Soft Comput (2012) 16:1707–1722
DOI 10.1007/s00500-012-0855-z

ORIGINAL PAPER

A novel two-stage hybrid swarm intelligence optimization algorithm and application

Wu Deng · Rong Chen · Bing He · Yaqing Liu · Lifeng Yin · Jinghuan Guo

Published online: 5 May 2012

Abstract This paper presents a novel two-stage hybrid swarm intelligence optimization algorithm called GA–PSO–ACO algorithm that combines the evolution ideas of the genetic algorithms, particle swarm optimization and ant colony optimization based on the compensation for solving the traveling salesman problem. In the proposed hybrid algorithm, the whole process is divided into two stages. In the first stage, we make use of the randomicity, rapidity and wholeness of the genetic algorithms and particle swarm optimization to obtain a series of sub-optimal solutions (rough searching) to adjust the initial allocation of pheromone in the ACO. In the second stage, we make use of these advantages of the parallel, positive feedback and high accuracy of solution to implement solving of whole problem (detailed searching). To verify the effectiveness and efficiency of the proposed hybrid algorithm, various scale benchmark problems from TSPLIB are tested to demonstrate the potential of the proposed two-stage hybrid swarm intelligence optimization algorithm. The simulation examples demonstrate that the GA–PSO–ACO algorithm can greatly improve the computing efficiency for solving the TSP and outperforms the Tabu Search, genetic algorithms, particle swarm optimization, ant colony optimization, PS–ACO and other methods in solution quality. And the experimental results demonstrate that convergence is faster and better when the scale of TSP increases.

Keywords Genetic algorithms · Particle swarm optimization · Ant colony optimization · Swarm intelligence · Traveling salesman problem · Two-stage hybrid algorithm

W. Deng (✉) · R. Chen · Y. Liu · J. Guo
Informational Science and Technology Institute,
Dalian Maritime University, 116026 Dalian, China
e-mail: dw7689@gmail.com

W. Deng · L. Yin
Software Institute, Dalian Jiaotong University,
116028 Dalian, China

W. Deng
Key Laboratory of Intelligent Computing and Signal Processing,
Anhui University, Ministry of Education, 230039 Hefei, China

W. Deng
State Key Laboratory of Power Transmission Equipment and System Security and New Technology, College of Electrical Engineering, Chongqing University, 400044 Chongqing, China

W. Deng · Y. Liu
Artificial Intelligence Key Laboratory of Sichuan Province,
Sichuan University of Science and Engineering,
643000 Zigong, China

W. Deng
Key Laboratory of Advanced Design and Intelligent Computing,
Dalian University, Ministry of Education, 116622 Dalian, China

B. He
Foreign Language Institute, Dalian Jiaotong University,
116028 Dalian, China

1 Introduction

The traveling salesman problem (TSP) (Held and Karp 1970) is a well-known combinatorial optimization problem as well as an NP-complete problem. The purpose of TSP is to find a route covering each city once and only once with a minimum route distance. As is well known, the computational complexity of NP-complete problem rises exponentially with the increasing of the number of cities. In recent years, since the TSP is a good ground for testing optimization techniques, many researchers in various fields such as artificial intelligence, biology, mathematics, physics, and operations research devote themselves to trying to find the efficient methods for solving the TSP, such as genetic algorithms (GAs) (Hua and Huang 2006), ant colony optimization (ACO) (Ellabib et al. 2007), simulated annealing (SA) (Lo and Hus 1998), neural networks (NN) (Masutti and de Castro 2009), particle swarm optimization (PSO) (Onwubolu and Clerc 2004), evolutionary algorithms (EA) (Shen and Zhang 2011), memetic computing (Acampora et al. 2011), etc. Besides, there are many practical applications of the TSP in the real world (Marinakis et al. 2010; Mehmet Ali and Kamoun 1993; Banaszak et al. 2009), such as data association, vehicle routing (with the additional constraints of vehicle's route, such as capacity's vehicles), data transmission in computer networks, job scheduling, DNA sequencing, drilling of printed circuits boards, clustering of data arrays, image processing and pattern recognition, analysis of the structure of crystals, transportation and logistics.

In this paper, we propose a new optimization method, called novel two-stage hybrid swarm intelligence optimization algorithm (GA–PSO–ACO), which is based on GAs, PSO and ACO. The proposed GA–PSO–ACO algorithm is divided into two stages. In the first stage, we use the randomicity, rapidity and wholeness of the PSO and GA to obtain a series of sub-optimal solutions (rough searching) through certain iterative times for adjusting the initial allocation of pheromone in ACO. In the second stage, we employ the advantages of the parallel, positive feedback and high accuracy of solution to accomplish solving whole problem (detailed searching) by using the initial allocation of pheromone in ACO in the first stage. We implement the proposed method by using TSP data sets. The GA–PSO–ACO algorithm achieves the better results and faster convergence in solving TSP with large-scale cities from 48 to 33,810.

The rest of this paper is organized as follows. Section 2 briefly describes the TSP. The section expatiates the concepts and characteristics of TSP and the mathematical formulation. Section 3 describes the related hybrid techniques for solving the TSP in recent decades. Section 4 briefly reviews the concepts of the swarm intelligence optimization algorithm, including genetic algorithms, particle swarm optimization, and ant colony optimization. Section 5 presents a new method named novel two-stage hybrid GA–PSO–ACO algorithm. Section 6 illustrates the detailed implementation steps of two-stage hybrid GA–PS–ACO algorithm to solve the TSP. Section 7 compares our experimental results with the recent algorithms that have been used to solve the TSP. Finally, the conclusions are discussed in Sect. 8.

2 Traveling salesmen problem

The TSP has attracted much attention from mathematicians and computer scientists, because it is easy to be described and difficult to be solved. The TSP problem can simply be described as: a search for the shortest closed tour that visits each city once and only once (Hoffman et al. 1985). The TSP can be represented by a complete directed graph $G = (N, A)$, where N is a set of n nodes (vertices), also called cities, and A is a set of arcs and $D = d_{ij}$ is the cost (distance) matrix associated with each $\mathrm{arc}(i,j) \in A$. The cost matrix D can be either symmetric or asymmetric. The TSP is to find a shortest closed tour visiting each of the $n = | N |$ nodes of G exactly once. The distances between the cities are independent of the direction of traversing the arcs, that is, $d_{ij} = d_{ji}$ for every pair of nodes in symmetric TSP. All TSP instances are taken from the TSPLIB benchmark in this paper.

Define the variables:

$$X_{ij} = \begin{cases} 1 & \text{if the arc}(i,j) \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

Objective function:

$$z = \min \sum_i \sum_j d_{ij} x_{ij} \tag{1}$$

The constraints are written as follows:

$$\sum_{i=1}^{n} x_{ij} = 1, \; j = 1, 2, 3, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \; i = 1, 2, 3, \ldots, n \tag{3}$$

$$x_{ij} \in \{0, 1\}, \; i,j = 1, 2, 3, \ldots, n \tag{4}$$

$$\sum_{i,j \in S}^{n} x_{ij} \le |S| - 1, 2 \le |S| \le N - 2 \tag{5}$$

In these formulations, Eq. (1) describes the total cost to be minimized. The Eqs. (2–5) are constraints' condition. Constraint (2) ensures that each position j is occupied by only one city, while constraint (3) guarantees that each city i is assigned one exact position. Constraint (4) describes the integrality constraints of variables zero–one $x_{ij}\,(x_{ij} \ge 0)$. Constraint (5) assures that each city in the final route will be visited once and that no sub-routes will be formed.

3 Related works

Swarm intelligence is an important research topic based on the collective behavior of decentralized and self-organized systems in computational intelligence. It consists of a population which simulates the animals' behavior in the real world. Now there are many swarm intelligence optimization algorithms, such as genetic algorithms, particle swarm optimization, ant colony optimization, bee colony algorithm, differential evolution, fish-warm algorithm,…,etc. In these swarm intelligence algorithms, the most popular and most widely used algorithms for solving the TSP are GAs, ACO and PSO. The parallelized technique and the idea of protected chromosomes are presented to reduce the searching space to speed up the processing time of GAs for solving the TSP (Adachi and Yoshida 1995). A genetic algorithm with an iterated local search capability is presented to further improve the solution quality for solving the TSP (Tasgetiren 2007). A technique that uses the Wang recurrent neural network with the "Winner Takes All" principle is presented to solve the TSP (Paulo et al. 2007). A genetic algorithm with reinforcement learning is presented to solve the TSP (Liu and Zeng 2009). A new mutation operator has been developed to increase genetic algorithm performance to find the shortest distance in the traveling salesman problem (Albayrak and Allahverdi 2011). Particle swarm optimization is presented to solve traveling salesman problem (Wang et al. 2003). Preserving diversity in particle swarm optimization is used to solve traveling salesman problem (Hendtlass 2003). Particle swarm optimization-based algorithms is presented for TSP and generalized TSP (Shi et al. 2007). A hybrid multi-swarm particle swarm optimization algorithm is presented to solve the probabilistic traveling salesman problem (Yannis and Magdalene 2010). Some improved ACO methods have been proposed, such as the ant colony system (Dorigo and Gambardella 1997), the MAX–MIN ant system (Stützle and Hoos 2000), the rank-based ant system (Bullnheimer et al. 1997), modified ACS with time windows (Cheng and Mao 2007; Sarhadi and Ghoseiri 2010), the KCC-Ants (Naimi and Taherinejad 2009), and memetic ant colony optimization algorithm (Mavrovouniotis and Yang 2011).

