the search space. The position of each particle is a potential solution. We could calculate the particle's fitness by putting its position into a designated objective function. When the fitness is higher, the corresponding X_i is "better". The ith particle's "flying" velocity is also a D-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ $(i = 1, 2, \ldots, m)$. Denote the best position of the ith particle as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the best position of the colony as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$, respectively. The PSO algorithm could be performed by the following equations:

$$X_i(k+1) = X_i(k) + V_i(k+1), \tag{1}$$

$$V_i(k+1) = wV_i(k) + c_1r_1\big(P_i - x_i(k)\big) + c_2r_2\big(P_g - x_i(k)\big), \tag{2}$$

where $i = 1, 2, \ldots, m$; w is the inertia coefficient which is a constant in the interval $[0, 1]$; c_1 and c_2 are learning rates which are nonnegative constants; r_1 and r_2 are generated randomly in the interval $[0, 1]$. The termination criterion for the iterations is determined according to whether the maximum generation or a designated value of the fitness of P_g is reached.

The algorithm described above can be considered as the conventional particle swarm optimization, which is applicable for continuous problems. However, it cannot be applied to discrete problems directly. Aiming at the discrete problems, Kennedy and Eberhart proposed a discrete binary version of PSO algorithm by defining the particles' trajectories and velocities in terms of changes

of probabilities that a bit will be in one state or the other [5]. Thus a particle moves in a state space restricted to 0 and 1 in each generation, where v_{id} represents the probability of bit x_{id} taking 1. Eq. (2) of the particle swarm formula remains unchanged formally, except that p_{id} and x_{id} are integers in $\{0, 1\}$. Eq. (1) is written as follows:

$$x_{id} = \begin{cases} 1 & \text{if } rand() < S(x_{id}), \\ 0 & \text{otherwise} \end{cases} \quad (d = 1, 2, \ldots, D), \tag{3}$$

where $S(\cdot)$ is a logistic transformation, which can constrain v_{id} to the interval $[0, 1]$, and $S(v_{id})$ can be considered as a probability.

3. Discrete PSO algorithm for TSP

3.1. Description of the discrete PSO algorithm for TSP

Clerc proposed a brief outline of the PSO method for solving TSP problems [15]. By adding a memory capacity to each particle in the PSO algorithm, Hendtlass [16] applied the PSO algorithm to solve small-size TSP problems, and improved its performance. Wang et al. [17] redefined the PSO operators by introducing the concepts of "Swap operator" and "Swap sequence", therefore the TSP problems could be solved by the PSO in another way. The sizes of cities in [16] and [17] are both 14 (both of them selected Burma14, a benchmark problem in TSPLIB with 14 cities), and that of [15] is 17 (it selected br17, a benchmark problem in TSPLIB with 17 cities). That is to say, the sizes of cities are rather limited in their algorithms.

To expand the scale of the solved problems, inspired by the "Swap operator" developed in [17], we introduce the permutation concept into our proposed discrete PSO algorithm for the TSP. Also an uncertainty searching strategy is added in the algorithm to speed up the convergence speed. For an m-ordered TSP problem, denote $X_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ as the ith particle's position in the population of PSO, which represents the traveling circle of $x_{i1} \to x_{i2} \to \cdots \to x_{im} \to x_{i1}$. The conventional PSO algorithm is defined by Eqs. (1)–(2). For TSP problem, Eq. (1) could be remained formally, while the meaning of the second item on the right hand of the equation, namely $V_i(k + 1)$, is changed. Therefore Eq. (2) should be redefined. First of all, the subtraction of two particle positions should be redefined. To express this problem, some concepts about permutation will be introduced.

Definition 1. A permutation is a bijection from a finite set X onto itself.

Permutations could be illustrated in relation notation. One can just arrange the "natural" ordering of the elements being permuted on a row, and the new ordering on another row:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \end{bmatrix}$$

stands for the permutation s of the set $\{1, 2, 3, 4, 5\}$ defined by $s(1) = 2$, $s(2) = 5$, $s(3) = 4$, $s(4) = 3$, $s(5) = 1$.

Definition 2. Let X be a set. A cycle is a permutation such that there exist distinct elements $a_1, a_2, \ldots, a_k$ of X such that

$$s(a_i) = \begin{cases} a_{i+1} & i = 1, \ldots, k-1, \\ a_1 & i = k \end{cases} \tag{4}$$

and

$$s(x) = x \quad \forall x \in X, x \notin \{a_i\} \quad (i = 1, 2, \ldots, k).$$

Definition 3. The order of a cycle is the number of elements of its nontrivial orbit. A cycle of order k is also called a k-cycle. A 1-cycle is an identity permutation. Therefore any permutation could be decomposed in *a product of disjoint cycles*. (Including 1-cycle.)

