Available online at www.elixirpublishers.com (Elixir International Journal)

Advanced Engineering Informatics

Elixir Adv. Engg. Info. 38 (2011) 4485-4491

ISSN: 2229-712X

# Improved swarm intelligence for solving symmetric travelling salesman problems having premature convergence

Ajit Kumar
Department of Information Science, S.J.C.E. Mysore, Visveswaraya Technological University, Karnataka.

**ARTICLE INFO**

**Article history:**
Received: 22 August 2011;
Received in revised form:
26 August 2011;
Accepted: 31 August 2011;

**Keywords**
Ant colony optimization,
Self organization,
Symmetric travelling salesman problem,
Pheromone trail,
Premature convergence,
Entropy,
Dynamic updating,
Optimal solution.

**ABSTRACT**
Ant colony optimization is unique convergence of Swarm Intelligence and Bio-Inspired Artificial Intelligence. Ants are social insects with limited skills that live in colonies able to solve complex problems. The intelligence of the global society arises from self organization mechanisms, based on the indirect communication between individuals through pheromones. The Symmetric Travelling Salesman Problem here presented is a typical case that requires a self organization type of algorithm, in order to cope with the problem dynamics. The simulation results show how the ant colony optimization is able to solve the different possible routing cases. The innovation in this entire proposal (paper) is that an attempt has been made to improve the existing ant colony algorithm and then apply to the problem set. Ant colony optimization (ACO) has been successfully applied to solve combinatorial optimization problems, but it still has some drawbacks such as stagnation behavior, long computational time, and premature convergence. These drawbacks are more evident when the problem size increases. In this paper, I have reported the analysis of using a reduced/lower pheromone trail bound and a dynamic updating rule for the heuristic parameters based on entropy to improve the efficiency of ACO in solving Traveling Salesman Problems (TSPs). TSPs are NP-hard problem. Even though the problem itself is simple, when the number of city is large, the search space will become extremely large and it becomes very difficult to find the optimal solution in a short time. From my simulations, it can be found that the proposed algorithm indeed has superior search performance over traditional ACO algorithms do.

## Introduction

The complex social behaviors of insects have been intensively studied in science and in research of computer technology. The attempt in the research of computer technology is to develop algorithms inspired by insect behavior to solve optimization problems [1]. These behaviors patterns can offer models for solving difficult combinatorial (distributed) optimization problems. Real ants which can indirectly communicate by pheromone information without using visual cues are able of finding the shortest path between food sources and their nest. The ants release pheromone on the ground while walking from their nest to food and then go back to the nest. Since a shorter path has a higher amount of pheromone in probability, ants will tend to choose a shorter path. Artificial ants imitate the behavior of real ants how they forage the food [1], but can solve much more complicated problem than real ants can. One of search algorithms with such concept is Ant Colony Optimization (ACO) [3]. ACO has been widely applied to solving various combinatorial optimization problems such as Traveling Salesman Problem (TSP) [2, 4, 5], Quadratic Assignment Problem (QAP) [3], Weapon-Target Assignment problems (WTA) [31, 32] etc.

Ant colony optimization (ACO) can be used to find the solutions of difficult combinatorial optimization problems. In ACO, artificial ants build solutions by moving on the problem graph and they deposit artificial pheromone on the graph in such a way that future artificial ants can build better solutions. Although ACO has very good searching capability in optimization problems, it has the problems of stagnation and premature convergence and those problems will be more obvious when the complexities of the considered problems increase. In this paper, I attempt to study issues in ACO so that the search efficiency can be improved. I consider the traveling salesman problem (TSP) as the benchmark. My approach differs from original ACO algorithms in two aspects. One is that I introduce a lower levels of pheromone trail bound into the ACO algorithm and the other is that the heuristic parameter used is not a pre-decided/fixed value.

For the first issue, the pheromone evaporation can be seen as an exploration mechanism and with this mechanism the trails will decrease in an exponential speed [1]. Thus, after iterations, the pheromone amounts for some edges may decrease to close to zero. This phenomenon will lead to stagnation behavior and indicates that the search stops to explore new possibility and no better tour can be formed. In my algorithm, in order to avoid such a situation, a lower pheromone trail limit is proposed. The simulation results indeed show the effectiveness of the approach. For the other issue, ants select paths depending on pheromone trail and heuristic information (heuristic visibility) [2, 5] when constructing tours. The pheromone is deposited on paths, once ants pass the paths. However, the constructed tours are not really the optimal solution in the initial stage and ants already deposit pheromone on the tours. Thus, later ants will tend to select the paths that have a higher amount of pheromone. In order to

**Tele:**
**E-mail addresses: ajitkumar.contact@gmail.com**

increase the diversity of paths in the initial stage, I propose to decrease the effects of the pheromone trails for paths selected in the early stage of learning. On the other hand, in the later stage of learning, the pheromone trails may have collected enough information to behavior as required and the effects of the pheromone trails may need to be emphasized. In this paper, I have proposed to use an adaptive heuristic parameter to resolve this dilemma. I shall discuss those ideas later.

This paper is organized as follows: Section 2 gives an introduction toTSP. Reviews of ACO are given in section 3. In section 4, the use of a lower pheromone trail bound is discussed. Then, a way of dynamically updating parameters based on entropy is also introduced in this section. In section 5, the proposed method is employed into several TSP problems and the results of our approach and of traditional ACO are reported. Finally, conclusions are given in section 6.