With the rapid development of the social economy, science and technology, the large-scale optimization problems are becoming too complicated to obtain satisfactory results by using the single swarm intelligence optimization algorithm. So it is necessary to utilize hybrid swarm intelligence optimization algorithm to solve all kinds of the complex large-scale optimization problems. The hybrid swarm intelligence optimization algorithm is to utilize the single swarm intelligence optimization algorithm of complementary advantages and the value-added information to overcome the insufficiencies of the single swarm intelligence optimization algorithm in order to enhance the efficiency of solving the complex large-scale optimization problems. Many researchers presented some personalization hybrid swarm intelligence optimization algorithms according to the practical problems on the applications of TSP in the real world in various fields, such as artificial intelligence, biology, mathematics, physics, and operations research. A new hybrid heuristic approach named ACOMAC algorithm is presented to solve the traveling salesman problem (Tsai et al. 2004). A new hybrid technique is presented for the optimization of large-domain electromagnetic problems (Grimaldi et al. 2005). A hybrid optimization method based on the ant colony and clonal selection principles is presented for a few benchmark optimization problems (Wang et al. 2007). A hybrid method based on combining two heuristic optimization techniques, genetic algorithms and particle swarm optimization is presented for the global optimization of multimodal functions (Kao and Zahara 2008). A hybrid method based on Nelder–Mead simplex search and particle swarm optimization is presented for constrained engineering design problems (Zahara and Kao 2009). A hybrid approach combining an improved genetic algorithm and optimization strategies is presented for the asymmetric TSP (Xing et al. 2008). A hybrid ant colony algorithm is presented for path planning in sparse graphs (Lim et al. 2008). A new grouping genetic algorithm approach is presented to solve the multiple TSP (Singh and Baghel 2009). A hybrid swarm intelligence algorithm is presented for the traveling salesman problem (Kuo et al. 2010). An efficient method based on hybrid genetic algorithm–particle swarm optimization (GA–PSO) is presented for various types of economic dispatch (ED) problem (Guvenc et al. 2011). A parallelized genetic ant colony system is presented for solving the TSP (Chen and Chien 2011). A hybrid PS–ACO algorithm is presented for solving the traveling salesman problem (Shuang et al. 2011). A method based on an adaptive simulated annealing algorithm with greedy search is presented for solving the traveling salesman problem (Geng et al. 2011).

Although these hybrid swarm intelligence optimization algorithms are widely used for solving the TSP, some insufficiencies exist for complex large-scale optimization problems, such as the long time, the slow premature convergence.

4 Swarm intelligence optimization algorithm

4.1 Genetic algorithms

Genetic algorithms (GAs) is a class of population-based stochastic search technique that solves problems by imitating processes observed during natural evolution. It is based on the principle of the survival and reproduction of the fitness. GAs continually exploit new and better solutions without any pre-assumptions, such as continuity and unimodality. GAs provided the parallel iterative algorithm with certain learning ability, which repeats evaluation, selection, crossover and mutation after initialization until the stopping criteria are reached (Holland 1975). It has been widely applied to function optimization, multi-objective optimization, TSP, and so on.

In the GAs, a population of candidate solutions is evolved at first. Each solution in candidate solutions is encoded as a binary string (chromosome). A performance function is used to evaluate the fitness value. For each iteration, a predetermined individuals' number will correspondingly produce fitness values associated with the chromosomes (Kao and Zahara 2008). A real-coded GAs is a genetic algorithm representation that uses a vector of floating-point numbers instead of 0s and 1s for implementing encoding of chromosome. With some modifications of the genetic operators, the real-coded GAs perform better than the binary-coded GAs for TSP. The crossover operator of a real-coded GAs is performed by the borrowing concept of convex combination. The random mutation operator is used to change the gene with a random number in the problem's domain (Fan et al. 2006; Chu et al. 2008).

Assume that we employ GAs to search for the largest fitness value with a given fitness function (Deng et al. 2012), shown in Fig. 1.

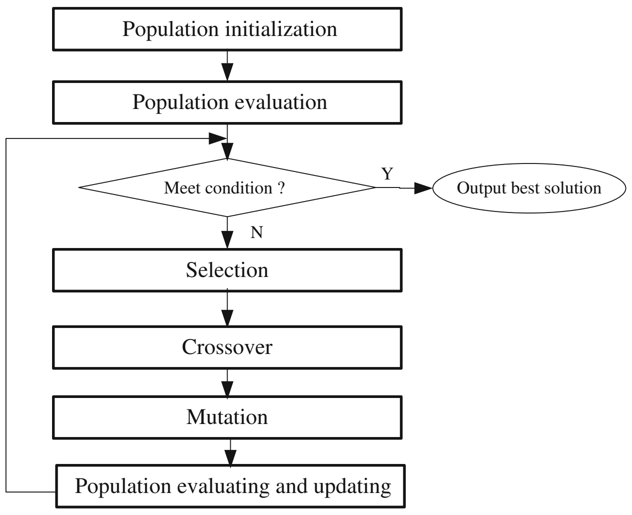

Fig. 1 Searching procedure of the GAs

4.2 Particle swarm optimization

Particle swarm optimization (PSO) is inspired by social behavior simulation, was originally designed and developed by Eberhart and Kennedy (1995). The PSO is a population-based search algorithm developed on basis of the simulation of the social behavior of birds within a flock. In the PSO, individuals are particles and are "flown" through hyperdimensional search space. They simulated birds' swarm behavior and made each particle in the swarm move according to its experience and the best experience of particle. Each particle represents a potential solution to the problem and searches around in a multi-dimensional search space. The PSO consists of a number of individuals which are denoted as particles. Each particle has a position and a velocity. Each particle is provided with a memory function and adjusts its trajectory according to the best-visited position and the global best position in the whole swarm. The PSO uses a fitness evaluation function to take each particle's position and set its fitness value in the course of optimization problem. The position of the individual's position of best fitness value is called the local best (*lbest*). The position of highest fitness value visited by the swarm is called the global best (*gbest*). In a D-dimensional research space, each particle is treated as a point. The best previous position of particle in the swarm is described as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$. The rate of the velocity for particle i is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^T$. The particle's new velocity and position is updated by the following equation (Assareh et al. 2010; Deng et al. 2011):

$$v_{id}^{k+1} = w \times v_{id}^{k} + c_1 \times rand_1 \times (pbest_{id} - x_{id}^{k}) + c_2 \times rand_2 \times (gbest_{gd} - x_{id}^{k}) \tag{6}$$

$$x_{id}^{t+1} = x_{id}^{t} + v^{t+1} \tag{7}$$

$$w = w_{\max} - (w_{\max} - w_{\min}) \times I/I_{\max} \tag{8}$$

where v_{id}^{k+1} and x_{id}^{t+1} are the velocity and position of the particle i at iterations d. The acceleration constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. In general, the value of c_1 and c_2 is [0, 4]. The $rand_1$ and $rand_2$ are random numbers uniformly distributed in [0,1] which denote remembrance ability for study. w is the inertia weight, $w_{\max}$ is the initial inertia weight of the velocity, $w_{\min}$ is the final inertia weight of the velocity. I is the current iteration times, $I_{\max}$ is the total iteration times.

