Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management
Istanbul, Turkey, July 3 – 6, 2012

Optimizing the Ant Colony Optimization Algorithm Using Neural Network for the Traveling Salesman Problem

Ayşe Hande Erol
Department of Industrial Engineering
Marmara University
Goztepe 34722, Turkey

Merve Er
Department of Industrial Engineering
Faculty of Engineering
Marmara University
Goztepe 34722, Turkey

Serol Bulkan
Department of Industrial Engineering
Faculty of Engineering
Marmara University
Goztepe 34722, Turkey

Abstract

Ant Colony Optimization (ACO) has been proved to be one of the most effective algorithms to solve a wide range of combinatorial optimization (or NP-hard) problems as the Travelling Salesman Problem (TSP). The first step of an ACO algorithm is setting the parameters that drive the algorithm. The basic parameters that are used in ACS algorithms are; number of ants, the relative importance (or weight) of pheromone, the relative importance of heuristics value, initial pheromone value, evaporation rate, and a parameter to control exploration or exploitation. Generally these parameters are set as discrete values, but past studies has shown that the behavior of the algorithm can be influenced by the modification of the parameters. In this paper, we propose a Neural Network (NN) to adapt two parameters of the ACO algorithm dynamically. The proposed hybrid algorithm is applied on the classical TSP, and tested on different TSP benchmark instances.

Keywords

Ant Colony optimization, evolutionary algorithms, neural network, parameter tuning in ACO, travelling salesman problem

1. Introduction

Problems in transportation and logistics; such as routing, scheduling, assignment etc. are very complex and hard combinatorial optimization problems. In recent years, advanced (soft) computing algorithms such as metaheuristics, Neural Networks (NN), and fuzzy logic are commonly used instead of traditional methods for solving these complex problems. Heuristic and metaheuristics algorithms can provide approximate solutions in a reasonable time for the practical applications of transportation problems.

One of the most common problems in areas such as logistics, distribution, and transportation industries / networks is the well-known Travelling Salesman Problem (TSP) (Applegate, et. al., 2006). The TSP is one of the most intensively investigated problems studied in both operations research, computer science, and transportation fields, and often comes up as a sub-problem in more complex combinatorial optimization problems. Given a finite set of

cities and their coordinates, TSP is the problem of finding a shortest closed tour that visits each city exactly once and turns back to the starting city. It has myriad of applications containing the movement of people, postal delivery, school bus routes, garbage collection etc. The solution of the TSP enables optimization of many other problems and transportation tasks, so it has significant value for transportation networks. The availability of extensive literature and the high applicability of TSP also make it ideal for a case study.

The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. Formally, the TSP can be stated as follows. The distances between *n* cities are stored in a distance matrix (D) with elements d_{ij} where *i, j= 1,…, n* and the diagonal elements d_{ii} are zero. A tour can be represented by a cyclic permutation Π of {*1, 2,…, n*} where Π(*i*) represents the city that follows city *i* on the tour. The TSP is then the optimization problem to find a permutation Π that minimizes the length of the tour denoted by (Hahsler and Hornik, 2007);

$$\sum_{i=1}^{n} d_{i\Pi(i)} \quad (1)$$

For this minimization task, the tour length of (n - 1)! vectors have to be compared. This results in a problem which is very hard to solve and in fact known to be NP-complete. According to Lenstra and Kan (1975), solving TSPs is an important part of applications in many areas including vehicle routing, computer wiring, machine sequencing and scheduling, frequency assignment in communication networks. Applications in statistical data analysis include ordering and clustering objects. For example, data analysis applications in psychology ranging from profile smoothing to finding an order in developmental data are presented by Hubert and Baker (Hubert and Baker, 1978).

