14

Particle Swarm and Ant Colony Algorithms and Their Applications in Chinese Traveling Salesman Problem

Shuang Cong, Yajun Jia and Ke Deng
University of Science and Technology of China
P. R. China

1. Introduction

Intelligent heuristic optimization methods have increasingly attracted the attentions and interests of many scholars in recent years. Such as genetic algorithm, ant colony algorithm, particle swarm optimization, simulated annealing, *etc.*. They have become effective tools to solve the TSP and other NP-hard combinatorial optimization problems. The particle swarm optimization (PSO) algorithm is a population-based evolutionary algorithm which was proposed by Eberhart and Kennedy in 1995 (Eberhart & Kennedy, 1995). The PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. No bird knows where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? An effective one is to follow the birds which are nearest to the food. The PSO firstly generates a random initial population, the population contains numbers of particles, each particle represents a potential solution of system, each particle is represented by three indexes: position, velocity, fitness. Firstly endows each particle a random velocity, in flight, it dynamically adjusts the velocity and position of particles through their own flight experience (personal best position), as well as their companions' (global best position). The evolutions of particles have a clear direction, the whole group will fly to the search region with higher fitness through continuous learning and updating. This process will be repeated until reach the default maximum iterations or the predetermined minimum fitness. The PSO is therefore in essence a fitness-based and group-based global optimization algorithm, whose advantage lies in the simplicity of algorithm, easy implementing, fast convergence and less parameters. Presently, the PSO has been widely applied in function optimization, neural network training, pattern classification, fuzzy system control and other applications. Whereas, like other intelligent optimization algorithms, the PSO may occur the phenomenon that particle oscillates in the vicinity of optimal solution during searching in the search space, therefore the entire particle swarm performs a strong "convergence", and it is easily trapped in local minimum points, which makes the swarm lose diversity. Thus it has the weakness of solving complex problems, and it is difficult to obtain a more accurate solution in the late evolution. Many scholars proposed some improved algorithms (Yuan et al., 2007; Xu *et al.*, 2008; Lovbjerg, 2001), which improve the search capabilities of the elementary PSO in different aspects.

Source: New Achievements in Evolutionary Computation, Book edited by: Peter Korosec,
ISBN 978-953-307-053-7, pp. 318, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

Bionics appeared in the mid 50's in 20th century, people were inspired from the mechanism of organic evolution, and put forward many new methods to solve complex optimization problems. In these methods, the evolutionary computation including evolution strategies, evolutionary programming, and genetic algorithms is the most remarkable. With people's research to biological group behavior and bio-social, algorithms based on swarm intelligence theory have appeared including Ant Colony Optimization (ACO). Since the Ant System (AS) which is the first algorithm in line with the ACO framework was put forward, the researchers have begun their attempts to improve the design. The first one is Elitist Strategy for Ant System (EAS). The EAS mainly gives special pheromone deposit to the artificial ants, which perform so far the best in constructing solutions followed by the Ant-Q algorithm which combines ant colony algorithm with the Q learning algorithm, uses the synergies of artificial ants. Then there appears the Ant Colony System (ACS), Rank based version AS (AS_{rank}) and Max-Min Ant System (MMAS). These three improvements have greatly improved the performance of AS, in particular the MMAS gets a lot of expansion and becomes an algorithm of highly practical application and one of the best ACO algorithms at present. In recent years, there have been some new improvements of ACO such as Approximate Nondeterministic Tree Search (ANTS). The ANTS is extended to a deterministic algorithm later, and it has a good performance in solving the Quadratic Assignment Problem (QAP); Another new improved algorithm is the Hyper-Cube Framework for ACO, and its purpose is automatically adjusting the value of pheromone trails to ensure that the pheromone trails lie always in the interval [0,1]. The current study for ACO has extended from TSP range to many other fields, and it has developed into solving the multi-dimensional and dynamic combinatorial optimization problems instead of the static one-dimensional optimization problem. The research of ACO has also developed from discrete domain into continuous domain. It has got fruitful research results in improving the performance of the ACO and grafting on bionic natural evolutionary algorithms or local search algorithms.

This chapter is divided into three parts. In part one, in order to solve the shortcoming of easily being trapped in local minimum points, we respectively introduced mutation and simulated annealing (SA) algorithm (Kang *et al.*,1998) to the PSO, and proposed a hybrid algorithm by combined with the advantages of the strong global search ability of PSO and good local search ability of SA. The hybrid algorithm proposed was applied to solve the Chinese Traveling Salesman Problem with 31 cities (C-TSP). The comparative study on the experimental results with SA, elementary PSO (Zhong *et al.*, 2007; Xiao *et al.*, 2004; Li *et al.*, 2008) and PSO with mutation were given. In part two, the mechanisms and properties of the five ant colony algorithms were synthesized, compared and analyzed including basic ant colony algorithm (ant system, AS), elitist strategy of ant system (EAS), a new rank-based version of the ant system (AS_{rank}), max-min ant system (MMAS) and ant colony system (ACS). The efficiency of five algorithms was also compared through their applications in the C-TSP. The investigations of the performances were done in the aspects of the effects of different parameters and the relations between parameters of the algorithms. The third part is conclusions.

2. PSO and its application in C-TSP

2.1 PSO with mutation

In the PSO, each single solution is a "bird" in the search space. The particles fly through the problem space by following the current optimum particles. In every iteration, each particle

is updated by following two "best" values. The first one is the best solution it has achieved so far, it is a *personal best position* denote by p_{ib}. Another "best" value, which is tracked by the particle swarm optimizer, is the best value obtained so far by any particle in the population, it is a *global best position* and defined as p_{gb}. All of particles have fitness values which are evaluated by the fitness function to be optimized. Each particle updates according to those two best values, and then a new generation of population is created.

Suppose that the searching space is *D* dimensional with *m* randomly initialized particles in it, the particle swarm can be indicated by following parameters: $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ stands for the location of particle *i* in the *D* dimensional space and it is also regarded as a potential solution, $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$ stands for the flight velocity of particle *i* in the *D* dimensional space, $p_i = (p_{i1}, p_{i2}, ..., p_{iD})$ stands for the personal best position of particle *i*, $p_i = (p_{g1}, p_{g2}, ..., p_{gD})$ stands for the global best position in the whole swarm. The particle updates its velocity and position with the following rules:

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 rand()(p_{ib} - x_{ib}^{k}) + c_2 rand()(p_{gb} - x_{ib}^{k}) \quad (1)$$

$$x_{ib}^{k+1} = x_{ib}^{k} + v_{ib}^{k+1} \quad (2)$$

in which, $i = 1, 2, \ldots, m$; $d = 1, 2, \ldots, D$; k is iteration number; c_1, c_2 are called *acceleration coefficients*, which are used to adjust the maximum flight step of personal best value and global best value, *rand*() returns a random number between (0,1); ω is inertia weight which affects the balance of global search ability and local search ability.

In Zhong *et al.* in 2007 a large number of experiments proved that once ω decreases linearly with the iteration, the convergence of algorithm would be significantly improved. Therefore here we let $\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) * k}{K}$, where k is the current iteration number, K is the maximum iteration number, $\omega_{\max}$ is the maximum inertia weight, $\omega_{\min}$ is the minimum inertia weight. The basic principles of Eq. (1) is that the velocity achieves information from the original velocity, personal best value and global best value, the number of information depends on ω, c_1 and c_2. The first part of Eq. (1) is called *memory term*, which denotes the impact of velocity and direction of previous iteration. The second part (the distance between current position of particle *i* and the personal best position) is called *self-awareness term*, which denotes the information that comes from its own experience. The third part (the distance between current position of particle *i* and the global best position) is called *population-awareness term*, which denotes the information that comes from another particles of the whole swarm, which reflects the knowledge sharing and cooperation. The PSO algorithm can be implemented in the following 6 steps:

Step 1. Initialize generation and all particles, viz. set the initial position *X* of each particle and the initial velocity *V* randomly.