4.3 Ant colony optimization (ACO)

Ant colony optimization (ACO) was introduced by Marco Dorigo (Colorni et al. 1991). The ACO is a metaheuristic inspired by the found food behavior of real ants with the shortest path. When ants move, ants will leave a chemical pheromone trail on the ground. Ants tend to choose the paths marked by the strongest pheromone concentration. The indirect communication of the ants is to pheromone trails in order to enable them to find shortest paths between their nest and food. The ACO algorithm is an essential system based on agents that simulates the natural cooperation and adaptation behavior of ants. The ACO simulates the method of real ants to rapidly establish the shortest route from a food source to their nest (Colorni et al. 1991). The idea of the ACO is to model the problem that is being solved, just as the search for a minimum cost path in a graph that uses artificial ants to search for good paths. The ACO consists of a number of cycles (iterations) of solution construction. A number of ants construct complete solutions by using heuristic information and the collected experiences of previous groups of ants in each iteration. These collected experiences are represented by using the pheromone trail which is deposited on the constituent elements of a solution (Colorni et al. 1991). Small quantities are deposited during the construction phase, while larger amounts are deposited at the end of each iteration in proportion to solution quality.

In the ACO algorithm, the ACO simulates the optimization of found food behavior of ant (Kaveh and Talatahari 2009). The ACO procedure is illustrated in Fig. 2.

In the ACO, we define a list of nodes which the kth ant cannot choose as the next node. This list is called **Tabu*k***, which includes all the customer nodes that have been visited by the kth ant until the current state in addition to all the depots except the one, which the current tour has been started from (Niknam et al. 2005). Assuming that there are n cities and m ants, at the same time assuming that the initial intensity of pheromone on each edge is set to a very small non-zero positive constant τ_0. In each cycle, each ant starts at a stochastic chosen city, then visits the other cities once and only once according to the transition rule based on the initial intensity of pheromone. When the ants complete the routes of one cycle, the length of one cycle will be computed. Then, the intensity of pheromone will be updated by using the pheromone update rule. The procedure of pheromone update rule is shown as follows (Chen and Chien 2011):

4.3.1 The transition rule

In the route, the kth ant starts from city r, the next city s is selected among the unvisited cities memorized in J_r^k according to the following formula (Chen and Chien 2011):

$$s = \arg\max_{u \in J_r^k} \left[\tau_i(r,u)^{\alpha} \cdot \eta(r,u)^{\beta}\right] \quad \text{if } q \le q_0 (\text{Exploitation}) \tag{9}$$

or visit the next city s with the probability $p_k(r,s)$,

$$p_k(r,s) = \begin{cases} \dfrac{\tau(r,s)^{\alpha} \cdot \eta(r,s)^{\beta}}{\sum_{u \in J_r^k} \tau(r,u)^{\alpha} \cdot \eta(r,u)^{\beta}} & \text{if } s \in J_r^k \\ 0 & \text{otherwise} \quad \text{if } q > q_0 \text{ (Bias exploitation)} \end{cases} \tag{10}$$

In formula (9 or 10), $p_k(r,s)$ is the transition probability (from city r to s for the kth ant in the ith group),$\tau(r,u)$ is the intensity of pheromone between city r and city u in the ith group, $\eta(r,u)$ is the length of the path between cities from city r to city u, J_r^k is the set of unvisited cities of the kth ant in the ith group, the parameter α and β are the control parameters for determining the weight of the trail intensity and the length of the path, q is a uniform probability randomly chosen value in [0, 1], and q_0 is a parameter between 0 and 1 and the higher q_0 the smaller the probability to make a random choice.

4.3.2 The pheromone update rule

In order to improve the future solutions, the pheromone trails of the ants must be updated to reflect the ants' performance and the quality of the solutions. An ant deposits pheromone trails on the arcs, it travels to update the pheromone trails. Trail updating includes local updating of trails after individual solutions have been generated and global updating of the best solution route after a predetermined number of solutions m has been accomplished

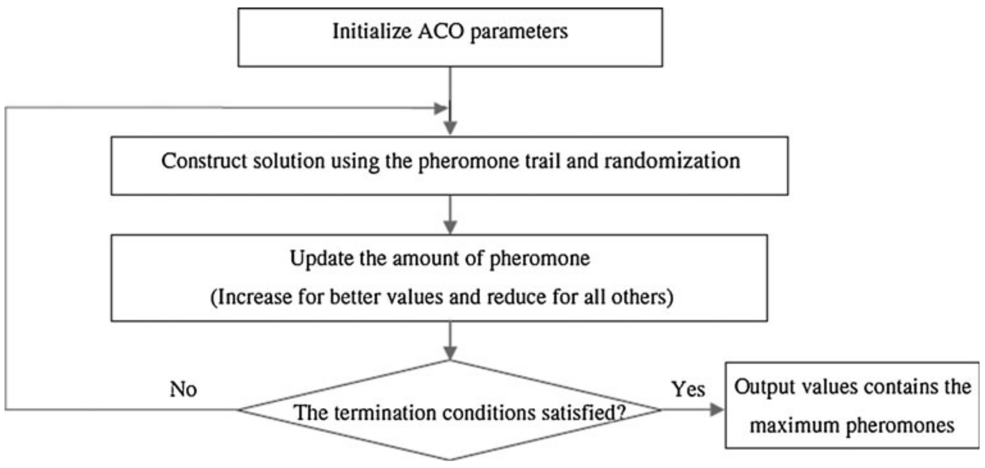
Initialize ACO parameters
Construct solution using the pheromone trail and randomization
Update the amount of pheromone
(Increase for better values and reduce for all others)
No
The termination conditions satisfied?
Yes
Output values contains the maximum pheromones

Fig. 2 Searching procedure of the ACO

(Taher and Babak 2010). This is done with the following local trail updating formula (Chen and Chien 2011):

$$\tau(r,u) = (1-\rho)\tau(r,s) + \sum_{k=1}^{m} \Delta\tau_k(r,s) \quad (11)$$

in the formula (11), $\rho\,(0<\rho<1)$ is the pheromone trial evaporation rate. $\Delta\tau_k(r,s)$ is the amount of pheromone trail added to the edge (r, s) by ant k between time t and $t+\Delta t$ in the tour. It is given by:

$$\Delta\tau_k(r,s) = \begin{cases} \frac{Q}{L_k} & (r,s)\in\pi_k \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

where Q is a constant parameter, L_k is the distance of the sequence π_k toured by ant in Δt.

This updating rule shows that the increments of pheromone increments only relate to the current search of the ant colony, which means that history experience is ignored and the valuable solutions have not been reinforced. This process will be repeated for a predetermined number of iterations. The best solution among all of the iterations is presented as an output of the ACO which should represent a good approximation of the optimal solution for the problem.

5 The framework of hybrid GA–PSO–ACO algorithm

As mentioned in the previous sections, researchers confirm that the PSO algorithm should be taken into account as a powerful technique for handling various kinds of optimization problems. So it is based on social adaptation of knowledge for working, and all individuals in population are considered to be the same generation. On the contrary, the GAs is based on evolution from generation to generation for working, so the individuals' changes are not considered in one single generation. The PSO and GAs have similar parallel characteristics in their interior, but have respective advantages when used to solve different optimization problems by simulation experiment and practical application. The ACO uses pheromone as an indirect communication medium among the individuals (ants) in a colony, and the converging procedure is a dynamic positive feedback of pheromone to the global optimum. Although the ACO algorithm is useful for discovering near-optimal solutions for some optimization problems, its feedback is that it takes long time to find such results and premature convergence in order to make it feasible for large-scale problems. In order to make full use of their respective excellent features, we propose a novel two-stage hybrid GA–PSO–CO based on the compensation by combining the evolution ideas of the GAs, PSO and ACO in this paper. The infrastructure and basic principle of the proposed hybrid algorithm is shown in Fig. 3.

The idea of the proposed hybrid algorithm is divided into two stages. In the first stage, we make use of the randomicity, rapidity, wholeness and parallels of the PSO and GAs to obtain a series of sub-optimal solutions (rough searching) through certain iterative times. And then these sub-optimal solutions are used to adjust the initial allocation of pheromone in the ACO. In the second stage, we make use of these advantages of the parallel, positive feedback and high accuracy of solution to implement solving of whole problem (detailed searching) by using the initial allocation of pheromone in the ACO.