Definition 4. Given a finite set $X = \{a_1, a_2, \ldots, a_n\}$, a transposition is a permutation f, such that there exist indices i, j such that $f(a_i) = a_j$, $f(a_j) = a_i$ and $f(a_k) = a_k$ for all other indices k. A transposition is 2-cycle. Any cycle could be decomposed in a product of transpositions. Therefore any permutation could be decomposed in a product of transpositions.

Given two n ordered sequences $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, denote the following permutation as

$$P_{AB} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix}. \tag{5}$$

Set k to be the minimum number of transpositions of which P_{AB} could be decomposed in a product, namely $P_{AB} = T_1 \cdot T_2 \cdots T_k$. Then we define the subtraction of $B - A = T_1 \cdot T_2 \cdots T_k$. The position of a particle could be considered as an m-ordered sequence, so the subtraction of two particle positions could be defined as above. While we know the start point and the traveling direction do not make any differences for a traveling circle, therefore a slide operator and a reverse operator are introduced to express these ideas.

Definition 5. For an m-ordered sequence $X = \{x_1, x_2, \ldots, x_m\}$, the slide operator acts on X is that: $SL(X, k) = \{x_{(1+k)\%m}, x_{(2+k)\%m}, \ldots, x_{(m+k)\%m}\}$.

Definition 6. For an m-ordered sequence $X = \{x_1, x_2, \ldots, x_m\}$, the reverse operator acts on X is that: $RE(X) = \{x_m, x_{m-1}, \ldots, x_1\}$.

For a particle X, we define $SL(X,k) = X$ and $RE(X) = X$, respectively.

Definition 7. Given two particle positions $X_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ and $X_j = \{x_{j1}, x_{j2}, \ldots, x_{jm}\}$, define their subtraction as:

$$\begin{aligned}&\text{sub\_par}(X_i, X_j) = X_j - X_i\\ &\quad= \min_k \big\{T_1T_2\cdots T_k \mid \\ &\qquad \exists l, st\, SL\big(P(X_i, X_j), l\big) = T_1T_2\cdots T_k \text{ or}\\ &\qquad SL\big(RE\big(P(X_i, X_j)\big), l\big) = T_1T_2\cdots T_k\big\}. \qquad (6)\end{aligned}$$

Set r to be a real value and T is a transposition, define

$$rT = \begin{cases} T & r \geqslant 0.5,\\ P_I & r < 0.5,\end{cases} \qquad (7)$$

here P_I is an identity permutation.

Redefine r_1 and r_2 in Eq. (2) as a real vector, whose dimensions are corresponding to the numbers of transpositions dotted with them. Therefore Eq. (2) could be rewritten as follows:

$$\begin{aligned}&V_i(k+1)\\ &\quad= wV_i(k) + c_1R_1^k\big(P_i^k - x_i(k)\big) + c_2R_2^k\big(P_g^k - x_i(k)\big)\\ &\quad= c_1R_1^k\big(P_i^k - x_i(k)\big) + c_2R_2^k\big(P_g^k - x_i(k)\big)\\ &\qquad + w\big(wV_1(k-1) + c_1R_1^{k-1}\big(P_i^{k-1} - x_i(k-1)\big)\\ &\qquad + c_2R_2^{k-1}\big(P_g^{k-1} - x_i(k-1)\big)\big)\\ &\quad= c_1R_1^k\big(P_i^k - x_i(k)\big) + c_2R_2^k\big(P_g^k - x_i(k)\big)\\ &\qquad + w\big(c_1R_1^{k-1}\big(P_i^{k-1} - x_i(k-1)\big)\\ &\qquad + c_2R_2^{k-1}\big(P_g^{k-1} - x_i(k-1)\big)\big)\\ &\qquad + w^2\big(wV_1(k-2) + c_1R_1^{k-2}\big(P_i - x_i(k-2)\big)\\ &\qquad + c_2R_2^{k-2}\big(P_g - x_i(k-2)\big)\big)\\ &\quad= \cdots.\end{aligned}$$

