Applied Soft Computing 10 (2010) 653–660

Contents lists available at ScienceDirect

Applied Soft Computing

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

An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem☆

Jingan Yang [a,b,*], Yanbin Zhuang [b,c]

[a] Institute of Artificial Intelligence, School of Computer and Information Sciences, Hefei University of Technology, Hefei 230009, PR China
[b] Changhzou Key Laboratory of Software Technology and Applications, Changzhou 213002, Jiangsu Province, PR China
[c] School of Computer and Information Engineering, Changzhou Institute of Technology, Changzhou 213002, Jiangsu Province, PR China

ARTICLE INFO

Article history:
Received 5 March 2008
Received in revised form 25 November 2008
Accepted 29 August 2009
Available online 17 October 2009

Keywords:
Ant colony optimization
Combinatorial optimization problem
Mobile agent routing problem
Premature convergence probability
Traveling salesman problems
Simulated annealing

ABSTRACT

This paper presents an improved ant colony optimization algorithm (IACO) for solving mobile agent routing problem. The ants cooperate using an indirect form of communication mediated by pheromone trails of scent and find the best solution to their tasks guided by both information (*exploitation*) which has been acquired and search (*exploration*) of the new route. Therefore the premature convergence probability of the system is lower. The IACO can solve successfully the mobile agent routing problem, and this method has some excellent properties of robustness, self-adaptation, parallelism, and positive feedback process owing to introducing the genetic operator into this algorithm and modifying *the global updating rules*. The experimental results have demonstrated that IACO has much higher convergence speed than that of genetic algorithm (GA), simulated annealing (SA), and basic ant colony algorithm, and can jump over the region of the local minimum, and escape from the *trap of a local minimum* successfully and achieve the best solutions. Therefore the quality of the solution is improved, and the whole system robustness is enhanced. The algorithm has been successfully integrated into our simulated humanoid robot system which won the fourth place of RoboCup2008 World Competition. The results of the proposed algorithm are found to be satisfactory.

1. Introduction

In recent years, neural networks, cell automation, and evolutionary computation have received increasing attention in the intelligent computation technologies. They are developed by simulating natural phenomenon. These technologies are highly parallel, self-adaptive, self-organized, and are full of vigor and vitality in computation intelligence.

Ant colony optimization (ACO) is a technique of problem solving inspired by the behavior of ants in finding paths from the nest to food and a new search *metaphor* for solving combinatorial optimization problems, and has been unexpectedly successful in recent years [1–5]. The ants cooperate using an indirect form of communication mediated by pheromone trails of scent and find an optimal solution to their tasks guided by both information (exploitation) which has been acquired and search (exploration) of the new route. Therefore the premature convergence probability of the system is lower. Dorigo et al. [6,7] proposed ant colony algorithm based on the *bionics* research achievement foundation and applied successfully this algorithm to solve different combinatorial optimization problems, such as the TSP problem [8–11], the quadratic assignment problem [12,13], the job-shop scheduling problem [14], sequential ordering problem [15], the vehicle routing problem (VRP)[16–19], and data mining [20] and network routing. Stützle and Hoos introduced Max–Min Ant System (MMAS) [21], a modification of an ant system applied to traveling salesman problem (TSP). These authors explicitly introduced two important user-defined parameters, a maximum and minimum trail levels, whose values are chosen in a problem-dependent way in order to restrict possible trail values to the interval $[\tau_{\min}, \tau_{\max}]$. Moreover, the MMAS controls the trail levels (initialized to the maximum value), only allowing the best ant at each iteration to update trails, thus providing a positive feedback on its results. Trails that never or rarely receive reinforcements will continuously lower their trail intensity of scent left on routes and will be selected more and more rarely by the ants, until they reach the value. The parameters, $\tau_{\min}$ and $\tau_{\max}$, are used to counteract *premature stagnation of search* for avoiding early convergence to a local minimum, maintaining some kind of *elitist strategy* at the same time. ACO is a new search *metaphor* for solving the combinatorial optimization problems, and has been successfully

☆ This paper is supported in part by Changhzou Key Laboratory of Software Technology and Applications, Jiangsu Province, PR China.
* Corresponding author at: Apt. 502, Building 204, South Campus, Hefei University of Technology, Hefei 230009, Anhui Province, PR China.
Tel.: +86 519 85211932; fax: +86 519 85217723.
E-mail address: jayang@mail.hf.ah.cn (J. Yang).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.08.040

applied to many complex optimization problems, especially applied to the complex discrete optimization problems. The essence of the ant colony optimization process lies in: (1) the selecting mechanism: the more *pheromone trail* on the route, the larger probability of choosing this route; (2) the updating mechanism: the pheromone on the route can get strong when ant's pass more and moreover simultaneously also gradually volatilizes and vanishes *as time goes on*; (3) the coordinated mechanism: the mutual communication and coordinated operation between the ants is performed in fact through the "pheromone". The ant colony algorithm has fully used such optimized mechanism to finally find the best solution through information exchange and mutual cooperation between the individuals, and enable this algorithm to have a very strong ability to discover the best solutions.