**Traveling Salesman Problems**

In computer science, the traveling salesman problem (TSP) is an old and well studied problem and is one of the combinatorial optimization problems. It is very easy to understand and to explain the behavior of the TSP. Nevertheless, when the problem size is large, it will become very difficult to find the optimal solution in a reasonable time. In fact, TSP is also known as a NP-hard problem.

Intuitively, TSP is to find the shortest trip of a salesman for a finite number of cities. A salesman is asked to start from a random city by visiting each city exactly once and then to return to the starting city [21]. A complete weighted graph $G = (N,E)$ can be used to represent a TSP, where $N$ is the set of $n$ cities and $E$ is the set of edges (paths) fully connecting all cities. TSP is also called the Hamiltonian circuit, which is a closed tour visiting each city in $G$ exactly once. Each edge $(i, j) \in E$ is assigned a cost $d_{ij}$, which is the distance between cities $i$ and $j$. $d_{ij}$ can be defined in the Euclidean space and is given as follows:

$d_{ij} = \sqrt{} (x_i - x_j)^2 + (y_i - y_j)^2$ . . . . . . . (1)

where $(x_i, y_i)$ and $(x_j, y_j)$ are the coordinates of city $i$ and of city $j$, respectively. The goal in TSP is to find a minimal length in the Hamiltonian circuit of the graph. In other words, TSP is to find a permutation $\pi$ of the city indices $\{1,2,\ldots,n\}$ whose traveling length $f(\pi)$ is the minimal, where $f(\pi)$ is defined as

$f(\pi) = \sum^{n-1}_{i=1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$ . . . . . (2)

In a symmetric TSP, the Hamiltonian graph is fully connected and its distance matrix is symmetric; that is, the lengths of edges satisfy $di\ j=dji$. For an asymmetric TSP, the length of an edge connecting two cities $i$ and $j$ depends on whether one goes from $i$ to $j$ or from $j$ to $i$, and in general $d_{ij} \neq d_{ji}$. Symmetric TSP has many applications such as in very large scale integration chip fabrication [11], in drilling holes for printed circuit boards [13], or in the positioning of X-ray devices [12].

Due to the simplicity of the TSP, it is easy to program an exhaustive search for the TSP. However, the complexities of the search space will rapidly grow with the city size. For $n$ cities, it has $(n-1)!$ solutions. For example, the 7-city problem has 360 solutions easily solved by using an exhaustive search method. However, the 100- city problem will have $(n-1)! \approx \sqrt{2\pi}.(n-1).[n-1/2.718]^{n-1} \approx 9.32\times10^{155}$ solutions. For such a number of solutions, an exhaustive search method may become infeasible. Hence, an approximate (heuristic) method is usually employed to find near-optimal solutions at relatively low computational cost [22, 23].

When the heuristic approaches are applied to the TSP, they can be classified as tour constructive heuristic and tour improvement heuristic (also called local optimization heuristics). Construction heuristic is an algorithm that determines a tour according to some construction rules, but does not try to improve upon this tour. Tour construction heuristic randomly chooses a city as the starting point and then utilizes some heuristic rules to add new cities on the tour so as to build a feasible TSP solution. For example, the ACO heuristic, the nearest neighbor heuristic and the greedy heuristic are three typical tour construction algorithms [29]. On the other hand, tour improvement heuristic utilizes some heuristic rules to exchange the edges on a tour so as to reduce the length of this tour until the optimal solution is found. Typical tour improvement heuristic is 2-opt, 3-opt [10]

**Ant Colony Optimization**

ACO algorithms [2–8] were firstly introduced by Dorigo and are novel metaheuristic optimization algorithms. ACO can be applied to solve distributed optimization problems among many cooperating simple agents that are not aware of their cooperative behavior. The study of ACO is to investigate the patterns derived from the observation of real ants' behavior. Real ants can indirectly communicate by pheromone information without using visual cues and are capable of finding the shortest path between food sources and their nests [15, 27]. The ants release pheromone on the ground while walking through a path. Thus, a pheromone trail is formed on the traversed way. Supposedly, a shorter path should have a higher amount of pheromone in probability. As a consequence, ants may tend to choose a shorter path. Artificial ants imitate the behavior of real ants how they forage the food, but can solve much more complicated problems than real ants can. A search algorithm with such concept is called Ant Colony Optimization (ACO) [14].

ACO is a metaheuristic in which a colony of artificial ants cooperates in finding good solutions of discrete optimization problems [19, 25, 26]. Cooperation is a key design component of ACO algorithms because the choice is to allocate the computational resources to a set of relatively simple agents (artificial ants) that communicate indirectly by pheromone information. The ACO metaheuristic is shown in pseudo code as follows :

```
Procedure: ACO metaheuristic
  while (not in termination condition)
    Construct Ant solutions
    Pheromone Update
    Daemon Actions                {optional}
end while
end procedure
```

Construction Ants Solutions manages a colony of ant that cooperatively and interactively visit adjacent states of the considered problem by moving through feasible neighbour nodes of the graph. The movements are based on a local ant-decision rule that makes use of pheromone trails and heuristic information. In this way, ants incrementally build solutions to the problem. Pheromone Update consists of pheromone evaporation and new pheromone deposition. Pheromone evaporation is a process of decreasing the intensity of pheromone trails over time. This process guides ants to explore possible paths, and it avoids too rapid convergence of the algorithm towards a suboptimal region. Pheromone Update is used to implement a useful form of forgetting which enables the ants to forage the promising area of the search space. The Daemon Actions procedure is used to implement centralized actions which cannot be performed by a single ant. Daemon Actions are optional for ACO algorithms. The idea is to collect useful global information that can be used to decide whether it is useful or not and to deposit additional pheromone to bias the search process from a non-local perspective.

