Expert Systems with Applications 38 (2011) 3873–3883

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Parallelized genetic ant colony systems for solving the traveling salesman problem

Shyi-Ming Chen *, Chih-Yao Chien

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, ROC

ARTICLE INFO

Keywords:
Traveling salesman problem
Ant colony systems
Genetic algorithms
Parallelization
Parallelized genetic ant colony systems

ABSTRACT

In this paper, we present a new method, called the parallelized genetic ant colony system (PGACS), for solving the traveling salesman problem. It consists of the genetic algorithm, including the new crossover operations and the hybrid mutation operations, and the ant colony systems with communication strategies. We also make an experiment with three classical data sets got from the TSP library to test the performance of the proposed method. The experiment results show that the performance of the proposed method is better than Chu et al.'s method (2004).

1. Introduction

The traveling salesman problem (TSP) (Lawler, Lenstra, & Shmoys, 1985) is a NP-complete problem. It tries to find a complete route with a minimal cost. Because the traveling salesman problem is a good background for testing the performance of optimization methods, some methods (Adachi & Yoshida, 1995; Baraglia, Hidalgo, & Perego, 2001; Chien & Chen, 2009; Ellabib, Calamai, & Basir, 2007; Fiechter, 1994; Freisleben & Merz, 1996; Hopfield & Tank, 1985; Kirkpatrick, Gelatt, & Vecchi, 1983; Liu & Zeng, 2009; Martin & Otto, 1996; Naimi & Taherinejad, 2009; Saadatmand-Tarzjan, Khademi, Akbarzadeh-T, & Moghaddan, 2007; Xie & Liu, 2009; Yi, Bi, Yang, & Tang, 2008) have been developed for solving the traveling salesman problem. Kirkpatrick et al. (1983) presented a simulated annealing method to solve the traveling salesman problem by simulating the annealing action in metallurgy. Hopfield and Tank (1985) presented a method based on neural networks to make routing decisions for solving the traveling salesman problem by mimicking the properties of biological neurons. Fiechter (1994) presented a tabu searching method with a parallelized mechanism to solve the traveling salesman problem. Adachi and Yoshida (1995) presented the parallelized techniques and the idea of protected chromosomes to reduce the searching space to speed up the processing time of genetic algorithm (GA) for solving the traveling salesman problem. Freisleben and Merz (1996) presented a hybrid method combining the GA and local search heuristics to find a near-optimum solution for solving the traveling salesman problem. Martin and Otto (1996) presented a method combining the simulated annealing (SA) with the local heuristics to solve the traveling salesman problem. Baraglia et al. (2001) presented a hybrid heuristic genetic algorithm for solving the traveling salesman problem. Ellabib et al. (2007) presented a method with exchange strategies for multiple ant colony systems to solve the traveling salesman problem. Saadatmand-Tarzjan et al. (2007) presented a novel constructive-optimizer neural network for solving the traveling salesman problem. Yi et al. (2008) presented a fast elastic net method for solving the traveling salesman problem. Naimi and Taherinejad (2009) presented an ant colony algorithm using a local updating process to solve the traveling salesman problem. Chien and Chen (2009) presented a method based on parallelized genetic ant colony systems to solve the traveling salesman problem. Liu and Zeng (2009) presented a method using genetic algorithms with reinforcement learning to solve the traveling salesman problem. Xie and Liu (2009) presented a method using multiagent optimization systems for solving the traveling salesman problem.

Swarm intelligence is an important research topic for solving optimization problems. It is inspired by the social behaviors of real insects or animals. Ant colony optimization (ACO) is the most popular one in this research topic. The original algorithm of ACO is known as the ant system (Dorigo, 1992) which was proposed by Dorigo to solve the traveling salesman problem. Since then, some algorithms based on the ACO are presented, such as the ant colony system (ACS) (Dorigo & Gambardella, 1997a, 1997b), the MMAS (Stützle & Hoos, 1996, 2000), the rank-based AS (Bullnheimer, Hartl, & Strauss, 1997) and KCC-Ants (Naimi & Taherinejad, 2009). These algorithms are all based on the idea of updating the pheromone information to search the shortest route. ACO algorithms are also used to solve other combinational optimization problems, such as the quadratic assignment (Stützle & Hoos, 2000) and the job scheduling (Merkle, Middendorf, & Schmeck, 2002; Merkle & Middendorf, 2003).

Genetic algorithms (GA) are important techniques in the research area of evolutionary computing. Because GA can search

* Corresponding author. Tel.: +886 2 27376417; fax: +886 2 27301081.
E-mail address: smchen@mail.ntust.edu.tw (S.-M. Chen).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.09.048

the domain space globally, it has been widely used to solve large-scale combinational optimization problems. As long as the fitness function is determined, we can obtain a good result by means of GA. Adachi and Yoshida (1995) presented protected chromosomes techniques to implement GA to solve the traveling salesman problem.

In this paper, we propose a new method, called the parallelized genetic ant colony system (PGACS) to solve the traveling salesman problem. First, we use the ant colony system to obtain the initial solutions. Then, we use the initial solutions as the initial population of GA to obtain better solutions. If GA searches a better solution, we use the global update of ACS to feedback the learning experience to ACS. After a certain number of cycles, we use the communication strategies (Chu, Roddick, & Pan, 2004) to exchange the learning experience between groups to speed up the convergence process. The performance of the proposed method is better than Chu et al.'s method (2004).

The rest of this paper is organized as follows. In Section 2, we briefly review the concepts of ant system (AS) and the ant colony optimization (ACO). In Section 3, we briefly review the concepts of genetic algorithms. In Section 4, we present a new method to solve the traveling salesman problem by parallelized genetic ant colony systems. In Section 5, we show the experiment results of the proposed method based on three classical TSP data sets. The conclusions are discussed in Section 6.

2. Ant colony optimization

In this section, we briefly review basic concepts of ant systems (AS) and the ant colony optimization (ACO). The ant system was proposed by Dorigo et al. (Colorni, Dorigo, & Maniezzo, 1991; Dorigo, 1992; Dorigo, Maniezzo, & Colorni, 1992, 1996). It is a cooperative population-based searching algorithm and is inspired by the foraging behavior of real ants. When a real ant searches the food from its nest, it will form a route based on the quantities of the pheromone on each edge in which it passed through. Each ant will deposit the pheromone on each traveled edges while searching, such that the pheromone level of those visited edges will be increased and the pheromone level on unvisited edges will be decayed. The other ants will search their own routes according to their experiences based on the pheromone levels in the edges that they choose to travel. The higher the pheromone level on an edge, the more the attractive to an ant. The more the ants passed the edge, the higher the pheromone level on this edge. The procedure of this searching behavior can be applied to solve the travel salesman problem. Assume that there are n cities and m ants and assume that the initial pheromone level on each edge is set to a very small non-zero positive constant τ_0. In each cycle, each ant starts at a randomly selected city and visits the rest of each city once and only once according to the transition rule based on the pheromone level on each edge. Therefore, the learning procedure is to update the level of pheromone. The ideas of ant systems to solve the traveling salesman problem are reviewed as follows (Dorigo and Gambardella, 1997; Dorigo et al., 1996):

(1) Transition rule: From city r, the next city s in the route is selected by ant k among the unvisited cities memorized in J_r^k according to the following rule:

$$s = \arg\max_{u \in J_r^k} \left[\tau(r,u)\cdot\eta(r,u)^{\beta}\right], \quad \text{if } q \leqslant q_0 \quad \text{(Exploitation)}$$

or select the next city s with the transition probability $P_k(r,s)$,

$$P_k(r,s) = \begin{cases} \frac{\tau(r,s)\cdot\eta(r,s)^{\beta}}{\sum_{u\in J_r^k}\tau(r,u)\cdot\eta(r,u)^{\beta}}, & \text{if } s \in J_r^k, \\ 0, & \text{otherwise,} \end{cases} \quad \text{if } q > q_0 \quad \text{(Bias exploitation)}$$

where J_r^k is the set of cities that remain to be traveled by the kth ant, $\tau(r,u)$ is the pheromone level between city r and city u, $\eta(r,u)$ is the inverse of the Euclidian distance from city r to city u, and the parameter β is the relative importance of the pheromone level versus the Euclidian distance.