As a new completely distributed computation tool, a mobile agent can overcome some shortcomings of *Client/Server* by transmitting their own code and states to the remote host computer and by the way in which it is carried out in local computers using the remote host computer. Thus a mobile agent is one of the main research directions of the distributed computation [22,23]. The mobile agent may move from one host computer to another host computer according to requirements, and complete the global distributed computation tasks using local computation ability and the resources of the various host computers. How do the mobile agent move between the host computers according to the kind of strategies in the process to execute tasks is an important question, how to complete the computation task efficiently and fast, also is the mobile agent routing problem discussed in this paper. This paper proposes an improved ant colony algorithm. This method investigates a mutation genetic operator and modifies the global updating rules and can solve some mobile agent routing problems, and escape from the trap of a local minimum, and speed up the convergence rate. Our algorithm has some excellent properties of robustness, self-adaptation, parallelism, and positive feedback process owing to introducing the genetic operator and modifying the global updating rules. Therefore the mobile agents can perform the distributed computation tasks with optimization efficiency at a fast rate. The experimental results here also have demonstrated that the improved ant colony system may solve some complex combinatorial optimization problems through cooperation between the ants (also regarded as an agent) and indicates some inherent advantages of robustness, self-adaptation, parallelism and positive feedback. The positive feedback process accounts for rapid discovery of good solutions, and the distributed computation avoids premature convergence, and the greedy heuristic helps find acceptable solutions in the early stages of the search process. But we also may introduce some specific questions-correlated domain knowledge, and provide some fully novel ideas for solving certain complex estimation problems.

The rest of this paper is organized as follows: Section 2 outlines an ant colony algorithm; Section 3 describes how to solve the mobile agent routing problem. We solve the mobile agent routing problem using the improved ant colony algorithm in Section 4 and show the experimental results in Section 5. Section 6 summarizes our algorithm and suggests our further work.

2. Ant colony algorithm

Ant colony optimization (ACO) is a promising metaheuristic and a great amount of research has been devoted to its empirical and theoretical analysis. The ACO has been and continues to be a fruitful paradigm for designing effective combinatorial optimization algorithms.

The ants can carry on indirect communication through a chemical substance *pheromone* [24–27], which is accumulative and also evaporative, but finally achieve the cooperative goal. The ants travel a shorter path on which the pheromone trail accumulates faster than on the longer one. Therefore, the faster the pheromone trails increase on the short path, then the greater the probability that the ants travel this path also. The pheromone trails can deposit unceasingly and evaporate as *the time goes on*. At the same time, the ants also can unceasingly secrete the pheromone in their travel process, thus the pheromone trails can be updated unceasingly. The pheromone trails on the path which few ants travel decrease more and more, but the pheromone trails on the path which more ants travel increase more and more.

When an ant is looking for a new food source, each ant is not directed by the pheromone trails and wanders randomly at the initial stage, therefore the path search indicates the complete randomness. Under the pheromone direction, the probability the successor ants travel the shorter path must also be higher than the probability the ants travel a longer path. These successor ants can secrete the new pheromone on the shorter path again in the traveling process. This causes the pheromone trails on the shorter path to continue increasing, but on the longer path that not many ants travel, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromone has to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. This forms a positive feedback process, and finally causes all ants to travel the shortest path, thus enabling the whole ant colony to find the shortest path from the ant nest to a food source.

Ants move by a stochastic local decision making based on two parameters, called trail of *scent* and *attractiveness*. Each ant incrementally constructs a solution to the problem as it moves. When an ant completes a solution or this solution is under construction, the ant evaluates the solution and modifies the trail value of this solution. This pheromone trail will direct the next ant search. The characteristics of the ant colony optimization algorithms are their explicit use of elements of previous solutions. In fact, they drive a constructive low-level solution, but including it in a population framework and randomizing the construction in a Monte Carlo method. Monte Carlo combination of the different solution elements is suggested also by genetic algorithms [28,29], but in the case of the ACO, the probability distribution is explicitly defined by previously obtained solution components. The particular way of defining components and associated probabilities is problem-specific, and can be designed in different ways, facing a trade-off between the specificity of the information used for the conditioning and the number of solutions which need to be constructed before effectively biasing the probability distribution to favor the emergence of good solutions [30].

3. The mobile agent routing problem

The distributed computation model based on a mobile agent routing problem can overcome not only drawbacks of the traditional *client/server* scheme, but also the flexibility, the stability, the extensibility of the ant colony system are suitable well for solving complex and modern distributed computation tasks. Thus the system may widely be applied to the distributed computation tasks, motion computation, the distributed information inspection as well as electronic commerce and so on. In the process for the mobile agents to carry out the distributed computation tasks, they are required to make careful choice to move to some host computer according to the task demand and the current network situation and complete one or several subtasks. After completing the subtasks, they move again to another host computer, and continue other subtasks, repeatedly until the

mobile agents complete the entire distributed computation tasks or fail. Therefore, for the distributed computation, and especially for the parallel processing of the distributed information, how to plan the motion route of a mobile agent appears especially important. Therefore, this paper defines the mobile agent routing problem as follows.

Definition 3.1. Suppose $\text{ant}_i\{i = 1, 2, \ldots, k, \ldots, l\}$ represents a set of all ants in an ant colony, then $\text{ant}_k \in \text{ant}_i$ means some ant, l is the number of the ants in the ant colony.