In ACO, when located at city *i*, ant *k* chooses the next city *j* according to the so-called *pseudorandom proportional* (ant-decision) rule given by

$$j = \begin{cases} \arg \max_{u \in allowed_k(i)} \{[\tau_{iu}]^{\alpha} \cdot [\eta_u]^{\beta}\} \\ \qquad\qquad \text{if } q \le q_0 \text{ (exploitation)} \\ J \qquad\qquad \text{otherwise (exploration)} \end{cases} \quad . . . . . . . . . \quad (3)$$

where $q$ is a random variable uniformly distributed in [0,1], $q_0$ ($0\le q_0 \le 1$) is a pre-defined parameter, $\tau_{iu}$ is the pheromone trail, $\eta_{iu} = 1/d_{iu}$ is the heuristic information (heuristic visibility), α and β are referred to as the visibility and the trail intensity and are two adjustable positive parameters that control the relative weights of the pheromone trail and of the heuristic information, and *allowed*$_k$*(i)* is the set of unvisited cities yet when ant *k* is located at city *i*. *J* is a random variable and is a tradeoff between visibility and trail intensity at iteration *t*. *J* is selected according to the transition probability given by

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in J_i^k} [\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}} & \text{if } j \in J_i^k \\ 0 & \text{otherwise.} \end{cases} \quad . . . \quad (4)$$

Global update is performed after iteration. When all ants have completed their tours, only the shortest tour is allowed to increase pheromone by the following equation:

$$\tau_{ij}(t) = (1-\psi)\cdot\tau_{ij}(t) + \psi\cdot\Delta\tau_{ij}^{elitist}(t) \quad . . . \quad (5)$$

where $0\le\psi\le 1$ is a decay parameter, $\Delta\tau^{elitist}{}_{ij} = (L_{elitist})^{-1}$, and $L_{elitist}$ is the length of the globally best tour. In Eq. (5), the deposited pheromone is discounted by a factor ψ. Global update gives the best tour higher reinforcement for the pheromone trails and the amount of the pheromone increases on the edges of the tour. While constructing a tour, ant *k* is located at city *i* and selects city $j \in allowed_k(t)$, the local update is performed by the following equation:

$\tau_{ij}(t)=(1\text{-}\rho)\cdot\tau_{ij}(t)+\rho\cdot\tau_0$ . . . . . . . (6)

where $0\le\rho\le 1$ is a decay parameter, $\tau_0 = (n \cdot L_{nn})^{-1}$ is the initial values of the pheromone trails, where *n* is the number of cities in the TSP and $L_{nn}$ is the cost produced by the nearest neighbor heuristic [16]. Eq. (6) is mainly to avoid very strong pheromone paths to be chosen by other ants and to increase the explorative probability for other paths. Once the edge between city *i* and city *j* has been visited by all ants, the local updating rule makes pheromone level diminish on the edge.

This rule enables the visited edges less and less attractive when ants traverse the edges and also indirectly increases the exploration of not yet visited edges. In other words, the role of the local updating rule is to shuffle the tours so that the early visited cities in one ant's tour may be possibly explored later in other ants' tours.

**The Proposed Mechanisms**

Although ACO has very good search capability for optimization problems, it may have the problems of stagnation and premature convergence. Those problems will be more evident when the complexities of the considered problems increase. In this research, two issues in ACO have been studied. The first one is to analyze the local update effects and the initial pheromone values. A lower pheromone trail bound is proposed in this research. The other is to investigate the characteristics of one heuristic parameter required in ACO. I then propose a way of dynamically modifying that parameter. This approach is based on the entropy measure of the pheromone in ACO.

**Analysis of Pheromone Trail Limit**

In ACO, the elitist strategy of global update enables that the pheromone trails of the currently best tour will continuously obtain positive feedback and other trails in the pheromone matrix evaporate in the process. It is easy to see that such an elitist strategy may lead to a stagnation behavior because the currently best tour may be a suboptimal tour and thus, the edges of this suboptimal tour will have an excessive growth of pheromone. Because the evaporation of the pheromone trails is in an exponential speed, the amount of pheromone for some unvisited edges will decrease to close to zero after iterations. If those unvisited paths edges are in the optimal path, the algorithm may not be able to visit them in the later search process. In that case,

I may say that the search is trapped into suboptimal solutions. In this study, in order to avoid such cases, a value $\tau_{\min}$ is considered as the minimal. I analyze two conditions for the selection of $\tau_{\min}$ in the following. First, the initial values of pheromone are set to $\tau_0 = (n \cdot L_{nn})^{-1}$, where $L_{nn}$ is produced by the nearest neighbour algorithm. Because the initial values of pheromone are the same for all trails, the tour length of the first iteration is close to $L_{nn}$. When the algorithm continues, $L_{elitist}$ will be shorter and may exceed $L_{nn}$. As the number of cities increases, ants find it very hard to find the optimal solution in relatively short iterations. However, if we set the minimal bound to $\tau_{\min} = (n \cdot Lelitist)^{-1}$, which is close to the initial pheromone values, this bound will not fulfill the principle of positive feedback because edges in poor solutions are also bounded to close to the initial value.