Step 2. For each particle, calculate the fitness value.

Step 3. For each particle, if the fitness value is better than the best fitness value $f(p_{ib})$ in history, set current value as the new p_{ib}.

Step 4. Choose the particle with the best fitness value of all the particles as p_{gb}.

Step 5. For each particle, calculate the particle velocity according to Eq. (1), and update particle position according to Eq. (2).

Step 6. If the maximum iteration or the minimum error criteria is not attained, return to *Step 2*; otherwise end the iteration.

The PSO has been successfully applied in many continuous optimization problems. The Traveling Salesman Problem (TSP) is a typical discrete combinatorial problem. If one wants to solve the TSP with PSO, some improvements of basic PSO must be done. In Huang *et al.*, in 2003 the concept of swap operator and swap sequence were introduced for solving the TSP. Suppose that the solution sequence of the TSP with n nodes is $S = (a_i), i = 1,...,n.$ The definition of *swap operator* $SO(i_1, i_2)$ is the points a_{i_1} and a_{i_2} in the solution sequence *S*. *Swap sequence* is an orderly sequence with one or more swap operators, meanwhile the order between swap operators is meaningful. Different swap sequence operate on the same solution may generate the same new solutions, the equivalent set of swap sequence is the set of swap sequence which has the same effect. Among all the equivalent sets of swap sequence, the swap sequence with least swap operators is called *basic swap sequence.* An array with *N* cities denotes the particle's position *X*, All the possible arrays compose the state space of the problem. Based on vectors, functions and operations defined above, the traditional updating equations will be changed in the following versions (Huang et al., 2003):

$$v_{id}^{k+1} = \omega v_{id}^{k} \oplus \alpha(P_{id} - X_{id}) \oplus \beta(P_{gd} - X_{id}) \tag{3}$$

$$x_{ib}^{k+1} = x_{ib}^{k} + v_{ib}^{k+1} \tag{4}$$

in which, $\alpha, \beta(\alpha, \beta) \in [0,1]$ are random numbers. $\alpha(P_{id} - X_{id})$ denotes that all the swap operators in the basic swap sequence $(P_{id} - X_{id})$ are reserved with a probability of α, similarly, $\beta(P_{gd} - X_{id})$ denotes that all the swap operators in the *basic swap sequence* $(P_{gd} - X_{id})$ are reserved with a probability of β. Thus the greater the value of α is, the more swap operators that $(P_{id} - X_{id})$ will be reserved, and the greater the impact of P_{id} is. Similarly, the greater the value of β is, the more swap operators that $(P_{gd} - X_{id})$ will reserve, and the greater the impact of P_{gd} is. The definition of operator " $\oplus$ " is the merger operator of two swap sequences. Operator "+" denotes the implementation of swap operation, operator "-" denotes to obtain the basic swap sequence of two sequences. For example : *A* = *(*1 2 3 4 5*), B* = *(*2 3 1 5 4*),* as can be seen that *A(1)* = *B(3)* = 1, so the first swap operator is *SO*(1,3), *B1* = *B* + *SO* (1,3), so one gets that *B1*: (1 3 2 5 4), *A*(2) = *B*1(3) = 1, so the second swap operator is *SO*(2,3), *B* = *B1* + *SO*(2,3), so one gets that *B2*: (1 2 3 5 4). Similarly, the third swap operator is *SO*(4,5), *B*3 = *B*2 + *SO*(4,5) = *A*, thus one gets a basic swap sequence: *SS* = *A* – *B* = (*SO*(1,3), *SO*(2,3), *SO*(4,5)).

The steps of the PSO algorithm for solving the TSP can be described as follows:

Step 1. Initialize generation and all the particles, set each particle a random initial solution and a random swap sequence.

Step 2. If the maximum iteration or the minimum error criteria is met, turn to *Step 5.*

Step 3. According to the particle's current position X_{id}^{k}, calculate the next position X_{id}^{k+1}, namely the new solution.

1. Calculate the difference between P_{id} and X_{id}, $A = P_{id} - X_{id}$, in which *A* is a basic swap sequence.

2. Calculate the difference between P_{gd} and X_{id}, $B = P_{gd} - X_{id}$, in which B is a basic swap sequence.
3. Calculate the velocity v_{id}^{k+1} according to Eq. (3), and convert the swap sequence v_{id}^{k+1} to a basic swap sequence.
4. Calculate the new solution x_{ib}^{k+1} according to Eq. (4).
5. If x_{ib}^{k+1} is better than P_{id}, set a new solution x_{ib}^{k+1} as new P_{id}.

Step 4. Choose the particle with the best fitness value of all the particles as P_{gd}, turn to *Step 2*.

Step 5. Show the result that obtained.

In the settlement of solving TSP, basic PSO generates new individual through Eq. (3) and Eq.(4), from which one can see that a basic swap sequence generated by Eq. (3) is in fact equivalent to the swap operator to a route, but the route between the two swap cities do not change, so it is easy to generate cross route which is illegal solution. To deal with the problem, inspired by the mutation operator in evolutionary algorithm, we add the mutation operator to the PSO. The specific approach is: after generating a new route by basic PSO approach during each iteration one does the mutation operator to the new route. More specific, change *Step 4* as follows:

Step 4: Generate two mutate cities randomly, then reverse the order of all the cities between two mutate cities. If the length of new route is less than the original route, set the new route as x_{ib}^{k+1}. Otherwise, maintain the original route unchanged.

2.2 A hybrid algorithm of PSO and SA

The SA algorithm derived from the principle of solid annealing. Firstly, heat the solid to a sufficiently high temperature, and then cool it slowly. This process is based on an analogy from thermodynamics where a system is slowly cooled in order to achieve its lowest energy state. According to Metropolis criteria, the probability of particles balance at temperature T is $e^{-\Delta E/(kT)}$, where E is the internal energy at temperature T; ΔE is the increment of internal energy; k is the Boltzmann constant. Once one converts the internal energy E to the objective function value, and the temperature T to control parameter t, the SA algorithm of solving combinatorial optimization problems may be obtained with the initial solution i and initial control parameter t by repeating the iteration of "generate new solution $\rightarrow$ calculate the difference of objective function $\rightarrow$ accept or discard" to the current solution, and gradually reduce the value of control parameter t. The current solution is the approximate to the optimal solution when algorithm is terminated. It is a stochastic heuristic search process based on Monte Carlo iterative method. The process is controlled by *Cooling Schedule*, which includes initial control parameter t, attenuation factor Δt, iteration number for each t and the termination condition .

The SA algorithm used to solve the TSP can be described as follows:

1. *Solution spaces*：Solution spaces are all the routes of visiting each city once. The solution can be denoted as $\{\omega_1, \omega_2, \ldots \omega_n\}$. $\omega_1, \ldots, \omega_n$ in an array that is composed of 1 to n, which denotes one walks to start from the city ω_1, and visits along with the cities $\omega_2, \ldots, \omega_n$ orderly, then returns to the city ω_1.
2. *Objective function*: Objective function is the total distance length of the route pass through all the cities. The objective function value of the optimal route is the least one. Objective function is also called fitness function.

3. *Criteria of new solution generation and acceptance*: Here we use reverse operator to generate new solution, more specifically, choose two different number *k* and *m* between 1 to *n* randomly, moreover, *k* is smaller than *m*, then one swaps the cities between *k* to *m*, that is to convert $\{\omega_1, \omega_2, \ldots \omega_k, \omega_{k+1}, ..., \omega_m, ..., \omega_n\}$ into $\{\omega_1, \omega_2, \ldots \omega_m, \omega_{m+1}, ..., \omega_{k+1}, \omega_k, ..., \omega_n\}$. Once the new route is generated, calculate the difference of distance length between new route and current route, if the length of new route is smaller, that is, $\Delta f = f(x_j) - f(x_i) \le 0$, set the new route as x_{ib}^{k+1}, if the length of new route is bigger, but $\exp(-\Delta f / t) > random(0,1)$, still set the new route as x_{ib}^{k+1}, otherwise maintain the current route unchanged.