Definition 3.2. τ_0 is defined as the initial pheromone value, $\tau_{ij}(t)$ means the *pheromone trail* laying on the edge (i, j) visited at time t, and $\Delta\tau_{ij}^k$ represents the amount of the trail contributions by ant k that used move (ij) to construct their solution.

Definition 3.3. In ant colony optimization, a combinatorial optimization problem is mapped on an undirected graph $G = (V, E)$, where $V = (0, 1, 2, \ldots, n-1)$ is the set of vertices and $E = \{(i, j)|i, j \in V\}$ is the set of unordered pairs of distinct vertices, called edges. The graph G is called constructed graph.

Definition 3.4. (Mobile agent routing problem—MARP): suppose we have n host computers in the distributed computation environments, their serial numbers are $0, 1, \ldots, n-1$ respectively and a mobile agent sets out from the host computer 0 for carrying out some task. Assume the time an mobile agent needs from the host computer i to the host computer j is $d(i, j)$, the probability that the mobile agent completes the task in the ith host computer is p_i, then time delay in the host computer i is t_i, and $p_0 = 0$, $t_0 = 0$. If an agent completes the task in some host computer, then the agent directly returns to the initial host computer 0, but no longer visits other host computers. If an agent cannot complete the task, the agent continues moving to another host computer until it completes the task or fails, and each host computer is visited at most one time on the way. For solving the mobile agent routing problem, we are required to propose an effective algorithm so that *time* for an agent to complete the task in a series of host computers is *shortest*.

4. Solving MARP and solution construction

From the definition of the MARP, as we know, when a mobile agent can not complete tasks in all host computers and finally returns to the initial host computer, then MARP would degenerate to TSP. Therefore TSP is an exceptional case of the MARP. Obviously the MARP is a complex combinatorial optimization question which has very high time complexity and the spatial complexity. So the methods for solving the MARP are required to have auto-adaptive, self-learning, distributed, and parallel characteristics. Thus within the time scope to be accepted we can reach the optimization solution or the approximate optimization solution of the problems. The ant colony algorithm discussed in paper not only has these congenital characteristics, moreover also has some advantages of positive feedback, related-domain knowledge that may be introduced and so on. Therefore ant colony algorithm is extremely suitable for solving the MARP and the combinatorial optimization problems.

4.1. Basic ant system

The algorithm that Dorigo first proposed is the basic ant algorithm [30]. Each ant may be used to represent one mobile agent when we apply the basic ant algorithm to solve the MARP. According to the definition of the MARP, the ants prefer to choose those routes on which there is higher pheromone trail concentration, that takes a shorter and is of a higher probability, but also must first consider these host computers which have high probability to complete the tasks, because it is very possible for them to complete already the pre-assigned tasks after the ants visited certain host computes. From this the ants can directly return to the initial host computer, but need not continue visiting other host computers. According to this basic idea and the probability given by Eq. (1), when the ant k travels, the probability that the ant k chooses the next host computer to be visited is:

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)][p_s/d(r,s)\cdot t_s]^\beta}{\sum_{u\in J_i(r)}[\tau(r,u)][p_u/d(r,u)\cdot t_u]^\beta} & s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where r is the current host computer which the ant is at, $p_k(r, s)$ is the probability which ant k chooses the s th host computer as the next destination. The $\tau(r, s)$ is the pheromone trail on the route between the r th host computer and the s th host computer (Suppose the pheromone trail on all routes is a constant at the initial trail level). $J_k(r)$ is the set of the host computers that remain to be visited by ant k positioned in the r th host computer, $\beta(0 \leq \beta \leq 1)$ is a parameter to control the trade off between visibility (*constructive heuristic*) and pheromone trail concentration. From (1) we have seen, the more pheromone trails on the route that the ants choose is, the more shorter the delay time is, the probability on which the route the ants choose is higher. This also just is the precise natural law which the actual ant colony system represents. The pheromone trails on the route can evaporate as time goes on, and also the pheromone trails in the route can be updated after all ants completed their tours respectively and returned to the initial host computer. The updated scope is in inverseproportion to the respective total time length:

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s) \quad (2)$$

where ρ is a user-defined parameter, m is the number of the ants, and the updated amount of pheromone laid by ant k, $\Delta\tau_k(r, s)$, is computed as

$$\Delta\tau_k(r,s) = \begin{cases} \dfrac{1}{T_k} & \text{if } (r,s) \in \text{tour done by ant } k \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where T_k is the total time for ant k to complete its tour. We can simulate the pheromone trail intensity changes caused by the pheromone volatility in the actual ant colony system and the new pheromone in the routes unceasingly secreted by the ants using Eq. (2). Precisely owing to the pheromone existence which can cause the indirect exchange between the ants (using the choice probability of Eq. (1) and the updated rules of Eq. (2)), thus the whole ant colony system can find the optimization routes. After all ants complete their tour, the amount of pheromone trail is updated according to the rule:

$$\tau_{ij}(t+1) = \rho\cdot\tau_{ij}(t) + \sum_{k=1}\Delta\tau_{ij}^k \quad (4)$$

where ρ is a user-defined parameter that controls the speed of evaporation such that $(1-\rho)$ represents the evaporation of trail between t and $t+1$, $\tau_{ij}(t)$-pheromone trail on the edge (i, j) at time t, and we have

$$\Delta\tau_{ij} = \sum_{k=1}^{m}\Delta\tau_{ij}^k \quad (5)$$

where $\Delta\tau_{ij}^k$ is the amount of trail laid on edge (i, j) by the k th ant, which can be computed as

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{Q}{L_k} & \text{if the ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

where Q is a constant, L_k is the cost of the k th ant's tour (typically length). The ant colony system simply iterates a main loop where m ants construct in parallel their solutions, thereafter updating the pheromone trail levels. The performance of the algorithm depends on the correct tuning of several parameters, namely: β, relative importance of trail and attractiveness, ρ, trail persistence, $t_{ij}(0)$, initial trail level, m, number of ants, and Q, used for defining to be of high quality solutions with low cost.