Secondly, the effects of the local updating rule are to make visited edges less and less attractive. In fact, for any $\tau_{ij}$, it will satisfy

$\lim_{k\to\infty} \tau_{ij}(k) = \tau_0$ . . . . . . . . . . . . . (7)

where *k* is the number of ants, $\tau_0$ is the initial values of pheromone.

Eq. (7) shows that $\tau_0$ is the lowest bound and is asymptotically converged when using the local updating rule. From this viewpoint, it strengthens my assertion about the minimal bound of the pheromone trails will rapidly reach $\tau_0$ in the first selection of $\tau_{\min}$. In my implementation, τmin is selected as:

$\tau_{\min} = 1/(c \cdot n^2 \cdot L_{elitist})$. . . . . . . . . . . (8)

where $L_{elitist}$ is the length of the currently best tour, *n* is the number of cities, and *c* is a constant and is determined heuristically. In my selection, I only give a value greater than zero and the minimal value of trails will decrease as the number of cities increases in order to ensure complete search. In order to see which value for *c* is proper, three TSP instances KoA100, KroB150, and KroA200 taken from TSPLIB [30] are used as the benchmarks. The experimental results are shown in **Table 1** and a good value for *c* is found to be 2. It should be notice that the value of *c* makes no significant influence on the search performance as shown in the table.

Figure 1 shows the change history of the pheromone value on an edge (edge$_{(1,2)}$) in the KroA100 TSP instance while using different values for the minimal bound. The solid red line is the initial pheromone value $\tau_0$ and the bold dotted light-blue line is the lower pheromone trail bound $\tau_{min} = (2\cdot n^2 \cdot Llitist\ )^{-1}$. The purple, green and bold dotted blue lines are the variation of pheromone on edge (1,2) while using a lower bound, $\tau_{min} = (2\cdot n^2 \cdot Llitist\ )^{-1}$, and $\tau_{min} = 0$ (i.e., no bound is used), $\tau_{min} = (n\cdot Llitist\ )^{-1}$, respectively. It can be found that when we set a suitably minimal bound, the pheromone amount (the purple line) is always higher than that of without a minimal bound (the green line) especially the portion marked in the figure (after 500 iterations).

The pheromone trail will become very small and close to zero after a period of updating when a minimal bound is not used. However, if a unsuitably minimal bound is used, the pheromone amount (the bold blue line) may exceed τ0 (the red line) after iterations and it will be bounded at τ0.

Although the stagnation situation does not emerge, the pheromone information will not be able to guide ants to bias the search and ants will blindly search in a large solution space. Table 2 shows the search performance for different bounds used. In parentheses, it is the relative error to the best solution from TSPLIB.

Those results are taken for an average for 30 runs. Obviously, the performance is worse than ACO does when an unsuitably minimal bound is set.

**Table 1.** The experiment results while using various values of *c* in Eq. (8).

| $q_0 = 0.7, \alpha = 1, \beta = 3, \rho = \psi = 0.1$, $m = 30$ (ants number), 30 runs, 5000 iterations/run | | | | |
|---|---|---|---|---|
| | KroA100 | KroB150 | KroA200 | Average error |
| $c = 1$ | 21358.52 (0.36%) | 26386.73 (0.98%) | 29623.32 (0.87%) | 0.74% |
| $c = 1.5$ | 21352.77 (0.33%) | 29387.16 (0.98%) | 29628.65 (0.89%) | 0.73% |
| $c = 2$ | 21347.11 (0.31%) | 26378.30 (0.95%) | 29627.13 (0.88%) | 0.71% |
| $c = 2.5$ | 21364.16 (0.39%) | 26389.32 (0.99%) | 29636.56 (0.91%) | 0.76% |
| $c = 3$ | 21381.67 (0.47%) | 26427.56 (1.14%) | 29622.44 (0.86%) | 0.82% |

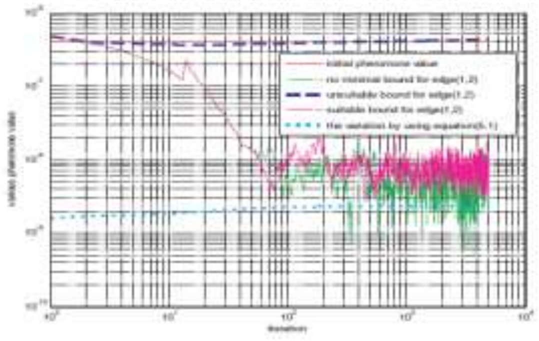

**Fig. 1.** The change history of the pheromone on edge (edge$_{(1,2)}$) in the KroA100 TSP instance while using different values for the minimal bound.