TSP belongs to the NP-hard problem class and cannot be solved optimally in a polynomial time (Papadimitriou, 1997; Garey and Johnson, 1979). Therefore, the TSP has provided a remarkable source for testing solution algorithms for combinatorial optimization problems, and many heuristics have been proposed to find near-optimal solutions for it. The development of computational methods to solve the TSP is an active field of research, and Applegate et. al. (2006) proposed a comprehensive review about solving TSP. These methods can be classified into two broad categories, exact algorithms which are guaranteed to output optimal tours and heuristics which generate good quality tours within a reasonable execution time. The former category includes cutting plane algorithms proposed by Dantzig et. al. in 1954; Grőtschel and Padburg in 1985; Hong in 1972, branch and bound algorithms proposed by Balas in 1965; Held and Karp in 1970; Lin in 1965, branch and cut algorithms proposed by Hong in 1972; Crowder and Padberg in 1980; Grőtschel and Holland in 1991; Padberg and Rinaldi in 1991 among other techniques. The latter include several construction heuristics such as the nearest neighbor heuristic, the nearest, farthest, and cheapest insertion heuristics, Christofides' heuristic, as well as improvement heuristics such as local search proposed by Lin and Kernighan in 1973, tabu search proposed by Fiechter in 1990; Gendreau et. al. in 1994; Knox, in 1994; Potvin et. al. in 1996; Tsubakitani and Evans in 1998, simulated annealing proposed by Cerny in 1985, genetic algorithms proposed by Nguyen in 2004, and swarm algorithms proposed by Goldbarg et. al. in 2008; Wang et. al. in 2003 (Ghosh and Basu, 2011).

One of the most successful algorithms for the TSP is the Ant Colony Optimization (ACO) metaheuristic (Dorigo and Caro, 1999). ACO algorithms mimic the real ants' capabilities of finding the shortest path between the nest and a food source. Therefore the first ACO algorithm, named "Ant System (AS)", was originally applied on a path optimization problem; the TSP (Dorigo, et. al., 1991; Dorigo, 1992). Since then different variations and extensions of the basic AS algorithm have been proposed, and a lot of researchers have studied on the potential and improvements of the ACO algorithms (Dorigo and Caro, 1999).

One of the significant issues that affect the performance of the ACO algorithm is the parameter selection. The ACO algorithms start by initializing some parameter values that are used in the solution construction and pheromone management phases of the algorithm. In the literature, most of the studies use hand-tuned parameters for ACO algorithms. But the values of these parameters may change both the execution time and the quality of the solution. In this study, a systematic approach is proposed to determine the parameters of the ACO algorithm. The performance of the ACO is enhanced by using a Neural Network algorithm. The NN takes the results obtained by the ACO algorithm for the used parameters, and update them in order to send to the ACO as feedback information. In the next section, the ACO metaheuristic will be introduced briefly, and the working principle of the ACO algorithm will be

given. Then the importance of the parameter selection in ACO will be emphasized with the support of some literature review. The third section explains the methodology of the proposed hybrid algorithm and gives empirical results. Finally the last section evaluates the solutions, and gives concluding remarks and possible future work.

2. Ant Colony Optimization and Parameter Selection

ACO has emerged in the early 90s as a nature inspired metaheuristic framework to solve hard combinatorial optimization problems (Dorigo and Caro, 1999; Blum, 2005). This class of optimization algorithms studies the systems in which artificial ants that take inspiration from the real ant's behavior stand for multi-agent methods. In other words, ACO brings the indirect communication (by pheromone) and self organization properties of ant colonies from nature to optimization problems. Since it was first proposed by Dorigo et. al., a lot of researchers have studied the potential and the improvements of the ACO algorithms, and its adaptation to different types of problems. Therefore, there are many extensions and variations of the basic AS algorithm in the literature (see Dorigo and Stützle, 2004 and Mullen, et. al., 2009).

Ant Colony System (ACS) algorithm is one of the most successful (efficient) variants of the original AS algorithm (Dorigo and Gambardella, 1996). AS differs from the basic AS algorithm with three properties:

1. the decision rule,
2. addition of a local pheromone updating procedure, and,
3. the restriction of the global pheromone updating only to the best solution.

Dorigo and Gambardella (1996) applied the ACS algorithm to the TSP. Each ant *k* constructs a solution by iteratively moving from city *i* to city *j* according to the pseudo-random proportional rule (a probabilistic state transition rule) that is given as:

$$j = \begin{cases} \arg\max_{l \in N_i^k} \left\{ \tau_{il} [\eta_{il}]^\beta \right\} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases} \tag{2}$$

where *j* is the random proportional rule, τ_{il} and η_{il} are the pheromone level and heuristic value between city *i* and city *l*, *q* is a random variable uniformly distributed between [0, 1], q_0 is a parameter ($0 \le q_0 \le 1$) to control exploration and exploitation, β is a parameter which determines the relative importance of the pheromone versus the heuristic value, and *J* is the action obtained by using the AS rule. The transition rule used by the AS gives the probability for ant *k* to move from city *i* to city *j* at time *t*, $p_{ij}^k(t)$, and is calculated as follows:

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}{\sum_{k \in allowed_k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where allowed *k* represents the feasible nodes that ant *k* can move to from city *i*, and α and β represent adjustable parameters that control the influence of the pheromone versus the heuristic information. The heuristic value, η is problem specific information that guides the ants' search, and changes according to the problem type.