6 The hybrid GA–PSO–ACO algorithm for the TSP

In the proposed hybrid GA–PSO–ACO algorithm, the hybrid algorithm is initialized by a population of random solutions and searches for the optimization solution by the search space. During this course, an evolution of this solution is performed by integrating the GAs, PSO and ACO. The proposed hybrid algorithm is presented as follows:

Step1: initialization
Initialize randomly the individuals of the population according to the limit of each unit including individual dimensions, searching points, and velocities. These individuals must be feasible candidate solutions that satisfy the operation constraints. For N-dimensional problem, the size of population is $4N$ individuals which are randomly generated. These individuals are regarded as chromosomes in the GAs and particles in the PSO.
Step 2: evaluation
The evaluation function f (called fitness) must be defined for evaluating each individual's fitness value. For emphasizing the best chromosome and faster convergence of the iteration process, the evaluation value is normalized in the range [0, 1]. The $4N$ individuals are sorted according to the evaluated fitness value. Then the population is divided into two subgroups with the equal individuals according to the sorted individuals. For an actual problem, a fitness function will be selected at first.
Step 3: selection operation in the GAs
Selection operation is to select two parent strings for generating new strings. In this paper, the top $2N$ individuals are selected to construct one subgroup. Parent strings are selected according to the fitness value of string.
Step 4: crossover operation in the GAs
Crossover operation is to exchange the position of two encoded strings in order to obtain next-generation encoded string. The two-point crossover method is to

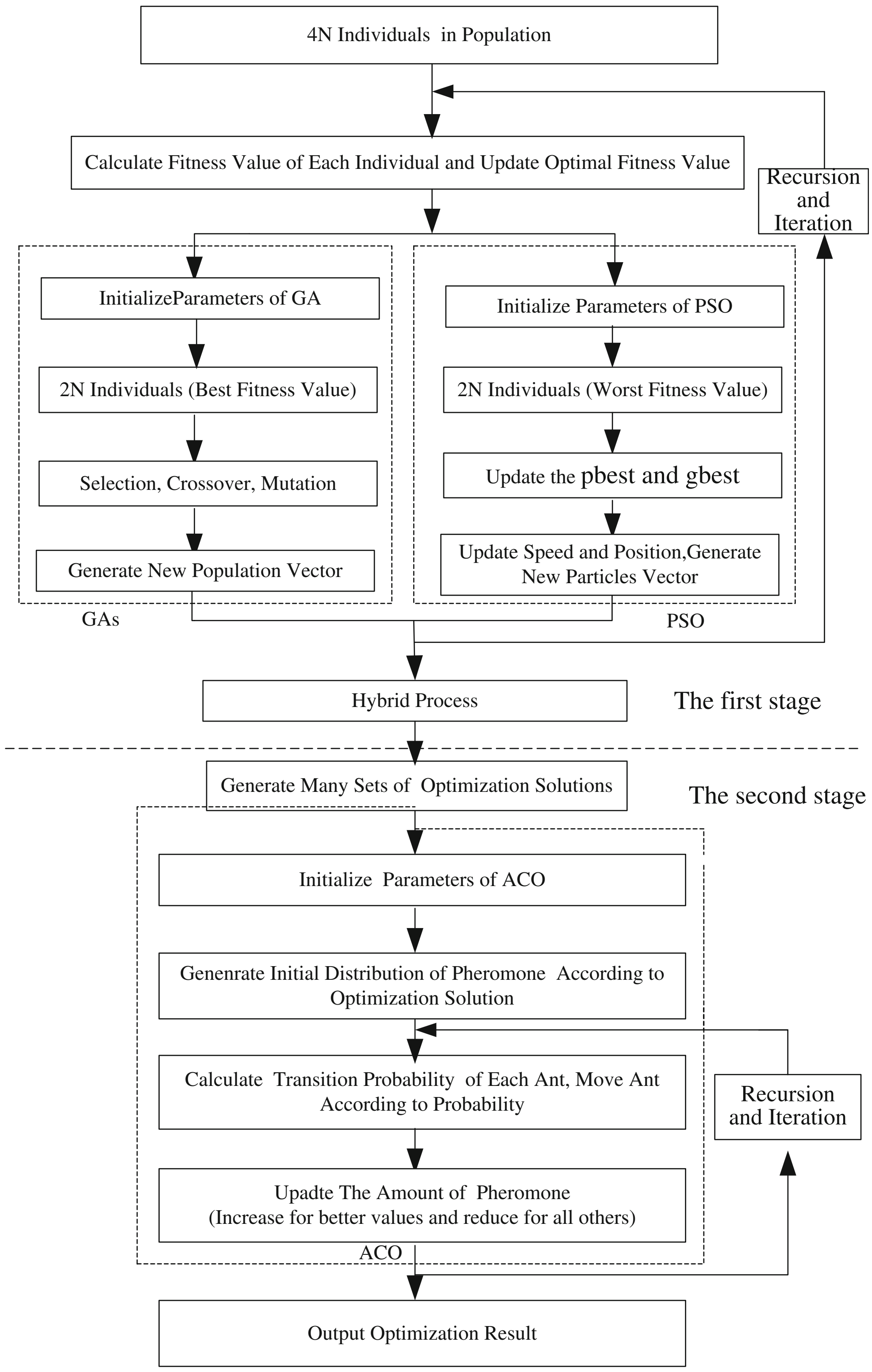
4N Individuals in Population
Calculate Fitness Value of Each Individual and Update Optimal Fitness Value
Recursion and Iteration
InitializeParameters of GA
2N Individuals (Best Fitness Value)
Selection, Crossover, Mutation
Generate New Population Vector
GAs
Initialize Parameters of PSO
2N Individuals (Worst Fitness Value)
Update the pbest and gbest
Update Speed and Position,Generate New Particles Vector
PSO
Hybrid Process
The first stage
Generate Many Sets of Optimization Solutions
The second stage
Initialize Parameters of ACO
Genenrate Initial Distribution of Pheromone According to Optimization Solution
Calculate Transition Probability of Each Ant, Move Ant According to Probability
Recursion and Iteration
Upadte The Amount of Pheromone (Increase for better values and reduce for all others)
ACO
Output Optimization Result

Fig. 3 The framework of two-stage hybrid GA–PS–ACO algorithm

set two crossover points at random, take a section between the points from one parent string and other sections outside the points from the other parent string, then recombine them. The selected genes consist of a substring. The top 2*N* individuals are matched, respectively, by pairs and crossed randomly according to the crossover probability (Pc = 100 %). The crossover operation method is shown as:

$$x_i^{'} = p_i x_i + (1 - p_i) x_{i+1} \quad i = 1, 2, 3, \ldots, 2N - 1 \tag{13}$$

$$x_i^{'} = p_i x_i + (1 - p_i) x_1 \quad i = 2N \tag{14}$$

In formula (13) and (14), p_i denotes the proportional distributed random number between 0 and 1. x_i denotes the position of the *i*th particle.
Step 5: mutation operation in the GAs
Mutation operation is to change elements of one string which came from the crossover operation. In this paper, the top 2*N* individuals are implemented mutation operation with one mutation probability (Pm = 20%). Inversion mutation method is adopted to implement the mutation operation. The mutation formula is shown as:

$$x_k^{'} = x_k + rand \times N(0, 1) \tag{15}$$

where x_i denotes the position of the *i*th chromosome.
Step 6: update speed in the PSO
The bottom 2*N* individuals are fed into PSO method to create 2*N* new individuals by selection of the global best particle and the neighborhood best particles for updating the velocity and position. The global best particle of the population is determined according to the sorted fitness values. The neighborhood best particles are selected by first evenly dividing the 2*N* particles into *N* neighborhoods and assigning the particle with the better fitness value in each neighborhood as the neighborhood best particle. The velocity of the particle is updated according to the Eq. (6).
Step 7: update position in the PSO
The position of the particle is updated according to the Eq. (7).
Step 8: update the local best value (*pbest*) in the PSO
For each particle, its fitness value is compared with the best position particle fitness value. If the fitness value of the best position particle is good, the fitness value of the best position is regarded as the current best position and *pbest* value is updated.
Step 9: update the global best particle (*gbest*) in the PSO
For each particle, its fitness value is compared with the best global position particle fitness value. If the fitness value of the best global position particle is good, the fitness value of best global position is regarded as the current best global position and *gbest* value is updated.
Step 10: hybrid process
The best 4*N* individuals are selected from next generation by the PSO and GAs.
Step 11: generate a series of optimization solutions
The obtained fitness degrees are compared in order to obtain the global optimal value *gbest*. If the result does not meet the termination criteria or reach iteration times, the step 2 to step 10 will be repeated until the current iteration reaches the predetermined maximum iteration.
Step 12: initialize parameters of the ACO
Generate *g* groups of ants (G0, G1… and Gg), where each group has *N* ants and each ant will choose a city as its starting city. According to the TSP, the position and initial allocation of pheromone is renewedly set in the ACO. The position of the *k*th ant of the *i*th group is corresponding to the optimal position of PSO or GA. The initial pheromone level between any two cities is set $T(i)$ according to the following equation:

$$T(i) = k \cdot a^{-f(X_i)} \tag{16}$$

where k is a constant ($k > 0$), $0 < \mathrm{a} < 1$, $f(X_i)$ is the value of the objective function. The values of the *a* and *k* are according to the actual problem.
Step 13: calculate transition probability of each ant according to probability
The *k*th ant of the *i*th group constructs its traveling sequence of cities (from city *i*, the next city *j*) using the following transition rule(x).
Step 14
Compute the length of the path traveled by each ant.
Step 15: update the amount of pheromone
The *k*th ant of the *i*th group allocates its traveling sequence of cities (from city *i*, the next city *j*) using the local pheromone update rule(y) according to the length of its path.
Step 16: output optimization result
Compute whether a better solution is obtained in this time step than the last; if so, then perform a global update on the solution and empty the Tabu value; repeat step 13 to step16.