In this implementation of the use of lower trail bounds, an interesting phenomenon exists. Table 3 shows the executing times for two TSP instances with the lower bound and without the lower bound. Their search performances

**Table 2.** The results of using different $\tau_{min}$ or without using a minimal bound.

| $q_0 = 0.7, \alpha = 1, \beta = 3, \rho = \psi = 0.1$, $m = 30$ (ants number), 30 runs, 5000 iterations/run | | | |
|---|---|---|---|
| Instance | $\tau_{min} = (n \cdot L_{Litist})^{-1}$ | $\tau_{min} = (2 \cdot n^2 \cdot L_{litist})^{-1}$ | $\tau_{min} = 0$ |
| KroA100 | 23243.66 (9.22%) | 21347.11 (0.31%) | 21374.74 (0.44%) |

**Table 3.** The results of time spent for the d198 and KroA200 instances.

| 10 runs, 60000 iterations/run | | d198 CPU time (minutes) | KroA200 CPU time (minutes) |
|---|---|---|---|
| ACO | $\beta = 2$ | 694.17 | 1052.68 |
| | $\beta = 3$ | 866.28 | 1297.58 |
| | $\beta = 4$ | 1084.88 | 1563.07 |
| | $\beta = 5$ | 1299.88 | 1617.68 |
| Our Method | | **621.85** | **702.67** |

**Dynamic Heuristic Parameter**

In ACO algorithms, Eq. (3) or (4) plays an important role in selecting solution paths. In the equation, there are two terms, the pheromone trail ($\tau_{iu}$) and the heuristic information ($\eta_{iu}$). α and β are two adjustable positive parameters controlling the relative weights of the pheromone trail and of the heuristic information. The heuristic information is essential in generating superior quality tours in the initial search stages. It is because the values of the pheromone trails do not have much information in the early stage of learning and cannot guide the artificial ants in constructing good tours. In this situation, the heuristic parameter may be set to a large value. On the other hand, in the later stage, the heuristic parameter may need a small value because the pheromone trails may have collected enough information to behavior as required and the heuristic information may mislead the search due to its locality. Thus, in this situation, we may need a small value for the heuristic parameter. As mentioned earlier, the heuristic parameter is set as a constant in traditional ACO algorithms. In order to show the above phenomena, I consider several TSP instances for different fixed heuristic parameters. Fig. 2 shows the results from several TSP instances for different fixed heuristic parameters. Here, I have taken the average of 30 runs. In the initial phases of the search, a high value of heuristic parameter can always provide high quality tours. This means that the influence of pheromone is greatly reduced, and ants are able to search other paths in constructing feasible solutions. It is evident that a small value of the heuristic parameter may result in bad performance in the early stage of learning. Nevertheless, a small value of the heuristic parameter can have good performance when the search process lasts long enough. Thus, it is intuitive to use an adaptive heuristic parameter for ACO. In this study, I intend to propose a way of designing an adaptive heuristic parameter for ACO such that the search performance can be better.

Now, the problem is how the heuristic parameter adapts. Traditionally, parameters required in learning are adapted based on iteration numbers [33]. However, the iteration number may not truly reflect the learning effects. In my study, I propose to use the concept of entropy of the pheromone trails to measure the learning status and then the heuristic parameter is adapted according to this entropy measure.

The concept of entropy is known from Shannon's information theory [17, 18]. It is a measure of uncertainty concerning an event and is used to denote the degree of disorder in a system. Shannon's entropy represents the information regarding the probability of occurrence of an event. In ACO, pheromone is the basis of path selection, and the selection is uncertain in nature. Thus, we propose to consider the entropy information in ACO to estimate the variation of the pheromone matrix. Each trail is a discrete random variable in the pheromone matrix. The entropy of a random variable $X$ is defined as

$$H(X) = -\sum_{t=1}^{r} P(x_t)\log P(x_t) \quad . \ . \ . \ . \ . \ . \ . \ . \quad (9)$$

where $X$ represents the trails in the pheromone matrix.

For a symmetric $n$ cities TSP, there are $n\cdot(n\text{-}1)/2$ distinct pheromone trails and $r = n\cdot(n\text{-}1)/2$. It is easy to see that when the probability of each trail is the same, $H$ will be the maximum (denoted as $H$max) [20] and is given by

$$H_{\max} = -\sum_{i=1}^{r} P_i \log P_i = -\sum_{i=1}^{r} \frac{1}{r}\log\frac{1}{r} = \log r. \quad . \ (10)$$