For the inertia coefficient w is a constant less than 1, the items of the high ordered about w might be ignored. In this paper, the items of w-ordered more than 2 are all omitted. Therefore we have

$$\begin{aligned}&V_i(k+1)\\ &\quad= c_1R_1^k\big(P_i^k - x_i(k)\big) + c_2R_2^k\big(P_g^k - x_i(k)\big)\\ &\qquad + w\big(c_1R_1^{k-1}\big(P_i^{k-1} - x_i(k-1)\big)\\ &\qquad + c_2R_2^{k-1}\big(P_g^{k-1} - x_i(k-1)\big)\big). \qquad (8)\end{aligned}$$

```
for (i = 1; i < m − 3; i++)
for (j = i + 2; j < m; j++)
  if (crossover(line[i], line[j]) is true)
    for (k = 0; k < (j − i)/2; k++)
      swap(x_{j−k}, x_{i+k+1}).
```

Fig. 1. The pseudo-code of the delete-crossover process.

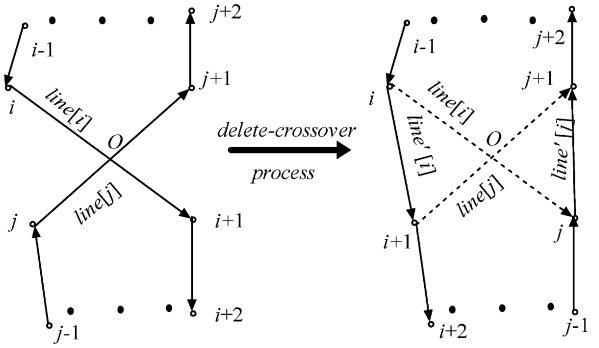

Fig. 2. Schematic diagram of the delete-crossover process.

According to Eqs. (1) and (8) the PSO algorithm for TSP could be proposed.

To speed up the convergence, a process is added to delete the crossover of traveling lines. For a particle position X, $line[i]$ refers the traveling line between the ith node and the $(i+1)$th node of X. Define a bool function $crossover(line[i], line[j])$ to judge whether $line[i]$ and $line[j]$ are crossed over and return *true* and *false* accordingly. Then the pseudo-code of the delete-crossover process is shown as Fig. 1 and the schematic diagram is shown by Fig. 2, respectively. From Fig. 2, it could be found that of the two traveling circles before and after the delete-crossover process, all the edges are unchanged, except that $line[i]$ and $line[j]$ are changed into $line'[i]$ and $line'[j]$. According to the theory that the sum of lengths of two sides in a triangle is larger than that of the third side, it is easy to draw that $line'[i] + line'[j] < line[i] + line[j]$. It means that the delete-crossover process could improve the performance of the particle. On the other hand, if the process is executed on all particles of the population, it should be much time-consuming. In the PSO program, all particles should learn from P_g, so that the improvement of P_g could affect all particles. Therefore the delete-crossover process only needs to be performed on P_g in each of iterations. Therefore the process is time-saving and little effectiveness lost at the same time.

In summary, the proposed discrete PSO algorithm for the TSP can be described as follows:

(1) Initialize the particle swarm: including the initializations of each particle's position and the evaluation of the fitness of each particle. At this stage, P_g is also searched and P_i of each particle is set as its initial position.
(2) Apply the PSO processes to each particle:
   (a) Execute the searching process for each particle according to Eqs. (1) and (8);
   (b) Execute the delete-crossover process for P_g;
(3) Judge the termination criterion, namely whether the iteration reaches the given number or the best fitness reaches the designated value. If the criterion is satisfied then stop the program, else go to Step 2.

3.2. Numerical results

To verify the validity of the proposed discrete PSO algorithm, some instances from the TSPLIB library [24] are selected for simulations. The experiments have been performed on a PC with 2 GHz processor and 128M memory. Each instance is run for 100 times.

Table 1 presents the numerical results. The second column stands for the best known optimal tour length for each problem (*Opt*), and the seventh for the relative error (*Err*), respectively, where the relative error is calculated as

$$Err = (Ave - Opt)/Opt \times 100\%. \tag{9}$$

Table 1 shows that the proposed discrete PSO method can be used to solve the TSP effectively. From Table 1 it can be seen that among the 5 test problems, the maximum relative error is 4.1673% and the average relative error is 3.5496%. It can be seen that our result is fairly good when the problem size is ranged from 50 to 80, whereas the results from the existing PSO-based algorithms [15–17] show that the sizes of the solved TSP problems were all smaller than 17 cites. It is obvious that the proposed algorithm has advantage on the solved problem size compared with the existing algorithms.