(2) Pheromone update rule:

$$\tau(r,u) \leftarrow (1-\rho)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s),$$

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k}, & \text{if } (r,s) \in \text{ the route performed by the } k\text{th ant,} \\ 0, & \text{otherwise.} \end{cases}$$

The process of AS for solving the TSP is shown in Fig. 1 (Colorni et al., 1991). An ant system requires a lot of computation time to process the transition probability. In order to improve the searching efficiency, some improved algorithms are proposed, such as the MAX–MIN ant system (MMAS) (Stützle & Hoos, 1996, 2000), the ant colony system (ACS) (Dorigo & Gambardella, 1997a, 1997b; Gambardella & Dorigo, 1996), the rank-based AS (Bullnheimer et al., 1997) and KCC-Ants (Naimi & Taherinejad, 2009).

ACS (Dorigo & Gambardella, 1997a, 1997b; Gambardella & Dorigo, 1996) is based on AS, but it updates the pheromone by two operations, i.e., the local update and the global update. The local update operation is performed after the ants moved to the next city, where its main idea is to diversify the searching space performed by the subsequence ants during the cycle. The global update is performed only for the shortest route after all ants complete their routes, where its main idea is that only the best ant could contribute to the colony. The update rules of ACS are shown as follows (Dorigo and Gambardella, 1997):

(1) Local pheromone update:

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\tau_0,$$

where ρ is the pheromone evaporation parameter and $0 < \rho < 1$.

Procedure Ant System Algorithm for Solving the TSP:
Step 1: Set parameters and initialize the pheromone on each edge.
Step 2: For each ant, choose the next city with the probability $P_k(r, s)$ given by the transition rule. If each ant travels all cities, go to **Step 3**. Otherwise, go to **Step 2**.
Step 3: Update the pheromone on each edge.
Step 4: If the termination condition met, then print the best route. Otherwise, go to **Step 2**.

Fig. 1. The process of AS for solving the TSP (Colorni et al., 1991).

(2) Global pheromone update:

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s)+\rho\cdot\Delta\tau_k(r,s),$$

$$\Delta\tau_k(r,s)=\begin{cases}\frac{Q}{L_{gb}}, & \text{if } (r,s)\in \text{global best route},\\ 0, & \text{otherwise},\end{cases}$$

where Q is a non-zero positive constant, L_{gb} is the length of the best route, and ρ is the pheromone evaporation parameter and $0<\rho<1$.

MMAS (Stützle & Hoos, 1996, 2000) is similar to ACS, but the pheromone level update operation is bounded by the pheromone level between $\tau_{\min}$ and $\tau_{\max}$. The update rules of MMAS are shown as follows:

$$\tau(r,s) \leftarrow \left[(1-\rho)\cdot\tau(r,s)+\Delta\tau^{best}(r,s)\right]_{\tau_{\min}}^{\tau_{\max}},$$

$$\Delta\tau^{best}(r,s)=\begin{cases}\frac{1}{L_{best}}, & \text{if } (r,s)\in \text{best tour},\\ 0, & \text{otherwise},\end{cases}$$

$$[x]_b^a=\begin{cases}a, & \text{if } x>a,\\ b, & \text{if } x<b,\\ x, & \text{otherwise},\end{cases}$$

where L_{best} is the shortest length of the tour and $\tau_{\min}$ and $\tau_{\max}$ are the lower bound and the upper bound of the pheromone, respectively.

3. Basic concepts of genetic algorithms

In this section, we briefly review basic concepts of genetic algorithms (Holland, 1975; Gen & Cheng, 1997). Genetic algorithm (GA) was proposed by Holland in 1975, where it is inspired by the biological evolution of species. In a GA system, it encodes the parameters of a solution into a numerical stream called the chromosome. The basic element of a chromosome is called the gene. In a GA system, it randomly generates a certain number of chromosomes as the initial population. Then, it randomly selects two chromosomes from the population to perform the crossover operation and the mutation operation repeatedly until the result satisfies the termination condition or the maximum number of generations is reached. After performing the crossover operation and the mutation operation, the system will calculate the fitness value of each chromosome. Chromosomes with higher fitness values will be selected into the gene pool for reproduction in the next generation. The main operations of GA are shown as follows (Holland, 1975):

(1) Selection: While performing the selection operations, the system uses either the tournament selection (Baker, 1985; Freisleben & Merz, 1996) or the roulette wheel selection (Baker, 1985; Freisleben & Merz, 1996) to select chromosomes with higher fitness values into the gene pool to perform the crossover operation. The process stops while the new population is full. The purpose of the selection operation is that the chromosomes with larger fitness values are chosen to participate in the production of the next generation.
(2) Crossover: While doing the crossover operation, the system randomly picks up two chromosomes as parents to perform the crossover operation. The system compares a randomly generated number with a predefined crossover rate between 0 and 1. If the randomly generated number is smaller than or equal to the crossover rate, then the system randomly chooses a crossover point and exchanges the genes after the crossover point to generate two offsprings. Then, the system puts these two offsprings into the gene pool, where the size of the gene pool is equal to the size of the population. Otherwise, the system puts these two chromosomes back into the gene pool. The crossover operation will be performed repeatedly until the gene pool is full. The purpose of the crossover operation is to spend the solution space in order to search better solutions.
(3) Mutation: After the crossover operation, the system chooses a certain number of chromosomes from the gene pool to perform the mutation operation. The system compares a randomly generated number with a predefined mutation rate between 0 and 1. If the randomly generated number is smaller than or equal to the mutation rate, then the system randomly selects some genes of the selected chromosome to mutate, i.e., to change the values of the genes. Then, it puts the mutated chromosome back into the gene pool. Otherwise, keep the chromosome unchanged and put it back into the gene pool. After the mutation operation is performed, the chosen chromosomes will be put back into the gene pool. It should be noted that not every generation will do the mutation. The purpose of the mutate operation is to prevent the GA system to fall into the local minimum.
(4) Calculate the fitness value: After the crossover and the mutation processes, the GA will calculate the fitness value of each chromosome by a fitness function, where chromosomes with higher fitness values will be selected into the gene pool for reproduction in the next generation. The GA stops if the terminated condition or the maximum number of generations is reached. Otherwise, it will return to the selection process to reproduce a new generation to search better solutions.