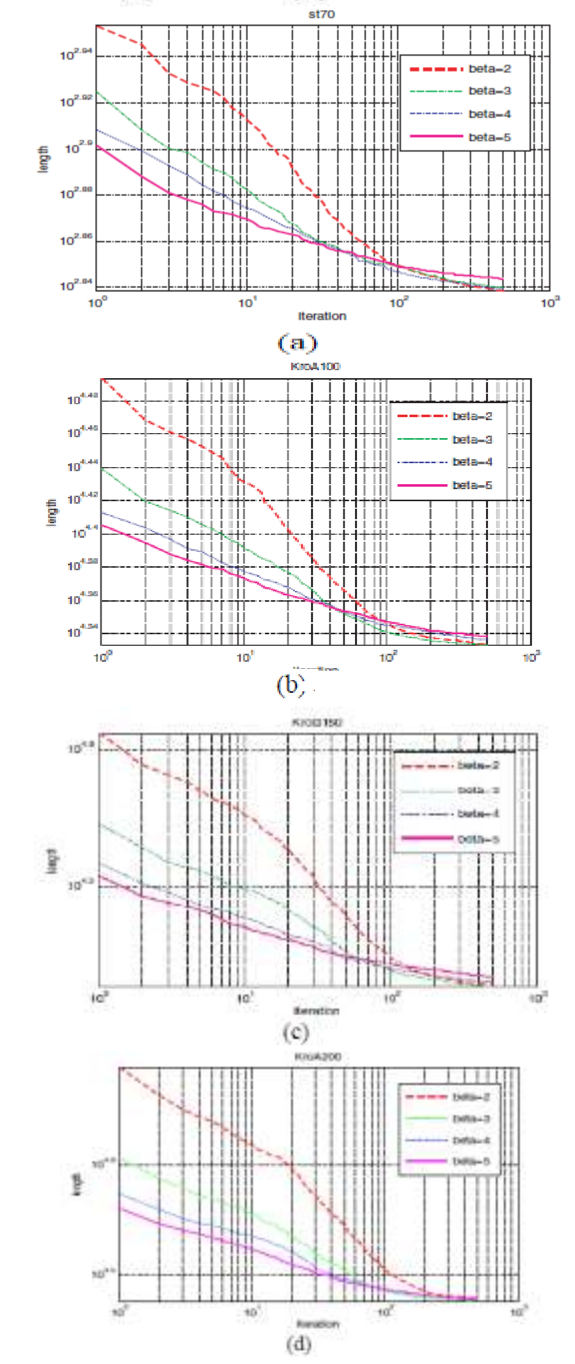

**Fig. 2.** The learning curves while using various fixed heuristic parameters for (a) the st70 instance, (b) the KroA100 instance, (c) the KroB150 instance, and (d) the KroA200 instance.

## The Proposed Algorithm

By combining the use of a lower limit bound and the entropy based adaptive heuristic parameter, the proposed ACO algorithm is given as follows:

```
Initialization: Set $\tau_{ij}(0) = \tau_0 = (nL_{nn})^{-1}$
Calculate the maximum entropy
    $H = -\sum_{i=1}^{r} P_i \log P_i = -\sum_{i=1}^{r} \frac{1}{r}\log\frac{1}{r}$
For $nc = 1$ to max_generations do
    While $k = 1$ to $m$ do
        Place ant $k$ on a randomly selected node;
    End While
    For $k = 1$ to $m$ do
        Construct tour $T^k(t)$ by applying $n-1$ loops the
          following steps:
        Produce a rand number $q \in [0,1]$, then according to
          the
        $j = \begin{cases} \arg\max_{u \in allowed_k(i)} \{[\tau_{iu}]\cdot[\eta_u]^\beta\} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases}$ ;
        Apply the local updating rule when an ant visits an
          edge
          $\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \rho\cdot\tau_0$;
        Compare the $\tau_{ij}$ and $\tau_{\min}$ if $\tau_{ij} < \tau_{\min}$ then setting
          $\tau_{ij} = \tau_{\min}$;
    End for
    Apply the global updating rule after ants complete their
      trips
        $\tau_{ij}(t) = (1-\psi)\tau_{ij}(t) + \psi\cdot\Delta\tau_{ij}(t)$;
    Compare the $\tau_{ij}$ and $\tau_{\min}$ if $\tau_{ij} < \tau_{\min}$ then setting
      $\tau_{ij} = \tau_{\min}$;
    Compute $H_{iteration}$ and Set $H' = H_{iteration}/H_{\max}$;
    Update the heuristic parameter by using Eq. (11)
End for
```

## Experimental Results

In order to demonstrate the superiority of the proposed method, I did not include local search [31, 31] in all ACO algorithms used. Several TSP problems are considered. They are st70, KroA100, KroB150, d198, KroA200, ts225, and lin318 obtained from the TSPLIB website [30] and these instances are run via UBUNTU 10.04. In my experiment, the evaluation of the distance for those TSP instances is computed with the floating-point numbers. The parameters are set to the following values: $\rho = \psi = 0.1$, $m = 30$ (ant number), $q_0 = 0.7$, $\alpha = 1$, $\beta = 2, 3, 4$, or 5, and $c = 2$. Owing to the randomness of the ACO algorithms, I take the average of the overall distances of all runs. Nevertheless, the number of runs and the number of iterations are different for different city numbers to clearly demonstrate the performance. It is because when the city number increases, the computational complexity is exponentially explored. As a consequence, the optimal solution is very hard to found in a short time. In order to ensure the convergence of the ACO algorithm, I have conducted more iterations for the case with a large city number.

The results are shown in Tables 4-10. All examples show that the proposed approach has much better search performance than traditional ACO algorithm does. It should be noticed that the use of the proposed dynamic heuristic

**Table 4.** The experiments for the 70-city TSP instance, st70.

| st70 (30 runs, 30000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 677.11 (0.31%) | 681.85 (1.01%) | 5.25 |
| | $\beta = 3$ | 677.11 (0.31%) | 680.75 (0.86%) | 4.98 |
| | $\beta = 4$ | 677.44 (0.36%) | 682.70 (1.14%) | 4.46 |
| | $\beta = 5$ | 681.55 (0.97%) | 685.34 (1.53%) | 1.90 |
| The proposed algorithm $A = 0.9, B = 0.75, C = 0.68$ | | **677.11 (0.31%)** | **679.61 (0.68%)** | **1.67** |
| The best solution from TSPLIB: 675 | | | | |

**Table 5.** The experiments for the 100-city TSP instance, KroA100.

| KroA100 (30 runs, 50000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 21285.44 (0.02%) | 21332.61 (0.24%) | 90.31 |
| | $\beta = 3$ | 21285.44 (0.02%) | 21324.88 (0.20%) | 89.94 |
| | $\beta = 4$ | 21307.42 (0.12%) | 21344.52 (0.29%) | 55.45 |
| | $\beta = 5$ | 21311.55 (0.14%) | 21397.78 (0.54%) | 58.95 |
| The proposed algorithm $A = 0.86, B = 0.7, C = 0.62$ | | **21285.44 (0.02%)** | **21302.11 (0.09%)** | **33.48** |
| The best solution from TSPLIB: 21282 | | | | |