Table 1
Results of the proposed algorithm for TSP problems

Problem	*Opt*	Best result	Worst result	*Err* (%)
EIL51	426	427	452	2.5751
BERLIN52	7542	7542	8362	3.8458
ST70	675	675	742	3.3422
EIL76	538	546	579	4.1673
PR76	108159	108280	124365	3.8176

4. Discrete PSO method for GTSP

4.1. Statement of generalized TSP (GTSP)

The generalized traveling salesman problem (GTSP) has been introduced by Henry-Labordere, Saskena and Srivastava in the context of computer record balancing and of visit sequencing through welfare agencies since 1960s. The GTSP has many extensive application fields, such as the covering tour problem, material flow system design, post-box collection, stochastic vehicle routing and arc routing, and many others [21,25].

In the GTSP problem, the candidate cities are divided into several groups. Some cities may belong to more than one group, while some other cities may just be in a single one. The aim of the GTSP is to seek the traveling cycle with the smallest length. According to the style of the cycle, there exist two different kinds of GTSP:

(1) the cycle passes exactly one vertex in each group and
(2) the cycle passes at least one vertex in each group.

For concision, only the first case is discussed in this paper.

4.2. Algorithm of the discrete PSO method for GTSP

For convenience, the generalized chromosome (GC) developed by Wu et al. [22] is introduced here. Assume that there are n candidate cities that are divided into m groups (including the singled cities). Namely, for the arbitrary given cities c_i $(i = 1, 2, \ldots, n)$, there must exist at least one group V_j $(j = 1, 2, \ldots, m)$ satisfies that $c_i \in V_j$. Then the traveling cycle includes m vertices and m lines which link the m vertices in a circle.

A group V_j $(j = 1, 2, \ldots, m)$ is said to be a super vertex if the number of cities belonging to V_j is larger than 1 and the cities belonging to V_j are called its elements. The city c_i $(i = 1, 2, \ldots, n)$ is said to be a scattering vertex if $V_j = \{c_i\}$ $(j = 1, 2, \ldots, m)$ and the number of cities belonging to V_j is equal to 1. Either a super vertex or scattering vertex is called a generalized vertex if it is dealt with in a uniform way.

Then the vertex groups could be divided into 2 classes according to their type, namely, the super vertex or the scattering one. Denote the number of the super vertices as $\widehat{m}$, that of the scattering vertices as $\widetilde{m}$, all the super vertices as $\{u_1, u_2, \ldots, u_{\widehat{m}}\}$, all the scattering ones as $\{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{\widetilde{m}}\}$, respectively. Then all the generalized vertices could be denoted as

$$\{w_1, w_2, \ldots, w_{\widehat{m}+\widetilde{m}}\},$$

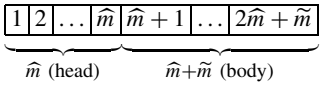

Fig. 3. Mode of the generalized chromosome.

where

$$w_k = \begin{cases} u_k & 1 \leqslant k \leqslant \widehat{m}, \\ \tilde{u}_{k-\widehat{m}} & \widehat{m}+1 \leqslant k \leqslant \widehat{m}+\widetilde{m}. \end{cases} \tag{10}$$

The GC designed by Wu et at. [22] is shown in Fig. 2. A GC consists of two parts. The left part contains $\widehat{m}$ genes and is named as a head; the right part contains $\widehat{m}+\widetilde{m}$ genes and is named as a body. When $0 < i \leqslant \widehat{m}$, the ith gene lies in the head part, which stores the index l ($l = 1, 2, \ldots, k_i$) of the element visited by the current tour in the super vertex u_i. When $\widehat{m} < i \leqslant \widehat{m}+(\widehat{m}+\widetilde{m})$, the ith gene lies in the body part, which stores the index k ($k \in \{1, 2, \ldots, \widehat{m}+\widetilde{m}\}$) of the generalized vertex.

Based on the coding technique developed by Wu et al. [23], remained as the "permutation" concept illustrated in Section 2 in the body part, we consider adding two local search processes and develop a discrete PSO method for the GTSP. Eq. (1) is still working in the method. However the particles' position X_i and velocity V_i are different from those of TSP problems for the existence of super vertices in GTSP problem. The code of X_i could be denoted as:

$$\begin{aligned} X_i &= \{x_{i1}, \ldots, x_{i\widehat{m}}, x_{i\widehat{m}+1}, \ldots, x_{i,2\widehat{m}+\widetilde{m}}\} \\ &= \{X^H, X^B\}, \end{aligned} \tag{11}$$

where

$$X^H = \{x_{i1}, \ldots, x_{i\widehat{m}}\}, \tag{12}$$

and

$$X^B = \{x_{i\widehat{m}+1}, \ldots, x_{i,2\widehat{m}+\widetilde{m}}\}. \tag{13}$$