The procedure of GA is shown in Fig. 2 (Holland, 1975).

Procedure Genetic Algorithm:
Step 1: Set the initial population.
Step 2: Evaluate the fitness value of each chromosome.
Step 3: /*Searching process*/:
1) Select the best-ranking chromosomes for reproduction. /*Selection*/
2) Randomly select two chromosomes as parents to do the crossover to breed the offspring. /*Crossover*/
3) Randomly select a certain number of chromosomes of the offspring for mutation. /*Mutation*/
4) Evaluate the fitness value of each offspring. /*Evaluation*/
Step 4: If the termination condition or the maximum number of generations is reached, then the chromosome having the highest fitness value is the best solution. Otherwise, go to **Step 3**.

Fig. 2. The procedure of GA algorithm (Holland, 1975).

4. The proposed parallelized genetic ant colony systems

In this section, we present a new method called the parallelized genetic ant colony system (PGACS) for solving the traveling salesman problem. First, we use the ant colony system to produce the initial population of the genetic algorithm. Then, we use the genetic algorithm to find a better solution based on the proposed bone-crossover operations, the traditional-crossover operations (Holland, 1975), and the route-mutation operations (Holland, 1975) and the pheromone-mutation operations (Li & Song, 2006). If the GA finds a better solution than the best solution that the ACS has got previously, it updates the pheromone level of this route with the global update operation. After a predefined number of cycles, the proposed method uses the communication strategies (Chu et al., 2004) to exchange the pheromone experience, where "a cycle" consists of "one execution for ACS" and "a couple of executions for GA". We use a "cycle counter" to count the execution times of the proposed PGACS. The flowchart of the proposed PGACS is shown in Fig. 3. The proposed PGACS is presented as follows:

Step 1: /* *Initialization*/ Generate g groups of ants, i.e., $G_1, G_2, \ldots,$ and G_g, where each group has N ants. Let each ant randomly selects a city as its starting city. Let the initial pheromone between any two cities be τ_0. Set the cycle counter to 0.

Step 2: /**ACS-Movement**/ Choose the next city s for the kth ant in the ith group according to the following equation:

$$s = \begin{cases} \arg\max_{u \in J_{k,i}(r)} \left[\tau_i(r,u) \cdot \eta(r,u)^{\beta}\right], & \text{if } q \leqslant q_0 \text{ (Exploitation)}, \\ \end{cases} \tag{1}$$

or visit the next city s with the probability $P_{k,i}(r,s)$,

$$P_{k,i}(r,s) = \begin{cases} \frac{\left[\tau_i(r,s)\cdot\eta(r,s)^{\beta}\right]}{\sum_{u \in J_k(r)} \left[\tau_i(r,u)\cdot\eta(r,u)^{\beta}\right]}, & \text{if } s \in J_{k,g}(r) \\ 0, & \text{otherwise} \end{cases} \quad \text{if } q > q_0 \text{ (Bias exploitation)}, \tag{2}$$

where $P_{k,i}(r,s)$ is the transition probability from city r to city s for the kth ant in the ith group, $\tau_i(r,u)$ is the pheromone level between city r and city u in the ith group, $\eta(r,u)$ is the heuristic information, which is defined as the inverse of the distance from city r to city u, $J_{k,i}(r)$ is the set of cities that remain to be traveled by the kth ant in the ith group, the parameter β controls the relative importance of the pheromone versus the heuristic information, q is a random constant deciding the kth ant in the ith group to choose the normal exploitation way or the biased exploitation way and q_0 is a constant between 0 and 1.

Step 3: /**ACS-local Pheromone Update**/ Update the pheromone level between any two cities for each group as follows:

$$\begin{aligned} &\tau_i(r,s) = (1-\rho)\cdot\tau_i(r,s) + \rho\cdot\tau_0, \\ &\text{if } \tau_i(r,s) < \tau^i_{\min}, \text{ then } \tau_i(r,s) = \tau^i_{\min}, \end{aligned} \tag{3}$$

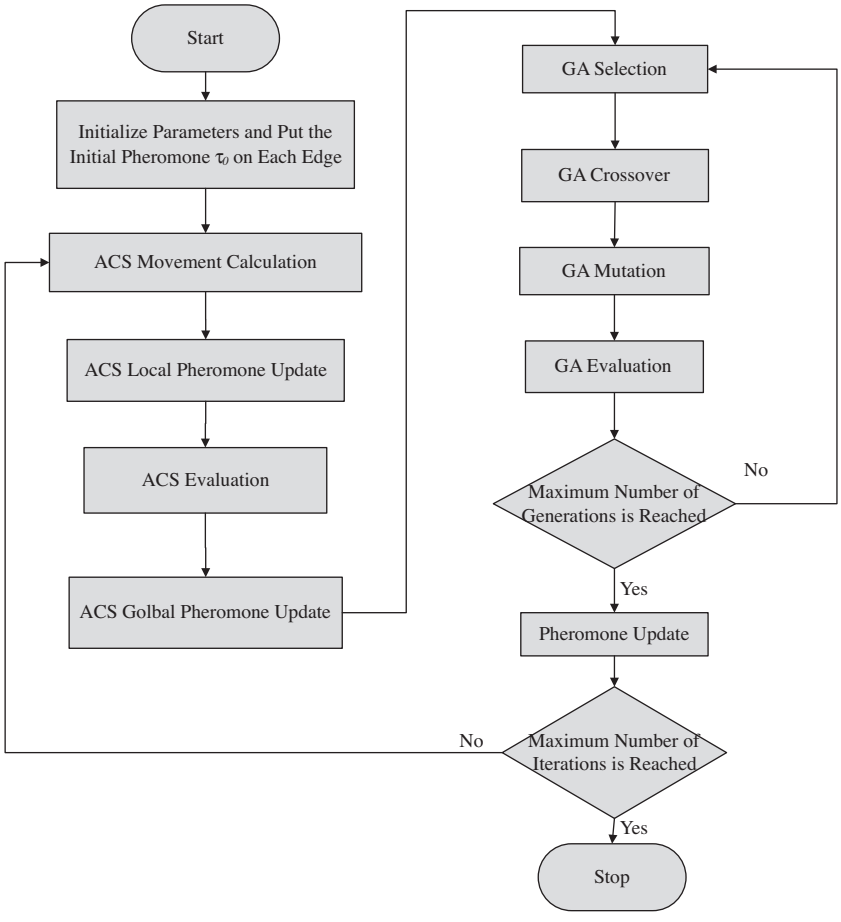

Fig. 3. The flowchart of the proposed PGACS.

where ρ is the pheromone evaporation parameter which is a constant between 0 and 1, and $\tau^i_{\min}$ is the lower bound of the pheromone level on each edge of the ith group (Stützle & Hoos, 1996, 2000).

Step 4: /*ACS-Evaluation*/ Calculate the total distance of the route performed by ants in each group.