4.2. Artificial ant colony systems

Artificial ant colony systems improve the basic ant algorithm in the following aspects:

(1) When an ant chooses the next host computer, it no longer completely obeys the previous experience, but it chooses the shorter routes with the higher pheromone trail intensity based on the certain probability by itself:

$$S = \begin{cases} \arg\max \left\{ [\tau(r,u)] \left[\dfrac{p_u}{d(r,u)\cdot T_u} \right]^{\beta} \right\} & \text{if } q \le q_0 \\ \text{choose according to (1)} & \text{otherwise} \end{cases}$$

where q is a random variable uniformly distributed in [0,1] ($q \in [0,1]$), q_0 is a parameter which indicates a balance factor between the information which is accumulated in the moving process of an ant and the path chosen by itself.

(2) Only the ants which find the shorter route currently are allowed to update the pheromone trail density, but not all ants are allowed to update the pheromone trail when they return to the initial host computer, this is called *global update rule*.

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau_g(r,s) \tag{6}$$

where

$$\Delta\tau_g(r,s) = \begin{cases} \dfrac{1}{T_{gb}} & \text{if } (r,s) \in \text{global-optimal path.} \\ 0 & \text{otherwise} \end{cases}$$

Here T_{gb} is the shortest time for ant to complete its tour.

(3) The ants are updating the pheromone trail intensity in the routes which they pass through in the process when the ants march forward (but this is not likely basic ant algorithm, they update the pheromone intensity again after the ants return to the initial host computer), this is called local updated rule.

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau_l(r,s) \tag{7}$$

where $\Delta\tau_l(r,s)$ may be set to the initial pheromone trail concentration τ_0.

4.3. Improved ant colony algorithm

In this section, we focus our attention on investigating a mutation genetic operator by introducing a genetic algorithm into ant colony system and modifying global updating rules based on thorough research on working mechanism of ant colony system, and can solve a mobile agent routing problem. Therefore mobile agents can perform the distributed computation tasks with the optimization efficiency and much higher convergence speed. For ant colony systems have improved the basic ant algorithm. So this system causes an ant no longer to be limited to pheromone trail which is already accumulated in its march process, but makes use of both the information (*exploitation*) which has been acquired and the search (*exploration*) of new routes, thus greatly promoting the system robustness. However in the experimental process, we found that the ant colony system still fortuitously falls into a local minimum sometimes. This case causes the route which the ant completes no longer *to evolve in the direction of the optimization solution* so that the whole system presents *premature convergence*. For ant colony system to escape from a local minimum and avoiding this premature convergence, we consider adding some kind of stochastic perturbation to the solution when the local minimum situation appears. Therefore the solution can escape from *the trap of a local minimum*, and continues to evolve in the direction of the optimization. This paper considers solving this problem in the following two aspects.

4.3.1. Application of genetic algorithm to our algorithm

We introduced genetic algorithm into our algorithm by combining two kinds of the optimization algorithms including the ant colony algorithm and genetic algorithm which all originate from the bionics principle. Genetic algorithm is actually a nonlinear optimal problem and produces inevitably local minima. The parameters of the search space in GA are encoded in the form of a *chromosome-like* structure. A group of these chromosomes constitutes a population. An index of *merit* (fitness value) is assigned to each individual chromosome according to a defined fitness function. A new generation is evolved by a selection technique, in which there is a larger probability of the fittest individuals being chosen. These chosen chromosomes are used as the parents in the construction of the next generation. A filial generation is produced as a result of reproduction operators applied to parents. There are two main reproduction operators, namely, *crossover* and *mutation*. Crossover operation is to choose stochastically two individual p_1 and p_2 from the initial population and to take p_1 and p_2 to be the father generation and produce each components of filial generation individual c only with some probability. Notable crossover techniques include *the single-point*, *the two-point*, and *the uniform types*. Mutation involves the modification of the value of each gene in the chromosome with some probability. The role of mutation is to restore unexplored or lost genetic material into the population to prevent *the premature convergence* of the GA to *suboptimal solutions*. Many filial generations are repeatedly produced until a predefined convergence level is reached. Suppose in the evolutionary process, the solutions fall into *the local minimum* for lack of global information of the environments the route of an ant is: $\{c_0,\ldots,c_{i-1},c_i,c_{i+1},\ldots,c_{j-1},c_j,c_{j+1},\ldots\}$. If we have two host computers c_i and c_j, and make