Similarly, we have

$$V_i = \{V^H, V^B\}, \tag{14}$$

where

$$V^H = \{v_{i1}, \ldots, v_{i\widehat{m}}\}, \tag{15}$$

and

$$V^B = \{v_{i\widehat{m}+1}, \ldots, v_{i,2\widehat{m}+\widetilde{m}}\}. \tag{16}$$

The calculation of the velocity of the body part is the same as Section 2, namely that

$$\begin{aligned} V_i^B(k+1) &= c_1 R_1^{B,k}\big(P_i^{B,k} - x_i^B(k)\big) \\ &\quad + c_2 R_2^{B,k}\big(P_g^{B,k} - x_i^B(k)\big) \\ &\quad + w\big(c_1 R_1^{B,k-1}\big(P_i^{B,k-1} - x_i^B(k-1)\big)\big) \\ &\quad + c_2 R_2^{B,k-1}\big(P_g^{B,k-1} - x_i^B(k-1)\big), \end{aligned} \tag{17}$$

where the subtraction of different particles' positions is defined by Eq. (6). While X^H is not a sequence without repetition, so we give the updating formula of the velocity of X^H as follows:

$$\begin{aligned} V_i^H(k+1) &= \mathit{Round}\big\{w V_i^H(k) + c_1 R_1^H\big(P_i^H - x_i^H(k)\big) \\ &\quad + c_2 R_2^H\big(P_g^H - x_i^H(k)\big)\big\}, \end{aligned} \tag{18}$$

where $\mathit{Round}(\cdot)$ is the vector function to round the object real vector to an integer one.

To speed up the convergence, two local search techniques are used in the proposed algorithm. A local search to the head is added first. For each gene of the head, the best index number in the corresponding super vertex is searched first, and then the original index is replaced with the best one. The above local search process is named by local search I, which could be considered as a greed search. The second local search acts on the body, which is used to exchange two generalized vertices' positions randomly in each of iterations. If the fitness increases then the exchange is remained, else the original particle keeps unchanged. This is named by local search II and could be considered as a randomly search, correspondingly.

In summary, the proposed discrete PSO method for GTSP can be described as follows:

(1) Initialize the particle swarm: including the initializations of each particle's head and body, and the evaluation of the fitness of each particle. Then P_g is searched and P_i of each particle is set as its initial position.
(2) Apply PSO processes to each particle:
   (a) Execute the searching process for each particle according to Eq. (1) and Eqs. (17)–(18);
   (b) Execute local search I;
   (c) Execute local search II.
(3) Judge the stop criterion, namely whether the iteration reaches the given number or the best fitness reaches the given value. If the criterion is satisfied then stop the program, else go to Step 2.

4.3. Numerical results

19 instances from the TSPLIB library [24] have been selected to examine the validity of the proposed algorithm for solving GTSP problems. These instances were originally generated for testing the standard TSP algorithms. To test the GTSP algorithms, Fischetti et al. [22] provided an exact separation algorithm for "generalized subtour elimination constrains" by addressing a set of

Table 2
Results of the proposed algorithm for benchmark test problems

Problem	*Opt*	Best result	Worst result	*Err* (%)
11EIL51	174	176	183	3.74
14ST70	316	315	331	0.62
16EIL76	209	214	232	5.46
16PR76	64825	64925	68073	1.36
20KROA100	9711	9758	10839	3.26
20KROB100	10328	10328	11596	3.90
20KROC100	9554	9572	11228	3.13
20KROD100	9450	9450	10427	2.46
20KROE100	9523	9523	10264	3.34
21EIL101	249	254	304	9.69
21LIN105	8213	8213	8989	2.49
22PR107	27898	27901	32974	1.56
25PR124	36605	36782	41804	2.20
29PR144	45886	45890	52510	4.04
30KROA150	11018	11085	13204	7.93
30KROB150	12196	12538	14700	8.75
31PR152	51576	51872	56217	4.38
32U159	22664	22795	28244	7.29
40D198	10557	10693	11779	5.20

inequalities, so that we could get the same partition results at different running provided that the data order are the same (the details could be referenced at Section 3 of [22]). Therefore the exact separation algorithm can be used to generate test data for different algorithms. In the following experiments, the population size is taken as 80.