The SA algorithm in hybrid algorithm of PSO and SA proposed calculates alternately by two steps:

1. Generate a new solution by stochastic perturbation and calculate the change of the objective function.
2. Decide whether the new solution is accepted or not. In the case at a high temperature, the solution that increases the objective function may be accepted by means of decreasing the temperature slowly, which may avoid to trap in local minima. In such a way the algorithm can converge to the global best solution.

The nature of basic PSO is the use of individual and global maximum to guide the position of the next iteration, which can converge fast at the early iteration. But, after several iterations current solution, personal best value and global best value tend to the same, which leads to the loss of population diversity, and makes the solution be trapped in local minima. In order to avoid this phenomenon, inspired by the SA algorithm, we redesign the algorithm's framework: when basic PSO converges to a solution p_g, use the solution p_g as the initial solution of SA, accept the new solution in accordance with the Metropolis criteria. If there is such a solution *y* satisfies $f(y) < f(p_g)$, that is, the solution calculated by basic PSO is not the global optimal solution. Then a new solution *y* can be used to randomly replace a particle in the swarm, and the evolution of PSO continues, which can increase the population diversity as well as retain the previous operation procedure. If there is not such a solution *y*, then let $f(y) < f(p_g)$, that is, no better solution than p_g has been found until the convergence of SA, which indicates that p_g is the global optimal solution.

2.3 The C-TSP application and results analysis

The TSP is a well-known combinatorial optimization problem with typical NP-hard and is often used to verify the superiority of some intelligent heuristic algorithm. The mathematical description of the TSP is: Given a list of *n* cities in order of visiting as $X = \{x_1, x_2, ..., x_n\}$ and $x_{n+1} = x_1$, the task is to find the shortest possible tour distance of $\min \sum_{i=1, x \in \Omega}^{n} d_{x_i x_{i+1}}$ that visits each city exactly once. The TSP can be modeled as a graph: the graph's vertices correspond to cities and the graph's edges correspond to connections between cities, the length of an edge is the corresponding connection's distance. A TSP tour is now a Hamiltonian cycle in the graph, and an optimal TSP tour is the shortest Hamiltonian cycle.

Chinese Traveling Salesman Problem (C-TSP) is a typical symmetric TSP problem. Its simple description is: Given a list of 31 Chinese capital cities and their pairwise distances, the task is

to find the shortest possible tour that visits each city exactly once. The C-TSP is a medium-scale TSP problem, which has $(31-1)!/2 = 1.326*10^{32}$ possible routes.

The algorithms of solving TSP problem are divided into two categories: Exact algorithms and approximation algorithms. The exact algorithms used frequently includes branch and bound algorithms, linear programming and exhaustion method, *etc.*. The running time for this approach lies within a polynomial factor of $O(n!)$, the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. Approximation algorithm is divided into tour route construction algorithm, tour route optimization algorithm and heuristic algorithm, in which heuristic algorithm is the most spectacular including genetic algorithm, ant colony algorithm, particle swarm optimization, simulated annealing, differential evolution, *etc.*. Currently, to C-TSP problem, simulated annealing, improved genetic algorithm, differential evolution all achieve the optimal solution of 15404 *km*. The SA has the advantages of high, quality, robust, easy to achieve but with slow convergence.

The hybrid algorithm proposed is applied into the C-TSP. At the same time, basic PSO, SA and PSO with mutation are also applied to do the comparisons. The programming language is Matlab 7.0, and it runs on Win XP with Intel Core2 Duo 2.10 GHz CPU. We use the discrete PSO (DPSO) that proposed by Huang *et al.* in 2003 as the basic PSO, parameter settings are as follows: Particle number m = 200, maximum inertia weight $w_{\max}$ = 0.99, minimum inertia weight $w_{\min}$ = 0.09, acceleration coefficient $c1$ = 0.8, $c2$ = 0.4, iteration number k = 2000. The parameters of mutation in the PSO are the same as in DPSO. The parameter settings of SA are as follows: Initial temperature T_0 = 5000, termination temperature T_f = 1, cycle constant L = 31000, attenuation factor α = 0.99. The parameter settings of the hybrid algorithm are as follows : Particle number m = 200, maximum inertia weight $w_{\max}$ = 0.99, minimum inertia weight $w_{\min}$ = 0.09, acceleration coefficients $c1$ = 0.8, $c2$ = 0.4, initial temperature T_0 = 5000, termination temperature T_f = 1, cycle constant L = 31000, attenuation factor α = 0.95. Each algorithm has been run for 20 times, the numerical results of four algorithms are listed in Table 1, in which "Worst" denotes the worst solution in 20 runs, "Best" represents the best solution in 20 runs, "Average " is the average fitness in 20 runs, "Optimal rate" denotes the rate that gets the times of the optimal solution (15404) over 20 times runs.

One can see from Table 1 that the result obtained by DPSO is not satisfactory. It is unable to find the optimal solution of 15404, and its average value of solutions is also away from the optimal solution, which is because current solution X_{id} , personal best value p_{ib} and global best value p_{gb} tend to the same after several iterations. The DPSO is difficult to get new effective edge and new effective route, and it is easy to be trapped in local minima. On the other hand, the SA, PSO with mutation and the hybrid algorithm can obtain the optimal solution. The average and worst distances of hybrid algorithm proposed are 15453.4 and 15587, respectively, which are the smallest in all those of the four algorithms. The solutions obtained by the hybrid algorithm proposed are the best and its optimal solution rate is 20%, which is the highest. In order to further compare SA, PSO with mutation and the hybrid algorithm, we have studied the three algorithms in the fitness curve when finding the optimal solution. The results are shown in Figures 1-3, in which the x axes is the iteration number in unite of times, and the y axes is the global best route distance which is just the solution of C-TSP.

Algorithm	Worst	Best	Average	Optimal rate
DPSO	20152	16665	18035.3	0
PSO With Mutation	16194	15404	15662.3	0.1
SA	15606	15404	15467.8	0.15
Hybrid Algorithm	15587	15404	15453.4	0.2

Table 1. Experiment Results

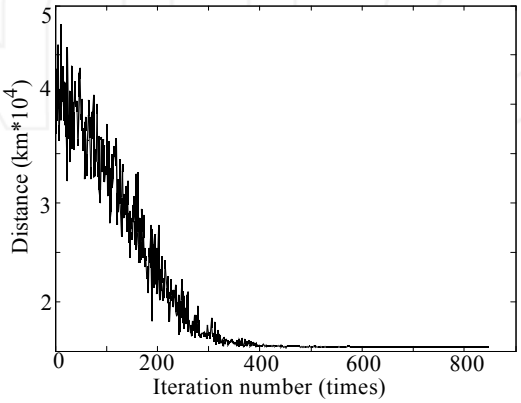


Fig. 1. The C-TSP optimization procedure of SA

From Figure 1 it is clear that SA converges to the optimal solution in about 500 iterations. Figure 2 indicates that PSO with mutation algorithm has oscillation at the early iteration, and it converges to the optimal solution in about 1000 iterations. From Figure 3 one can see that the hybrid algorithm converges to the optimal solution in about 100 iterations. The