When an ant moves from node *i* to node *j*, a local pheromone updating is performed by the evaporation of a small amount of pheromone from path (*i*, *j*) according to the following formula:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \tag{4}$$

where τ_0 is the initial pheromone value, and ρ is the pheromone evaporation rate. While τ_0 value is set to a user-defined small constant in AS algorithm, in the ACS algorithm it is usually found with a Nearest Neighbor (NN)

heuristic (Solomon, 1987). Also after a cycle has been completed, a global updating is applied to edges belonging to the best solution. The pheromone trails on these paths are updated as follows:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{bs}(t,t+1) \tag{5}$$

and $\Delta\tau_{ij}^{bs}$ is calculated as follows:

$$\Delta\tau_{ij}^{bs} = \begin{cases} \dfrac{1}{c^{bs}} & if\ (i,j) \in T^{bs} \\ 0 & otherwise \end{cases} \tag{6}$$

where T^{bs} is the best solution and c^{bs} is its cost/length.

Most of the algorithms in the literature focus on the enhancement of the pheromone update mechanism of the basic AS algorithm (how to compute and how to update), parallelization strategies, or the improvement of the initial or the final solution by additional methods. However the existing literature about the importance and the effects of parameter selection in the performance of ACO algorithms is not reach. But past researches show that the performance of the algorithms also depends on the correct tuning of the used parameters (Dorigo, et. al., 1991 and Dorigo, et. al., 1996). While the pheromone level will have a greater influence in the ants' decisions when $\alpha>\beta$, the edge visibility will have a greater influence otherwise. Also, while a high value for α may cause stagnation, a high value for β leads to a greedy search (for search stagnation we refer to Dorigo, et. al., 1996).

Some researchers have studied driving the optimal combinations of the adjustable parameters of ACO to increase the success of the algorithm (Dorigo, et. al., 1996; Dorigo, et. al., 1991; Dorigo, et. al., 1996; White, et. al., 1998; Pilat and White, 2002; Zaitar and Hiyassat, 2005). Dorigo et. al. (1991) experimentally determined the best values of α, β, ρ, and Q (quantity of trail laid by ants) parameters for three AS algorithms, and compared the three models by running them ten times using the best parameters set (the influence of Q parameter is found negligible). The ant-cycle algorithm has obtained better results than the other two algorithms, and its behavior is investigated for different combinations of parameters α and β. The results that are obtained by running the algorithm ten times and for 2500 cycles for each set of parameters are summarized in Figure 1. They also investigated the stagnation behavior that is the situation in which all the ants find the same solution.

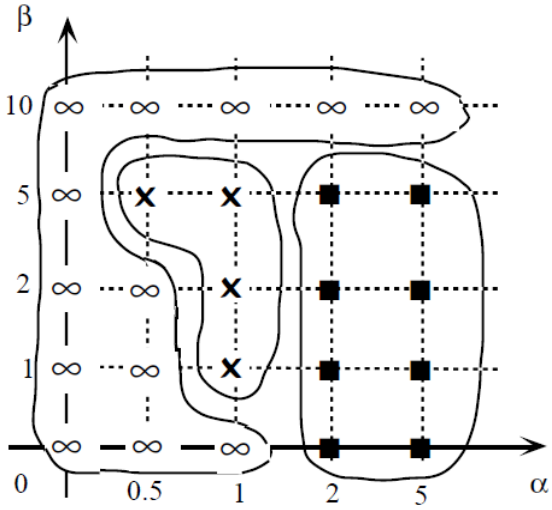

Figure 1: Ant-cycle Behavior for Different Combinations of α-β Parameters (Dorigo, et. al., 1991)
■ - the algorithm does not find very good solutions and enters stagnation behavior
∞ - the algorithm does not find very good solutions without entering stagnation behavior
x - the algorithm finds very good solutions without entering stagnation behavior

Pilat and White (2002) use a GA to evolve experimental variable values used in ACS-TSP. The suggested values for β, ρ, and q_0 are 6, 0.2, and 0.7 respectively. Zaitar and Hiyassat (2005) also use the standard GA in optimizing the ACO parameters, and apply it on the classical TSP. First a random population of chromosomes is generated, and then the values for both α and β parameters are calculated. Then the ACO algorithm is implemented for each

chromosome, and the shortest path is selected. Both the ACO and GA are implemented iteratively. The outputs of the ACO-GA for a certain set of cities are collected in order to find the optimal parameter values. The average values of α, β, and ρ are 1.5729, 1.3896, and 0.52 respectively. Zhangqi et. al. (2011) apply GA to optimize and configure parameters of the basic ant colony algorithm. Simulation results show that the improved ant colony algorithm is superior to the basic ant colony algorithm in terms of stability and the quality of the solution for mobile robot path planning problem.