Table 2 presents the numerical results. The second column stands for the exact optimal tour length for each problem from Ref. [22]. Table 2 shows that the proposed discrete PSO method can be used to solve the GTSP effectively. From Table 2 it can be seen that among the 19 test problems, the maximum relative error is 9.69% and the average relative error of all the testing instances is 4.25%. Therefore the numerical results are fairly good.

5. Conclusions

Focused on the TSP problems, a novel discrete PSO algorithm has been presented by adding an uncertain strategy into the approach. Furthermore, the algorithm is extended for solving the generalized TSP problems by introducing the generalized chromosome technique. To the best of our knowledge, it is the first time that the PSO-based algorithm has been used to solve the GTSP problems. Some benchmark problems are used to examine the effectiveness of the proposed algorithms. Numerical results show that the proposed algorithms are effective. It has also been shown that the proposed algorithms can solve larger size problems than those solved using the existing algorithms.

Acknowledgements

The first two authors are grateful to the support of the National Natural Science Foundation of China under Grant Nos. 60433020 and 60673023, the science-technology development project for international collaboration of Jilin Province of China under Grant No. 20050705-2, the doctoral funds of the National Education Ministry of China under Grant No. 20030183060, and "985" Project of Jilin University.

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[2] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, Evolutionary Programming 7 (1998) 601–610.
[3] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.
[4] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.
[5] J.F. Chang, S.C. Chu, J.F. Roddick, J.S. Pan, A parallel particle swarm optimization algorithm with communication strategies, Journal of Information Science and Engineering 4 (21) (2005) 809–818.
[6] J. Kennedy, R.C. Eberhart, Discrete binary version of the particle swarm algorithm, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 5, Orlando, Florida, USA, 1997, pp. 4104–4108.
[7] M. Clerc, in: Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem New Optimization Techniques in Engineering, Springer, 2004, pp. 219–239.
[8] R.E. Bellman, Dynamic programming treatment of the traveling salesman problem, Journal of the ACM 9 (1962) 61–63.
[9] M. Bellmore, G.L. Nemhauser, The traveling salesman problem: a survey, Operations Research 16 (1968) 538–558.
[10] D.E. Goldberg, Messy genetic algorithms: motivation, analysis, and first results, Complex Systems 3 (1989) 493–530.
[11] G. Ausiello, M. Demange, L. Laura, V. Paschos, Algorithms for the on-line quota traveling salesman problem, Information Process Letters 92 (2004) 89–94.
[12] T. Munakata, Y. Nakamura, Temperature control for simulated annealing, Physical Review E 64 (2001) 046127.
[13] F.V. Fomin, A. Lingas, Approximation algorithms for time-dependent orienteering, Information Process Letters 83 (2002) 57–62.
[14] L. Huang, C.G. Zhou, K.P. Wang, Hybrid ant colony algorithm for traveling salesman problem, Progress in Natural Science 4 (13) (2003) 295–299.
[15] M. Clerc, Discrete particle swarm optimization illustrated by the traveling salesman problem, http://www.mauriceclerc.net, 2000.
[16] T. Hendtlass, Preserving Diversity in Particle Swarm Optimization, in: Lecture Notes in Computer Science, vol. 2718, Springer, 2003, pp. 4104–4108.

[17] K.P. Wang, L. Huang, C.G. Zhou, W. Pang, Particle swarm optimization for traveling salesman problem, International Conference on Machine Learning and Cybernetics 3 (2003) 1583–1585.
[18] A.L. Henry-Labordere, The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem, RIRO B 2 (1969) 43–49.
[19] J.P. Saskena, Mathematical model for scheduling clients through welfare agencies, CORS J. 8 (1970) 185–200.
[20] S.S. Srivastava, S. Kumar, R.C. Garg, P. Sen, Generalized traveling salesman problem through *n* sets of nodes, CORS J. 7 (1969) 97–101.
[21] Y.N. Lien, E. Ma, B.W.-S. Wah, Transformation of the generalized traveling-salesman problem into the standard traveling-salesman problem, Information Sciences 74 (1993) 177–189.
[22] M. Fischetti, J.J. Salazar, P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, Operations Research 45 (1997) 378–394.
[23] C.G. Wu, Y.C. Liang, H.P. Lee, C. Lu, A generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining, Physical Review E 70 (2004), 016701-1–13.
[24] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA Journal on Computing 3 (4) (1991) 376–384.
[25] G. Laporte, A. Asef-vaziri, C. Sriskandarajah, Some applications of the generalized traveling salesman problem, Journal of the Operational Research Society 47 (1996) 1461–1467.