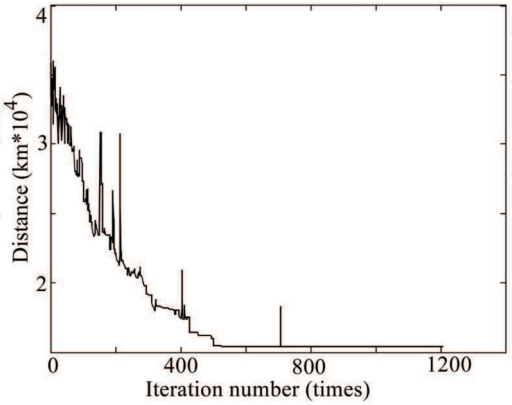


Fig. 2. The C-TSP optimization procedure of PSO with mutation

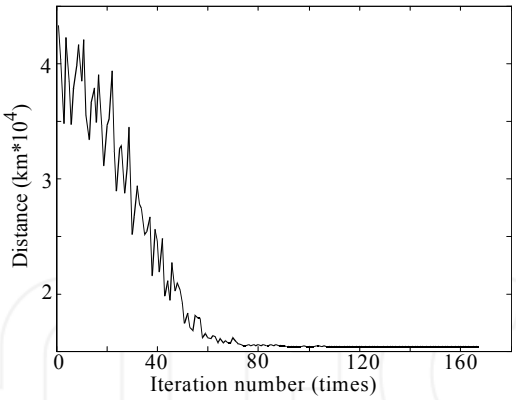


Fig. 3. The C-TSP optimization procedure of hybrid algorithm proposed

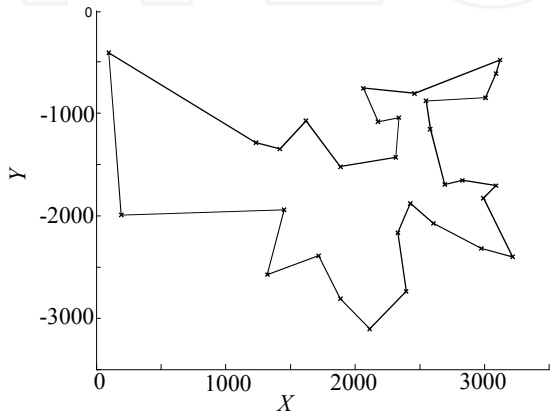


Fig. 4. Optimal route of C-TSP

problem of local optimal solution is not only been solved, but also the convergence speed is greatly decreased. Figure 4 is the optimal route made by the hybrid algorithm for C-TSP problem. The optimal route is: Beijing - Harbin - Changchun - Shenyang - Tianjin - Jinan - Hefei - Nanjing - Shanghai - Hangzhou - Taipei - Fuzhou - Nanchang - Wuhan - Changsha - Guangzhou - Haikou - Nanning - Guiyang - Kunming - Chengdu - Lhasa - Urumchi - Xining - Lanzhou - Yinchuan - Xian - Zhengzhou - Shijiazhuang - Taiyuan - Hohhot – Beijing. The total distance length of the optimal route is 15404 *km*.

3. Ant colony optimization algorithms and their improvements to the C-TSP application

3.1 Ant colony optimization algorithms

Artificial ants of the ACO algorithms build solutions by performing random walk which use a certain stochastic rules on a completely connected graph $G_C = (C, L)$ whose nodes are the components *C*, and the set *L* fully connect the components *C*. Each connection of the map

$G_C = (C, L)$ can be associated pheromone trail τ_{ij}, and heuristic information η_{ij} (The subscripts *i* and *j* are labeling of the nodes on the map).

It is important to note that artificial ants are in parallel movement independently. Although each ant is complex enough to find a (probably poor) solution to the problem under consideration, good-quality solutions can only emerge as the result of the collective interaction among the ants. This collaborative interaction is obtained via indirect communication mediated by the information ants read or write in the variables storing pheromone trail values. To some extent, this is a distributed learning process in which the single ant is not self-adaptive but, on the contrary, it can modify adaptively the way represented and perceived by other ants. Informally an ACO algorithm can be imagined as the interplay of three procedures (Dorigo & Stützle, 2004): Construction of Ants Solutions, Pheromone updating, and Daemon Actions.

1. Construction of Ants Solutions manages a colony of ants that concurrently and asynchronously visit adjacent states of the problem considered by moving through neighbor nodes of the problem's construction graph G_C. They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information. In such a way, ants incrementally build solutions of optimization problem. Once an ant has built a solution, the ant evaluates the solution that will be used by the Pheromone updating procedure to decide how much pheromone to deposit.
2. Pheromone updating is the procedure in which the pheromone trails are modified. The trails' values move either increase, as ants deposit pheromone on the components or connections they use, or decrease due to pheromone evaporation. From a practical point of view, the deposit of new pheromone increases the probability whose components/ connections are used by either many ants or at least one ant. A very good solution produced will be used again in future ants. Differently, pheromone evaporation carries out a forgetting factor in order to avoid a too rapid convergence to a sub-optimal region, so pheromone evaporation has the advantage of generating new search areas.
3. Daemon Actions procedure is used to centralize the actions which can not be performed by single ants. Examples of daemon actions are the activation of a local optimization procedure, or the collection of global information used to decide whether it is useful or not to deposit additional pheromone to update the search process.

These three procedures should interact and take into account the characteristics of the problem considered. The TSP can be represented by a complete weighed graph $G_C = (C, L)$ with *C* being the set of nodes representing the cities, and *L* being the set of arcs. Each arc $(i, j) \in L$ is assigned a value d_{ij}, which is the distance between cities *i* and *j*. In the symmetric TSP, $d_{ij} = d_{ji}$ holds for all the arcs in *L*; but in the general case of the asymmetric TSP, the distance between a pair of nodes *i*, *j* is dependent on the direction of traversing the arc, that is, there is at least one arc (*i*, *j*) for which $d_{ij} \neq d_{ji}$. More formally, TSP is described as: A finite set $C\{c_1, c_2, \cdots, c_N\}$ of components is given, where *N* is the number of components. Set $L = \{l_{ij} \mid c_i, c_j \in C\}$ fully connects the components C. $d_{ij}(i, j = 1, 2, \cdots n)$ is the Euclid distance of arc l_{ij}:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \qquad \forall (i, j) \in L \tag{5}$$

In the TSP, $G = (C, L)$ is a directed graph and the goal is to find a minimum length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed path visiting each

of the n nodes of G exactly once. In another way, an optimal solution to the TSP is a permutation R of the node indices $\{c_1, c_2, \cdots, c_n\}$ such that the length $F(R)$ is minimal, where $F(R)$ is given by

$$F(R) = \sum_{i=1}^{n-1} d_{R(i)R(i+1)} + d_{R(n)R(1)} \tag{6}$$

The ACO can be applied to the TSP in a straightforward way. The construction graph $G_C = (C, L)$, where the set L fully connects the components C, is identical to the problem graph; the set of states of the problem corresponds to the set of all possible partial tours; and the constraints Ω enforce that the ants construct only feasible tours that correspond to permutations of the city indexes. In all available ACO algorithms for the TSP, the pheromone trails are associated with arcs, so the τ_{ij} refers to the desirability of visiting city j directly after city i. The heuristic information is chosen as $\eta_{ij} = 1/d_{ij}$ thus the heuristic desirability of going from city i directly to city j is inversely proportional to the distance between the two cities. If there is the length of arc (i, j) equal to 0, then put the η_{ij} set to a very small value. For implementation purposes, pheromone trails are usually collected into a pheromone matrix whose elements are the arcs' pheromone trails. This can be done analogously for the heuristic information. Tours are constructed by applying the following simple construction procedure to each ant (Dorigo & Stützle, 2004): (1) choose a initial city according to some criterion. (2) make use of pheromone and heuristic values to probabilistically construct a tour by iteratively adding cities, to which the ant has not visited yet, until all cities have been visited; (3) go back to the initial city. After all ants have completed their tours, they may deposit pheromone on the tours they have followed. Sometimes, adding Daemon Actions such as the local search will improve the performance of algorithm.

