International Journal of Innovative
Computing, Information and Control
Volume 8, Number 4, April 2012

ICIC International ©2012 ISSN 1349-4198
pp. 2789–2802

IMPROVING PERFORMANCE OF ACO ALGORITHMS USING CROSSOVER MECHANISM BASED ON BEST TOURS GRAPH

Aybars Ugur[1] and Dogan Aydin[2]

[1]Department of Computer Engineering
Ege University
Bornova-Izmir 35100, Turkey
aybars.ugur@ege.edu.tr

[2]Department of Computer Engineering
Dumlupinar University
Kutahya 43030, Turkey
dogan.aydin@dpu.edu.tr

Received November 2010; revised March 2011

Abstract. *Several algorithms have been proposed that are based on the ant colony optimization (ACO) meta-heuristic in literature. This paper proposes an extra data structure that we called best tours graph feeding the pheromone trail information for ACO algorithms. Best tours graph is a table that blends the information on the global best tours encountered statistically during iterations and includes the strengths of edges. The table is crossover of favorable edges, and the technique provides a simple crossover mechanism. A powerful pheromone reinforcement mechanism is also developed based on the best tours table to increase the performance of ACO algorithms in this study. Algorithms are tested on Traveling Salesman Problem using TSPLIB. Our experiments and comparisons show that the method improves the performance of almost all original ACO algorithms.*
Keywords: Ant colony optimization algorithms, Pheromone updating, Meta-heuristic, Crossover, Graph, Traveling salesman problem

1. **Introduction.** Several algorithms based on Ant Colony Optimization (ACO) meta-heuristic have been proposed in literature. The swarm intelligence is defined by Bonabeau et al. [1] as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies". Ant Colony Optimization (ACO) Algorithms, Artificial Bee Colony (ABC) Algorithms [2,3], which mimic the food foraging behavior of swarms of honey bees, Particle Swarm Optimization (PSO) Algorithms, which simulate the social behavior of bird flocking or fish schooling, and Artificial Immune System (AIS) Algorithms inspired by the principles and processes of the vertebrate are examples of swarm intelligence.

Ant Colony Optimization as a swarm intelligence method is a mathematical model and a probabilistic technique for solving computational optimization problems. The model is based on the foraging behavior of some ant species. Ants lay down pheromone trails in finding the shortest path between the colony and the food. These trails represent the quality of the ways that are used to find favorable paths by colony members.

Different ACO algorithms have been developed for solving combinatorial optimization problems such as routing (traveling salesman, vehicle routing, sequential ordering) [4-6], assignment (quadratic assignment, timetabling, graph coloring) [7-9] and scheduling problems [10-13]. Recently, ACO algorithms have been applied to continuous optimization [14-18]. Some published studies about using ACO algorithms to solve engineering problems can be found in [19-23].

One of the most important combinatorial optimization and global search heuristics problems is the Traveling Salesman (Salesperson) Problem (TSP). Euclidean TSP, which is a sub-case of metric TSP, is an NP-hard problem. All of the possible permutations of cities can be listed, and the total distance for each permutation can be calculated to find the shortest route [24]. In this case, the total number of permutations is $(n-1)!$ and Big O time is $O(n!)$. For example, a 50-city TSP has 6.08×10E+62 and a 100-city TSP has 9.33×10E+155 possible solutions. A brute-force (or exhaustive) search examines all the possible solutions and search space rapidly increases with the problem size. Rather than enumerating all possibilities, approximation algorithms are usually employed to find near-optimal solutions quickly for large problem sizes.

The TSP has many applications in transportation systems [25]. Arranging efficient bus routes that pick people up from specified locations can be modeled as a TSP. Development of efficient TSP solvers is also important in path planning for airlines, delivery trucks, postal carriers and computer networks. Since the 1950s, many researchers (computer scientists, mathematicians, biologists and others) have attempted to develop TSP solving methods, such as evolutionary methods, genetic algorithms, ACO algorithms, neural networks, tabu search, simulated annealing and greedy algorithms, to find the cheapest tour or shortest route for a given collection of cities.

Evolutionary algorithms (EA), such as genetic algorithms (GA) and ACO, have been applied to the TSP, since they allow an efficient evolution toward quality sub/optimal solutions. Information concerning good solutions in the current iteration/cycle is forwarded to the next exploring generation, to refine the search toward greater optimality. Traits of genes obtained through the use of genetic operators, such as crossover and mutation, are forwarded to the offspring of the new generation. In ACO, the positive feedback of pheromone deposits on arcs comprising more optimal node-arc tours (paths) allows the next cycle/iteration to progress toward a solution of greater optimality [26].

Crossover in GA, which is a genetic operator used to vary chromosomes, selects genes from parent chromosomes and produces new offspring over generations. The idea behind crossover is that the new chromosome may be better than all of the parents if it takes the good characteristics from each of the parents, as in natural phenomenon. The new chromosome is then a better solution and has more performance; so selecting the best chromosomes from the population to be parents affects the general performance of the genetic algorithms. Over the years, numerous variants of crossover have been developed in the GA literature.

We proposed an extra data structure, which we called best tours graph (table), for feeding the pheromone trail information. The best tours graph blends the information about the global best tours encountered statistically during iterations. Original ACO algorithms have a weakness of crossover with respect to genetic algorithms. The method we proposed provides a simple and effective crossover mechanism. We also developed powerful pheromone reinforcement mechanism based on the best tours table to increase the performance of TSP solving ACO algorithms.

The remainder of the paper is organized as follows. Section 2 reviews basic principles of ACO algorithms and the related work on pheromone updating mechanisms. Section 3 describes the mechanism that we developed and the details of the data structure that contributed to previous best tours. The experimental study is presented in Section 4. Conclusions are given in Section 5.