$$\begin{aligned} &\frac{1}{d(c_{i-1},c_j)\cdot t_{c_j}} + \frac{1}{d(c_j,c_{i+1})\cdot t_{c_{i+1}}} + \frac{1}{d(c_{j-1},c_i)\cdot t_{c_i}} \\ &\quad + \frac{1}{d(c_i,c_{j+1})\cdot t_{c_{j+1}}} > \frac{1}{d(c_{i-1},c_i)\cdot t_{c_i}} + \frac{1}{d(c_i,c_{i+1})\cdot t_{c_{i+1}}} \\ &\quad + \frac{1}{d(c_{j-1},c_j)\cdot t_{c_j}} + \frac{1}{d(c_j,c_{j+1})\cdot t_{c_{j+1}}} \end{aligned} \tag{8}$$

then this solution mutates to $\{c_0,\ldots,c_{i-1},c_j,c_{i+1},\ldots,c_{j-1},c_i,c_{j+1},\ldots\}$. After this mutation, the algorithm is allowed to escape from the trap of a local minimum, jumps over the region containing a local minimum point, and continues approaching the optimization solution. This not only improves the convergence rate of the

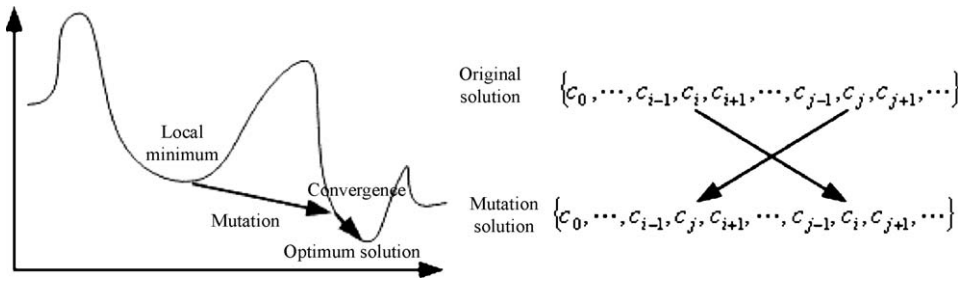

Fig. 1. After mutation of the local minimum, the algorithm is allowed to escape from the trap of a local minimum, jumps over the region containing a local minimum point, and continues approaching the optimization solution.

GA, but also prevents the solution from trapping into a local optimum point, and the quality of the solution is improved. Therefore the whole system robustness is promoted. This process is shown in Fig. 1.

4.3.2. Revising the global update rules

Once all ants have constructed their solutions, pheromone trails are updated as usually in ACO algorithms: first, pheromone trails are reduced by a constant factor to simulate evaporation; then, some pheromone is laid on components of the best solution. Pheromone trails are updated usually at each iteration, increasing the level of those that facilitate moves that were a part of "*good*" solutions, while decreasing all other "*bad*" solutions when all ants have completed their solution, respectively. For the current optimization solution is very possible to correspond to the local minimum, if we only make the ant which gives the current optimization solution perform the global update, then it is very possible for the whole system to fall into the local minimum. Therefore what this paper considers is: not only that optimal ant in the current cycle can perform the global update, but "*l*" *optimal ants* simultaneously can update the pheromone trail level laid in the route when they have completed their tours. Namely, Eq. (6) can be revised as

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s) + \sum_{i=1}^{m}\rho\cdot\Delta\tau_i(r,s) \tag{9}$$

where

$$\Delta\tau_i(r,s) = \begin{cases} \dfrac{1}{i\cdot T_i} & \text{if } (r,s)\in \text{global-optimal path.} \\ 0 & \text{otherwise} \end{cases}$$

where T_i is the time for ith optimal solution to complete the task. Through suitably promoting the multiplicity of the solutions like this, the solution determined by using IACO can escape from the trap of a local minimum to a great extent.

4.4. Algorithm description

When we solve the MARP problems using the ant colony algorithm, at the preliminary stage, because there is not the pheromone trail in the paths, the ants all almost choose randomly their respective paths. The ants secrete the pheromone on the

```
1. /* Initialize */
For every edge (r, s) τ(r, s):= τ0 End-for
For k:=1 to m do
    Let host computer 0 be a starting point of the k-th ant
    Jk(0):={1, ..., n-1}
    rk := 0
End-for
2. /* construction of solutions. Route of the k-th ant is stored in Tourk */
For i:=1 to n-1 do
    If i<n-1 then
        For k:=1 to m do
            If the k-th ant does not complete the task Then
                Choose next host computer sk to be accessed according to (1) and (7)
                Jk(sk):= Jk(rk) - sk
                Tourk(i):=(rk ,sk )
            End-if
        End-for
    Else
        For k:=1 to m do
            sk := 0
            Tourk(i):=(rk ,0 )
        End-for
    End-if
    For k:=1 to m do
        Locally update pheromone trail τ(rk,sk) according to (7)
        rk := sk
    End-for
End-for
3. /* global update rules */
For k:=1 to m do
    calculate Tk /* Tk is to take time of the k-th ant */
End-for
Order Tk (from small to large)
globally update according to (9)
4. /* circulation control */
If satisfying end conditions Then
    Return to shortest path
Else
    If fall into local minimum, then
        mutate according to (8)
    End-if
    goto 2
End-if
```

Fig. 2. The MARP question-solving algorithm in pseudo-code for revising the global update rules and preventing the solution from trapping into a local optimum point.

Table 1
Comparison of our algorithm with other methods.