3. Optimizing the Ant Colony Optimization Algorithm using Neural Network

The parameters of the ACO have a critical impact on the performance of the algorithm. A lot of testing is needed to determine the best combination of the ACO parameters. This paper introduces NN to optimize the parameters of the ACO algorithm. The NN takes the results obtained by the ACO algorithm for the used parameters (α and β), and update them in order to send to the ACO as feedback information. NN is used for choosing best parameters to minimize the length of the tour for TSP instances, where as the ACO serves as the evaluation algorithm. Hybridizing ant algorithm with NN is shown in Figure 2.

First the ACO algorithm is run with different α and β parameters for 50 times. Then the generated solutions with the given parameters are given to a NN as an input data set and the NN algorithm predicts the best parameter values. The pseudo code of the proposed algorithm is shown in Figure 3.

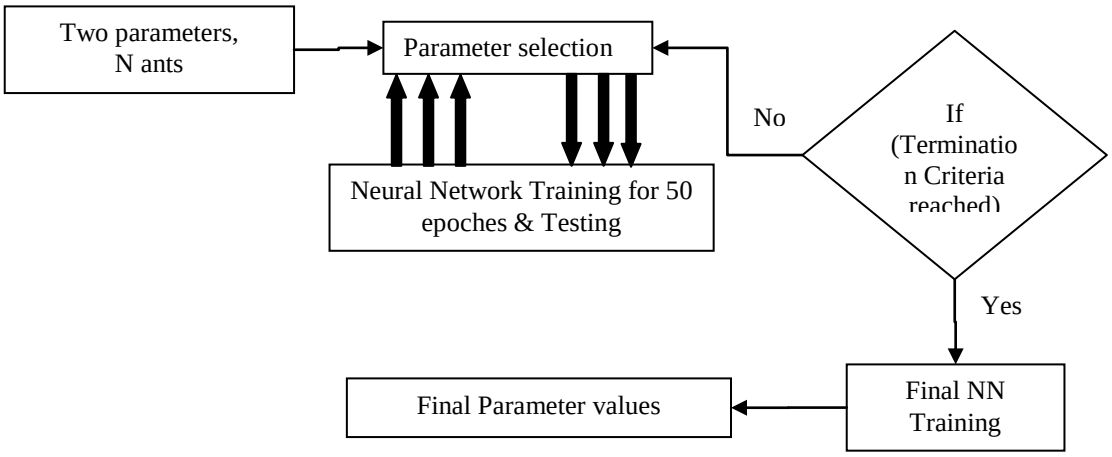

Figure 2: Hybridizing ant algorithm with NN (adapted from Sivagaminathan and Ramakrishnan, 2007)

```
Begin  ACO-NN ()
    do i=1 to sample size
    Randomize parameter alpha(α) and beta(β)
    while termination condition not met
        do each ant
            Generate solution
            Local pheromone update
        else
        Global pheromone update
    end if
    else
    Apply NN with the generated samples
    Run ACO with the parameters predicted by the NN
end if
Best is the ideal solution
end
```

Figure 3: The pseudo code of the proposed ACO-NN algorithm

The proposed ACO-NN hybrid algorithm is applied on a set of TSP benchmark instances. Input files are the problem sets taken from the TSP Library (TSPLIB). Symmetric TSP instances are used in this study. The algorithm is developed on Eclipse Classic 3.5.2 and JCreator LE 4.50 by using Java programming language. The program ran on a computer with Intel ® Core(TM) 2 Duo 1.83Ghz CPU and 1 GB-RAM.

One type of TSP instances is chosen and trained by NN algorithm. The best α and β combination which is given by NN algorithm is 0.886 and 4, respectively. According to these parameters other symmetric problem instances are solved. As a summary of results; the minimum and maximum values of performance measure for TSP instances in TSPLIB is shown in Table 1.