3.2 Ant System (AS)

Ant System is created by Marco Dorigo and others in 1991 (Dorigo & Stützle, 2004; Dorido *et al.*, 1991; Colorini *et al.*, 1992; Dorigo 1992), and it is the first algorithm which is in line with the ACO algorithm framework. Initially three different versions of AS were proposed which are called *ant-density*, *ant-quantity*, and *ant-cycle*. These three versions are different on pheromone updating. Whereas in the ant-density and ant–quantity versions the ants update the pheromone directly after a move from one city to an adjacent city. In the ant-cycle version the pheromone deposited by each ant is set to be a function of the tour quality. The version of ant-cycle considers the quality of complete solution and uses pheromone updating method which has overall mechanism. Ant-cycle is better than the other two versions which just consider the single-step path information. Nowadays, when referring to AS, one actually refers to ant-cycle (AS described in this chapter also refers the ant-cycle version). The principle of AS is introduced as follows.

3.2.1 Tour construction

In AS, m artificial ants concurrently build a tour of the TSP. First, the m ants are put on randomly n chosen cities, which is also known as the scale of the problem. At each

construction step, ant *k* applies a probabilistic action choice rule to decide which city to visit next. Evidently the next visit city *j* must be in the feasible neighborhood of ant *k*. Due to visit each city only once, so this neighborhood structure N_i^k is restricted by M^k which is used to store information of ant *k* about the path it followed so far. The following is the path chosen by the probability formula:

$$P_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{if } j \in N_i^k; \\ 0, & \text{otherwise}; \end{cases} \tag{7}$$

By this probabilistic rule, the probability of choosing a particular arc(*i*, *j*) increases with the value of the associated pheromone trail τ_{ij} and of the heuristic information value η_{ij}. The heuristic information value $\eta_{ij} = 1/d_{ij}$ represents a pre-given inspiration information which describes the objective conditions of the path outside. Pheromone trail τ_{ij} is the key factor in the tour construction and it represents experience which comes from the previous generation. Last, α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information. Each ant *k* has a memory storage M^k and it records in accordance with the order in which they visit all the cities visited. This M^k is used to define the feasible neighborhood N_i^k in the construction rule. In addition, the memory M^k allows ant *k* both to compute the length of the tour T^k it generated and to retrace the path to deposit pheromone. Although the solution of the whole construction is parallel, there are two different ways of implementing it: parallel and sequential solution construction. In the parallel implementation, at each construction step all the ants move from their current city to the next one, while in the sequential implementation an ant builds a complete tour before the next one starts to build another one. These two methods are equivalent in AS, because the pheromones are released only after *m* ants constructing a complete solution, they do not significantly influence the algorithm's behavior. However, in every step of ants moving if the local pheromone updating is added, then the effect of these two methods is different, such as ACS.

3.2.2 Pheromone trails updating

After all the *m* ants have constructed their tours, the pheromone trails are updated. First step is pheromone evaporation, and each edge of the pheromone is to evaporate according to pheromone evaporation rate ρ. Pheromone evaporation is implemented by

$$\tau_{ij} \leftarrow (1-\rho) * \tau_{ij}, \quad \forall (i,j) \in L \tag{8}$$

The parameter ρ ($0 \le \rho \le 1$), which represents the pheromone evaporation rate, is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to forget bad decisions previously taken. Actually if an arc is not chosen by the *m* ants then its associated pheromone value decreases exponentially in the number of iterations.

After pheromone evaporation, all the *m* ants deposit pheromone on the arcs they have crossed in their tours:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad , \qquad \forall (i,j) \in L \tag{9}$$

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant *k* deposits on the arcs it has visited. It is defined as:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q / C^{k} \ , & \text{if arc } (i,j) \text{ belongs to } T^{k}; \\ 0 \quad , & \text{otherwise;} \end{cases} \tag{10}$$

where C^{k}, the length of the tour T^{k} built by the *k*-th ant, is computed as the sum of the lengths of the arcs belonging to T^{k}. By means of Eq. (10), the better an ant's solution is, the more pheromone the arcs belonging to this tour receive. This ensures the probability of choosing good path.

3.3 Elitist Ant System (EAS)

Elitist Ant System is the first improvement on the initial AS, which is called elitist strategy for Ant system (EAS) (Dorigo & Stützle, 2004; Dorido *et al.*, 1991; Colorini *et al.*, 1992; Dorigo, 1992). EAS gives the ant which has constructed so far the best path solution the elite logo, sets the so far the best path T^{bs}, and provides strong additional reinforcement to the arcs that is belong to the best tour T^{bs} found since the start of the algorithm.

In tour construction, the methods in EAS is the same as the methods in AS. In pheromone updating, the pheromone evaporation formula is the same as (3.4).The additional reinforcement of tour T^{bs} is achieved by adding a quantity e / C^{bs} to its arcs, where *e* is a parameter that defines the weight given to the best-so-far tour T^{bs}, and C^{bs} is its length. The following is the equation for the pheromone deposit:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e * \Delta\tau_{ij}^{bs}, \qquad \forall (i,j) \in L \tag{11}$$

where $\Delta\tau_{ij}^{k}$ is defined as in Eq. (10) in AS and $\Delta\tau_{ij}^{bs}$ is defined as follows:

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1 / C^{bs}, & \text{if arc } (i,j) \text{ belongs to } T^{bs}; \\ 0 \quad , & \text{otherwise.} \end{cases} \tag{12}$$

The key is that the EAS has adopted a daemon action, which is the additional incentive of the elite ants. Although this operation belongs to the pheromone updating steps, it is a kind of additional guidance to the overall operation. One can image like this: After all the ants including the elite ant depositing pheromone on the arcs they have crossed over their tour, the elite ant give the T^{bs} additional release of pheromones.

Compared with EAS, the pheromone updating mechanism in the AS is weak indeed. Sometimes, the optimal path may be with a very small difference between the paths which are not so satisfactory, and the mechanism in the AS can not make a good distinction between them. This is because of the simple form of the pheromone depositing formula which allows all ants use the same weight for depositing pheromone. Usually, the level for the algorithm to explore the overall optimal solution is not enough. The EAS with the

parameter *e* determining the weight gives the best-so-far tour, that is, the best-so-far solution has been improved in the course of the search status, and the algorithm attempts to search a better solution which around the best-so-far solution. From the other side of the coin, the EAS concentrates a smaller search space which is compressed from original search space. Such a smaller search space may have better solutions. The mechanism increases the probability for finding overall optimal solution, and at the same time it also speeds up the convergence. Experiments later in this chapter show that parameter *e* needs to be selected with a reasonable value: an appropriate value for parameter *e* allows EAS to both find better tours and have a lower number of iterations. If parameter *e* is too small, the elitist strategy will not have much effect because of the low discrimination for better ants, while if the parameter *e* is too large, the algorithm will be too dependent on the initial best-so-far tour, and have rapid convergence to a small number of local optimal solutions, which weaken algorithm's ability of exploration.

3.4 Rank-Based Ant System (AS_{rank})

Another improvement over AS is the rank-based version of AS (Dorigo & Stützle, 2004; Bullnheimer *et al*., 1997): AS_{rank}. In AS_{rank}, before updating the pheromone trails, the ants are sorted by increasing tour length and the quantity of pheromone deposited by an ant is weighted according to the rank of the ant. Usually in each iteration, only the (*w*-1) best ranked ants and the ant produced the best-so-far tour (this ant is not necessarily is belong to the set of ants of the current algorithm iteration) are allowed to deposit pheromone. The best-so-far tour gives the strongest feedback, with weight *w*; the *r-th* best ant in the current iteration contributes the pheromone updating with the value $1/C^r$ multiplied by a weight given by max {0, w-r}.