2. **ACO Algorithms.**

ACO algorithms take inspiration from the behavior of real ant colonies to solve combinatorial optimization problems. They are based on a colony of artificial ants, that is, simple computational agents that work cooperatively and communicate

through artificial pheromone trails [27,28]. All Ant Colony Optimization algorithms share the same ideas. The basic ACO meta-heuristic is shown below [29]:

Algorithm 1 The Ant Colony Optimization Meta-heuristic

```
Set parameters, initialize pheromone trails
while termination condition is not met do
  ConstructAntSolutions
  ApplyLocalSearch (optional)
  UpdatePheromones
end while
```

FIGURE 1. The ant colony optimization meta-heuristic

ACO algorithms are essentially construction algorithms: In each algorithm iteration, every ant constructs a solution to the problem by traveling on a construction graph. Each edge of the graph, representing the possible steps the ant can make, is associated with two kinds of information that guide the ant movement [1]:

- Heuristic information, which measures the heuristic preference of moving from node i to node j, i.e., of traveling the edge a_{ij}. It is denoted by η_{ij}. This information is not modified by the ants during the algorithm run.
- (Artificial) pheromone trail information, which measures the "learned desirability" of the movement and mimics the real pheromone that natural ants deposit. This information is modified during the algorithm run depending on the solutions found by the ants. It is denoted by τ_{ij}.

All m ants complete their tours using the rule of selecting next city (state transition rule) in the ConstructAndSolutions step of ACO algorithm for TSP. The ApplyLocalSearch step is optional, and solutions obtained by ants can be improved by local search algorithms. The aim of the UpdatePheromones phase is to increase the pheromone values of promising solutions and decrease the undesirable ones.

The ACO meta-heuristic can be easily applied to TSP, which is one of the most studied NP-hard problems. This leads researchers developing techniques to use the TSP for benchmarking. The ACO algorithms proposed for the TSP are Ant System (AS), Elitist Ant System (ASe) [30], Ant Colony System (ACS) [31], Rank-based Ant System (RBAS) [32], Best-Worst Ant System (BWAS) [33] and Max-Min Ant System (MMAS) [34].

2.1. **Ant system.** Ant System (AS) is the first ACO algorithm, and three versions of it were proposed. Ants update the pheromone directly after a move from one city to an adjacent one in ant-density and ant-quantity algorithms. In ant-cycle, the pheromone update was done only after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality [35].

Preliminary experiments on a set of benchmark problems [30,36,37] have shown that the ant-cycle's performance was much better than those of the other two algorithms. Consequently, research on AS was directed toward a better understanding of the characteristics of the ant-cycle, which is now known as Ant System, while the other two algorithms were abandoned [30].

Tour Construction: Initially, each ant is put into a randomly chosen city. The algorithm executes $t_{\max}$ iterations. m ants build a tour executing n steps, in which a probabilistic decision (state transition) rule is applied for each iteration. In the construction of a solution, each ant selects the next city to be visited through a stochastic

mechanism. The choosing probability of ant k to go from city i to city j at the tth iteration of the algorithm is calculated as shown in Equation (1).

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}][\eta_{ij}]}{\sum_{h\in\Omega}[\tau_{ih}][\eta_{ih}]} & \text{if } j \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where τ_{ij} is the amount of pheromone trail between nodes i and j, η_{ij} is the desirability or visibility of edge (i, j) ($\eta_{ij} = 1/d_{ij}$), α is a parameter to control the influence of τ_{ij}, β is a parameter to control the influence of η_{ij}, d_{ij} is the distance between city i and city j, Ω is the set of unvisited cities.

τ_{ij}, the intensity of the trail, can be interpreted as an adaptive memory and is regulated by a constant parameter α. η_{ij} can be interpreted as a measure of desirability and is called visibility; its influence is controlled by a constant parameter β. It is important to note that the constant adjustable parameters, α and β, change the algorithm performance dramatically.

Pheromone Updating: After all ants finish their tours the pheromone trails are updated. The quantity of pheromone per tour is the same for all artificial ants as an assumption, so more pheromone is laid per unit length on shorter tours. First, AS evaporates some value of pheromone from all edges and deposits, as Equation (2) demonstrates:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2}$$

where ρ ($0 \leq \rho \leq 1$) is the evaporation rate or coefficient of decay and provides to forget previous bad decisions, and $\Delta\tau_{ij}^{k}(t) = 1/L^{k}(t)$ if edge(i, j) is used by ant k.

So, each ant in the colony retraces the path it has followed and deposits $1/L_k$ (L_k is the length of its tour) amount of pheromone independently.

2.2. **Other ant colony optimization algorithms.** Different ant colony optimization algorithms that can be applied to the well-known TSP are proposed in the literature. This subsection discusses these methods and briefly explains their differences with respect to AS.

Elitist Ant System (ASe) is an extended version of Ant System proposed by the same authors. It uses an elitist strategy that employs a number of elitist ants, which gives a strong additional reinforcement to the edges belonging to the global best tour. Global best tour is the best tour found since the start of the algorithm.

Ant Colony System (ACS) introduces three major modifications. First, ACS uses pseudo-random proportional action choice rule, which is more aggressive. Second, it allows pheromone addition only to the edges belonging to the global best solution. Third, ants apply local pheromone trial update (online step-by-step pheromone trial update). Both evaporation and deposit are applied to edges immediately after having crossed for moving from city i to city j. This ensures that not all ants follow the same path and increases the exploration of not-yet visited edges.

Rank-based Ant System (RBAS) is similar to elitist strategy but employs a number of best ants of each iteration to deposit pheromone. Ants are sorted by tour length for this aim.

Best-Worst Ant System (BWAS) uses the transition rule of AS. Main modifications with respect to AS are

- Global best and current worst solutions are used to update positively and negatively.
- Restarts the search process when stagnant.
- Each component of the pheromone matrix is mutated.

Max-Min Ant System (MMAS) includes three main modifications with respect to AS:

- Only global best or iteration best tour is allowed to add pheromone.
- Range $[\tau_{\min}, \tau_{\max}]$ of the pheromone trails is limited to avoid stagnation.
- All edges (about pheromone trails) are initialized to the upper trail limit.

3. **The Proposed Method: A Crossover Mechanism for ACO Algorithms.** The method that we proposed is based on the Best Tours Graph (or best tours table) that we introduced for ACO algorithms. A best tours graph includes all the edges encountered in global best solutions. This is a framework for the pheromone table and also more stable than it. Recent global best solutions are better than the previous ones, so we developed an updating mechanism for adjusting weights by also considering this issue. Details and the pheromone updating mechanism based on the best tours graph are explained in the next subsections.

3.1. **Best tours table: construction and updating.** The best tours table that we proposed is an extra graph supporting the pheromone information of ACO algorithms. The weight of an edge of the graph represents the information about the strength of this edge and shows how much the edge is a candidate for the solution. ψ is the amount of strength on edge (i, j). The strength of any edge (ψ) is calculated by blending its existing information in the global best tours during iterations. It is important to record how many times and when this edge was a part of global best solutions. A best tours table is composed from these values and is a function of the best tours encountered.

Initially all edges are set to 0. This table (graph) is updated when a new global best solution is found. Updating the Best Tours table is a three-step process:

- Strength Decrease: This phase evaporates some value of strength by Equation (3):

$$\psi_{ij}(t) = (1 - \rho_{BT})\psi_{ij}(t) \tag{3}$$

  where ρ_{BT} $(0 \leq \rho_{BT} \leq 1)$ is the evaporation rate of best tours table from all edges. Parameter is introduced with this method and controls the evaporations for updating Best Tours Table. Evaporation must be applied, because the importance of previous best solutions must be decreased when a better tour is found.
- Strength Increase: Weights of corresponding edges belonging to the new global best tour are incremented by 1.
- Normalizing: All weights of the graph are normalized by dividing the maximum value in the table. This limits the values in $[0, 1]$ in the table and provides ease of control for next phases.

Assuming that the best tour found in the first iteration as shown in the graph of Figure 2 on the right, related weights of edges are updated, as in Figure 2 on the left, with our algorithm according to the best tour. Since the matrix is symmetric for symmetric TSP, only its upper half (upper triangle) is stored.

Assuming the second global best tour shown in the graph of Figure 3 is found after some iterations, the best tour table in Figure 3 is obtained with $\rho_{BT} = 0.2$.

This mechanism is provided by evaporating the previous table, adding the effect of new best tour and normalizing. After these operations all weights are in the range of $[0, 1]$. Corresponding edge weights of a new global best tour are increased by 1 before normalization, and this provides the edges of the new global best solution have the maximum values for next iterations. Ants are forced to find solutions around the new global best tour, but also previous global best tours are taken into account. This table is the version after crossover of global best tours and the result of a kind of statistical blending.

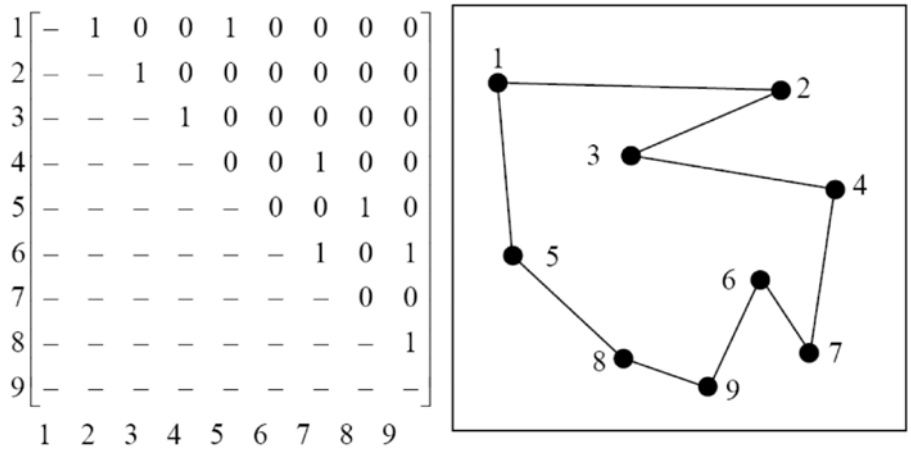

FIGURE 2. The best tour and best tours table after the first iteration

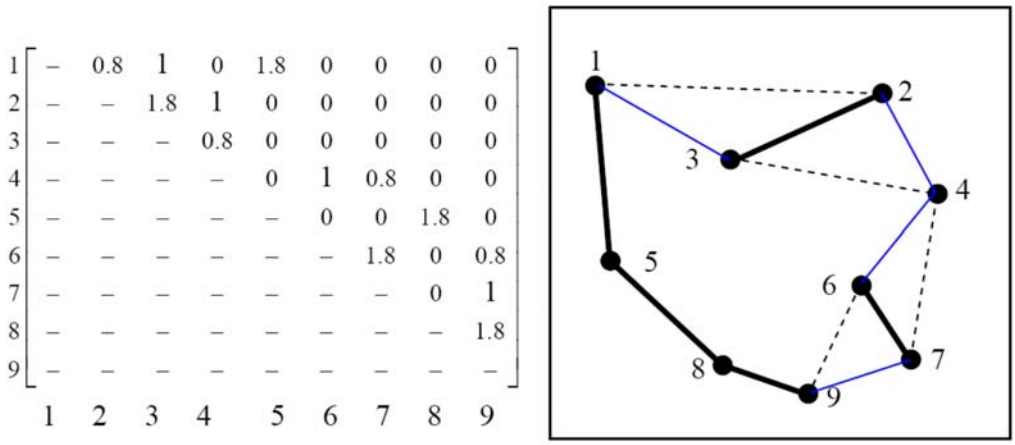

FIGURE 3. Second best tour found and best tours table after updating. Blue lines represent the new edges of the tour. Dashed lines represent the previous edges that are not in the new best tour.

We designed the best tours table to be more stable as a backbone of the pheromone table. Main roads are crystallized as finding new global best tours. Because its edges proved themselves during previous iterations as parts of global best tours, these are candidates for the solution. Feeding the pheromone information with this information obtained during the algorithm improves the performance.

The best tours graph is more dynamic for the first iterations than the later iterations generally, because best solutions are more distant from the ideal best solution for the first iterations and ants can find better solutions easily. The best tours graph is sparser for the first iterations than for the later iterations. As the number of global best solutions encountered increases during iterations, the number of edges (which have values different from 0) of the graph increases. The evolution of the best tours table for a random instance having 20 cities is recorded by our environment [38] during the iterations, as shown in Figure 4.

3.2. **Pheromone updating mechanism.** First of all, pheromone updating rules of the related ACO algorithm are applied for all iterations. After that, the pheromone table is also reinforced by the best tours table, in which one of the two conditions is met:

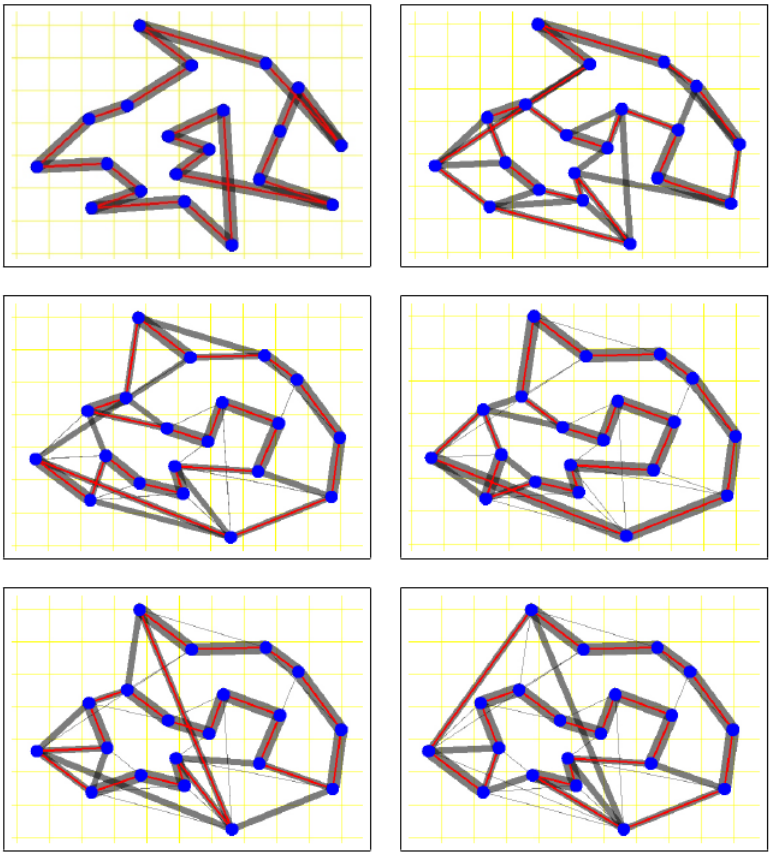

FIGURE 4. Changes in the best tours table for a 20 city random sample. Each figure shows the best tour table values after new best tour has found. Red and gray lines indicate the best found tour and table values, respectively. Thicknesses of gray lines represent the quantities of the table values.

- When new global best solution is found, the pheromone table is reinforced by the best tours table. It is applied using Equation (4) for the next RL_1 iterations.

$$\tau_{ij}(t+1) = \tau_{ij}(t) * \psi_{ij}(t) * k_b * (T_{NN}/T_{best}) \tag{4}$$

  where k_b is the coefficient to control the effect of the best tours table on the pheromone table, T_{NN} is the tour length of the TSP calculated by using nearest neighbor algorithm, and T_{best} is the tour length of the new best tour found.

  RL_1 controls the running length of best tours table reinforcement. For example, after the new best tour is found, the best tours table reinforces the pheromone table for RL1 iterations.
- After some stagnant iterations (S_{itr}) all pheromone trails on edges are reinitialized, similar to the Max-Min Ant System, and after that, the best tours table reinforces the pheromone table for RL_2 iterations. For example, after some iterations without a new best tour encountered, the pheromone table is initialized. After initialization, the pheromone table is reconstructed by combining the related ACO algorithm and the effect of the best tours table using the equation $k_b * T_{NN}/T_{best}$. In this way, exploration of new best tours around previous best tours is increased.

Out of these two conditions, the original pheromone updating mechanism of the related ACO algorithm is applied. The original ACO algorithm continues until one of these two conditions is met. Figure 5 shows an example of the proposed updating mechanism during iterations. After the new global best solution is found (at iteration 100 for this example),

RL_1 period is started. The best tours table reinforces the pheromone table for RL_1 ($RL_1 = 30$ for this example) iterations. After iteration 130, the original ACO algorithm continues up to iteration 350 (= period starting point + period length). It is assumed that a new best tour is not found in this period. After that point, all edges (about pheromone trails) are initialized and RL_2 period is started; thus, the best tours table reinforces the pheromone table for (for example) $RL_2 = 40$ iterations. Periods of first iterations are not shown in Figure 5 for simplification and understandability. The probability of finding new best tours is high during the first several iterations, and many RL_1 periods may be restarted before the previous one is completed.

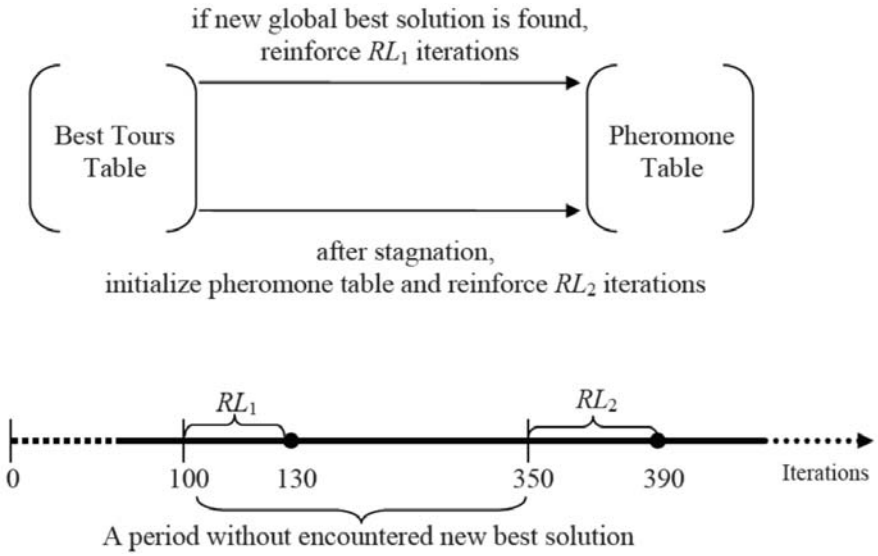

FIGURE 5. Periods of pheromone updating mechanism during iterations (assuming a new best tour was found at iteration 100) show the difference of proposed mechanism from the original ACO algorithms

4. **Experimental Results.**

4.1. **Parameter setting.**

ACO algorithms have a number of parameters that influence algorithm search behavior, making the performance of ACO algorithms strongly dependent on the particular values of the parameters. Another important fact is that one algorithm can outperform another because of optimizing its parameters successfully. Facilitating more meaningful comparative studies and avoiding unfair comparisons, automatic parameter tuning algorithms are very useful and have been highly recommended in recent years. We used one of the well-known parameter tuning algorithms, Iterative F-race [39,40], for configuring the ACO parameters in this study.

F-race is based on racing algorithms in machine learning. It starts with finite sets of parameter configurations and iteratively runs target algorithms with all existing parameter configurations on a different problem instance. Some bad configurations are eliminated using a series of pair-wise testing after each iteration if a significant difference exists among the configurations after using a non-parametric Friedman test. This process continues until some good candidates are alive or the maximum budget is reached. In Iterative F-race, F-race is performed at each iteration but the new candidate configuration set of a new F-race is generated around the best parameter configurations obtained from previous F-race executions. More information about Iterative F-race can be found in [39,40].

We have found the best possible parameter configuration for ACO algorithms as follows: We have generated small TSP instances randomly and used each original ACO algorithm in the Iterative F-race as sample instances and target algorithms, respectively. After obtaining the parameter configuration of original ACO algorithms, we optimized additional parameters coming from our proposed approach for each ACO variant. Iterative F-race also has some parameters to be set. We have set the values of those parameters, maximum budget (M) and constant parameter μ, as 5000 and 6 respectively. Under these conditions, the best parameter configuration sets obtained with Iterative F-race are shown in Table 1.

TABLE 1. Parameters and their corresponding values obtained with iterative F-race

Parameters	**AS**	**ASe**	**RBAS**	**ACS**
m	100	32	75	21
α	0.661	0.5399	0.4135	0.925
β	3.862	6.218	5.314	8.293
ρ	0.9292	0.4615	0.4607	0.3831
r (rank limit for RBAS)	—	—	8	—
e (number of elitist ants for ASe)	—	10	—	—
S_{itr} (Stagnant iterations)	292	163	280	226
ρ_{BT}	0.71	0.78	0.69	0.74
RL_1	64	18	63	55
RL_2	57	66	37	78
k_b	5	7	6	5

4.2. **Comparison results.** The best tours table and mechanism that we proposed are tested on numerous benchmark problems from TSPLIB with satisfactory results, and comparisons with the original ACO algorithms are presented in this section.

We set parameter values as indicated in Table 1 for all the experiments in this section. All the investigated algorithms were run 10 times, each of which terminates after n.10000 tour constructions on each TSPLIB instance. Our analyses are based on average values or errors ($Error_{ave} = ((Result_{ave} - Optimum)/Optimum) * 100$).

In the first stage, we compared the original ACO algorithms with improved versions that we proposed (AS+BT, ASe+BT, RBAS+BT, ACS+BT; +BT indicates "the algorithm with best tour tables"). It is important to note that we did not adapt our mechanism to MMAS and BWAS, which have pheromone smoothing or/and specific restart mechanisms themselves. Therefore, our mechanism is not necessary or useful for MMAS and BWAS.

Table 2 shows the comparison results. In Table 2, it is clearly seen that the proposed mechanism is valuable and increases performances of original ACO algorithms in almost all instances.

We also used box-plots that indicate the distribution of the ACO algorithms' errors on different TSP instance size ranges. These box-plots are shown in Figure 6. For all instances, the overall performance of ACO algorithms with best tour tables performance are better than original ones, but ACS+BT improvement on ACS algorithms seems very small. To see that clearly, correlation plots that illustrate relative performance between ACO algorithms with and without the proposed approach are shown in Figure 7. The coordinates of each point are the results obtained with the original ACO algorithm (on x-axis) and results obtained with ACO algorithms using the proposed method (on y-axis)

TABLE 2. Effects of best tours (BT) table and updating mechanism on original ACO algorithms for selected TSPLIB instances. Improved results with the best tours table mechanism on original ACO versions or optimum results are shown in boldface.

Instance	AS	AS+BT	ASe	ASe+BT	ACS	ACS+BT	RBAS	RBAS+BT	Optimum
Eil51	439.3	428.1	432	427	427	426.9	429.8	427.3	426
Berlin52	7569.8	7542.3	7542	7542	7542	7542	7542	7542	7542
Eil76	556.7	539.1	549.5	538.7	539.7	538.2	542.7	538	538
Rat99	1278.1	1228.8	1248.3	1218.9	1213.5	1211	1228.1	1215.8	1211
KroA100	22552.8	21417.1	22153.8	21434.2	21424.5	21394.7	21779.4	21340.1	21282
Lin105	14753.9	14565.9	14521.7	14464.2	14379	14379	14443.8	14439.7	14379
D198	17033	16908.7	16639.6	16540.8	16149.1	16292.2	16430.3	16297.4	15780
Tsp225	4399.2	4244.1	4210.1	4094.5	3959.6	3954	4215.3	4083.3	3916
Lin318	46844.9	45827	45908	44763.2	43790.8	42760.6	45890.4	44417	42029

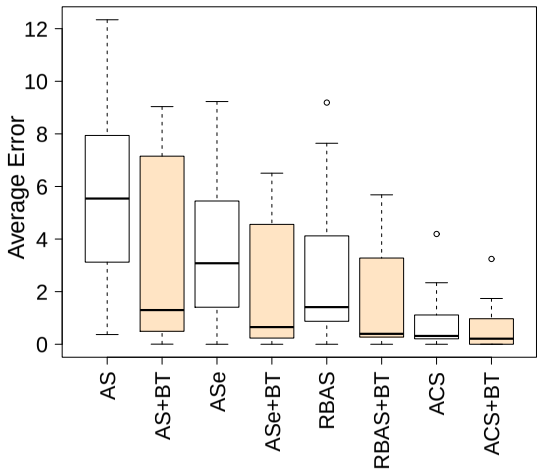

FIGURE 6. The distribution of the errors obtained on TSPLIB instances

TABLE 3. Pair-wise tests of ACO with BT algorithms with original ACO algorithms. The p-value (< 0.05) is highlighted in boldface.

	Wilcoxon p	Bonferroni-Dunn p	Holm p	Hochberg p	Unadjusted p
AS+BT vs AS	0.001953	0.0006215	0.0006215	0.0006215	0.0006215
ASe+BT vs ASe	0.007133	0.00138	0.00138	0.002761	0.00138
RBAS+BT vs RBAS	0.007133	0.007359	0.007359	0.01472	0.007359
ACS+BT vs ACS	0.1024	0.2044	0.2044	0.2044	0.2044

on a TSPLIB instance. Thus, results obtained by the best tours table better than original algorithms in a particular instance appear on the right portions of correlation plots.

To understand clearly the significant difference between an ACO algorithm with BT and without BT, we need to perform a statistical analysis [41]. We conducted a pair-wise Wilcoxon's test [41], which was adjusted to testing by a method of Holm and post-hoc analysis of Friedman's tests with Bonferroni-Dunn's [42], Holm's [43] and Hochberg's procedures [44]. Table 3 lists the pair-wise testing of ACO with BT algorithms with original ACO algorithms at %5 level. Table 3 indicates that almost all ACO algorithms with BT significantly outperform for all statistical tests except ACS algorithms.

Finally, we have plotted the relative improvements between the algorithms (Figure 8). For small-size instances ($n \leq 100$), the influence of the proposed approach on algorithm

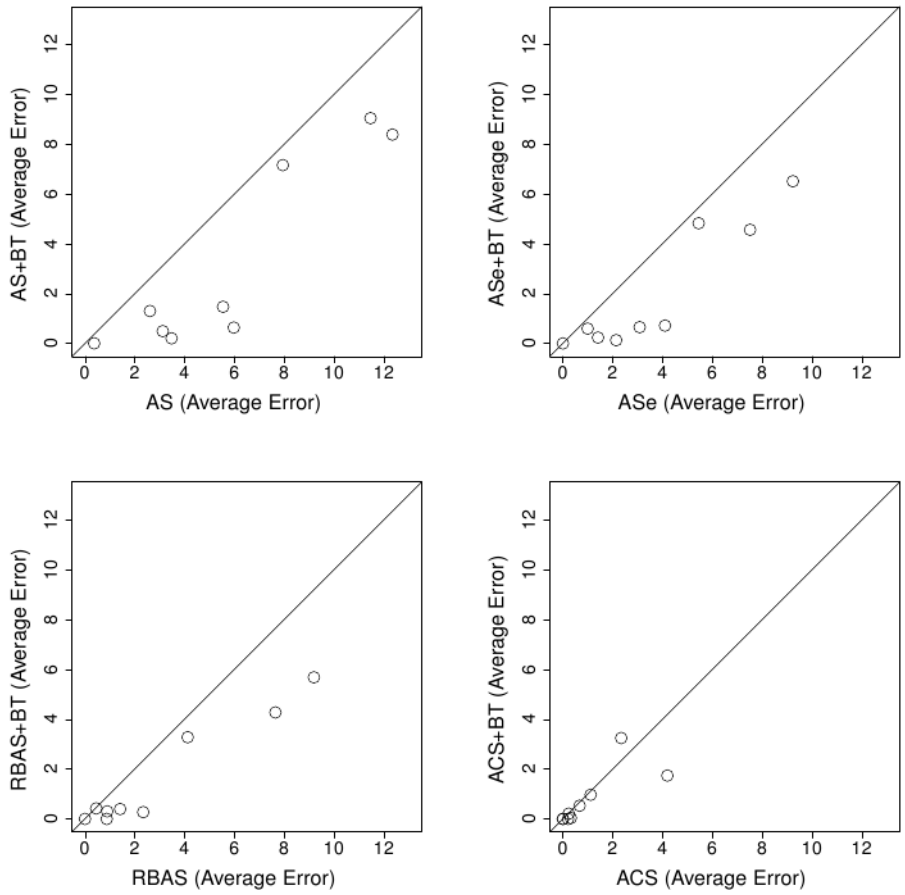

FIGURE 7. The correlation plots between ACO with BT and original ACO algorithms. Each point represents a TSPLIB instance.

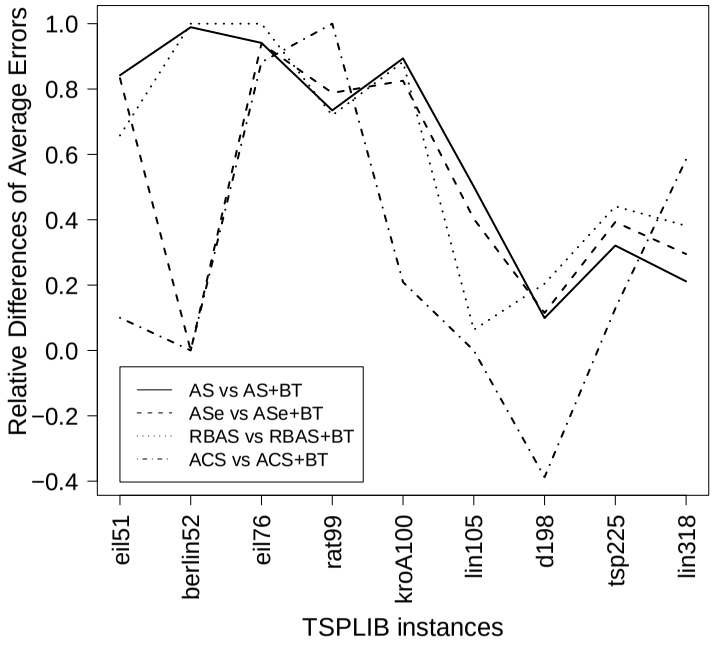

FIGURE 8. Relative differences between average errors obtained with the proposed approach and average errors obtained with original ACO algorithms

search behavior can be seen clearly; thus, it improves the algorithm performance significantly. But on large-size or hard instances ($n > 100$), the effect of BT on the ACO algorithm disappears gradually.

5. **Conclusions.** The main contribution of this paper is to propose the first method using extra graphs, which is called best tours table, for feeding pheromone trails for ACO algorithms. The mechanism developed for construction and updating of the best tours table and the powerful pheromone updating mechanism that can be applied to all ACO algorithms based on the best tours table are the second and third contributions. The space complexity of this graph is very low and $O(n^2)$, which equals the space complexity of graphs for trial information. We designed the algorithm so that additional time complexity is also very low. The best tours graph is updated only when a new global best solution is found and the pheromone table is reinforced by the best tours graph for only a small number of iterations. The best tours table gathering statistical information about global best tours encountered provides valuable edges or roads for ants. A bit more statistics brings new capabilities to problem-solving techniques.

Crossover is a powerful operator of Genetic Algorithms. We designed and developed an efficient crossover mechanism for ACO algorithms in this study. Significant improvements are obtained for TSP instances. But the effect of the proposed approach decreases on large-size TSP instances or ACS algorithms. In the future, we will attempt to improve the proposed approach robustness on all varieties of TSP instances and ACO variants. We will also adapt the crossover mechanism that we designed in this study for solving TSP to other combinatorial optimization problems.

REFERENCES

[1] E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Intelligence*, Oxford University Press, NewYork, NY, 1999.

[2] P.-W. Tsai, J.-S. Pan, B.-Y. Liao and S.-C. Chu, Enhanced artificial bee colony optimization, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5081-5095, 2009.

[3] J. A. Pacurib, G. M. M. Seno and J. P. T. Yusiong, Solving the Sudoku puzzle problem using improved artificial bee colony algorithm, *ICIC Express Letters*, vol.4, no.3(A), pp.733-738, 2010.

[4] L. M. Gambardella and M. Dorigo, Ant colony system hybridized with a new local search for the sequential ordering problem, *INFORMS Journal on Computing*, vol.12, no.3, pp.237-255, 2000.

[5] L. M. Gambardella, E. D. Taillard and G. Agazzi, MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows, *Proc. of New Ideas in Optimization*, pp.63-76, 1999.

[6] M. Reimann, K. Doerner and R. F. Hartl, D-ants: Savings based ants divide and conquer the vehicle routing problems, *Computers & Operations Research*, vol.31, no.4, pp.563-591, 2004.

[7] D. Costa and A. Hertz, Ants can colour graphs, *Journal of the Operations Research Society*, vol.48, pp.295-305, 1997.

[8] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS Journal on Computing*, vol.11, no.4, pp.358-369, 1999.

[9] K. Socha, J. Knowles and M. Sampels, A MAX-MIN ant system for the university timetabling problem, *Proc. of ANTS, LNCS*, Brussels, vol.2463, 2002.

[10] M. L. Besten, T. Stuetzle and M. Dorigo, Ant colony optimization for the total weighted tardiness problem, *Proc. of PPSN-VI, LNCS*, pp.611-620, 2000.

[11] C. Blum, Beam-ACO-hybridizing ant colony optimization with beam search: An application to open shop scheduling, *Computers & Operations Research*, vol.32, no.6, pp.1565-1591, 2005.

[12] D. Merkle and M. Middendorf, Ant colony optimization with global pheromone evaluation for scheduling a single machine, *Applied Intelligence*, vol.18, no.1, pp.105-111, 2003.

[13] D. Merkle, M. Middendorf and H. Schmeck, Ant colony optimization for resource-constrained project scheduling, *IEEE Transactions on Evolutionary Computing*, vol.6, no.4, pp.333-346, 2002.

[14] J. Dreo and P. Siarry, An ant colony algorithm aimed at dynamic continuous optimization, *Applied Mathematics and Computation*, vol.181, no.1, pp.457-467, 2006.

[15] P. S. Shelokar, P. Siarry, V. K. Jayaraman et al., Particle swarm and ant colony algorithms hybridized for improved continuous optimization, *Applied Mathematics and Computation*, vol.188, no.1, pp.129-142, 2007.
[16] K. Socha, ACO for continuous and mixed-variable optimization, *Proc. of ANTS 2004, LNCS*, Brussels, vol.3172, pp.25-36, 2004.
[17] K. Socha and C. Blum, An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training, *Neural Computing & Applications*, vol.16, no.3, pp.235-247, 2007.
[18] K. Socha and M. Dorigo, A colony optimization for continuous domains, *European Journal of Operational Research*, vol.185, no.3, pp.1155-1173, 2008.
[19] S. N. Kuan, H. L. Ong and K. M. Ng, Solving the feeder bus network design problem by genetic algorithms and ant colony optimization, *Advances in Engineering Software*, vol.37, no.6, pp.351-359, 2006.
[20] L. D. S. Coelho and V. C. Mariani, Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, *Expert Systems with Applications*, vol.34, no.3, pp.1905-1913, 2008.
[21] S. S. Kim, I.-H. Kim, V. Mani and H. J. Kim, Ant colony optimization for SONET ring loading problem, *International Journal of Innovative Computing, Information and Control*, vol.4, no.7, pp.1617-1626, 2008.
[22] J. Song and D. H. Kim, Dynamic subcarrier and bit allocation for throughput maximization in multiuser OFDM systems using ant colony optimal algorithm, *International Journal of Innovative Computing, Information and Control*, vol.6, no.10, pp.4705-4718, 2010.
[23] P. S. Shelokar, V. K. Jayaraman and B. D. Kulkarni, An ant colony classifier system: Application to some process engineering problems, *Computers & Chemical Engineering*, vol.28, no.9, pp.1577-1584, 2004.
[24] A. Ugur, Path planning on a cuboid using genetic algorithms, *Information Sciences*, vol.178, no.16, pp.3275-3287, 2008.
[25] I. Ellabib, H. Calamai and O. A. Basir, Exchange strategies for multiple ant colony system, *Information Sciences*, vol.177, no.5, pp.1248-1264, 2007.
[26] B. Fox, W. Xiang and H. P. Lee, Industrial applications of the ant colony optimization algorithm, *International Journal of Advanced Manufacturing Technology*, vol.31, pp.805-814, 2007.
[27] M. Dorigo and T. Stuetzle, *Ant Colony Optimization*, MIT Press, Cambridge, MA, 2004.
[28] C. G. Martinez, O. Cordon and F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, *European Journal of Operational Research*, vol.180, no.1, pp.116-148, 2007.
[29] M. Dorigo, M. Birattari and T. Stuetzle, Ant colony optimization: Artificial ants as a computational intelligence technique, *IEEE Computational Intelligence Magazine*, vol.1, no.4, pp.28-39, 2006.
[30] 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.
[31] M. Dorigo and L. M. Gambardella, Ant colonies for the traveling salesman problem, *Bio Systems*, vol.43, pp.73-81, 1997.
[32] H. Bullnheimer, R. F. Hartl and C. Strauss, A new ranked-based version of the ant system: A computational study, *Central European Journal for Operational Research*, vol.7, pp.25-38, 1999.
[33] O. Cordon, I. F. Viana and L. Moreno, New ACO model integrating evolutionary computation concepts: The best-worst ant system, *Proc. of ANTS*, Brussels, pp.22-29, 2000.
[34] T. Stuetzle and H. H. Hoos, MAX-MIN ant system, *Future Generation Computer Systems*, vol.16, no.8, pp.889-914, 2000.
[35] T. Stuetzle and M. Dorigo, ACO algorithms for the traveling salesman problem, in *Evolutionary Algorithms in Engineering and Computer Science*, K. Miettinen, M. Makela, P. Neittaanmaki and J. Periaux (eds.), Wiley, 1999.
[36] D. Dorigo, *Optimization, Learning and natural Algorithms*, Ph.D. Thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT, 1991 (in Italian).
[37] M. Dorigo, V. Maniezzo and A. Colorni, Positive feedback as a search strategy, *Technical Report 91-016*, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.
[38] A. Ugur and D. Aydin, An interactive simulation and analysis software for solving TSP using ant colony optimization algorithms, *Advances in Engineering Software*, vol.40, pp.341-349, 2009.
[39] P. Balaprakash, M. Birattari and T. Stuetzle, Improvement strategies for the F-race algorithm: Sampling design and iterative refinement, *Proc. of the 4th International Conference on Hybrid Metaheuristics, LNCS*, Berlin, Germany, vol.4771, pp.108-122, 2007.

[40] M. Birattari, Z. Yuan, P. Balaprakash and T. Stuetzle, F-race and iterated F-race: An overview, *Experimental Methods for the Analysis of Optimization Algorithms*, pp.311-336, 2010.
[41] S. Garcia, D. Molina, M. Lozano and F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization, *Journal of Heuristics*, vol.15, no.6, pp.617-644, 2009.
[42] J. H. Zar, *Biostatistical Analysis*, Prentice Hall, Englewood Cliffs, 1999.
[43] S. Holm, A simple sequantially rejective multiple test procedure, *Scandinavian Journal of Statistics*, vol.6, pp.65-70, 1979.
[44] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, *Biometrika*, vol.75, pp.800-803, 1988.