Algorithms	Basic ant algorithm	Ant colony algorithm	Improved ant colony algorithm	Genetic algorithm	Simulated algorithm
Optimal solutions	154	132	121	162	152
Average best solutions	160	137	125	167	179
Normal errors	0.12	0.09	0.04	0.08	0.09
Iteration numbers	8724	8147	6909	5926	5872
Time(s) to find BS	453.67	397.13	156.03	352.39	205.24

paths when they complete a tour in the process of constructing the solution, and a superior solution of each generation can update the pheromone trail on the paths when the ants complete a tour according to the global update rule. These processes realize pheromone-based indirect and asynchronous communication among ants mediated by an environment, and form a positive feedback where effective fragments of solutions will receive a greater amount of pheromone and in turn a larger number of ants will choose the fragments of solutions and deposit pheromone. This urges the pheromone trail density in the shorter path to continue promoting in turn again. So repeatedly, constituting a positive feedback process which causes almost all ants to select the shortest path and solve the optimization solution. The algorithm for solving MARP problems is shown in Fig. 2.

5. Experimental results and performance comparison

5.1. Experimental Results based on IACO

In order to examine the performance of the algorithm proposed in this paper, we complete some contrast experiments. First we imitate the topology of the actual network and produce the required data: Number n of the host computers, the time delay $d(i, j)$ between the host computers, the probability p_i and the host computer operating time t_i. The MARP question is solved by using basic ant algorithm, the ant colony algorithm, the improved ant colony algorithm, and the simulated annealing respectively. Then the obtained results are compared with the current results. In order to give one objective contrast standard, we also complete one experiment for solving the MARP question using the standard genetic algorithm. By means of the adjustment, the parameters were set as follows: $n = 30$, $\beta = 0.8$, $q_0 = 0.9$, $\rho = 0.1$, $\tau = 0.1$, $m = 10$. Each test iterates 10,000 times, and takes the average results of 10 experiments. Performance contrast of the experiment results of the various algorithms is shown in Table 1. The optimization result of each item is indicated with hold body. The evolutionary curves of the optimization solutions and the worst solution of four algorithms are presented in Figs. 3 and 4. Fig. 3 exhibits that the IACO has the fastest rate of convergence because the number of generations up to convergence is the smallest. The algorithm converging through smaller generations has better convergence performance because all the algorithms have the same population size in the experiment. On average best solution, IACO has lower solution costs than all the other algorithms at every iteration.

5.2. Performance comparison of IACO with other algorithms

The simulated annealing (SA), tabu search (TS), genetic algorithm (GA), and ant colony system (ACS) are four of the main algorithms for solving combinatorial optimization problems of intelligent systems. Performance comparison of our algorithm with the basic ant algorithm, ant colony algorithm, GA and SA is shown in Table 1. Evolutionary curves of the best solution and the worst solution are presented in Figs. 3 and 4 repectively. The improved ant colony algorithm proposed in this paper not only can solve optimization solutions, but the IACO is robust, easy to escape from the trap of a local minimum, and has much higher convergence speed than that of GA and SA.

As a instance, we have evaluated some complex combinatorial optimization problems $G_1 \sim G_4$ proposed in [31] using GA, SA and our algorithms. As seen in Table 2, the solutions found using IACO algorithm approach the best solutions more fast than using the GA and the SA algorithms. After reaching the best solutions these solutions can stabilize in nearby the best solutions or are the approximate best solutions. The performance comparison of calculating $G_1 \sim G_4$ using IACO, GA, and SA algorithms is shown in Table 2. The results in Table 2 have demonstrated that the best solutions of $G_1 \sim G_4$ calculated by IACO approaches most to their theoretical best solutions (-15.000, 7049.331, 680.630, and 0.054) respectively. In our experiments, we take the number of the initial solutions: $N = 20$, the number of the individuals in population:

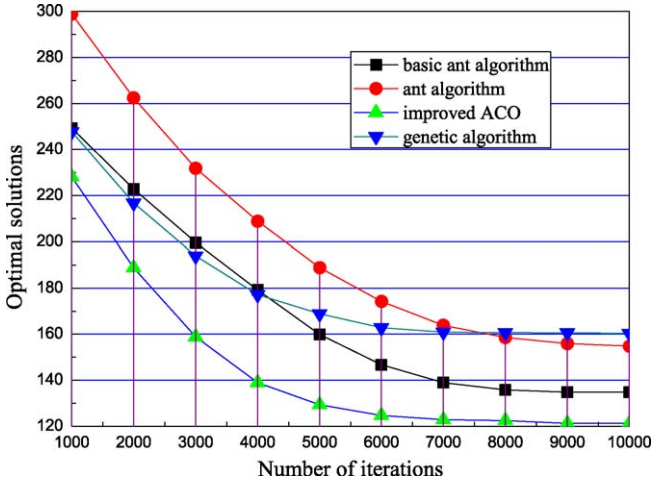

Fig. 3. Evolutionary curves of the optimization solutions.