7 Experiment simulation and analysis

In order to demonstrate the effectiveness and performance of the proposed algorithm, we have implemented the proposed algorithm using MATLAB 2009a on a 2.5G Pentium (R) Dual-Core CPU E5200 PC with 35 datasets obtained from TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/). According to TSPLIB, the distance between any two cities is calculated by the Euclidian distance and then rounded off after the decimal point in this paper.

7.1 Parameters configuration

In this paper, the parameters of these algorithms are selected after thorough testing. A number of different alternative values were tested for all instances (we started with some classic values that have already been used in other studies papers, and then we modified these values until the selected values are chosen). The selected ones are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. Thus, the selected parameters for the GAs, PAO, ACO, PS–ACO and GA–PSO–ACO algorithms are shown in Table 1 (Shuang et al. 2011). The first column represents the parameter name. The second column represents the parameters values of the GAs. The third column represents the parameters values of the PSO. The fourth column represents the parameters values of the ACO. The fifth column represents the parameters values of PS–ACO. The sixth column represents the parameters values of the GA–PSO–ACO.

7.2 Experimental results and analysis

In this subsection, the Tabu Search, GAs, PSO, ACO and PS–ACO are performed on 9 TSP benchmark instances and GA–PSO–ACO algorithm is performed on 35 TSP benchmark instances from TSPLIB with cities scale from 48 to 33,810. We coded each algorithm in Matlab language. The comparison to be performed here will take into account the total cost of the solutions found by the algorithms, the computational cost of each algorithm, and the percentage deviation of the solutions found in relation to the best known solutions. All algorithms were run 20 times for each instance and the results presented include the best, worst and average solutions found. The number of cities in each instance is the number following the letters that name the instances, for example, in eil51 the number of cities is 51. One thousand generations were evolved each time. The simulated experimental results are listed in Tables 2, 3, 4, 5, 6, 7, 8, 9 and 10. In these tables, optimal solution denotes the optimal tour length as reported in TSPLIB. Best denotes the best solution found by each algorithm. Worst denotes the worst solution found by each algorithm. Average denotes the average value of the total run solutions. Error denotes the percent difference of the solution. A better algorithm is considered to be those whose values of best, average and error are smaller than those of other algorithms.

The results of the GA–PSO–ACO algorithm are also compared with the results of a number of implementations of the Tabu Search, GAs, PSO, ACO and PS–ACO in Tables 2, 3, 4, 5, 6, 7, 8, 9 and 10. In these implementations, the same instances are used as in this paper and comparisons of the results can be performed. It should be noted that the comparisons are based on quality of the results. As it can be observed that the values of best, average of the GA–PSO–ACO algorithm are the best among six algorithms for all experiments. We can also see that the ability to improve the values of average and error of the GA–PSO–ACO algorithm is the best among the six algorithms.

To verify the effectiveness and efficiency of the proposed algorithm, the GA–PSO–ACO algorithm is performed on 35 TSP benchmark instances from TSPLIB with cities scale from 48 to 33,810. The GA–PSO–ACO is executed on each TSP instances with 20 runs, and the results are listed in Table 11.

As can be seen in Table 11, for the 35 TSP instances with our algorithm GA–PSO–ACO, the experiment values are close to the optimal solution of tour. In addition, for TSP instances eil51, the GA–PSO–ACO algorithm can find the best known solutions 426. Particularly, for TSP

Table 1 The parameters setting used for GA, PSO, ACO, PS–ACO and GA–PSO–ACO algorithms

Parameter	GAs	PSO	ACO	PS–ACO	GA–PSO–ACO
Population size	100	100	100	100	100
Iteration times	1,000	1,000	1,000	1,000	1,000
Inertia value	N/A	N/A	N/A	N/A	0.80
Maximum velocity	N/A	0.50	N/A	0.50	0.50
Learn factor	N/A	$c_1 = c_2 = 2$	N/A	$c_1 = c_2 = 2$	$c_1 = c_2 = 2$
Crossover operator	Single point	N/A	N/A	N/A	Single point
Crossover rate	0.90	N/A	N/A	N/A	0.90
Mutation operator	Real value	N/A	N/A	N/A	Real value
Mutation rate	0.01	N/A	N/A	N/A	0.01
β	N/A	N/A	2.0	2.0	2.0
Evaporation coefficient	N/A	N/A	0.05	0.05	0.05
Q	N/A	N/A	100	100	100
q_0	N/A	N/A	0.90	0.90	0.90
R_0 for crossover strategy	N/A	N/A	0.33	0.33	0.33
Proportion of GA–PSO	N/A	N/A	N/A	N/A	0.50

Table 2 The comparisons for att48

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	48	33,522	34,198	35,886	34,978	676	2.017
GAs	48	33,522	34,572	36,299	35,002	1,050	3.132
PSO	48	33,522	34,759	37,672	36,179	1,237	3.690
ACO	48	33,522	34,357	35,197	34,460	835	2.491
PS–ACO	48	33,522	33,641	34,730	33,956	119	0.354
GA–PSO–ACO	48	33,522	**33,524**	34,164	33,662	2	0.006

Table 3 The comparisons for eil51

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	51	426	445.44	472.76	459.32	19.44	4.563
GAs	51	426	446.96	537.67	481.07	20.96	4.920
PSO	51	426	447.51	454.99	446.53	21.51	5.049
ACO	51	426	436.85	454.99	446.53	10.85	2.547
PS–ACO	51	426	427.40	442.51	433.09	1.74	0.408
GA–PSO–ACO	51	426	**426**	436.20	431.83	0.00	0.000

Table 4 The comparisons for berlin52

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	52	7,542	7,976.84	8,286.68	8,014.60	434.84	5.766
GAs	52	7,542	8,201.17	8,443.02	8,376.55	659.17	8.740
PSO	52	7,542	8,197.79	8,589.31	8,319.51	655.79	8.695
ACO	52	7,542	7,647.55	7,780.57	7,732.31	105.55	1.399
PS–ACO	52	7,542	7,568.54	7,618.31	7,586.42	26.54	0.352
GA–PSO–ACO	52	7,542	**7,544.37**	7,544.37	7,544.37	2.37	0.031

Table 5 The comparisons for st70

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	70	675	702.27	738.45	718.56	27.56	4.083
GAs	70	675	715.43	744.31	731.72	40.43	5.990
PSO	70	675	720.41	753.29	741.09	45.41	6.727
ACO	70	675	697.76	716.83	705.58	22.76	3.372
PS–ACO	70	675	684.16	710.47	698.75	9.16	1.357
GA–PSO–ACO	70	675	**679.60**	704.25	694.60	4.60	0.681

Table 6 The comparisons for pr76

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	76	108,159	110,941	130,637	122,104	2,782	2.572
GAs	76	108,159	115,329	124,851	120,245	7,170	6.629
PSO	76	108,159	118,038	126,583	122,735	9,879	9.134
ACO	76	108,159	110,517	120,922	114,964	2,358	2.180
PS–ACO	76	108,159	109,244	113,120	110,162	1,085	1.003
GA–PSO–ACO	76	108,159	**109,206**	112,443	110,023	1,074	0.968

Table 7 The comparisons for eil101

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	101	629	667.43	709.11	685.49	38.43	6.110
GAs	101	629	682.37	745.33	706.25	53.37	8.485
PSO	101	629	687.32	779.11	731.58	58.32	9.272
ACO	101	629	649.87	695.18	664.07	20.87	3.318
PS–ACO	101	629	637.65	674.07	651.36	8.65	1.375
GA–PSO–ACO	101	629	**633.07**	641.17	637.93	4.07	0.647