Table 1: Summary of results using ACO-NN algorithm for TSP instances

Problem name	Performance over Best Known Solution (%)		Best solution of application	Best known solutions	Average CPU times (seconds)
	Min	Max			
ulysses16.tsp	0	0.034	**6859**	6859*	10.953
ulysses22.tsp	0	0.030	**7013**	7013*	15.578
fri26.tsp	0	0.082	**937**	937*	17.313
bays29.tsp	0	0.053	**2020**	2020*	14.765
bayg29.tsp	0	0.088	**1610**	1610*	34.543
att48.tsp	0	0.034	**10648**	10648*	57.340
hk48.tsp	0	0.045	**11461**	11461*	32.231
berlin52.tsp	0	0.065	**7542**	7542*	54.346
burma14.tsp	0	0.032	**3323**	3323*	57.453
eil51.tsp	0	0.053	**426**	426*	24.357
eil76.tsp	0	0.045	**538**	538*	43.456
eil101.tsp	0	0.089	**629**	629*	59.234
st70.tsp	0.019	0.035	**688**	675*	35.621
brg180.tsp	0.012	0.088	1975	1950*	57.459
ch130.tsp	0.033	0.088	6310	6110*	69.594
ch150.tsp	0.039	0.088	6789	6528*	69.976

** Optimum solutions*

For symmetric problem instances, the proposed ACO-NN algorithm finds either optimum solutions or above the best known solutions by 5% on the average.

4. Conclusions and Future Research

Determining the best combination of the ACO parameters (α and β) is a critical impact on the performance of the algorithm. For increasing the performance of the algorithm, these parameters must be estimated and used as good as possible by NN. In this study, NN is proposed for the adjustment of parameters of the ACO algorithm. The use of NN for parameter selection has improved the solution quality of ACO algorithm. This improvement justifies the development of the ACO-NN algorithm. Same algorithm will be applied for asymmetric problem instances in a future study. Also, all sets of TSP instances will be used in training NN algorithm instead of one type of TSP instance.

References

1. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J., The Travelling Salesman Problem: A Computational Study, *Princeton Series in Applied Mathematics*, Princeton University Press, Princeton, NJ, 2006.
2. Hubert, L.J., Baker, F.B., Applications of Combinatorial Programming to Data Analysis: The Traveling Salesman and Related Problems, Psychometrika, 43(1), 81-91, 1978.
3. Botee, H.F., Bonabeau, E., Evolving Ant Colony Optimization, *Adv. Complex Systems*, 1, 149-159, 1998.
4. Blum, C., Ant Colony Optimization: Introduction and Recent Trends, *Physics of Life Reviews*, 2(4), 353-373, 2005.
5. Dorigo, M., Caro, D.C., The Ant Colony Optimization Meta-heuristic, New Ideas in Optimization, Corne et al. editors, McGraw Hill, London, UK, 11- 32, 1999.
6. Dorigo, M., Maniezzo, V., Colorni, A., Positive feedback as a search strategy, *Technical Report 91-016*, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.
7. Dorigo, M., Optimizing Learning and Natural Algorithms, *PhD thesis*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1992.
8. Dorigo, M., Stützle, T., Ant Colony Optimization, *MIT Press*, Cambridge, MA, USA, 2004.
9. Dorigo, M., Gambardella, L.M., Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, *Technical Report 96-5*, IRIDIA, Universit´e Libre de Bruxelles, 1996.
10. Dorigo, M., Maniezzo, V., Colorni, A., The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems*, Man and Cybernetics, B26 (1), 29-41, 1996.
11. Dorigo, M., Maniezzo, V., Colorni, A., Ant System: An Autocatalytic Optimizing Process, *Technical Report 91-016 Revised*, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.
12. Ghosh, D., Basu, S., Diversified Local Search for the Traveling Salesman Problem, *Working Paper Series*, Indian Institute of Management Ahmedabad, India, W.P. No. 2011-01-03, 2011.
13. Hahsler, M., Hornik, K., TSP-Infrastructure for the Traveling Salesperson Problem, *Journal of Statistical Software*, 23(2), 1-21, ISSN 1548-7660, 2007.
14. Hubert, L.J., Baker, F.B., Applications of Combinatorial Programming to Data Analysis: The Traveling Salesman and Related Problems, *Psychometrika*, 43(1), 81-91, 1978.
15. Lenstra, J., Kan., A.R., