PAPER

# An Improved Model of Ant Colony Optimization Using a Novel Pheromone Update Strategy

**Pooia LALBAKHSH**[†*], ***Member***, **Bahram ZAERI**[††], ***Nonmember***, ***and*** **Ali LALBAKHSH**[†††a)], ***Member***

**SUMMARY** The paper introduces a novel pheromone update strategy to improve the functionality of ant colony optimization algorithms. This modification tries to extend the search area by an optimistic reinforcement strategy in which not only the most desirable sub-solution is reinforced in each step, but some of the other partial solutions with acceptable levels of optimality are also favored. therefore, it improves the desire for the other potential solutions to be selected by the following artificial ants towards a more exhaustive algorithm by increasing the overall exploration. The modifications can be adopted in all ant-based optimization algorithms; however, this paper focuses on two static problems of travelling salesman problem and classification rule mining. To work on these challenging problems we considered two ACO algorithms of ACS (Ant Colony System) and AntMiner 3.0 and modified their pheromone update strategy. As shown by simulation experiments, the novel pheromone update method can improve the behavior of both algorithms regarding almost all the performance evaluation metrics.

***key words:*** *ant colony optimization, ant colony system, ant-miner, classification rule mining, learning automata, reinforcement learning*

## 1. Introduction

Ant colony optimization (ACO) was adopted to solve different problems in both classic computer literature and real world challenges just after its introduction to the world [1]. Although the underlying theory of this evolutionary method is pretty simple and follows the philosophy similar to many other iterative evolutionary approaches, its multiagent potential provides a scalable and flexible structure to tackle different kinds of problems even better than some other evolutionary methods. Inspired by real foraging ants, ACO takes advantage of independent software agents simulating real foraging ants to find partial solutions. These artificial ants also make use of an indirect communication strategy which is called stigmergy. This strategy uses the environment as a shared medium for the exploring ants to inform each other about their experience [2]. The platform of stigmergy is the advantage of ACO over other schemes. This feature eliminates the need for centralized control and prepares an infrastructure to encompass both local and global information for the search process. In other words, artificial ants in ACO are not confined to a special area of the problem space; they can spread throughout the search space based on the predefined plans and/or heuristic actions to find better sub-solutions and accumulate their local/non-local experience onto artificial pheromone trails which can be further used by the whole colony. In almost any ACO algorithm, problem is modeled as a connected graph together with the required data structures for the process of stigmergy. Artificial ants traverse the graph edges to build a partial solution and store the related optimization data on nodes. This information is used by other ants to choose better alternatives to converge to the problem solution [1].

Although the artificial ants are mostly reputed as software artificial agents, graph nodes are also important agents sometimes much more active than artificial ants [3]. Deeper conceptual study on the graph nodes reveals their capabilities and intelligence in terms of learning and decision making [4]. In ACO, graph nodes can be studied as learning automata (LA) in which action set for each automaton contains the goals towards which the artificial ants are triggered. For each automaton the environmental response is the goodness of the selected action evaluated through a problem dependent evaluation function. We will use the LA model to study the characteristics of ACO pheromone update and to describe the theoretical aspects of our proposal.

Our proposed algorithm deals with the pheromone update phase of ACO which impacts the efficiency of the algorithm. Finding better solutions for a specific problem with a huge or dynamic search space closely depends on how timely and extensively the algorithm covers the search space and how the learning process converges to the global optima. The novel strategy proposed in this paper tries to increase exploration towards potential areas of the problem space according to previously gathered optimization data. To analyze the efficiency of our proposed algorithm, we focused on two ACO static algorithms which are ACS which faces travelling salesman problem and Ant-Miner which is used for classification rule mining through data sets and equipped them with our new pheromone update strategy. As simulation results show, the proposed approach works well in both cases in almost all evaluation metrics.

## 2. General Ant Colony Optimization Approach

As a general optimization method, ant colony optimization can be applied to solve an extensive category of optimization problems. Although details of each algorithm should be adjusted according to the nature of the problem, the whole

Manuscript received May 8, 2013.

[†]The author is with the Islamic Azad University-Borujerd Branch, Borujerd, Lorestan, Iran.

[††]The author is with the Young Researchers Club, Arak Branch, Islamic Azad University-Arak Branch, Arak, Iran.

[†††]The author is with the Islamic Azad University, Kermanshah Branch, Kermanshah, Iran.

[*]Presently, with the Modirane Danesh Mehvar Computer Co.

a) E-mail: ali.lalbakhsh@yahoo.com

DOI: 10.1587/transinf.E96.D.2309

anatomy is fixed as three main phases illustrated in Algorithm 1 [1]. This structure is then customized and articulated according to different application domains.

**Algorithm 1** General Procedure of ACO Metaheuristic

```
Procedure ACO_Metaheuristic
    Schedule_Activities
        Construct_Solutions
        Update_Pheromone
        Apply_DeamonActions
    End_Schedule_Activities
End_Procedure
```

The processes of generating exploring ants and forwarding them through the problem search space are accomplished in the phase of *Construct_Solutions*. According to the algorithm, artificial ants may use forward-backward model or forward model. In forward-backward model after constructing a partial solution by forward ants, a backward ant is generated corresponding to each forward ant to trace back and update pheromone values for the corresponding solution [5], [6], while in forward model no backward ant is generated and the forward ant itself is in charge of updating pheromone values during or after constructing a subsolution. Ant-based routing algorithms mostly use forward-backward model while both algorithms in this paper use the forward model. At the core of the *Construct_Solutions* phase, is a function to evaluate each partial solution. This function is responsible for accepting or rejecting a solution according to the predefined metrics. Each acceptance leads to a pheromone update process mentioned as *Update_Pheromone* phase in Algorithm 1, allowing the corresponding ant to add its experience in the form of increasing a pheromone value as an attracting signal for the other ants to get informed about the feasibility of the corresponding partial solution. In its basic form, pheromone update is accomplished by the following relation[1]:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau \tag{1}$$

Where $\tau_{ij}$ is the pheromone value assigned to the transition of state $i$ to state $j$, and $\Delta\tau$ is the reward or reinforcement factor dedicated to this transition. To control pheromone values, different approaches may be adopted. Some of these strategies are using probabilistic quantities bounded in the closed interval of [0,1] [5], [6], defining upper and lower limits for pheromone values [7], and making use of a pheromone evaporation process [1]. *Apply_DeamonAction* procedure is an optional phase which cannot be accomplished by artificial ants and needs centralized actions. The purpose of daemon action which can be implemented as a local search is to refine current partial solution to obtain better results [8].

To narrow the subject and focus on our proposed approach in the next two sections we study Ant Colony System (ACS) and Ant-Miner as two instances of static ACO algorithms.

## 3. Ant Colony System for Travelling Salesman Problem

In travelling salesman problem (TSP), having $N$ distinct cities, the purpose is to find a tour involving all cities that minimizes tour length formulated as [8]:

$$\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}) \tag{2}$$

Where $d(c_i, c_j)$ is the distance between city $i$ to city $j$ and $\pi$ is the resulted tour. This paper considers symmetric TSP in which the following assumption is considered:

$$d(c_i, c_j) = d(c_j, c_i);\ for 1 \le i \le N, 1 \le j \le N \tag{3}$$

Several algorithms have been proposed to solve TSP using ACO [7], [9]–[17] while we selected Ant Colony System or ACS algorithm because of its acceptable results and scalability. Since in this paper we are dealing with huge graphs, scalability is an important factor which is presented by ACS using a limited number of exploring ants independent from the number of cities[†]. Considering Algorithm 1, in the *Construct_Solutions* phase of ACS, artificial ants are generated and triggered from a source city to construct a *Hamiltonian* tour by means of pheromone data and some heuristic values by adding nodes that are not yet included in the tour. Selecting the next city from the current one is accomplished by the following relation called pseudorandom proportional rule:

$$J = \begin{cases} argmax_{i \in N_i^k}\{\tau_{il}[\eta_{il}]^\beta\} & \text{if } q \le q_0 \\ p_{ij}^k & \text{otherwise} \end{cases} \tag{4}$$

Where $q$ is a random variable uniformely distributed in the closed interval of [0,1], and parameter $q_0$ is defined as $0 \le q_0 \le 1$. $p_{ij}^k$ is determined according to:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad , j \in N_i^k \tag{5}$$

Where $N_i^k$ corresponds to the neighboring nodes of node $i$ which are not yet visited by ant $k$, $\alpha$ and $\beta$ are two parameters for balancing the importance of pheromone value and heuristic information respectively, and $\eta_{il}$ is a heuristic value calculated as:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{6}$$

In ACS, the *Update_Pheromone* phase consists of two update processes namely: global update and local update. In

[†]In many other proposed algorithms number of exploring ants are set equal to the number of nodes. Although this strategy works well with small graphs, it is not applicable for huge graphs with huge numbers of nodes causing inapplicable algorithm runtimes.

the sense of global update, at the end of each iteration, a pheromone update process is accomplished according to (7) by only the ant which has caused the best tour:

$$\tau_{ij} \leftarrow f(x) = \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs} & \forall(i,j) \in T^{bs} \\ \tau_{ij} & \text{otherwise} \end{cases} \quad (7)$$

Where $\rho$ is the evaporation factor, $\tau_{ij}$ corresponds to the current amount of pheromone on the link $(i, j)$, $T^{bs}$ is the best-so-far tour in the iteration, and $\Delta\tau_{ij}^{bs}$ is the reward given to the iteration best tour as:

$$\Delta\tau_{ij}^{bs} = \frac{1}{c^{bs}} \quad (8)$$

Where $c^{bs}$ refers to the cost of the best tour in the iteration.

In local pheromone update, update process is done on the last traversed edge after each tour construction by the corresponding ant through the following relation.

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \quad (9)$$

Where $\xi$ is the pheromone decay factor bounded in (0,1] and $\tau_0$ is the initial pheromone value. As a daemon action, ACS uses a version of 3-opt tour improvement heuristic to improve the results of the tour construction phase. More detailed descriptions on ACS can be found in [1], [15], [16].

## 4. Ant-Miner: An ACO Algorithm for Classification Rule Mining

The purpose of rule mining is to extract knowledge from data. More precisely, a classification rule mining algorithm tries to find precise and comprehensive rules throughout data repositories for classification. Considering ant colony optimization, the algorithm is designed as a supervised evolutionary learning algorithm working on structured datasets with several attributes, their values, and classification categories. The search space graph is conceptually considered in runtime according to the terms and attributes of datasets.

The first ACO rule mining algorithm was Ant-Miner reported by *Parpinelli, Lopes, and Freitas* to discover classification rules [18]. This algorithm not only considers the characteristics of ACO such as stigmergy, evaporation, and heuristic action, it also benefits from the concept of daemon action in the form of rule pruning. The overall framework of Ant-Miner is similar to Algorithm 1; however, a more detailed algorithm for Ant-Miner can be presented as Algorithm 2.

Like other ACO algorithms, pheromone tables are identically initialized at the beginning with predetermined values followed by the three phases of the ACO model. Each classification rule contains a condition part as the antecedent and a predicted class.

$$\text{if} < term_1 \text{ AND } term_2 \text{ AND } term_3 \text{ AND } \ldots \ term_i > \text{then} \\ < class_k >$$

where $term_i$ refers to a selected term and $class_k$ refers to a selected class from the training set.

Each term is a triple <*attribute, operator, value*> where *value* is a value belonging to the domain of *attribute*, and *operator* corresponds to a rational operator. In all datasets used in our simulations, the operator "=" is only considered.

**Algorithm 2** A general pseudo code for Ant-Miner:

```
TrainingSet = AllTrainingCases;
While (number of uncovered cases in the training set >
      MaxUncoveredCases)
Begin
    i = 0;
    initialize_trails ( );
    Repeat
        i = i + 1;
        Ant_i incrementally constructs a rule;
        Prune_rule ( );
        Update_trail (Ant_i);
    Until (i ≥ number of ants) OR (Ant_i has constructed samerule
as the previous T ants);
    Select the best rule among all constructed rules;
    Remove the cases correctly covered by the selectedrule from training
set;
End;
```

To evaluate and control the rule construction process, Ant-Miner uses the quality factor $Q$ to remove irrelevant terms in a process called rule pruning. The quality factor is calculated as:

$$Q = \frac{TP}{TP+FN} \times \frac{TN}{FP+TN} \quad (10)$$

In this relation $TP$ is the number of cases involved in the rule that have a class predicted by the rule. $FP$ reflects the number of cases covered by the rule having a class different from the class predicted by the rule. $FN$ shows the number of cases that are not covered by the rule but that have a class predicted by the rule, and finally $TN$ is the number of cases that are not covered by the rule and do not have the class predicted by the rule. According to relation (10), the value of $Q$ is bounded in the closed interval of [0, 1]. Larger values for $Q$ result in higher rule qualities. In Ant-Miner after constructing a classification rule by each ant, pheromone update phase is performed in two steps. First, the amount of pheromone for each $term_{ij}$ occurring in the rule found by the ant is increased proportional to the quality of that rule. Then, pheromone evaporation is carried out by decreasing the amount of pheromone values associated with each $term_{ij}$ that does not occur in the rule.

Three versions of Ant-Miner are proposed as Ant-Miner 1.0, Ant-Miner 2.0, and Ant-Miner 3.0. We considered the latest version according to its superiorities [18]–[20].

In Ant-Miner 3.0 the pheromone initialization is accomplished through the following relation equally for all pheromone tables:

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i} \quad (11)$$

Where $a$ is the number of attributes, and $b$ refers to the number of values in the domain of attribute $i$. In the phase of rule construction, Ant-miner 3.0 adopts a transition rule as illustrated in the following code to improve exploration.

**While** ($q_1 \leq \varphi$)
**Begin**
**If** $q_2 \leq \sum_{j \in J_i} p_{ij}$ **Then** Choose $term_{ij}$ ;
**End;**
Choose $term_{ij}$ with the $Max(p_{ij})$;

$q_1$ and $q_2$ are randomly generated numbers, and $\varphi$ is a parameter bounded in [0,1]. $J_i$ refers to the number of $i^{th}$ attribute values, and the probability of $P_{ij}$ is calculated as:

$$p_{ij}(t) = \frac{\tau_{ij}(t)\eta_{ij}}{\sum_i^a \sum_j^{b_i} \tau_{ij}(t)\eta_{ij}}, \quad \forall i \in I \tag{12}$$

Where $\eta_{ij}$ is a heuristic value for $term_{ij}$, and $\tau_{ij}(t)$ is the amount of pheromone available on the relation between attribute $i$ and value $j$ at time $t$. $I$ refers to a set of attributes that are not used by the ants. After constructing a rule, pheromone update phase is performed according to the following update rule:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + (1 - \frac{1}{1+Q})\tau_{ij}(t) \tag{13}$$

Where $\rho$ is the pheromone evaporation factor for controlling the influence of the history on the current pheromone trail determined as a constant value of 0.1 and $Q$ is the rule quality. More details on ant-based classification rule mining can be found in [18]–[21].

## 5. Modifications on ACO Pheromone Update

As mentioned above, pheromone update is a key process in all ACO algorithms which guarantees the convergence accuracy and speed towards the solution. Our strategy concentrates on how to update pheromone trails to increase exploration to find better partial solutions. In our theoretical discussions, we focus on reinforcement learning strategy adopted behind ACO algorithms with the aid of learning automata theory. Using this well defined model, we study the way ACS accomplishes the pheromone update phase and then propose our novel pheromone update strategy.

We start by the general form of pheromone update strategy which can be illustrated as the following relation:

$$\tau_{ij}(n+1) = (1-\rho)\tau_{ij}(n) + \Delta \quad \forall i,j \in s^* \tag{14}$$

Where $\rho$ is the evaporation factor and $s^*$ corresponds to set of the selected sub-solutions. For desirable actions in ACS according to (9) $\Delta = \xi \times \tau_0$ and in *Ant-Miner* according to (13) $\Delta = (1 - \frac{1}{1+Q}) \times \tau_{ij}(t)$. In both algorithms for other actions $\Delta = 0$ since no pheromone modification is considered for the actions which are not selected. Assume a feasible sub-solution with pheromone value $\tau_{ij}$ where $\tau_{ij}$ is the highest value among constructed sub-solutions. Evidently, this solution is selected for the *Update_Pheromone* phase and the algorithm ignores all the other constructed solutions even with pheromone values as $\tau_{ij} - \epsilon$ (where $\epsilon$ is a very small value). This strategy implies a kind of crisp decision making for the constructed sub-solutions. In other words, it is not important how good a sub-solution is, if it is not the best one. We are to solve this problem with a more optimistic strategy to cover a group of feasible sub-solutions in each step to elevate the chance of finding better solutions.

The model which ACS and Ant-Miner follow can be well studied through the theory of learning automata (LA). The general non-linear form of learning in LA theory is stated as [22], [23]:

$$p_i(n+1) = (p_i(n) - (1-\beta(n)) \sum_{j \neq i} g_i(p_i(n)) - \beta(n) \sum_{j \neq i} h_j(p_i(n)) \quad \text{if } \alpha(n) = \alpha_i$$

$$p_i(n+1) = p_i(n) - (1-\beta(n))g_i(p_i(n)) + \beta(n)h_i(p_i(n) \quad \text{if } \alpha(n) \neq \alpha_i \tag{15}$$

In this model, $\beta(n)$ refers to the environmental response and the functions $g_i(.)$and $h_i(.)$ can be associated with reward and penalty respectively. Although some works are reported considering penalty on ACO algorithms [3], ACS and Ant-Miner like most of the other ACO algorithms are designed in reward-inaction form ignoring penalizing undesirable actions. Considering $\beta(n) = 0$ , penalty statement of $h_i(.)$ is removed and the following relation will be resulted from (15):

$$p_i(n+1) = p_i(n) - (1-\beta(n)) \sum_{j \neq i} g_i(p_i(n)) \text{ if } \alpha(n) = \alpha_i$$

$$p_i(n+1) = p_i(n) - (1-\beta(n))g_i(p_i(n)) \text{ if } \alpha(n) \neq \alpha_i \tag{16}$$

Defining $r = 1 - \beta(n)$ as the reinforcement factor, relation (16) can be summarized into the following:

$$p_i(n+1) = p_i(n) + r(1 - p_i(n)) \quad \text{if } \alpha(n) = \alpha_i$$
$$p_i(n+1) = p_i(n) - r(p_i(n)) \quad \text{if } \alpha(n) \neq \alpha_i \tag{17}$$

To use the LA model, the pheromone update model of ACO should be mapped onto a probability model. Such an approach is used in ant-based routing algorithms in which all routing tables involve probability values [5], [6], [24]. We simply use (18) to transform pheromone values into the corresponding probability values. Since in ACO, problem space is modeled as a graph, $p_i(n)$ in all relations is replaced with $p_{ij}(n)$[†].

$$p_{ij}(n) = \frac{\tau_{ij}(n)}{s_i(n)} \tag{18}$$

[†]Each action in the problem space graph corresponds to selecting a link from node $i$ as the current node to a next node $j$, so $p_{ij}(n)$ corresponds to the probability of an action which selects the link connecting node $i$ to node $j$.

Where $\tau_{ij}(n)$corresponds to the amount of pheromone for the link connecting node $i$ to $j$, and $s_i(n)$ is the summation of all pheromone values as:

$$s_i(n) = \sum_{j=1}^{N} \tau_{ij}(n) \tag{19}$$

in (19) $N$ corresponds to all existing selections (actions) in the $i^{th}$ node of the problem space graph.

Considering (14), and (18), the following integrated relations for reward-inaction reinforcement learning is achieved:

$$p_{ij}(n+1) = \frac{\tau_{ij}(n+1)}{s_i(n+1)} = \frac{(1-\rho)\tau_{ij}(n)}{(1-\rho)s_i(n)+\Delta} \tag{20}$$

and using (17) we have:

$$\frac{\tau_{ij}(n+1)}{s_i(n+1)} = \frac{(1-\rho)\tau_{ij}(n)}{(1-\rho)s_i(n)+\Delta} = \frac{\tau_{ij}(n)}{s_i(n)} - r\frac{\tau_{ij}(n)}{s_i(n)} \tag{21}$$

Finally, the reinforcement factor $r$ is emerged as:

$$r = \frac{\Delta}{(1-\rho)s_i(n)+\Delta} \tag{22}$$

The above equation reveals the independency of the reinforcement factor from path probabilities, showing the linear behavior of the applied update strategy. In other words, the desirable action in ACO is reinforced regardless of its current desirability. We call this situation as linear pheromone update in ACO.

Our novel model is proposed according to the following pheromone update model in which not only the desirable action is reinforced, some other potential actions are also reinforced with a different value of $\hat{\Delta}$.

$$\begin{aligned} \tau_{ij}(n+1) &= (1-\rho)\tau_{ij}(n) + \Delta \quad \forall i, j \in s^* \\ \tau_{ij}(n+1) &= (1-\rho)\tau_{ij}(n) + \hat{\Delta} \quad \forall i, j \notin s^* \end{aligned} \tag{23}$$

By extending the model we have:

$$p_{ij}(n+1) = \frac{\tau_{ij}(n+1)}{s_i(n+1)} = \frac{(1-\rho)\tau_{ij}(n) + \hat{\Delta}}{(1-\rho)s_i(n)+\Delta_s} \quad \forall i, j \notin s^* \tag{24}$$

where $\hat{\Delta} = \frac{\Delta}{N}$ , $\Delta_s = \Delta + N\hat{\Delta}$ and $N > 1$ is a constant value determined by the algorithm. It should be noted that, by the above definition, $\hat{\Delta}$ is a fraction of $\Delta$ so $\hat{\Delta} < \Delta$ which avoids algorithm divergence preventing undesirable actions to get more reinforcement than the most feasible one. Similar to (21), by extracting reinforcement factor from (24) we have:

$$r = \frac{\Delta_s \tau_{ij} - \hat{\Delta} s_i(n)}{((1-\rho)s_i(n)+\Delta_s)\tau_{ij}(n)} \tag{25}$$

The above relation shows that the resulted reinforcement factor is a function of $\tau_{ij}(n)$ and $r$ depends on the current amount of pheromone. This condition shows that reinforcement factor varies based on different actions. We refer to this situation as nonlinear pheromone update; therefore, the following definitions can be expressed based on the above math works:

*Difinition 1.* In ACO algorithms, a pheromone update strategy is linear if the pheromone update function is independent from the current value of the pheromone related to the selected action.

*Difinition 2.* In ACO algorithms, a pheromone update strategy is non-linear if the pheromone update function is a function of the current value of the pheromone related to the selected action.

Nonlinear pheromone update not only reinforces the current automaton towards better results, it also encourages the other potential automata to select better actions. We focus on the concept of exploration while adopting nonlinear update strategy into both ACS and Ant-Miner. We try to form a local obligation towards selecting better actions by defining a kind of coordinated exploration throughout the system.

## 6. Adopting Nonlinear Pheromone Update on ACS and Ant-Miner

Since our strategy deals with the update phase, we only modify the update pheromone phase in both algorithms. We embark by travelling salesman problem and then classification rule mining will be taken into account.

### 6.1 ACS with Nonlinear Pheromone Update

Our proposed strategy in ACS uses the same strategy as the original algorithm in tour construction phase in which each ant $k$ uses a pseudorandom proportional rule defined as (4) to move from city $i$ to city $j$ with parameters $\alpha = 1$ and $\beta = 3$. We ignored ACS local pheromone update and only the global update is considered in which only the ant that is responsible for the best-so-far tour is allowed to add pheromone after the iteration. In this step, nonlinear update plays its role through which both selected and some of the potential non-selected solutions are updated. In nonlinear method, pheromone update for best-so-far solution is accomplished according to (26) in which $c_{bs}$ refers to the cost of the best-so-far tour and $\rho_1$ corresponds to the evaporation rate for the best solution, and $s^*$ refers to the feasible solution.

$$\tau_{ij}(t+1) = (1-\rho_1)\tau_{ij}(t) + \Delta \quad \forall i, j \in s^* \tag{26}$$

$$\Delta = \frac{1}{c^{bs}} \tag{27}$$

On the other hand, for the other tours, the update strategy is triggered using a similar equation but with different parameters as:

$$\tau_{ij}(t+1) = (1-\rho_2)\tau_{ij}(t) + \hat{\Delta} \quad \forall i, j \in \ddot{s} \tag{28}$$

$$\hat{\Delta} = \frac{\Delta}{N} \quad \text{for } N > 1 \tag{29}$$

Where $\rho_2$ corresponds to the evaporation rate, $\ddot{s}$ is the $N$ feasible solutions ranked after the best solution, $N$ itself is a constant value determined according to the algorithm. To coordinate exploration toward better results, $N$ is defined as the number of neighbors with larger amount of pheromone. In our simulations we considered $N = 20$; therefore, in addition to the best solution, some other 20 top solutions are also favored. Relation (28) is the ingredient added to ACS to favor feasible alternatives which improve the exploration. As simulation results show, not only this strategy results in better tour lengths, it also reduces the required number of iterations and algorithm's run time.

### 6.2 Ant-Miner with Nonlinear Pheromone Update

The major anatomy of our non-linear Ant-Miner follows Ant-Miner 3.0, but we used the update relation proposed in Ant-Miner 1.0 because its simplicity copes well with the nonlinear pheromone update model. It also takes advantage of a rule pruning strategy as a daemon action according to relation (10).

In our algorithm, pheromone update strategy (13) for the selected term is changed into (30) similar to Ant-Miner 1.0 [18]. This is a simple form of ACO pheromone update in comparison with the complicated form used in Ant-Miner 3.0.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + Q\tau_{ij}(t) \quad \forall i, j \in s^* \tag{30}$$

Where $Q$ corresponds to rule quality calculated through(10). For the other terms, pheromone values are increased as:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \frac{Q}{2 \times N}\tau_{ij}(n) \quad \forall i, j \notin s^* \tag{31}$$

Where $N$ corresponds to the number of attributes in the training set. This strategy favors other terms to be considered for participating in the current rule. Favoring other terms by the quality factor also prevents stagnation in this algorithm. Similar to non-linear ACS, by using non-linear pheromone update in Ant-Miner much more extensive domain of terms are considered which can result in more accurate and comprehensive rules.

## 7. Simulation and Performance Evaluation

We simulated nonlinear update strategy on original versions of ACS and Ant-Miner 3.0 algorithms considering the modifications described in the previous section. The main skeleton of the previous algorithms are kept unchanged. For the traveling salesman problem we considered two challenges of memory consumption and algorithm's run time in the selection of a competing algorithm. We had to dismiss all approaches using $n$ exploring ants where $n$ is equal to the number of cities because of their inapplicable memory consumption and runtimes in huge graphs. We selected ACS+NN algorithm proposed in [25] considering its improvements and applicability in huge search spaces. In this algorithm the nearest neighbor (NN) strategy is used as a kind of local search method to boost the efficiency of ACS by selecting nearest neighbors in craeting the initializing tour. Having such an initialization, the time taken by the ants to find the shortest path will be reduced. On the other hand, this scheme of initialization heps the ants to find better tours by means of these initial suggestions. For an extensive evaluation we simulated four algorithms namely, ACS+NN, ACS with non-linear pheromone update, Ant-Miner 3.0, and finally Ant-Miner with non-linear pheromone update. Each pair is run on same input simulation data on the same machine to have accurate simulation results.

### 7.1 Simulation Environments

For travelling salesman problem we used an open source software package developed by *Thomas Sttzle* available online at: *http://www.aco-metaheuristic.org/aco-code*. We coded ACS+NN on the original code applying the nearest neighbor strategy in the initialization phase. To develop the nonlinear version, the pheromone update phase of the original code was modified and replaced with the novel nonlinear update function. Simulations were run on 6 selected graphs from TSPLIB graph repository available at: *http:// www.iwr.uni-heidelberg.de /groups /comopt/software /TSPLIB95 /tsp/* having 198, 1291, 3038, 5934, 7397, and finaly 11849 nodes. The characteristics and structures of these graphs can be found in Table 1. Parameter initialization was identically accomplished for both ACS+NN and Non-linear ACS according to the values illustrated in Table 2[†].

For classification rule mining, experiments are ac-

**Table 1** Graphs used in traveling salesman problem experiments (EUC_2D corresponds to 2-dimensional Euclidean distance and CEIL_2D refers to the rounded up EUC_2D used in distance computation).

| Graph | No. of Cities | Type | Optimum Solution | Origin |
|---|---|---|---|---|
| d198 | 198 | EUC_2D | 15780 | TSPLIB |
| d1291 | 1291 | EUC_2D | 50801 | TSPLIB |
| pcb3038 | 3038 | EUC_2D | 137694 | TSPLIB |
| rl5934 | 5934 | EUC_2D | 556045 | TSPLIB |
| pla7397 | 7397 | CEIL_2D | 23260728 | TSPLIB |
| rl11849 | 11849 | EUC_2D | 923288 | TSPLIB |

**Table 2** Initialization of constant parameters for both ACS+NN and Nonlinear ACS.

| Parameter | Value | Algorithms |
|---|---|---|
| Max Time | 10s | ACS+NN / Nonlinear ACS |
| No.Ants | 25 | ACS+NN / Nonlinear ACS |
| Neighbors Bound | 20 | Nonlinear ACS |
| $\xi$ | 0.1 | ACS+NN |
| $\alpha$ | 1.0 | ACS+NN / Nonlinear ACS |
| $\beta$ | 2.0 | ACS+NN / Nonlinear ACS |
| $\rho$ | 0.5 | ACS+NN / Nonlinear ACS |

[†]in Table 2 some parameters are used for just one of the algorithms because it does not exist in the other one. For example the parameter *Neighbors Bound* refers to the $N$ best sub-solutions that the nonlinear update strategy considers for its pheromone update process.

complished through *GUIAntMiner* software, developed in JAVA by *Fernando Meyer* under the orientation of *Rafael Stubs Parpinelli* who is the initiator of ant-based rule mining [18]. The code can be downloaded from *http://sourceforge.net*. Simulation results were generated using five standard databases from the *UCI Irvine Machine Learning Dataset Repository* used to analyzed rule mining algorithms (*http://sourceforge.net*). The characteristics and structures of these datasets can be found in Table 3. To train the algorithm about 70% of datasets were used as training sets. Parameter initialization was identically accomplished for both AntMiner 3.0 and Non-linear AntMiner according to the values illustrated in Table 4. All algorithms were simulated on a Double Core Pentium PC 3.00 GHz with 4 GB RAM under Windows 7 32-bit operating system.

### 7.2 Performance Evaluation Metrics

To evaluate nonlinear pheromone update in solving travelling salesman problem, four major metrics were considered for the original and the nonlinear algorithm. These metrics are:

- *Average branching factor* which estimates the exploration of the algorithm by analyzing the distribution of pheromone trail values.
- *Tour length information* a group of data about the best generated tours for both algorithms such as the best try, average best tour length, standard deviation and error.
- *Average iteration for the best tour* which shows the average required iterations to construct the best tour. This metric shows how good artificial ants are performing to obtain the best tour which reveals the convergence speed of the algorithm.
- *Average best experienced time* which shows how fast the algorithm finds the feasible solution.

To evaluate the results of non-linear pheromone update on Ant-Miner in classification rule mining, three major metrics were analyzed which are:

- *Number of discovered rules* which shows the number of discovered classification rules for each data set.
- *Total elapsed time* which reflects the algorithm's run time for each data set.
- *Accuracy rate* which shows how accurate the discovered rules are according to datasets.

To calculate *Accuracy Rate* the folllowing calculation was used:

$$AccuracyRate = \frac{TP + TN}{TP + FN + FP + TN} \tag{32}$$

Where all the acronyms are the same as mentioned in Sect. 4.

### 7.3 Simulation Results and Diagrams for ACS+NN and Nonlinear ACS

To comprehensively evaluate the proposed nonlinear version of ACS, we simulated both the ACS+NN and our algorithm on 6 completely connected graphs mentioned in Table 1. The results are obtained from 30 independent simulation runs and for both algorithms 25 artificial ants were considered. Figure 1 shows the average branching factor for ACS+NN and ACS with non-linear pheromone update. As shown in this figure, the non-linear approach outstrips the ACS+NN in the sense of exploration in a steady way independent from the number of cities. The reason for this improvement is reinforcing more trails which prevents artificial ants to be trapped in local optima in the huge problem search spaces. As shown in the figure, the nonlinear approach improves the branching factor by upto 12% in some cases.

Figure 2 shows the average iteration for the best tour resulted by both algorithms. The non-linear approach shows its efficiency in convergence speed since less iteration is required to find the desirable solution. For this metric, improvements in some cases are more than 40%. Figure 3 diagrams the average best experienced time resulted by both algorithms. As shown in the diagram, the non-linear approach experiences much less time than ACS+NN caused

**Table 3** Datasets used in the experiments (Tic. stands for Tic-Tac-Toe; L.B.C. stands for Ljubljana Breast Cancer; W.B.C. stands for Wisconsin Breast Cancer; D.T.Y. stands for Dermatology; H.P stands for Hepatits; Cat. stands for Categorical; Con. stands for Contineous).

| Data Set | No. of Cases | Cat. Attributes | Con. Attributes | Classes |
|---|---|---|---|---|
| Tic. | 282 | 9 | NA | 2 |
| L.B.C | 683 | NA | 9 | 2 |
| W.B.C | 985 | 9 | NA | 2 |
| D.T.Y | 366 | 33 | 1 | 6 |
| H.P | 155 | 13 | 6 | 2 |

**Table 4** Initialization of constant parameters for both AntMiner 3.0 and Nonlinear AntMiner.

| Parameter | Value | Algorithms |
|---|---|---|
| Number of Ants | 10 | AntMiner 3.0 / Nonlinear AntMiner |
| Min Case Per Rule | 5 | AntMiner 3.0 / Nonlinear AntMiner |
| Max Uncovered Cases | 10 | AntMiner 3.0 / Nonlinear AntMiner |
| Rules for Convergence | 10 | AntMiner 3.0 / Nonlinear AntMiner |
| Number of Iterations | 100 | AntMiner 3.0 / Nonlinear AntMiner |
| $\rho$ | 0.1 | AntMiner 3.0 / Nonlinear AntMiner |
| $\varphi$ | 0.4 | AntMiner 3.0 / Nonlinear AntMiner |

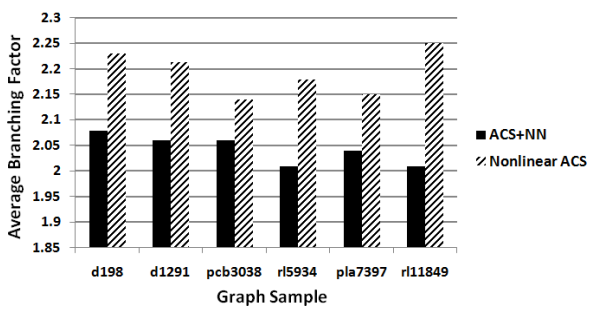



**Fig. 1** Average branching factor for ACS+NN and ACS with non-linear pheromone update for TSP.

by the less iteration required in the non-linear approach. algorithm's run time in some cases reduces by 39%.

Table 5. and Table 6. show the information about the tours generated by ACS+NN and Nonlinear ACS algorithms respectively. These tables contain the best try, average of the best tours, standard deviation for the best tour, and error which was calculated as the difference between the optimum solution and the solution found by the algorithms. The information show the improvements caused by the nonlinear strategy in comparison with the ACS+NN algorithm.

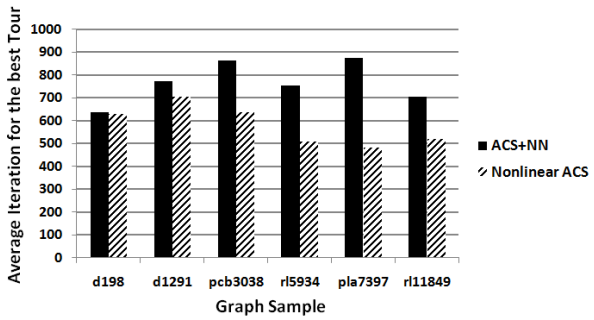



**Fig. 2** Average iteration for the best tour resulted for ACS+NN and ACS with non-linear pheromone update for TSP.

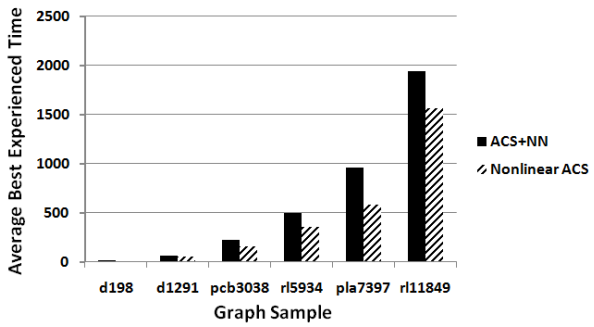



**Fig. 3** Average best computational time for ACS+NN and the non-linear algorithm for TSP.

**Table 5** Tour information for ACS+NN algorithm (Std-Dev stands for Standard Deviation).

| Graph | Best Solution | Average Best | Std-Dev | Error |
|---|---|---|---|---|
| d198 | 16353 | 16700.37 | 209.86 | 573 |
| d1291 | 55336 | 56669.63 | 935.68 | 4535 |
| pcb3038 | 163459 | 166798.53 | 1452.48 | 25765 |
| rl5934 | 648899 | 663131.87 | 6964.85 | 92854 |
| pla7397 | 27711693 | 28210545.33 | 227682.09 | 4450965 |
| rl11849 | 1103806 | 1119696.73 | 6977.67 | 180518 |

**Table 6** Tour information for ACS with nonlinear pheromone update(Std-Dev stands for Standard Deviation).

| Graph | Best Solution | Average Best | Std-Dev | Error |
|---|---|---|---|---|
| d198 | 16258 | 16655.77 | 228.25 | 478 |
| d1291 | 54793 | 55803.60 | 710.92 | 3992 |
| pcb3038 | 162793 | 164650.70 | 1011.12 | 25099 |
| rl5934 | 639483 | 648758.47 | 3826.23 | 83438 |
| pla7397 | 27438467 | 27847044.50 | 159528.34 | 4177739 |
| rl11849 | 1082890 | 1093732.73 | 5192.01 | 159602 |

Table 7 shows the percentage of improvements of the nonlinear strategy in comparison with ACS+NN algorithm for four important factors namely: average branching factor, average iteration for the best tour, average best experienced time, and finally the best tour. As shown in the table all these factos are improved in all problem samples.

### 7.4 Simulation Results for Ant-Miner

Simulation results for Ant-Miner 3.0 and Ant-Miner with nonlinear pheromone update are generated through 30 independent simulation runs with similar conditions for both algorithms. Table 8 contains accuracy rates generated by both algorithms. As figured in the table, the non-linear strategy is able to find more accurate rules than Ant-Miner 3.0. Table 9 involves the number of discovered rules for both algorithms. Evidently, for a rule mining algorithm it is preferred to find much more comprehensive rules, so smaller number of rules is considered as superiority. As shown in Table 9, in majority of datasets the non-linear approach results in fewer constructed rules. Table 10 presents the elapsed time for the process of rule construction for both algorithms. As shown in the table, the non-linear strategy reduces the required time for rule construction. This improvement is significant when the algorithm deals with large datasets like W.B.C and L.B.C. Detailed results for improvements caused

**Table 7** Improvements of ACS with non-linear pheromone update with respect to ACS+NN (ABF stands for Average Branching Factor, BT stands for Best Tour, ABI stands for Average Best Iteration, ABR stands for Average Best Run time).

| Graph | ABF | BT | ABI | ABR |
|---|---|---|---|---|
| d198 | 7.2% | 0.5% | 0.9% | 13.7% |
| d1291 | 7.4% | 1.0% | 8.5% | 10.3% |
| pcb3038 | 3.8% | 0.4% | 26% | 26.6% |
| rl5934 | 8.4% | 1.4% | 32% | 28.1% |
| pla7397 | 5.4% | 1% | 44.5% | 39.5% |
| rl11849 | 11.9% | 1.9% | 26% | 19% |

**Table 8** Comparison of accuracy rates for Ant-Miner 3.0 and Ant-Miner with nonlinear pheromone update (The acronyms are similar to those used in Table 3).

| Algorithm | Tic. | L.B.C | W.B.C | D.T.Y | H.P |
|---|---|---|---|---|---|
| Ant-Miner 3.0 | 70.53 | 75.02 | 90.66 | 97.28 | 81.13 |
| Nonlinear Ant-Miner | 74.73 | 76.62 | 92.99 | 97.53 | 83.5 |

**Table 9** Comparison of the number of discovered rules for both algorithms (The acronyms are similar to those used in Table 3).

| Algorithm | Tic. | L.B.C | W.B.C | D.T.Y | H.P |
|---|---|---|---|---|---|
| Ant-Miner 3.0 | 8.5 | 6.4 | 12.4 | 7.4 | 5.9 |
| Nonlinear Ant-Miner | 8.0 | 6.3 | 12.4 | 7.6 | 5.1 |

**Table 10** Comparison of total elapsed time for Ant-Miner 3.0 and Ant-Miner with non-linear pheromone update in seconds (The acronyms are similar to those used in Table 3).

| Algorithm | Tic. | L.B.C | W.B.C | D.T.Y | H.P |
|---|---|---|---|---|---|
| Ant-Miner 3.0 | 9 | 3 | 31 | 96 | 5 |
| Nonlinear Ant-Miner | 9 | 2 | 19 | 52 | 5 |

**Table 11** Improvements of Nonlinear AntMiner with respect to the AntMiner 3.0 (The acronyms are similar to those used in Table 3).

| Datasets | Accuracy Rate | No.Discovered Rules | Elapsed Time |
|---|---|---|---|
| Tic. | 6.0% | 5.9% | 0.0% |
| L.B.C | 2.1% | 1.5% | 33.0% |
| W.B.C | 2.5% | 0.0% | 38% |
| D.T.Y | 0.2% | –2.7% | 45% |
| H.P | 2.9% | 13.5% | 0.0% |

by our proposed rule mining algorithm are presented in Table 11.

## 8. Conclusion

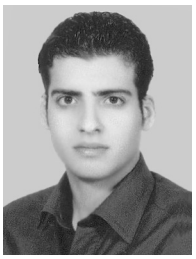
Our proposed strategy focuses on the concept of exploration which is one of the key points in swarm-based evolutionary algorithms particularly when the algorithm deals with huge search spaces. The way the algorithm controls exploration determines how it covers the problem search space. This issue is much more critical when the algorithm deals with an instance of *NP-hard* problems such as TSP and rule mining with a huge or dynamic search space. The proposed non-linear pheromone update strategy tries to increase exploration but in a controlled way to cover the potential areas of the problem space to find more feasible solutions. As simulation results show, our new idea improves both metrics of algorithms' runtimes and accuracy.

### References

[1] M. Dorigo and T. Sttzle, Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.
[2] M. Dorigo, M. Bonabeau, and E. Theraulaz, "Ant algorithms and stigmergy," Future Generation Computer Systems, vol.16, no.8, pp.851–871, 2000.
[3] P. Lalbakhsh, B. Zaeri, A. Lalbakhsh, and M.N. Fesharaki, "AntNet with reward–penalty reinforcement learning," Proc. 2nd Int. Conf. Computational Intelligence, Communication Systems, and Networks, pp.17–21, Liverpool, UK, 2010.
[4] K. Verbeeck and A. Now, "Colonies of learning automata," IEEE Trans. Syst., Man. Cybern. B, Cybern., vol.32, no.6, pp.772–780, 2002.
[5] G. Di Caro, D. Frederick, and L.M. Gambardella, "AntHocNet:An adaptive nature-inspired algorithm for routing in mobile ad hoc networks," European Transactions on Telecommunicationsm, vol.16, pp.443–455, 2005.
[6] G.Di Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," J. Artificial Intelligence Research, vol.9, pp.317–365, 1998.
[7] T. Sttzle and H.H. Hoos, "Max-Min ant system," Future Generation Computer Systems, vol.16, no.8, pp.889–914, 2000.
[8] E. Aarts, Local Search in Combinatorial Optimization, John Wiley & Sons, 1999.
[9] B. Bullnheimer, R.F. Hartl, and C. Strauss, "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, 1999.
[10] O. Cordon, I.F. De Viana, F. Herrera, and L. Moreno, "A new ACO model integrating evolutionary computation concepts: The best-worst ant system," Proc. 2nd Int. Workshop on Ant Algorithms, Brussels, pp.22–29, 2000.
[11] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.
[12] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," Technical Report 91-016 revised, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.
[13] M. Dorigo and V. Maniezzo, "Ant system: optimization by a colony of cooperating agents," IEEE Trans. Syst., Man. Cybern., B: Cybern., vol.26, no.1, pp.29–41, 1996.
[14] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1992.
[15] M. Dorigo and L.M. Gambardella, "Ant colonies for the travelling salesman problem," biosystems, vol.43, no.2, pp.73–81, 1997.
[16] M. Dorigo and L.M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem," IEEE Trans. Evol. Comput., vol.1, no.1, pp.53–66, 1997.
[17] T. Sttzle and H.H. Hoos, "The max-min ant system and local search for the travelling salesman problem," Proc. IEEE Int. Conf. Evolutionary Computation, NJ, USA, pp.309–314, 1997.
[18] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., vol.6, no.4, pp.312–332, 2002.
[19] B. Liu, H.A. Abbas, and B. McKay, "Density-based heuristic for rule discovery with AntMiner," Proc. 6th Australian-Japan Joint Workshop on Intelligent and Evolutionary Systems, pp.180–184, Canberra, Australia, 2002.
[20] B. Liu, H.A. Abbas, and B. McKay, "Rule discovery with ant colony optimization," Proc. IEEE/WIC Int. Conf. Intelligent Agent Technology, China, pp.83–88, 2003.
[21] P. Lalbakhsh, M.S.K. Fasaei, B. Zaeri, and M.N. Fesharaki, "Focusing on rule quality and pheromone evaporation to improve ACO rule mining," Proc. IEEE Int. Symp. Computers and Informatics, pp.108–112, Kuala Lumpur, Malaysia, 2011.
[22] K. Narendra and M.A.L. Thathachar, Learning Automata: An Introduction, Prentice Hall, New Jersey, USA, 1989.
[23] J. Oomen and S. Mirsa, "Cybernetics and learning automata," Springer Handbook of Automation- Part B, vol.12, no.1, pp.221–235, 2009.
[24] G. Di Caro, Ant colony optimization and its application to adaptive routing in telecommunication networks, PhD Thesis, Polytechnic School, Universit Libre de Bruxelles, Brussels, Belgium, 2005.
[25] C.F. Tsai, C.W. Tsai, and C.C. Tseng, "A new hybrid heuristic approach for solving large ttraveling salesman problem," Inf. Sci., vol.166, Issues 1–4, pp.67–81, 2004.

**Pooia Lalbakhsh** was born in Kermanshah, Iran in 1981. He received the B.Sc. degree in Computer Hardware Engineering from Islamic Azad University, Maybod Branch, in 2003 and M.Sc. in Computer Architecture from Islamic Azad University, Science and Research Branch, Tehran in 2006. His research interests include Swarm Intelligence, Ant Colony Optimization, and Network Centric Warfare.

**Bahram Zaeri** was born in Borujerd, Iran in 1986. He received the B.Sc. degree in Computer Software Engineering from Islamic Azad University, Borujerd Branch, in 2005.



**Ali Lalbakhsh** was born in Tehran, Iranin 1986. He received the B.Sc and M.Sc. degree in Tlecommunication Engineering from Islamic Azad University.