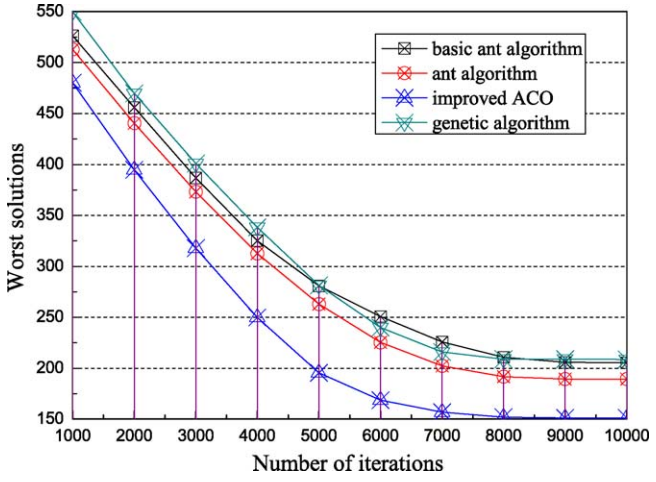

Fig. 4. Evolutionary curves of the worst solutions.

Table 2
Performance comparison of IACO with GA and SA applied to MARP.

Exams	Iteration times			Time(s) to find BSs			Average BSs		
	GA	SA	IACO	GA	SA	IACO	GA	SA	IACO
G_1	894	598	472	60.08	53.86	40.76	-14.136	-14.571	-14.957
G_2	4918	3067	2007	352.39	205.24	156.03	8063.12	7128.06	7051.12
G_3	5828	4679	4182	451.73	384.12	342.54	682.114	687.293	680.203
G_4	5301	4023	3894	411.48	365.40	319.84	0.056	0.055	0.0557

Table 3
The experimental results with different parameters of IACO.

Parameter setup				Iteration numbers and best solutions (BSs)	
q_0	ρ	$p_{\text{crossover}}$	p_{mutation}	Iter. numbers to reach BS	Aver. BSs of each test
0.2	0.1	0.2	0.3	601.2	-14.947
0.4	0.1	0.2	0.3	513.4	-14.948
0.6	0.1	0.2	0.3	547.8	-14.951
0.7	0.3	0.4	0.5	383.8	-14.962
0.8	0.4	0.6	0.7	605.6	-14.968
0.8	0.3	0.8	0.8	637.6	-14.974

$M = 40$, and perform 30 tests repeatedly to each problem under the combination of the different parameter values using IACO, and then take its average result.

5.3. Comparison of the results with the different parameters

Table 3 shows the influence of the different parameter values upon the experimental results of G_1. As can be seen, the iteration number is smallest (383.8) and the average best solutions is 14.962, when $q_0 = 0.7$, $\rho = 0.3$, $p_{\text{crossover}} = 0.4$, $p_{\text{mutation}} = 0.5$.

Our algorithm can avoid the premature convergence, escape from the trap of a local minima, jump over the region containing a local minimum point by crossover and mutation, and perform behavior switching and obstacle avoidance effectively by evolutionary learning and some results have been successfully integrated into our CZU2008 simulated humanoid robot soccer team for *evolutionary global path optimization* which won the fourth place of *RoboCup2008* World Competition.

6. Conclusion and future work

Ant colony optimization has been and continues to be a fruitful paradigm for designing effective combinatorial optimization solution algorithms. After more than ten years of researches, both its application effectiveness and its theoretical groundings have been demonstrated, making ACO one of the most successful paradigm in the metaheuristic area. This paper proposes an improved ant colony algorithm by introducing the mutation operator based on the ant colony algorithm, and improve global update rule. Therefore our algorithm not only inherits many advantages of the ant colony algorithms which integrate self-learning, distributed, parallel computing into one system by allocating a processor to each colony, but also can improve performance by allowing the algorithm to escape from the trap of a local minimum, and speed up the convergence rate. Thus this algorithm is suitable for solving the complex combinatorial optimization problems. At the same time, this paper also defines a very important mobile agent routing problem in mobile agent technology, i.e., proposes an improved ant colony optimization algorithm including a mutation genetic operator by introducing genetic algorithm into the ant colony systems based on thorough research on working mechanism of ant colony system, and solves some mobile agent routing problems using this algorithm. So mobile agents are able to carry out and complete the distributed computing tasks with the optimization efficiency and the fastest convergence rate. This algorithm has a very important significance for enhancing the performance of the mobile agent system and improving the distributed computing model. The experimental results based on the improved ant colony algorithm for solving the mobile agent routing problem have demonstrated that the algorithm is more suitable for solving the combinatorial optimization problems. This ant colony system is not only stable and may achieve the optimization solution or the approximate optimization solution, and has faster convergence rate, but the possibility for this algorithm to fall into a local minimum is extremely low, because at each local minimum, this new algorithm can perform a stochastic mutation of current solutions to escape that local minimum. Therefore the algorithm proposed in this paper is one of the many outstanding optimization algorithms. In contrast with the experimental results of the previous methods and other method, this algorithm has a strong robustness, high self-adaptation, and a fast convergence rate. But the method proposed in this paper is also required to be improved in the following aspects:

(1) the performance improvement and parameters analysis of ACO algorithm still remain in the experimental stage for lack of solid theoretical support;
(2) make the thorough research on the working mechanism of ACO algorithm for reducing its algorithm complexity;
(3) combine ACO algorithm with the local search strategy and improve further the quality of the optimization solutions;
(4) conduct research on the parallel mechanism of the ant colony optimization algorithms so that it improves the efficiency of the algorithm used in the intelligent systems.