Table 8 The comparisons for kroA200

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	200	29,368	31,289	33,438	32,219	1,921	6.541
GAs	200	29,368	32,261	34,572	33,158	2,893	9.851
PSO	200	29,368	32,350	34,526	33,132	2,982	10.154
ACO	200	29,368	31,669	33,839	32,434	2,301	7.835
PS–ACO	200	29,368	30,190	33,626	31,927	822	2.799
GA–PSO–ACO	200	29,368	**29,731**	33,228	31,015	363	1.221

Table 9 The comparisons for rat783

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	783	8,806	9,185	9,407	9,294	397	4.304
GAs	783	8,806	9,217	9,423	9,315	411	4.667
PSO	783	8,806	9,316	9,516	9,423	510	5.792
ACO	783	8,806	9,093	9,403	9,246	287	3.259
PS–ACO	783	8,806	9,041	9,387	9,177	235	2.669
GA–PSO–ACO	783	8,806	**9,030**	9,316	9,126	224	2.545

Table 10 The comparisons for pr1002

Algorithm	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
Tabu Search	1,002	259,045	267,617	269,419	268,715	8,572	3.309
GAs	1,002	259,045	274,124	278,547	276,812	15,079	5.821
PSO	1,002	259,045	278,476	282,591	280,738	19,431	7.501
ACO	1,002	259,045	268,502	269,859	268,934	9,457	3.651
PS–ACO	1,002	259,045	266,213	268,549	267,576	7,168	2.767
GA–PSO–ACO	1,002	259,045	265,987	268,512	266,774	6,942	2.680

instances att48, berlin52 and eil101, the new best known solutions 33,524, 7,544.37 and 633.07 are approaching to the best known solutions 33,522, 7,542 and 629.

We select 14 TSP instances from using 35 TSP instances in order to validate the effectiveness of the proposed method. Table 12 is a comparison of the experimental results of the proposed method with other method [Somhom's method (Somhom et al. 1997), Cochrane's method (Cochrane and Beasley 2003), Masutti's method (Masutti and de Castro 2009)]. PDbest is the percentage deviation of the found best solution. PDav is the percentage deviation of the found average solution. From Table 12, we can see that the percentage deviations of the found best solution of the GA–PSO–ACO method are better than other method for the data sets eil51, rad100, eil101, ch130, kroA150, lin318, rat575, rat783, d1655. The percentage deviations of the found average solution of the GA–PSO–ACO method are better than other method for the data sets eil51, berlin52, eil76, rad100, kroD100, eil101, ch130, kroA150, lin318, rat575, rat783, d1655. So Table 12 shows the proposed

Table 11 Results of GA–PSO–ACO for 35 TSP benchmark instances from TSPLIB

No.	Instances	Scale	Optimal solution	Best	Worst	Average	Difference	Error (%)
1	att48	48	33,522	33,524	34,164	33,662	2	0.006
2	eil51	51	426	426	434.20	431.84	0	0.000
3	berlin52	52	7,542	7,544.37	7,544.37	7,544.37	2.37	0.031
4	st70	70	675	679.60	704.25	694.60	4.60	0.681
5	eil76	76	538	545.39	555.98	550.16	7.39	1.373
6	pr76	76	108,159	109,206	112,443	110,023	1,074	0.968
7	rat 99	99	1,211	1,218	1,307	1,275	7	0.575
8	rad100	100	7,910	7,936	8,115	8,039	26	0.329
9	kroD100	100	21,294	21,394	21,753	21,484	100	0.470
10	eil101	101	629	633.07	641.17	637.93	4.07	0.647
11	lin105	105	14,379	14,397	14,926	14,521	18	0.125
12	pr107	107	44,303	44,316	44,981	44,589	13	0.029
13	pr124	124	59,030	59,051	61,259	60,157	21	0.036
14	bier127	127	118,282	118,476	12,273	120,301	194	0.164
15	ch130	130	6,110	6,121.15	6,317.53	6,203.47	11.15	0.183
16	pr144	144	58,537	58,595	58,753	58,662	58	0.099
17	kroA150	150	26,524	26,676	26,904	26,803	152	0.573
18	pr152	152	73,682	73,861	74,147	73,989	179	0.243
19	u159	159	42,080	42,395	42,673	42,506	315	0.749
20	rat195	195	2,323	2,341	2,387	2,362	18	0.775
21	kroA200	200	29,368	29,731	33,228	31,015	363	1.221
22	gil262	262	2,378	2,399	2,478	2,439	21	0.883
23	pr299	299	48,191	48,662	48,965	48,763	471	0.978
24	lin318	318	42,029	42,633	42,857	42,771	604	1.438
25	rd400	400	15,281	15,464	15,548	15,503	183	1.197
26	pcb442	442	50,778	51,414	51,538	51,494	626	1.252
27	rat575	575	6,773	6,912	6,983	6,952	137	2.047
28	u724	724	41,910	42,657	42,759	42,713	747	1.783
29	rat783	783	8,806	9,030	9,316	9,126	224	2.545
30	pr1002	1,002	259,045	265,987	268,512	266,774	6,942	2.680
31	d1291	1,291	50,801	52,378	52,562	52,443	1,577	3.104
32	d1655	1,655	62,128	64,401	65,954	65,241	2,273	3.658
33	nl4461	4,461	182,566	18,933	198,741	192,574	6,764	3.705
34	brd14051	14,051	469,385	490,432	512,592	503,560	21,047	4.484
35	pla33810	33,810	66,048,945	70,299,195	75,858,356	72,420,147	4,250,250	6.435

GA–PSO–ACO method is better than other method in the mass for each TSP dataset.

Figures 4 and 5 are two comparisons of the percentage deviations of the best solution and the average solution to the best known solution for different methods. As can be seen in Figs. 4, 5, the proposed GA–PSO–ACO method obtains smaller percentage deviations than the other presented methods, such as Somhom's method (Somhom et al. 1997), Cochrane's method (Cochrane and Beasley 2003), Masutti's method (Masutti and de Castro 2009). Figure 6 illustrates some of the best routes found by the GA–PSO–ACO for TSP and their costs (their route lengths). Note that the way the network grows, like an expanding ring, reduces the possibility of crossings in the routes, which are characteristic of locally optimal routes.

8 Conclusion

In this paper, we have presented a novel two-stage hybrid swarm intelligence optimization algorithm (genetic algorithms and particle swarm optimization with ant colony optimization) named GA–PSO–ACO algorithm for the

Table 12 The comparison of the experimental results of the proposed method with other method

Instances	Optimal solution	Somhom's (Somhom et al. 1997)		Cochrane's (Cochrane and Beasley 2003)		Masutti's (Masutti and de Castro 2009)		The proposed method	
		PDbest	PDav	PDbest	PDav	PDbest	PDav	PDbest	PDav
eil51	426	1.64	3.43	0.94	2.89	0.23	2.69	0.00	1.37
berlin52	7,542	0.00	5.81	0.00	7.01	0.00	5.18	0.03	0.03
eil76	538	1.49	5.46	2.04	4.35	0.56	3.41	1.373	2.26
rad100	7,910	1.28	3.99	1.19	3.64	0.91	3.66	0.33	1.63
kroD100	21,294	0.89	2.28	0.80	1.87	0.38	1.89	0.47	0.89
eil101	629	1.27	4.17	1.11	3.78	1.43	3.12	0.65	1.42
lin105	43,279	0.62	2.75	0.00	1.08	0.00	0.15	0.13	0.99
ch130	6,110	1.02	2.82	1.13	2.82	0.57	2.82	0.18	1.53
kroA150	26,524	1.47	4.61	1.55	3.06	0.58	3.14	0.57	1.05
kroA200	29,368	0.86	3.70	0.92	3.27	0.79	2.80	1.22	5.61
lin318	42,029	2.17	4.16	2.65	4.31	1.92	3.97	0.44	1.29
rat575	6,773	N/A	N/A	4.89	6.20	4.05	5.06	1.05	1.76
rat783	8,806	N/A	N/A	5.66	6.57	5.00	6.11	2.55	3.63
d1655	62,128	N/A	N/A	9.10	10.09	14.15	16.07	3.66	5.01