Step 5: /*ACS-Global Pheromone Update*/ Update the pheromone level between cities for each group, shown as follows:

$$\begin{aligned}
&\tau_i(r,s) = (1-\rho)\cdot\tau_i(r,s) + \rho\cdot\Delta\tau_i(r,s),\\
&\Delta\tau_i(r,s) = \begin{cases} \frac{1}{L_{group_i_best}}, & \text{if } (r,s)\in \text{best route of the } i\text{th group},\\ 0, & \text{otherwise},\end{cases}\\
&\text{if } \tau_i(r,s) > \tau^i_{\max}, \text{ then } \tau_i(r,s) = \tau^i_{\max},
\end{aligned} \tag{4}$$

where $L_{group\_i\_best}$ is the shortest length of the ith group and $\tau^i_{\max}$ is the upper bound of the pheromone level on each edge of the ith group (Stützle & Hoos, 1996, 2000).

Step 6: /*GA-Selection*/ Let the sequence of cities searched by an ant form a chromosome and let the cities in the city sequence be genes. For the ith group, select m predominant ants from the ith group and select n predominant ants among G groups into the gene pool to participate in the reproduction process for the new population. The

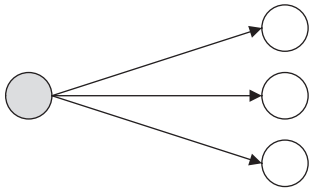

Fig. 5. Strategy 1.

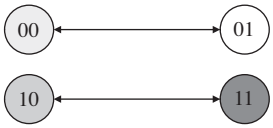

Fig. 6. Strategy 2.

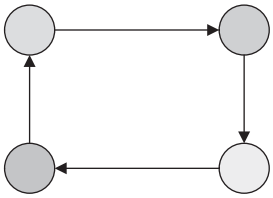

Fig. 7. Strategy 3.

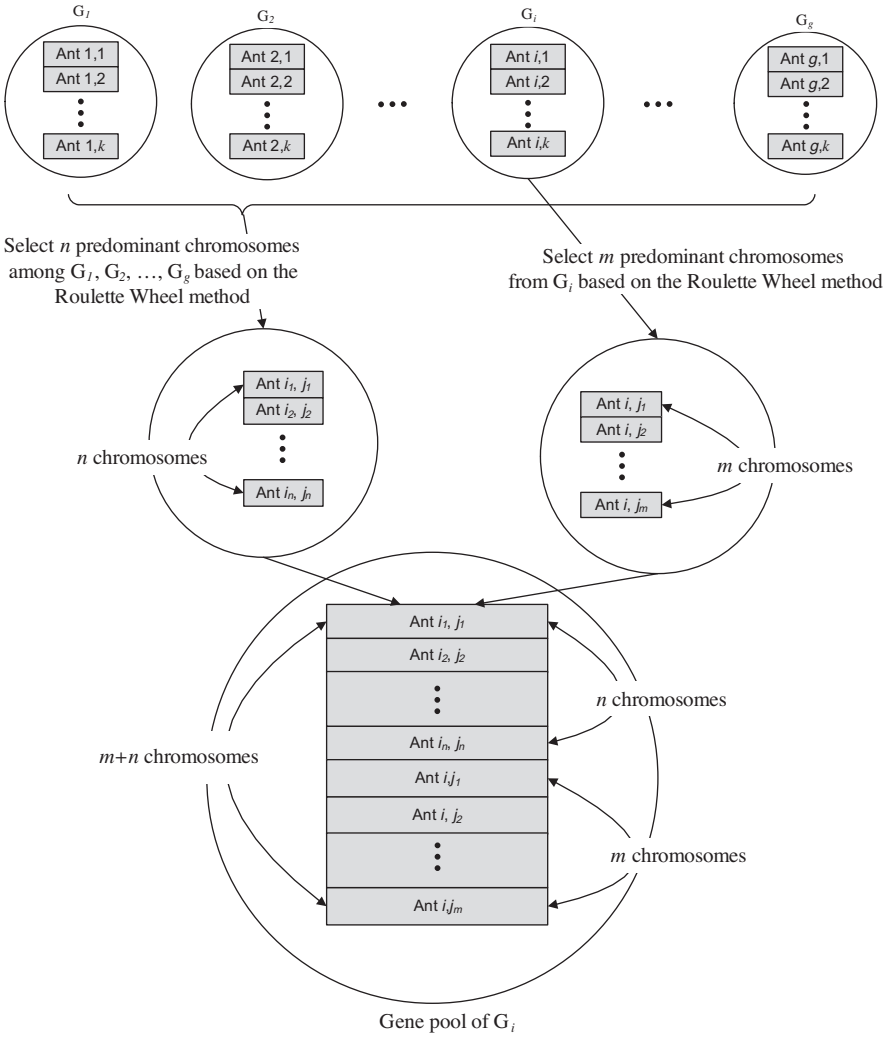

Fig. 4. The selection process of group G_i of the proposed PGACS.

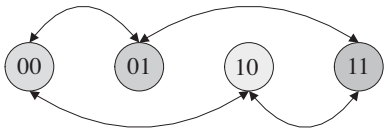

Fig. 8. Strategy 4.

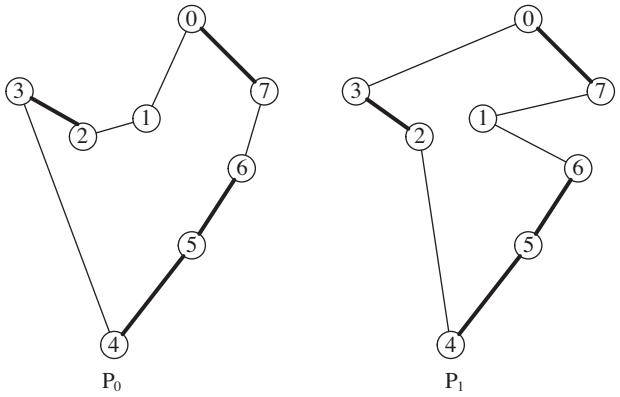

Fig. 9. Routes of P_0 and P_1, where the thicker edges are bones of P_0 and P_1.

shorter the route length the ant performs, the more predominant the ant is. The details of the selection process are shown in Fig. 4.

Step 7: /*GA-Crossover*/ For the *i*th group, randomly select two ants, called the chromosomes in GA, as the parents to perform the crossover operation using the two strategies shown as follows:

(1) *Bone-Crossover*: If $r \leqslant R_0$, where R_0 is a constant between 0 and 1 and r is a random parameter to decide which crossover mode that the system takes, then choose the bone-crossover. First, the system finds the intersection set of two ants. Each connected part in the intersection is called a *bone*. Then, it chooses the longest bone as the base-bone of their offsprings. Then, starting with the base-bone's end, select the next city according the following rules:
   (i) If the start city is an element of a bone in the bone set, then connect the bone with the base-bone to form a new one and eliminate it from the bone set.
   (ii) If the cities connected with the start city are already traveled, then choose the nearest city to travel.
   (iii) If the start city is not an element of a bone in the bone set, then find the connected cities in the ant's parents and choose the city with the highest pheromone level on the path to connect.

(2) *Traditional-Crossover* (Holland, 1975): If $r > R_0$, where R_0 is a constant between 0 and 1 and r is a random parameter to decide which crossover mode that the system takes, then choose the traditional-crossover. Randomly select two crossover points, where the crossover operation is performed by exchanging the segment of the chromosomes between two crossover points of the selected two chromosomes. Then, it eliminates the duplicate cities outside the two crossover points or appends the missing cities to the end of the offsprings.