Acknowledgements

The authors would like to thank the editors and the reviewers for their valuable comments and suggestions for improving the quality of this paper.

References

[1] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, USA, 2004, July.
[2] S.C. Zhan, J. Xu, J. Wu, The optimization selection on the parameters of the ant colony algorithm, Bulletin of Science and Technology 19 (5) (2003) 381–386.

[3] M. Dorigo, G.D. Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC, IEEE Press, Piscataway, NJ, (1999), pp. 1470–1477.
[4] M. Dorigo, M. Birattari, T. Stützle, Ant colony optimization, IEEE Computational Intelligence Magazine 1 (4) (2006) 28–39.
[5] T. Stützle, M. Dorigo, A short convergence proof for a class of ant colony optimization algorithms, IEEE Transctions on Evolutionary Computation 6 (4) (2002) 358–365.
[6] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics: Part B 26 (1) (1996) 29–41.
[7] M. Dorigo, G.D. Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artificial Life 5 (2) (1999) 137–172.
[8] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[9] M. Dorigo, L.M. Gambardell, Ant colonies for the traveling salesman problem, Biological Systems 43 (2) (1997) 73–81.
[10] L. Bianchi, L.M. Gambardella, M. Dorigo, An ant colony optimization approach to the probabilistic traveling salesman problem, in: Proceedings of the PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, 2002.
[11] K.S. Hung, S.F. Su, Z.J. Lee, Improving ant colony optimization algorithms for solving traveling salesman problems, Journal of Advanced Computational Intelligence and Intelligent Informatics 11 (4) (2007) 433–434.
[12] V. Maniezzo, A. Colorni, The ant system applied to the quadratic assignment problem, IEEE Transactions on Knowledge and Data Engineering 11 (5) (1999) 769–778.
[13] L.M. Gambardella, E. Taillard, M. Dorigo, Ant colonies for the quadratic assignment problem, Journal of the Operational Research Society 50 (1999) 167–176.
[14] A. Colorni, M. Dorigo, V. Maniezzo, M. Trubia, Ant system for job-shop scheduling, JORBEL-Belgian, Journal of Operations Research, Statistics and Computer Science 34 (1) (1994) 39–53.
[15] L.M. Gambardella, M. Dorigo, HAS-SOP: an hybrid ant system for the sequential ordering problem. Technique Report No. IDSIA 97-11, Lugano: IDSIA, 1997.
[16] P. Toth, D. Vigo, The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial & Applied Mathematics, Philadelphia, 2001.
[17] K.M. Sim, W.H. Sun, Ant colony optimization for routing and load-balancing: Survey and new directions, IEEE Transactions of Systems, Man, and Cybernetics, Part A: System and Humans 33 (5) (2003) 560–572.
[18] B.M. Ombuki, B. Ross, F. Hanshar, Multi-objective genetic algorithms for vehicle routing problem with time windows, Applied Intelligence 24 (1) (2006) 17–30.
[19] L.M. Gambardella, É. Taillard, G. Agazzi, in: D. Corne, M. Dorigo, F. Glover (Eds.), MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, London, McGraw Hill, pp. 63–76, 1999.
[20] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transactions on the Evolutionary Computations 6 (4) (2002) 321–332.
[21] T. Stützle, H. Hoos, Improvements on the ant system: introducing max–min ant system, in: Proceedings of the ICANNGA'97, International Conference on ANN and Genetic Algorithms, Springer Verlag, Vienna, 1997.
[22] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: F. Varela, P. Bourgine (Eds.), Proceedings of the European Conference on Artificial Life (ECAL'91, Paris, France), Elsevier Publishing, Amsterdam, pp. 134–142, 1991.
[23] Z.H. Luo, J.A. Yang, Research on the contributed computing model based on mobile agents, Minicomputer Systems 23 (3) (2002) 300–305, March.
[24] A.J. Kulkarnia, K. Tai, Probability collectives: a multi-agent approach for solving combinatorial optimization problems. Applied Soft Computing, online publication, 2009.
[25] C. Blum, M. Dorigo, The hyper-cube framework for ant colony optimization, IEEE Transactions on Systems, Man and Cybernetics: Part B 34 (2) (2004) 1161–1172.
[26] C. Blum, M. Dorigo, Search biasin ant colony optimization: on the role of competition-balanced systems, IEEE Transactions on Evolutionary Computation 9 (2) (2005) 159–174, April.
[27] C. Blum, M. Dorigo, Deception in ant colony optimization. Lecture Notes in Computer Science, vol. 3172, Springer/Heidelberg, Berlin, 2004pp.119–130.
[28] M.O. Beatrice, Brian Ross. Ombuki, Hanshar. Franklin, Multi-objective genetic algorithms for vehicle routing problem with time windows, Applied Intelligence 24 (1) (2006) 17–30.
[29] Z.J. Lee, S.F. Su, C.C. Chuang, K.H. Liu, Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment, Applied Soft Computing 8 (1) (2008) 55–78, January.
[30] S. Favuzza, G. Graditi, E. Sanseverino, Adaptive and dynamic ant colony search algorithm for optimal distribution systems reinforcement strategy, Applied Intelligence 24 (1) (2006) 31–42, February.
[31] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs, 3rd ed., Springer–Verlag/Heidelberg Press, Berlin, 1996, pp. 261–262.