In tour construction, the methods in AS_{rank} are the same as the methods in AS. In pheromone updating, the pheromone evaporation formula is the same as in Eq. (8). The AS_{rank} pheromone update rule is:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in L \tag{13}$$

where $\Delta\tau_{ij}^{r} = 1/C^r$ and $\Delta\tau_{ij}^{bs} = 1/C^{bs}$; C^r is the length of *r-th* solution and C^{bs} is the same as in Eq. (12)

Compared with AS the AS_{rank} selects *w* ants to deposit pheromone according to the rank of solutions' quality, which is a new improved formula. It completely abolishes the *national pheromone deposit* mechanism, in other word, only the ant who has constructed a good enough solution can deposit pheromone and the amount of pheromone to deposit is decided by the rank. It can reduce the operation of the pheromone and get rid of bad solutions directly. The pheromone depositing mechanism in AS_{rank} is a group of elite ants award, and it is better than the mechanism in AS which just depends on the reciprocal of the tours. Totally, the performance of AS_{rank} is much better than AS.

AS_{rank} and EAS are different on the reward strategy of elitist ants. EAS just give the best-so-far solution an additional incentive, although it can find the good solutions, indirectly it is greatly weakened or even abandoned the second-best-so-far solutions whose neighborhood may have better solutions. In $AS_{rank,}$ the algorithm gives a group of elitist ants award, but the award is according to the rank of solutions. On the one hand the mechanism takes into

account the importance of the best-so-far solution, which ensures that the experience of the leading elitist ant is retained; on the other hand it considers the neighborhood of sub-optimal solutions, that is, it increases ability to explore the optimal solution.

3.5 MAX-MIN Ant System (MMAS)

Max-Min Ant System is a constructive amendment to AS (Dorigo & Stützle, 2004; Stützle & Hoos, 1996; Stützle & Hoos, 1997; Stützle & Hoos, 2000). It is one of the best ACO algorithms. There are four main modifications with respect to AS:

1. It strongly exploits the best tours found: only either the iteration-best ant, that is, the ant produced the best tour in the current iteration, or the best-so-far ant is allowed to deposit pheromone.
2. It limits the possible range of pheromone trail values in the interval $[\tau_{\min}, \tau_{\max}]$.
3. The pheromone trails are initialized to the upper pheromone trail limit together with a small pheromone evaporation rate.
4. Pheromone trails are reinitialized when the system approaches are stagnated or when no improved tour has been generated for a certain number of consecutive iterations.

The first point strongly exploits the best tours found, but may lead to a stagnation situation in which all the ants follow the same tour. To increase the effect, the second modification is introduced by MMAS. It makes that the pheromone in each arc does not accumulate too large or consume too small, so that it could ensure the sustainability of exploration. The third point makes MMAS have a stronger ability to explore at the initial stage. The fourth point introduces a new mechanism which can be applied to all the ACO algorithms, that is, through the resumption of initial pheromone and re-search thus it increases the possibility of finding the optimal solution.

Since only the optimal solution is allowed to deposit pheromones, the formula about pheromones depositing is very concise:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \qquad \forall (i,j) \in L \tag{14}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}$. C^{best} is the length of optimal tour. It can be the best-so-far tour or iteration-best tour.

In general, in MMAS implementations both the iteration-best and the best-so-far updating rules are used, in an alternate way. Obviously, the choice of the relative frequency with which the two pheromone updating rules are applied has an influence on how greedy the search is: When pheromone updating is always performed by the best-so-far ant, the search focuses very quickly around the best-so-far solution, whereas when it is the iteration-best ant that updates pheromones, the number of arcs received pheromone is larger and the search is less directed.

In MMAS, there are two default daemon actions. One is the pheromone trails limits, the other one is the pheromone trails re-initialization. In MMAS, lower and upper limits $\tau_{\min}$ and $\tau_{\max}$ on the possible pheromone values of any arc are imposed in order to limit the probability p_{ij} of selecting a city *j* when an ant is in city *i* in the interval $[P_{\min}, P_{\max}]$, so that it could avoid searching stagnation and enhance the ability to explore. In the long run, the upper pheromone trail limit on any arc is bounded by $1/\rho C^*$, where C^* is the length of the optimal tour. Based on this result, MMAS uses an estimate of this value of $1/\rho C^{bs}$, to define

$\tau_{\max}$: each time a new best-so-far tour is found, the value of $\tau_{\max}$ is updated. The lower pheromone trail limit is set to $\tau_{\min} = \tau_{\max} / a$, where a is a parameter which decides the proportion of upper limit and lower limit. In MMAS, in order to avoid stagnation, the lower pheromone trail limits play a more important role than upper limits. Pheromone trail re-initialization is typically triggered when the algorithm approaches the stagnation behavior or if no improved tour of a given number of algorithm iterations is found. It can increase the exploration of paths that have only a small probability of being chosen. By the way, the pheromone trail re-initialization also increases the probability of finding the global optimal solution (equivalent to cumulative probability).

3.6 Ant Colony System (ACS)

Ant Colony System (ACS) is an extension of AS (Dorigo & Stützle, 2004; Dorigo & Gambardella, 1997). The ACS exploits the search experience accumulated by the ants more strongly than AS. The rule is called pseudorandom proportional rule. When located at city *i*, ant *k* moves to a city *j*, the rule is given by

$$j = \begin{cases} \arg \max_{l \in N_i^k} \{\tau_{il}[\eta_{il}]^{\beta}\}, & \text{if } q \le q_0; \\ J, & \text{otherwise;} \end{cases} \tag{15}$$

where q is a random variable uniformly distributed in $[0,1]$, J is a random variable selected according to the probability distribution given by Eq.(5) with a = 1. $q_0 (0 \le q_0 \le 1)$ is a parameter with which the ant makes the best possible move as indicated by the learned pheromone trails from the heuristic information, while with probability $(1 - q_0)$ it performs a biased exploration of the arcs.

This pseudorandom proportional has strong artificial operability because the parameters q_0 can be set to guide the algorithm's preference. By tuning the parameter q_0, it is allowed to modify the degree of exploration and to select whether to concentrate the search of the system around the best-so-far solution or to explore other tours.

In ACS only the best-so-far ant is allowed to update pheromone after each iteration including the pheromone deposit and pheromone evaporation. In each time an ant uses an arc to move from city *i* to city *j*, which is called the local pheromone updating to remove some pheromone from the arc to increase the exploration of alternative paths. The global pheromone trail updating is described as the following equation:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \Delta \tau_{ij}^{bs}, \qquad \forall (i,j) \in T^{bs} \tag{16}$$

where $\Delta \tau_{ij}^{bs} = 1 / C^{bs}$. C^{bs} is the length of the best-so-far tour T^{bs}. It is worth to note that the deposited pheromone is discounted by a factor ρ, which results in the new pheromone trail becoming a weighted average between the old pheromone value and the amount of pheromone deposited.

The local pheromone trail updating is described as the following equation:

$$\tau_{ij} \leftarrow (1-\varepsilon)\tau_{ij} + \varepsilon \tau_0, \qquad \forall (i,j) \in L \tag{17}$$

where ε, $0 \le \varepsilon \le 1$, ε is the local pheromone evaporation rate and the τ_0 is the initial pheromone. The effect of the local updating rule is to reduce pheromone trail of an ant in some arc so that the arc becomes less desirable for the following ants. The role of local updating is to strengthen the capacity of artificial ant exploration.

3.7 Application of five ACO algorithms in C-TSP and results analysis

In order to demonstrate and verify the properties of five ACO algorithms mentioned above, in this section, we'll apply them in the Chinese TSP with 31 capital cities, and compare their advantages and disadvantages in the medium-scale discrete optimization problem.

The important parameters of Ant System are as follows. *N*: the number of cities; *m*: the number of artificial ants; *a*: the parameter that controls the relative importance of pheromone trail; β: the parameter that controls the relative importance of heuristic information; ρ: pheromone evaporation rate; *q*: a constant represents the weight of the deposited pheromone; τ_0: initial pheromone trail; NC: preset number of iterations. In addition, in its improved algorithms the addition parameters are as follows. The *e* in EAS : a parameter that defines the weight given to the best-so-far tour. The *w* in AS_{rank}: the number of the artificial ants who are allowed to add pheromone. The *τ_proportion, τ_max, τ_min* and *nowbest_p* in MMAS: *τ_proportion* is a parameter which decides the proportion of upper limit and lower limit; *τ_max* is the upper limit of pheromone; *τ_min* is the lower limit of pheromone; *nowbest_p* is the frequency of selecting best-so-far tours rule of depositing pheromone. The *pbest* and *local_p* in ACS: *pbest* is a probability of choosing right path; *local_p* is the local pheromone evaporation rate.

A candidate list is first built to restrict the number of available choices considered at each construction step. In general, candidate lists contain a number of the best rated choices according to some heuristic criterion. First, configure for each city a nearest neighbor list which records the other cities sorted in ascending order by distance; Second, build the candidate lists for each city and set the parameter $nn(0 \le nn \le n)$ which decides the number of the nearest neighbors needed; Last, get the cities which are previous *nn* cities in the nearest neighbor list into the candidate list for each city. When an ant constructs solution, it gives priority to the candidate list of cities. In fact, the ant usually considers from the first city in the candidate lists that has not been visited and selects with random probability rule. When all the cities in the candidate list have been visited by an ant, the ant will consider other cities and select the city which has a maximum value of $[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}$, that is, the ant selects the city which is the best experience-oriented one. Set the parameter *nn* to a constant which is below the number of cities *n*, especially for a small value, the algorithm's speed will be improved significantly. The mechanism is feasible, because in TSP a good path can not appear a city *i* connects another city *j* which has a long distances from city *i*. In other words, the ant in city *i* should choose the city *j* which nears the city *i*. It is worth to note that the parameter *nn* is important for candidate list. If the value of *nn* is too large, the effect of speeding up algorithm will be weakened. On the other hand, a too small *nn* will make the performance of algorithm very poor. However, it should be noted that the use of truncated nearest-neighbor lists can make it impossible to find the global optimal solution. The global optimal solution does not mean to be the combination of cities in the candidate lists. Perhaps

in order to achieve the best, ants in some cities should choose far way cities to go and these cities are not in departure cities' candidate lists.

The steps of Ant System (AS) algorithm is as follows (in the case no candidate lists):

Step 1. Enter an actual TSP, get the scale n of the problem, and transform the instance into a symmetric distance matrix distance (set the diagonal elements with small values to prevent the situation of NAN.)

Step 2. Initialize all the parameters, including m, a, β, ρ, q, τ_0 and NC. Set the iteration number to 0.

Step 3. Initialize storage variable including best-so-far solution *nowbest_opt* = 2 * *vicinity* (*vicinity* is the solution comes form the nearest algorithm), best-so-far path *nowbest_path* = zeros (1, n), pheromone tails matrix τ = ones(n) * τ_0 , and the matrix which describes the importance of heuristic information τ_β = dist.^(- β).

Step 4. Begin to circulate, and set the iteration number nc = nc +1.

Step 5. The starting cities of ants are randomly distributed as begin_city = randperm (n); initialize tabu list *tabu* = ones (m, n) and taboo the starting city; initialize path matrix *path* = zeros (m, n) and add the starting city into the first column; build the matrix that describes the importance of pheromone τ_α = τ.^ a; build the comprehensive weight matrix $\tau_d = \tau_\alpha .* \tau_\beta$.

Step 6. Ant walking steps *step*=*step*+1 (initialize *step* = 0 , and $step \le n-1$).

Step 7. Artificial ants' label k = k +1 (initialize k = 0, and $k \le m$)

Step 8. Choose the next city in accordance with the probability formula, taboo the selected city in the ant taboo list *tabu* (k, :), and add this chosen city into the ant path sequence *path* (k, *step* +1).

Step 9. If $k < m$, then turn to step 7; otherwise turn to step 10.

Step 10. If $step \le n-1$, then turn to step 6; otherwise turn to step 11.

Step 11. According to the pheromone updating formula of AS, update the pheromone trails τ, and update the optimal solution as best-so-far solution *nowbest_opt* and the optimal path as best-so-far tour *nowbest_path* which includes the cities of the best-so-far tour.

Step 12. If the iteration number nc < NC, then turn to step 4; otherwise end the operation, export the data and image results.

Remark: The steps described above are just the AS algorithm. The improved algorithms mentioned above should have some changes, which are mainly in the steps of the parameter settings, constructing solutions and pheromone updating. In the process of building a solution, they use parallel mechanism. To add the candidate lists to the algorithm, first of all one needs to add a step in one of the first two steps in the algorithm: list the neighbor cities in ascending order by distance and configure nearest neighbor list for each city, set the parameter *Neighbor_num* of the candidate list, and then get the candidate list *Neighbor_list* from the nearest neighbor list; followed by modifying the step 8: first of all, begin to consider the first city in the candidate list that does not be visited, and select the target city of the next step with probability rules. When the cities of the candidate list have all been visited one compares the values $\tau_d = \tau_\alpha .* \tau_\beta$ of all the other remaining cities, and select the largest one as the next target city; taboo the selected city in the ant taboo list *tabu* (k,:), and add this city into the ant path sequence *path* (k, step +1).

Table 2 is the summary of the experimental results of the five ACO algorithms including adding the mechanism of the candidate list and pheromone re-initialization, *etc.*. Each system runs for 100 times. In Table 2, the algorithm with "_C" has the mechanism of candidate list, the algorithm with "_R" has the mechanism of pheromone re-initialization, Best or Worst mean the best or worst solution of the 100 solutions. "Opt. rate" is the rate of global optimal solution, Average is the average solution of all solutions. "Average converg." is the average convergence iteration number, and Average time is the average time in 100 running solutions. Relative error is described as follows:

$$relative\ error = \frac{|\ average\ \ solution - global\ optimal\ \ solution\ |}{global\ optimal\ \ solution} \tag{18}$$

where the *global optimal solution* = 15404, and the average solution is the average solution in 100 running solutions.

Algorithm	Best	Worst	Opt. rate	Average	Relative error	Average converg.	Itera. No.	Average time
AS	15420	15669	0	15569.05	1.071%	1405.95	3000	12.221
AS_C	15420	15620	0	15548.3	0.937%	1380.11	3000	6.044
EAS	15404	15625	48%	15447.4	0.282%	1620.48	4000	16.409
EAS_C	15404	15593	52%	15437.62	0.218%	1606.95	4000	7.954
AS_{rank}	15404	15593	63%	15413.74	0.063%	1857.19	4000	16.662
AS_{rank}_C	15404	15520	65%	15408.05	0.026%	1747.07	4000	8.349
MMAS	15404	15593	55%	15428.54	0.159%	2371.96	5000	20.386
MMAS_C	15404	15593	57%	15424.32	0.132%	2166.56	5000	10.384
MMAS_R	15404	15593	68%	15418.48	0.094%	2984.5	8000	23.951
MMAS_C_R	15404	15520	73%	15418.75	0.096%	2655.68	8000	10.799
ACS	15404	15779	40%	15442.42	0.249%	2889.67	10000	5.546
ACS_C	15404	15745	40%	15445.51	0.269%	2708.9	10000	4.817

Table 2. The summary of the test results when ACO algorithms are applied to the C-TSP

One can obtain from Table 2 the following results:

1. In the test of all 12 kinds of algorithms, from the column "Best" solutions one can see except AS and AS_C which add the mechanism of candidate list to AS, the other 10 kinds of algorithms can detect the global optimal solution 15404.
2. Along all the algorithms, from the column "Opt. rate" one can see max-min ant system which has the mechanisms of the candidate list and pheromone initialization added in MMAS_C_R owns the highest rate of excellent, that is 73%, followed by MMAS_R;

Except AS and AS_C, ACS and ACS_C which add the mechanism of candidate list to ACS have the worse performance, that is 40%.