**Table 6.** The experiments for the 150-city TSP instance, KroB150.

| KroB150 (20 runs, 55000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 26127.35 (−0.01%) | 26336.26 (0.79%) | 224.02 |
| | $\beta = 3$ | 26127.35 (−0.01%) | 26420.66 (1.11%) | 182.53 |
| | $\beta = 4$ | 26131.23 (0.005%) | 26370.28 (0.92%) | 204.21 |
| | $\beta = 5$ | 26144.16 (0.05%) | 26383.64 (0.97%) | 195.81 |
| The proposed algorithm $A = 0.83, B = 0.68, C = 0.58$ | | **26127.35 (−0.01%)** | **26251.81 (0.47%)** | **97.72** |
| The best solution from TSPLIB: 26130 | | | | |

**Table 7.** The experiments for the 198-city TSP instance, d198.

| d198 (10 runs, 60000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 15844.51 (0.41%) | 15918.22 (0.86%) | 57.00 |
| | $\beta = 3$ | 15876.05 (0.61%) | 15956.61 (1.12%) | 73.05 |
| | $\beta = 4$ | 15861.68 (0.52%) | 15995.52 (1.37%) | 72.24 |
| | $\beta = 5$ | 15974.41 (1.23%) | 16046.16 (1.69%) | 79.05 |
| The proposed algorithm $A = 0.85, B = 0.7, C = 0.58$ | | **15839.61 (0.38%)** | **15879.28 (0.63%)** | **20.64** |
| The best solution from TSPLIB: 15780 | | | | |

**Table 8.** The experiments for the 200-city TSP instance, KroA200.

| KroA200 (10 runs, 60000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 29405.72 (0.13%) | 29653.23 (0.97%) | 211.07 |
| | $\beta = 3$ | 29422.24 (0.18%) | 29614.99 (0.84%) | 116.43 |
| | $\beta = 4$ | 29454.81 (0.29%) | 29545.02 (0.60%) | 82.52 |
| | $\beta = 5$ | 29480.16 (0.38%) | 29585.81 (0.74%) | 125.69 |
| The proposed algorithm $A = 0.85, B = 0.6, C = 0.5$ | | **29369.41 (0.005%)** | **29430.45 (0.21%)** | **55.12** |
| The best solution from TSPLIB: 29368 | | | | |

**Table 9.** The experiments for the 225-city TSP instance, ts225.

| ts225 (10 runs, 70000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 126936.80 (0.23%) | 127434.39 (0.62%) | 408.11 |
| | $\beta = 3$ | 127104.85 (0.36%) | 127564.55 (0.73%) | 370.93 |
| | $\beta = 4$ | 127467.24 (0.65%) | 128511.21 (1.48%) | 578.26 |
| | $\beta = 5$ | 128256.02 (1.27%) | 128918.11 (1.79%) | 372.18 |
| The proposed algorithm $A = 0.85, B = 0.68, C = 0.56$ | | **126645.93 (0.0023%)** | **126857.57 (0.17%)** | **100.71** |
| The best solution from TSPLIB: 126643 | | | | |

**Table 10.** The experiments for the 318-city TSP instance, lin318.

| Lin318 (5 runs, 80000/run) | | Optimal solution | Average solution | Standard deviation |
|---|---|---|---|---|
| ACO | $\beta = 2$ | 43148.5 (2.66%) | 43379.1 (3.21%) | 283.14 |
| | $\beta = 3$ | 42899.2 (2.07%) | 43054.9 (2.44%) | 129.21 |
| | $\beta = 4$ | 42410.2 (0.91%) | 42877.2 (2.02%) | 275.48 |
| | $\beta = 5$ | 42470.7 (1.05%) | 42829.6 (1.91%) | 314.61 |
| The proposed algorithm $A = 0.85, B = 0.58, C = 0.48$ | | **42170.5 (0.34%)** | **42378.3 (0.83%)** | **124.85** |
| The best solution from TSPLIB: 42029 | | | | |

parameter will not introduce much computational burden into the algorithm. In my simulation, the CPU time of this approach is almost the same as that of using fixed heuristic parameter. Thus, the CPU times are not included in those tables.

## Conclusions

In this paper, I have proposed a dynamic updating rule for the heuristic parameters based on entropy to improve the efficiency of ACO. I have also proposed to use a lower pheromone trail bound in the algorithm. Various analyses are conducted n our study and reported in this paper. From my experimental results, the proposed method demonstrates excellent performance and is much better than traditional ACO algorithms. It can also be found that the proposed dynamic update of the heuristic parameters based on entropy will generate high quality tours and it can guide ants toward the effective solutions space in the initial search stages.

## References:

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Swarm intelligence from natural to artificial systems,” Oxford University Press, 1999.

[2] M. Dorigo and L. M. Gambardella, “The colony system: A cooperative learning approach to the traveling salesman problem,” IEEE Transactions on Evolutionary Computation, Vol.1, No.1, April,1997.