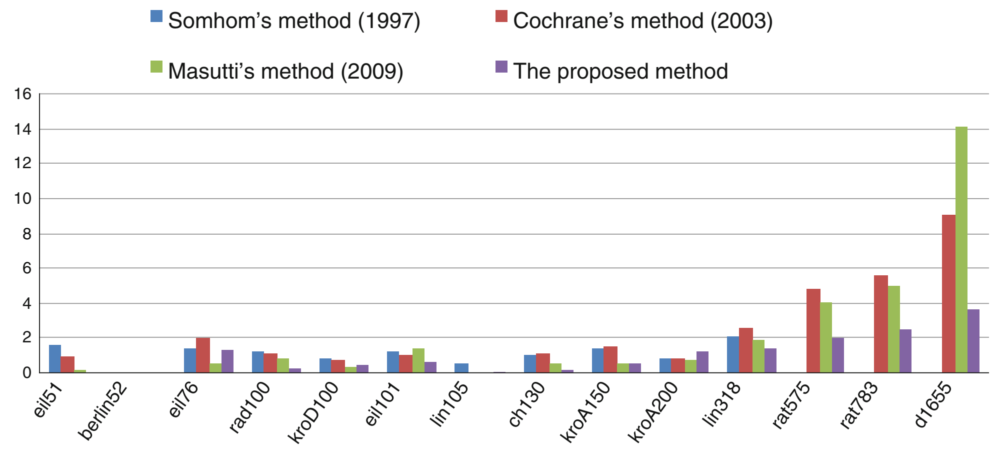

Fig. 4 Percentage deviations of the best solution (PDbest) found to each TSP dataset for different methods

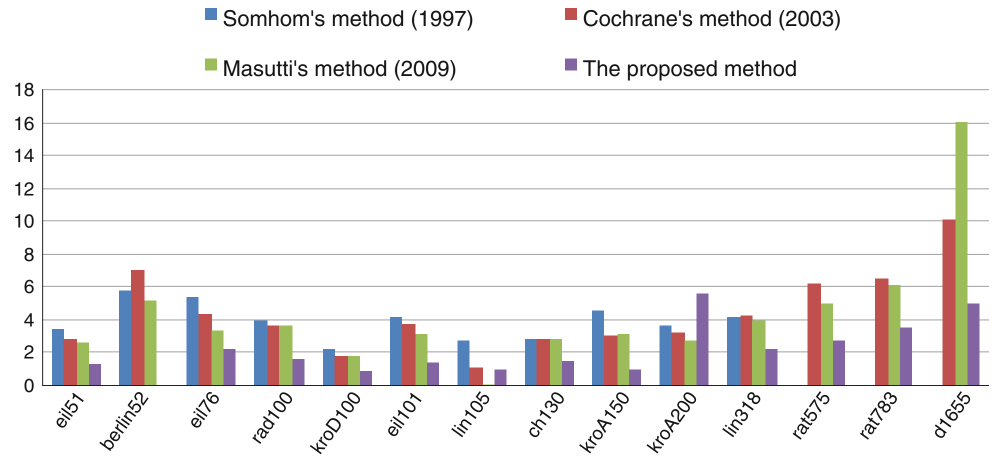

Fig. 5 Percentage deviations of the average solution (PDav) found to each TSP dataset for different methods

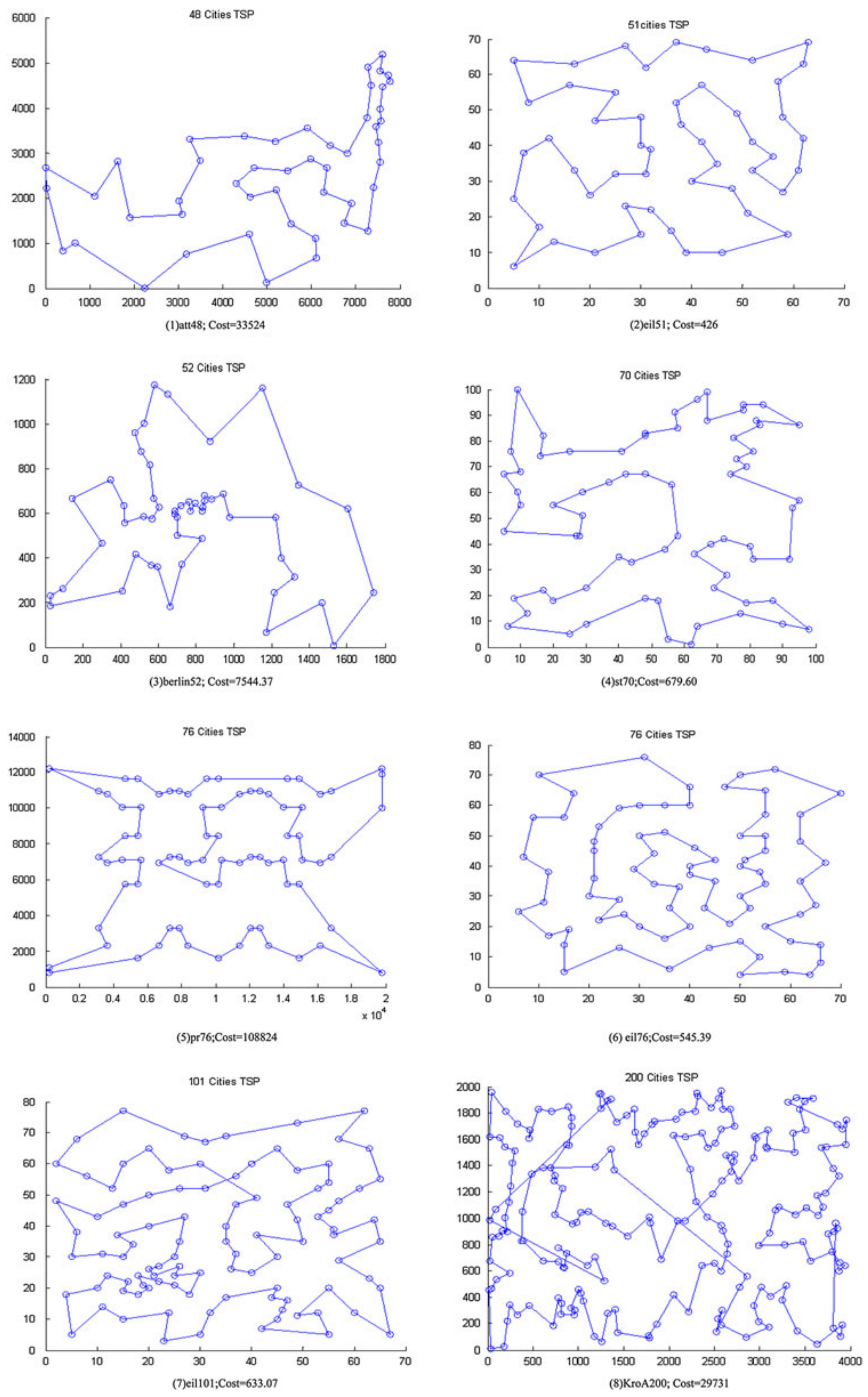

Fig. 6 Some of the best routes found by GA–PSO–ACO for TSP and their costs

traveling salesman problem (TSP). Although the GAs, PSO and ACO can usually find the better solutions for the TSP, their solutions are still infected by the randomly initializing population and parameter configuration, etc. With the cooperative evolution scheme of GAs, PSO and ACO, we can improve the best solution and average solution quality. Because the PSO provides the global best experience, the GAs are provided with the best individual to survive in the next generation and the ACO is provided with the procedure of converging to the global optimum, we ensure that the best solution of the GA–PSO–ACO algorithm will be better after exchanging several individuals from the GAs, PSO and ACO. From the experimental results, we can see that the best solution quality and average solution quality of GA–PSO–ACO algorithm is really better than those of the Tabu Search, GAs, PSO, ACO and PS–ACO, respectively. The result was obtained by testing the 35 data sets from the TSPLIB with cities scale from 48 to 33,810 and by comparing the experimental results of the proposed method with the Tabu Search, GAs, PSO, ACO and PS–ACO. And we also find that the best solution and average solution quality of the hybrid PS–ACO algorithm is better than that of the Tabu Search, GA, PSO and ACO, respectively. There is a comparison of the experimental results of the proposed method with other method [Somhom's method (Somhom et al. 1997), Cochrane's method (Cochrane and Beasley 2003), Masutti's method (Masutti and de Castro 2009)]. As a whole, we can see that the order of solution quality for the six algorithms is GA–PSO–ACO > PS–ACO > ACO > Tabu Search > GAs > PSO. However, the GA–PSO–ACO algorithm takes longer CPU time than the other five algorithms for the same TSP instances.

So, some future works exist for us. We will apply the GA–PSO–ACO algorithm to other practical problems, such as the vehicle routing problem, the job-shop scheduling problem, the flow-shop scheduling problem, logistics and data transmission in computer networks, etc. Besides, since better solutions are obtained with shorter CPU time, it can be concluded that the GA–PSO–ACO algorithm is greatly improved.