3. Compare these five algorithms without any mechanism, from the "Opt. rate " one can also see that rank based version AS, AS_{rank} has the highest rate of excellent, that is 63%, followed by MMAS, EAS, ACS, and AS.
4. Compare these five algorithms without any mechanism to those five algorithms with mechanisms of the candidate list respectively, that is, compare XXX with XXX_N, one can see from the optimal rate and average running time except ACS and ACS_C are not obvious, the mechanism of the candidate list slightly improves the performance of the algorithms and greatly reduces the running time.
5. Compare the four cases of MMAS, that is, MMAS, MMAS_C, MMAS_R and MMAS_C_R, one can see from the optimal rate that every mechanism can improve the performance of algorithms, and max-min ant system only with the mechanisms of pheromone initialization the candidate list has a better performance than only with the mechanisms of the candidate list. Of course, max-min ant system with two mechanisms at the same time has the best performance.

4. Conclusions

This chapter has analyzed the characteristics of the SA and PSO for solving the C-TSP problem. Combined the fast convergence speed of PSO with the good local search ability of SA, a hybrid algorithm has been proposed. Numerical simulations show that the proposed algorithm is more efficient in C-TSP than single PSO and SA, respectively. Generally speaking, when ACO algorithms are applied to the TSP instance C-TSP, elitist strategy for ant system, rank based version AS, max-min ant system, ant colony system show better performance, they have a certain percentage to find the global optimal solution, but ant system fails to find global optimal solution. In all these systems, max-min ant system which has the mechanisms of the candidate list and pheromone initialization added in shows the best performance in the C-TSP.

5. Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 60774098.

6. References

Eberhart, R. C. & Kennedy, J. (1995). A New Optimizer Using Particles Swarm Theory. *Proceedings of the 6th Int'1 Symposium of Micro Machine and Human Science*, pp. 39-43, 0-7803-2676-8, Nagoya, Oct. 4-6

Yuan, Z. L.; Yang, L. L. & Liao, L. (2007). Chaotic Particle Swarm Optimization Algorithm for Traveling Salesman Problem. *Proceedings of the IEEE International Conference on Automation and Logistics*, pp.1121-1124, 978-1-4244-1531-1, Jinan, China, Aug. 18-21

Xu, Y. H., Wang, Q. W. etc. (2008). An Improved Discrete Particle Swarm Optimization Based on Cooperative Swarms. *International Conference on Web Intelligence and*

Intelligent Agent Technology, pp. 79-82, 978-0-7695-3496-1, Sydney, Australia, Dec. 9-12

Lovbjerg, M.; Rasmussen, T. K. & Krink, T. (2001). Hybrid Particle Swarm Optimizaters with Breeding and Subpopulations. *Proc of the Third Genetic and Evolutionary Computation Conference,* Vol. 24, pp. 469-476, San Francisco, USA

Kang, L. H. & Xie,Y.(1998). Non-numerical Parallel Algorithms—Simulated Annealing Algorithm, Science Press, 703003736, Beijing

Zhong, W. L.; Zhang, J. & Chen, W. N.(2007). A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem. *IEEE Congress on Evolutionary Computation*, pp. 3283-3287, 978-1-4244-1339-3, Singapore, Sept. 25-28

Xiao, J.M.; Li, J.J. & Wang, X.H. (2004). A Modified Particle Swarm Optimization for Travelings Salesman Problems. *Computer Engineering and Applications*, Vol. 40, No. 35: pp. 50-52

Li, L. L.; Zhu, Z. K. & Wang, W.F. (2008). A Reinforced Self-Escape Discrete Particle Swarm Optimization for TSP. *Second International Conference on Genetic and Evolutionary Computing,* pp. 467-470, 978-0-7695-3334-6, Jinzhou, China, Sept. 25-26

Huang, L.; Wang, K. P. & Zhou, C.G.(2003). Particle Swarm Optimization For Traveling Salesman Problems. *journal of JiLin university (science edition)*, Vol. 41, No.10: pp. 477-480

Dorigo, M.; & Stützle, T.(2004). Ant Colony Optimization. MIT Press, 978-0-262-04219-2 Boston

Dorigo, M.; Maniezzo, V. & Colorni, A.(1991). Positive feedback as a search strategy. Technical report, *Dipartmento di Elettronica*, Politecnico di Milano, Milan, pp. 91-96

Colorini, A.; Dorigo, M. & Maniezzo, V.(1992). Distributed optimization by ant colonies. In F.J. Varela&P.Bourgine (Eds), *Proceedings of the First European Conference on Artificial Life*, pp.134-142. Cambrige, MA, MIT Press

Dorigo, M. (1992) Optimization, Learning and Natural Algorithms [in Italian].PhD thesis, *Dipartimento di Elettronica,* Politecnico di Milano, Milan

Bullnheimer, B.; Hartl, R.F. & Strauss, C. (1997) A new rank based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, Vol. 7, No. 1: pp. 25-38

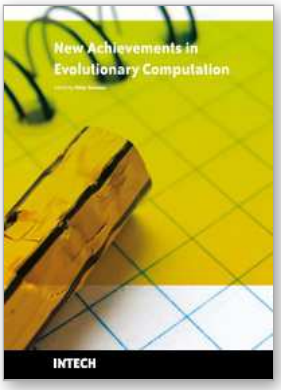
Stützle, T. & Hoos, H. H. (1996). Improving the Ant System: A detailed report on the MAX-MIN Ant System. *Technical report AIDA-96-12*, FG Intellektik, FB Informatik, TU Darmstadt, Germany

Stützle, T. & Hoos, H. H.(1997). The MAX-MIN Ant System and local search for the traveling salesman problem. In T.Bäck, Z.Michalewicz, & X.Yao (Eds.), *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97),* pp.309-314, Piscataway, NJ, IEEE Press

Stützle, T. & Hoos, H. H. (2000). MAX-MIN Ant System. *Future Generation Computer Systems*, Vol. 16, No. 8: pp. 889-914

Dorigo, M. & Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. *BioSystems*, Vol. 43, No.2: pp. 73-81

Dorigo, M. & Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Comtutation*, Vol. 1, No. 1: pp. 53-66



New Achievements in Evolutionary Computation

Edited by Peter Korosec

ISBN 978-953-307-053-7
Hard cover, 318 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Evolutionary computation has been widely used in computer science for decades. Even though it started as far back as the 1960s with simulated evolution, the subject is still evolving. During this time, new metaheuristic optimization approaches, like evolutionary algorithms, genetic algorithms, swarm intelligence, etc., were being developed and new fields of usage in artificial intelligence, machine learning, combinatorial and numerical optimization, etc., were being explored. However, even with so much work done, novel research into new techniques and new areas of usage is far from over. This book presents some new theoretical as well as practical aspects of evolutionary computation. This book will be of great value to undergraduates, graduate students, researchers in computer science, and anyone else with an interest in learning about the latest developments in evolutionary computation.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Shuang Cong, Yajun Jia and Ke Deng (2010). Particle Swarm and Ant Colony Algorithms and Their Applications in Chinese Traveling Salesman Problem, New Achievements in Evolutionary Computation, Peter Korosec (Ed.), ISBN: 978-953-307-053-7, InTech, Available from: http://www.intechopen.com/books/new-achievements-in-evolutionary-computation/particle-swarm-and-ant-colony-algorithms-and-their-applications-in-chinese-traveling-salesman-proble

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821