[3] A. Colorni, M. Dorigo, and V. Maniezzo, “Distributed optimization by ant colonies,” Proc. of ECAL91-European Conference on Artificial Life, pp. 134-142, 1991.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: Optimization by a colony of cooperating agents,” IEEE Transactions on System, Man, and Cybernetics, Part B, Vol.26, pp. 29-41, 1996.

[5] T. St¨utzle and H. H. Hoos, “Max-min ant system,” Future Generation Computer System, Vol.16, pp. 889-914, 2000.

[6] L. M. Gambardella and M. Dorigo, “Ant-Q: A reinforcement learning approach to the traveling salesman problem,” Proceeding of ML-95, Twelfth Intern. Conf. on Machine Learning, Morgan Kaufmann, pp. 252-260, 1995

[7] M. Dorigo and L. M. Gambardella, “A study of some properties of ant-Q,” Proceedings of PPSN-IV, Fourth

International Conference on Parallel Problem Solving from Nature, Vol.1141, pp. 656-665, Springer-Verlag, 1996.
[8] B. Bullnheimer, R. F. Hartl, and C. Strauss, “A new rank-based version of ant system: A computational study,” Central European Journal for Operations Research and Economics, Vol.7, No.1, pp. 25-38, 1999.
[9] T. St¨utzle and M. Dorigo, “Ant colony optimization,” MIT Press, 2004.
[10] S. Lin, “Computer solutions of the traveling salesman problem,” Bell Systems Technology Journal, Vol.44, pp. 2245-2269, 1965.
[11] B. H. Korte, “Applications of combinatorial optimization,” Mathematical programming: Recent developments and applications, Kluwer, Dordrecht, pp. 1-55, 1989.
[12] R. G. Bland and D. F. Shallcross, “Large traveling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation,” Operations Research Letters, Vol.8, pp. 125-128, 1989.
[13] G. Reinelt, “The traveling salesman: Computational solutions for TSP applications,” Lecture Note in Computer Science, Vol.840, Springer-Verlag, Berlin, 1994.
[14] M. Dorigo and G. D. Caro, “Ant colony optimization: A new metaheuristic,” Proceedings of the 1999 Congress on Evolutionary Computation, Vol.2, pp. 1470-1477, 1999.
[15] R. Beckers, J. L. Deneubourg, and S. Goss, “Trails and U-turn in the selection of the shortest path by the ant Lasius Niger,” Journal of Theoretical Biology, Vol.159, pp. 397-415, 1992.
[16] D. J. Rosenkranz, R. E. Stearns, and P. M. Lewis, “An analysis of several heuristics for the traveling salesman problem,” SIAM Journal on Computing, Vol.6, pp. 563-581, 1997.
[17] N. R. Pal and S. K. Pal, “Entropy: A new definition and its applications,” IEEE Transactions on System, Man, and Cybernetics, Vol.5, No.21, pp. 1260-1270, 1991.
[18] R. Bose, “Information theory, coding, and cryptography,” McGraw Hill, 2002.
[19] M. Dorigo and G. D. Caro, “Ant algorithm for discrete optimization,” Artificial Life, Vol.5, No.3, pp. 137-172, 1999.
[20] E. T. Jaynes, “On the rationale of the maximum entropy methods,” Proceedings of the IEEE, Vol.70, No.9, pp. 939-952, 1982.
[21] E. L. Lawer, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, “The traveling salesman problem,” Wiley, New Work, 1985.
[22] G. A. Croes, “A method for solving traveling salesman problems,” Operations Research, Vol.6, No.6, pp. 791-812, 1958.
[23] J. L. Bently, “Fast algorithms for geometric traveling salesman problems,” Journal on Computing, Vol.6, No.4, pp. 387-411, 1992.
[24] G. E. Robinson, “Regulation of division of labor in insect societies,”Annual Review of Entomology, Vol.37, pp. 637-665, 1992.
[25] L. Nemes and T. Roska, “A CNN model of oscillation and chaos in ant colonies: A case study,” IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Vol.42, No.10, pp. 741-745, 1995.
[26] V. Maniezzo and A. Coorni, “The ant system applied to the quadratic assignment problem,” IEEE Transactions on Knowledge and Data Engineering, Vol.11, pp. 769-778, 1999.
[27] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, “Selforganized shortcuts in the argentine ant,” Naturwissenchaftn, Vol.76, pp. 579-581, 1989.
[28] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling salesman problem,” Operations Research, Vol.21, No.2, pp. 498-516, 1973.
[29] E. Aarts and J. K. Lenstra, “Local search in combinatorial optimization,” John Wiley & Sons Inc., 1997.
[30]http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/
[31] Z.-J. Lee, S.-F. Su, and C.-Y. Lee, “An immunity based ant colony optimization algorithm for solving weapon-target assignment problems,” Applied Soft Computing, Vol.2, pp. 39-47, 2002.
[32] Z.-J. Lee, S.-F. Su, and C.-Y. Lee, “A heuristic based ant colony system applied to weapon-target assignment problems,” Journal of Applied Systems Studies, Vol.4, No.2, 2003.
[33] C.-C. Chuang, S.-F. Su, and C.-C. Hsiao, “The annealing robust backpropagation (ARBP) learning algorithm,” IEEE Trans. On Neural Networks, Vol.11, No.5, pp. 1067-1077, 2000.
[34] K.-S. Hung, “An Entropy-based Algorithm in Ant Colony Optimization for Traveling Salesman Problems,” Master Thesis, Dept. Electrical Engineer, National Taiwan University of Science and Technology, Spring 2006.