Acknowledgments The authors would like to thank all the reviewers for their constructive comments. This research was supported by the National Natural Science Foundation of China (61175056, 51175054, 60870009), the Visiting Scholarship of State Key Laboratory of Power Transmission Equipment and System Security and New Technology (Chongqing University) (2007DA10512711406), the Open Project Program of Artificial Intelligence Key Laboratory of Sichuan Province (Sichuan University of Science and Engineering), China (No. 2010RZ004, 2011RYJ03), the National High Technology Research and Development Program of China (863 Program) (2012AA040912), the Open Project Program of Key Laboratory of Intelligent Computing & Signal Processing of Ministry of Education, (Anhui University), China, the Open Project Program of Key Laboratory of advanced Design and Intelligent Computing (Dalian University), Ministry of Education, China (No. ADIC2010008), and the Open Project Program of Key Laboratory of Numerical Simulation in the Sichuan Provincial College (Neijiang Normal University) China (No. 2011SZFZ001), Young Key Teachers Foundation Projects of Dalian Maritime University (2011QN028, 2012QN031). The program for the initialization, study, training, and simulation of the proposed algorithm for the TSP in this article was written with the tool-box of MATLAB 2009 produced by the Math-Works, Inc.

References

Acampora G, Gaeta M, Loia V (2011) Combining multi agent paradigm and memetic computing for personalized and adaptive learning experiences. Comput Intell J 27(2):141–165

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, pp 1–20

Albayrak M, Allahverdi N (2011) Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms. Expert Syst Appl 38(3):1313–1320

Assareh E, Behrang MA, Assari MR, Ghanbarzadeh A (2010) Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran. Energy 35(12):5223–5229

Banaszak D, Dale GA, Watkins AN, Jordan JD(2009) An optical technique for detecting fatigue cracks in aerospace structures. In: Proc 18th ICIASF, pp 1–7

Bullnheimer B, Hartl RF, Strauss C((1997) A new rank based version of the ant system—a computational study. Cent Eur J Oper Res Econ 7(1):25–38

Chen SM, Chien CY (2011) Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Syst Appl 38(4):3873–3883

Cheng CB, Mao CP (2007) A modified ant colony system for solving the travelling salesman problem with time windows. Math Comput Model 46(9–10):1225–1235

Chu SC, Huang HC, Shi Y, Wu SY, Shieh CS (2008) Genetic watermarking for zerotree-based applications. Circuits Syst Signal Process 27(2):171–182

Cochrane EM, Beasley JE (2003) The co-adaptive neural network approach to the Euclidean traveling salesman problem. Neural Netw 16(10):1499–1525

Colorni A, Dorigo M, ManiezzoV (1991) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life, Paris, France, pp 134–142

Deng W, Li W, Yang XH (2011) A novel hybrid optimization algorithm of computational intelligence techniques for highway passenger volume prediction. Expert Syst Appl 38(4):4198–4205

Deng W, Chen R, Gao J et al (2012) A novel parallel hybrid intelligence optimization algorithm for function approximation problem. Comput Math Appl 63(1):325–336

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66

Ellabib I, Calamai P, Basir O (2007) Exchange strategies for multiple ant colony system. Inf Sci 177(5):1248–1264

Fan SKS, Liang YC, Zahara E (2006) A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search. Comput Ind Eng 50(4):401–425

Geng XT, Chen ZH, Yang W, Shi DQ, Zhao K (2011) Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Appl Soft Comput 11(1):3680–3689

Grimaldi AE, Gandelli A, Grimaccia F, Musseeta M, Zich RE (2005) A new hybrid technique for the optimization of large-domain electromagnetic problems. In: Antennas and propagation society international symposium, pp 61–64

Guvenc U, Duman S, Saracoglu B, Ozturk A (2011) A hybrid GA–PSO approach based on similarity for various types of economic dispatch problems. Electron Electr Eng Kaunas: Technologija 2(108):109–114

Held M, Karp RM (1970) The traveling salesman problem and minimum spanning trees. Oper Res 18:1138–1162

Hendtlass T (2003) Preserving diversity in particle swarm optimization. Lect Notes Comput Sci 2718:4104–4108

Hoffman AJ, Wolfe P (1985) History. In: Lawler L, Rinooy K, Shmoys D (eds) The traveling salesman problem. Wiley, Chichester, pp 1–16

Holland JH (1975) Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press

Hua Z, Huang F (2006) A variable-grouping based genetic algorithm for large-scale integer programming. Inf Sci 176(19):2869–2885

Kao YT, Zahara E (2008) A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl Soft Comput 8(2):849–857

Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(5–6):267–283

Kennedy J, Eberhart R (1995) Particle swarm optimization, Proceedings of the IEEE international conference on neural networks, IEEE Press, Piscataway, pp 1942–1948

Kuo H, Horng SJ, Kao TW, Lin TL et al (2010) Hybrid swarm intelligence algorithm for the travelling salesman problem. Expert Syst Appl 27(3):166–179

Lim KK, Ong YS, Lim MH, Chen XS, Agarwal A (2008) Hybrid ant colony algorithms for path planning in sparse graphs. Soft Comput 12(10):981–1004

Liu F, Zeng G (2009) Study of genetic algorithm with reinforcement learning to solve the TSP. Expert Syst Appl 36(3):6995–7001

Lo CC, Hus CC (1998) Annealing framework with learning memory. IEEE Trans Syst Man Cybern Part A 28(5):1–13

Marinakis Y, Marinaki M, Dounias G (2010) A hybrid particle swarm optimization algorithm for the vehicle routing problem. Eng Appl Artif Intell 23(4):463–472

Masutti TA, de Castro LN (2009) A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. Inf Sci 179(10):1454–1468

Mavrovouniotis M, Yang SX (2011) A memetic ant colony optimization algorithm for the dynamic travelling salesman problem. Soft Comput 15(7):1405–1425

Mehmet Ali MK, Kamoun F (1993) Neural networks for shortest tour computation and routing in computer networks. IEEE Trans Neural Netw 4(5):941–953

Naimi HM, Taherinejad N (2009) A new robust and efficient ant colony algorithms: using new interpretation of local updating process. Expert Syst Appl 36(1):481–488

Niknam T, Ranjbar AM, Shirani AR (2005) A new approach for distribution state estimation based on ant colony algorithm with regard to distributed generation. J Intell Fuzzy Syst 16(2):119–131

Onwubolu GC, Clerc M (2004) Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. Int J Prod Res 42(3):473–491

Paulo HS, Maria TAS, Sérgio S (2007) A new approach to solve the traveling salesman problem. Neurocomputing 70(4–6):1013–1021

Sarhadi H, Ghoseiri K (2010) An ant colony system approach for fuzzy traveling salesman problem with time windows. Int J Adv Manuf Technol 50(9–12):1203–1215

Shen G, Zhang YQ (2011) A new evolutionary algorithm using shadow price guided operators. Appl Soft Comput 11(2):1983–1992

Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf Process Lett 103(5):169–176

Shuang B, Chen JP, Li ZB (2011) Study on hybrid PS–ACO algorithm. Appl Intell 34(1):64–73

Singh A, Baghel AS (2009) A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Comput 13(1):95–101

Somhom S, Modares A, Enkawa T (1997) A self-organizing model for the traveling salesman problem. J Oper Res Soc 48(4–6): 919–928

Stützle T, Hoos HH (2000) MAX–MIN ant system. Future Gener Comput Syst 16(8):889–914

Taher N, Babak A (2010) An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. Appl Soft Comput 10(1):183–197

Tasgetiren MF, Suganthan PN, Pan QK, Liang YC (2007) A genetic algorithm for the generalized traveling salesman problem. In: Proceedings of the 2007 IEEE congress on evolutionary computation, pp 2382–2389

Tsai CF, Tsai CW, Tseng CC (2004) A new hybrid heuristic approach for solving large traveling salesman problem. Inf Sci 166(1–4): 67–81

Wang KP, Huang L, Zhou CG, Pang W (2003) Particle swarm optimization for traveling salesman problem. In: International conference on machine learning and cybernetics, pp 1583–1585

Wang X, Gao XZ, Ovaska SJ (2007) A hybrid optimization algorithm based on ant colony and immune principles. Int J Comput Sci Appl 4(3):30–44

Xing LN, Chen YW, Yang KW et al (2008) A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. Eng Appl Artif Intell 21(8):1370–1380

Yannis M, Magdalene M (2010) A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput Oper Res 37(3):432–442

Zahara E, Kao YT (2009) Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst Appl 36(2):3880–3886