Step 8: /*GA-Mutation*/ For the *i*th group, the mutation operation includes the following two strategies:

(1) *Route-Mutation*: Randomly select two segments S_0 and S_1 for mutation. The mutation is performed by exchanging S_0 and S_1 if the randomly generated number i_r is smaller than a predefined mutation rate ϕ_r.

(2) *Pheromone-Mutation*: If the randomly generated number i_p is smaller than ϕ_p, where ϕ_p is the pheromone mutation rate, then randomly select an edge between two cities and change its pheromone level to a random generated constant between τ_{min} and τ_{max}, where the values of i_r, i_p, ϕ_r and ϕ_p are between 0 and 1, respectively.

Step 9: /*GA-Evaluate*/ Calculate the fitness value of the offspring for each group, where the fitness value is equal to the total length of the route. If the maximum number of generations is reached, then go to *Step 10*. Otherwise, go to *Step 6*.

Step 10: /*Communication Pheromone Update*/ Use the seven communication strategies (Chu et al., 2004) to exchange the pheromone experience. For the *i*th group,

(1) Strategy 1: For every T_1 cycles, using the best group to update the pheromone level between cities for the other groups. The strategy is shown in Fig. 5.
(2) Strategy 2: For every T_2 cycles, using the *neighbor* to update the pheromone level between cities, where the *neighbor* is the group whose binary representation of the group number differs only the last significant bit from the processing group. The strategy is shown in Fig. 6.
(3) Strategy 3: For every T_3 cycles, using the *neighbor* to update the pheromone level between cities, where the *neighbor* is the group arranged as the ring structure. The strategy is shown in Fig. 7.
(4) Strategy 4: For every T_4 cycles, using the *neighbor* to update the pheromone level between cities, where the *neighbor* is the group whose binary representation of the group number differs only one bit from the processing group. The strategy is shown in Fig. 8.
(5) Strategy 5: Update the pheromone level between cities for each group using both Strategy 1 and Strategy 2.
(6) Strategy 6: Update the pheromone level between cities for each group using both Strategy 1 and Strategy 3.
(7) Strategy 7: Update the pheromone level between cities for each group using both Strategy 1 and Strategy 4.

Step 11: /*Termination*/ If the terminated condition or the maximum number of cycles is reached, then print out the best solution. Otherwise, go to *Step 2*.

Example 4.1. Let us consider a 7-cities TSP problem. Assume that there are two routes P_0 and P_1 performed by ACS and assume that P_0 = 0 1 2 3 4 5 6 7 and P_1 = 0 3 2 4 5 6 1 7, as shown in Fig. 9. When P_0 and P_1 are performed the crossover with the bone-crossover strategy, the intersection (i.e., the bone set) is {0 7, 2 3, 4 5 6}, as shown in Fig. 9. We take the longest bone as the base of their offsprings, i.e., {4 5 6}, as shown in Fig. 10 (a), and choose City 6 as the starting city. First, find those cities connected with the starting city "City 6", which are City 5 and City 7 in P_0 and City 5 and City 1 in P_1. Because City 5 is already in the base bone, we compare the pheromone levels of City 1 and City 7 with City 6, respectively. In this case, we choose City 7. Therefore, the new bone is {4 5 6 7}, as shown in Fig. 10 (b), and the new starting city is City 7. Then, we check whether the new starting city is in the bone set. Because City 7 is an element of the bone in the bone set, we choose the bone containing City 7, i.e., add {0 7} to form a new base bone. The new bone becomes {4 5 6 7 0}, as shown in Fig. 10 (c), and the new starting city is City 0. Then, because the starting city "City 0" is not in the base bone, we find the cities connected with City 0

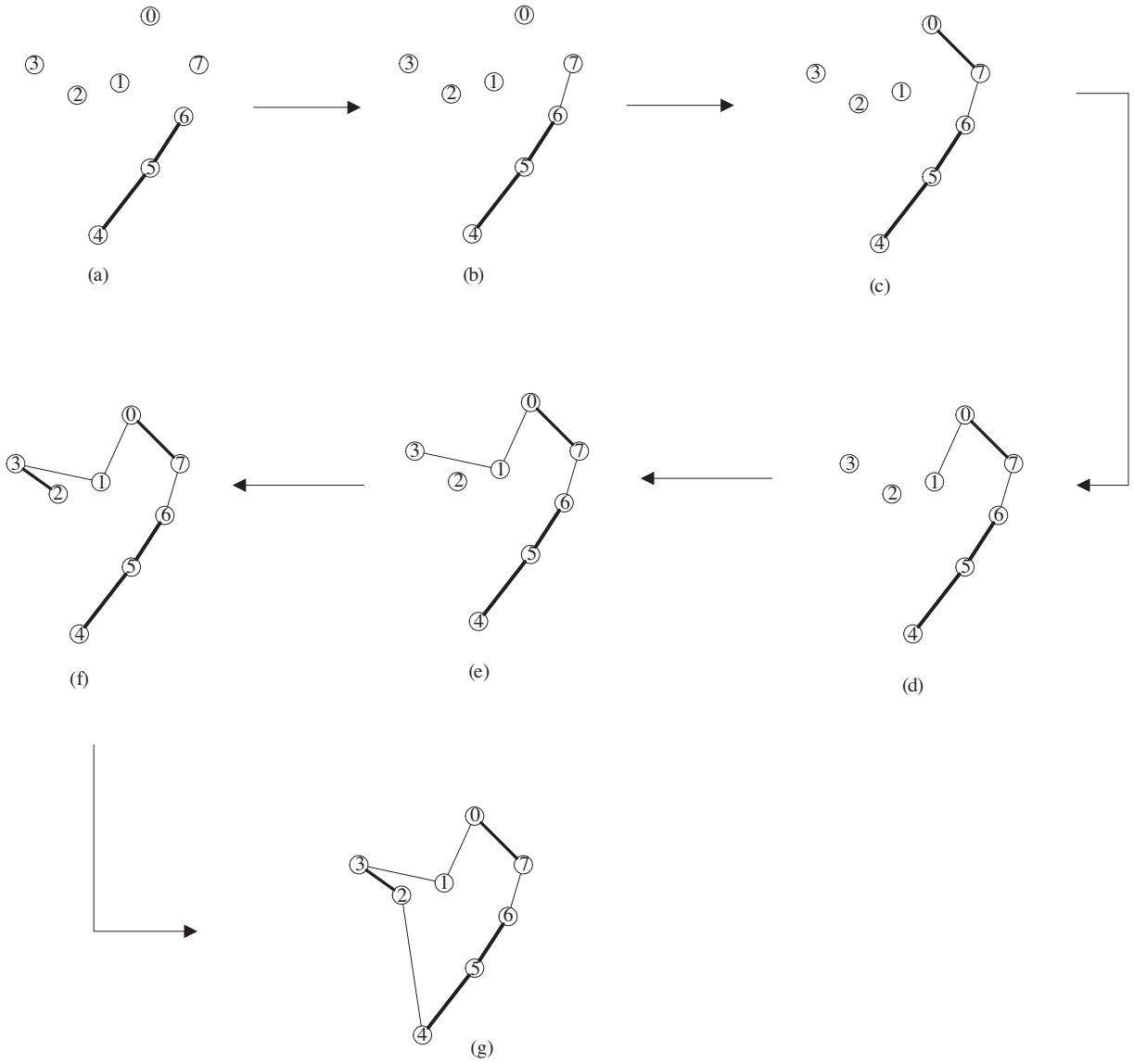

Fig. 10. The procedure of the bone-crossover strategy.

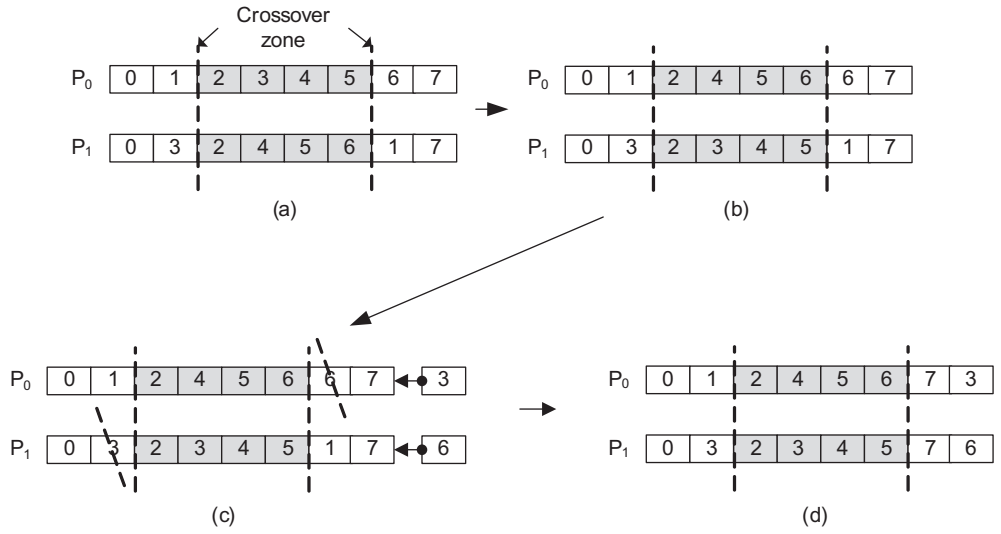

Fig. 11. The process of the normal-crossover strategy.

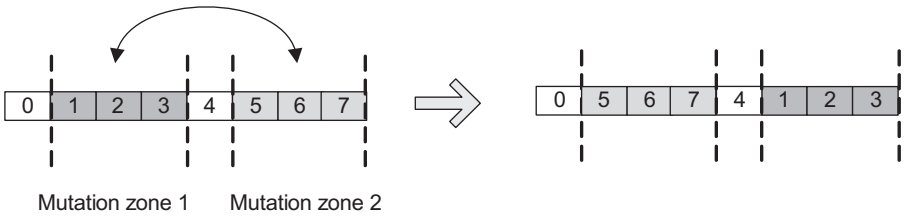

Fig. 12. The route mutation operation.

in P_0 and P_1, respectively, i.e. City 7 and City 1 in P_0 and City 7 and City 3 in P_1. Because City 7 is already in the base bone, we compare the pheromone levels of City 1 and City 3 with City 0, respectively. Then, the system chooses City 1 to join the base bone. Therefore, the new bone becomes {4 5 6 7 0 1}, as shown in Fig. 10 (d). These steps are performed repeatedly until the full route of the offspring is completed, which are shown from Fig. 10 (e) to Fig. 10 (g).

When P_0 and P_1 are performed the crossover using the normal-crossover strategy, the system randomly generates two crossover points, as shown in Fig. 11(a). Then, it exchanges the sequence of cities between two crossover points, as shown in Fig. 11(b). Because it is possible that some cities have already been traveled but are missed after the exchange process. Therefore, the system eliminates those duplicate cities outside the crossover zone and inserts those missing cities at the end of chromosomes, as shown in Fig. 11(c). Then, two offspring chromosomes are reproduced, as shown in Fig. 11(d).

When the crossover operation is finished, the offsprings will be performed the route mutation operation with the probability ϕ_r and will be performed the pheromone mutation operation with the probability ϕ_p. The route mutation operation is performed by randomly selecting two mutation zones with the same size and exchanges these two zones, as shown in Fig. 12. The pheromone

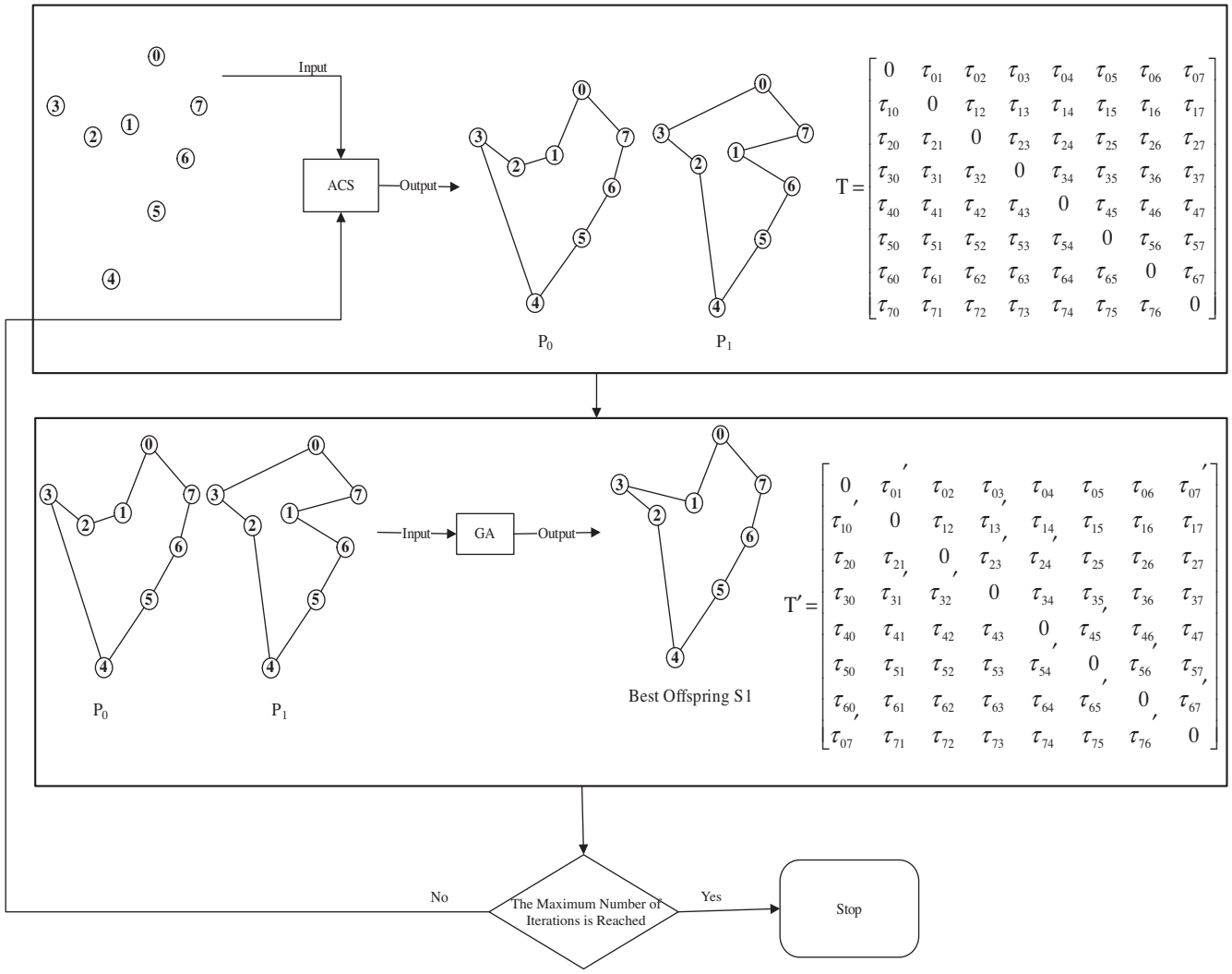

Fig. 13. An example of the search process and the feedback process of the proposed PGACS.

Table 1
Parameters settings.

Parameters	st70.tsp	eil101.tsp	tsp225.tsp
β	2.0	2.0	2.0
ρ	0.1	0.1	0.1
q_0	0.9	0.9	0.9
The number of GA generations	100	200	200
crossover rate	1.0	1.0	1.0
R_0	0.33	0.33	0.33
route_mutate_rate ϕ_r	0.3	0.3	0.3
pheromone_mutate_rate ϕ_p	0.2	0.2	0.2
τ_{min}	$\tau_{max}/20$ (Liu et al., 2007; Yang et al., 2006)		
τ_{max}	$1/(1-\rho)\cdot L_{iteration\_best}$ (Liu et al., 2007; Yang et al., 2006)		

(Note: $L_{iteration\_best}$ denotes the shortest tour length of the ith iteration.)

mutation operation is performed by randomly selecting an edge and change the pheromone level with a randomly generated number between τ_{min} and τ_{max}.

If GA finds a better solution than ACS, the system will perform the feedback procedure to update the pheromone map of ACS. In the following, we use an example to illustrate the feedback process of the 7-cities TSP problem. While the cycle counter is 0, the pheromone level on each edge in the pheromone map is τ_0 and the initial pheromone map is T_0. Assume that P_0 and P_1 are two routes of the traveling results of ACS. Then,

(i) We use the traveling results of ACS as the input of GA, i.e., P_0 and P_1 in this case. Assume that the best offspring of P_0 and P_1 is S_1 with the route "0 1 3 2 4 5 6 7" and assume that the

Table 2
A comparison of the average tour length of the proposed PGACS with the PACS (Chu et al., 2004) for the EIL101 data set with 4 groups and 20 ants in each group.

Seed	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5	Strategy 6	Strategy 7
PACS (Chu et al., 2004)							
1	657	648	649	654	646	646	647
2	657	650	643	647	641	660	648
3	644	655	641	653	641	646	648
4	645	651	651	647	643	642	650
5	641	648	648	651	644	647	647
6	656	648	645	648	647	651	644
7	642	649	646	651	650	647	645
8	646	653	658	647	653	647	652
9	645	651	652	649	652	649	646
10	643	654	645	652	646	646	650
Average	648	651	648	650	646	648	648
The proposed PGACS							
1	645	646	643	645	646	646	645
2	645	649	650	649	650	650	650
3	645	646	645	646	645	650	649
4	642	652	650	650	645	645	653
5	649	650	643	644	648	650	650
6	640	641	641	640	641	641	640
7	642	642	645	642	647	645	642
8	642	642	642	642	642	642	642
9	646	648	648	648	650	648	650
10	640	642	640	640	640	642	640
Average	643	645	644	644	645	645	646

Table 3
A comparison of the average tour length of the proposed PGACS with PACS (Chu et al., 2004) for the ST70 data set with 4 groups and 20 ants in each group.

Seed	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5	Strategy 6	Strategy 7
PACS (Chu et al., 2004)							
1	679	683	678	681	684	680	682
2	681	677	688	677	679	677	677
3	681	678	678	678	677	681	678
4	677	682	678	679	685	686	681
5	678	678	678	678	678	682	677
6	691	694	678	678	692	689	689
7	682	678	677	681	681	678	677
8	677	677	677	681	681	678	678
9	677	679	678	677	683	683	678
10	678	682	678	681	680	682	677
Average	680	681	679	679	682	682	679
The proposed PGACS							
1	677	677	687	677	687	687	677
2	677	677	677	677	677	677	677
3	677	677	677	677	677	677	677
4	677	677	677	677	677	677	677
5	677	677	677	677	677	677	677
6	677	677	677	677	677	677	677
7	677	677	677	677	677	677	677
8	677	677	677	677	677	677	677
9	677	677	677	677	677	677	677
10	684	684	684	684	684	684	684
Average	678	678	679	678	679	679	678

Table 4
A comparison of the average tour length of the proposed PGACS with the PACS (Chu et al., 2004) for the TSP225 data set with 4 groups and 20 ants in each group.

Seed	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5	Strategy 6	Strategy 7
PACS (Chu et al., 2004)							
1	3907	3913	3879	3903	3882	3866	3884
2	3903	3879	3883	3955	3916	3902	3891
3	3888	3900	3889	3953	3906	3878	3919
4	3892	3908	3889	3895	3871	3866	3919
5	3881	3888	3879	3879	3882	3878	3886
6	3942	3916	3892	3961	3883	3901	3882
7	3904	3876	3881	3881	3881	3892	3882
8	3950	3912	3950	3890	3957	3936	3952
9	3903	3903	3889	3896	3882	3904	3881
10	3877	3875	3877	3876	3884	3919	3895
Average	3905	3897	3891	3909	3894	3894	3899
The proposed PGACS							
1	3862	3872	3872	3872	3872	3872	3872
2	3930	3940	3940	3924	3940	3924	3924
3	3862	3859	3859	3865	3859	3859	3859
4	3864	3859	3864	3864	3864	3859	3859
5	3894	3881	3881	3904	3917	3917	3904
6	3876	3876	3876	3876	3876	3876	3876
7	3901	3859	3898	3896	3950	3899	3896
8	3859	3865	3859	3862	3859	3859	3862
9	3891	3885	3879	3879	3882	3879	3878
10	3882	3881	3881	3876	3876	3965	3876
Average	3882	3878	3881	3882	3890	3891	3881

route length is L_0, as shown in Fig. 13. We add the quantity $1/L_0$ of the pheromone to the original pheromone map T, i.e., let $\tau_{i,j\in S_1}(i,j)' \leftarrow \tau_{i,j\in S_1}(i,j) + \frac{1}{L_0}$. Therefore, a new pheromone map T' is generated.

(ii) After the pheromone update process, the system will check whether the cycle counter reaches the maximum number of cycles. If the cycle counter reaches the maximum number of cycles, the system will output the final best route and its length. If the cycle counter does not reach the maximum number of cycles, the system will increase the cycle counter by one and return to ACS with the new pheromone map as its input.

Fig. 13 shows an example of the search procedure and the feedback procedure of the propose PGACS.

5. Experimental results

In this section, we make a comparison of the experimental results of the proposed PGACS with PACS (Chu et al., 2004). We have implemented the proposed method using Microsoft Visual C++ 2005 in a Pentium 4 PC. We use three data sets of the traveling salesman problem, i.e., EIL101, ST70 and TSP225, to compare the performance of the proposed PGACS with PACS, where the parameter settings are shown in Table 1. The number of cycles for ST70 and EIL101 are 1000 and 1000, respectively, and the number of cycles for TSP225 is 2000. The strategy communication periods T_1, T_2, T_3 and T_4 are set to 30, 30, 30, and 30, respectively. Experimental results of the proposed PGACS are demonstrated by the average tour length with 10 seeds, where the number of groups of the ants is set to 4 and there are 20 ants in each group.

A comparison of the average tour length of the proposed PGACS with the PACS (Chu et al., 2004) for the EIL101 data set with 4 groups and 20 ants in each group is shown in Table 2, where the parameter "*seed*" shown in Table 2 is the random seed of the function "*srand*()" in the C++ language (Deitel & Deitel, 2001). From Table 2, we can see that the average tour lengths regarding the seven strategies of the proposed PGACS are all better than those of the PACS (Chu et al., 2004).

A comparison of the average tour length of the proposed PGACS with the PACS (Chu et al., 2004) for the ST70 data set with 4 groups and 20 ants in each group is shown in Table 3, where the parameter "*seed*" shown in Table 3 is the random seed of the function "*srand* ()" in the C++ language (Deitel & Deitel, 2001).

A comparison of the average tour length of the proposed PGACS with the PACS (Chu et al., 2004) for the TSP225 data set with 4 groups and 20 ants in each group is shown in Table 4, where the parameter "*seed*" shown in Table 4 is the random seed of the function "*srand*()" in the C++ language (Deitel & Deitel, 2001).

6. Conclusions

In this paper, we have presented a new method called the parallelized genetic ant colony systems (PGACS) for solving the traveling salesman problem. We also make an experiment with tree data sets get from the TSP library to test the performance of the proposed method. From Tables 2–4, we can see that the performance of the proposed method is better than Chu et al.'s method (2004). It provides us with a useful way to solve the traveling salesman problem.

Acknowledgements

This work was supported in part by the Natural Science Council, Republic of China, under Grant NSC 97-2221-E-011-108.

References

Adachi, N., & Yoshida, Y. (1995). Accelerating genetic algorithms: Protected chromosomes and parallel processing. In *Proceedings of the first international conference on genetic algorithms in engineering systems: innovations and applications*, (p. 414). Sheffield, UK.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the 1985 international conference on genetic algorithms* (pp. 101–111). Hillsdale, New Jersey, USA: L. Erlbaum Associates.

Baraglia, R., Hidalgo, J. I., & Perego, R. (2001). A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 5*(6), 613–622.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). A new rank based version of the ant systems – A computational study. *Central European Journal for Operations Research and Economics, 7*(1), 25–38.

Chien, C.Y., & Chen, S.M. (2009). A new method for solving the traveling salesman problem based on parallelized genetic ant colony systems. In *Proceedings of the 2009 international conference on machine learning and cybernetics* (pp. 2828–2833). Baoding, Hebei, China.

Chu, S. C., Roddick, J. F., & Pan, J. S. (2004). Ant colony system with communication strategies. *Information Sciences, 167*(1–4), 63–76.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life* (pp. 134–142). Milano, Italy.

Deitel, H. M., & Deitel, P. J. (2001). *C: How to program* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

Dorigo, M. (1992). Learning and Nature Algorithm (in Italian). Ph.D. Dissertation, Dipartimento di Electtonica, Politecnico di Milano, Italy.

Dorigo, M., & Gambardella, L. M. (1997a). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Dorigo, M., & Gambardella, L. M. (1997b). Ant colonies for the traveling salesman problem. *Biosystems, 43*(2), 73–81.

Dorigo, M., Maniezzo, V., & Colorni, A. (1992). Positive Feedback as A Search Strategy. Technical Report. Dipartimento di Electtonica (pp. 91–106). Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 26*(1), 29–41.

Ellabib, I., Calamai, P., & Basir, O. (2007). Exchange strategies for multiple ant colony system. *Information Sciences, 177*(5), 1248–1264.

Fiechter, L. (1994). A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics, 51*(3), 243–267.

Freisleben, B., & Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the 1996 IEEE international conference on evolutionary computation* (pp. 616–621). Nagoya, Japan.

Gambardella, L.M., & Dorigo, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the 1996 IEEE international conference on evolutionary computation* (pp. 622–627). Nagoya, Japan.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design* (1st ed.). NY: John Wiley & Sons.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Cambridge, MA: MIT Press.

Hopfield, J. J., & Tank, D. W. (1985). computation of decisions in optimization problems. *Biological Cybernetics, 52*(3), 141–152.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Lawler, E. L., Lenstra, J. K., & Shmoys, D. B. (1985). *The traveling salesman problem: A guided tour of combinatorial optimization*. Chichester: Wiley Series in Discrete Mathematics & Optimization.

Li, B., & Song, X. (2006). Improved ant colony algorithm with pheromone mutation and its application in flow-shop problems. In *Proceedings of the 6th world congress on intelligent control and automation* (pp. 3353–3356). Dalian, China.

Liu, X. H., Liu, W. J., & Jin, T. G. (2007). Improved MMAS or multi-process routes decision-making. In *Proceedings of the 2007 IEEE conference on industrial electronics and applications* (pp. 1426–1430). China: Harbin.

Liu, F., & Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Systems with Applications, 36*(3), 6995–7001.

Martin, O. C., & Otto, S. W. (1996). Combining simulated annealing with local search heuristics. *Annals of Operations Research, 63*(1), 57–75.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333–346.

Merkle, D., & Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence, 18*(1), 105–111.

Naimi, H. M., & Taherinejad, N. (2009). New robust and efficient ant colony algorithms: Using new interpretation of local updating process. *Expert Systems with Applications, 36*(1), 481–488.

Saadatmand-Tarzjan, M., Khademi, M., Akbarzadeh-T, M.-R., & Moghaddan, H. A. (2007). A novel constructive-optimizer neural network for the traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 37*(4), 754–770.

Stützle, T., & Hoos, H.H. (1996). Improving the Ant System: A Detail Report on the MAX–MIN Ant System. Technical Report. AIDA-96-12. FG Intellektik, FB Informatik, TU Darmstadt, Germany.

Stützle, T., & Hoos, H. H. (2000). MAX–MIN ant system. *Future Generation Computer Systems, 16*(8), 889–914.

Xie, X. F., & Liu, J. (2009). Multiagent optimization system for solving the traveling salesman problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 39*(2), 489–502.

Yang, H., Li, X., Bo, C., & Shao, X. (2006). A graphic clustering algorithm based on MMAS. In *Proceedings of the 2006 IEEE congress on evolutionary computation* (pp. 1592–1597). Canada: Vancouver, BC.

Yi, J., Bi, W., Yang, G., & Tang, Z. (2008). A fast elastic net method for traveling salesman problem. In *Proceedings of the 8th international conference on intelligent systems design and applications* (pp. 462–467). Taiwan: Kaohsiung.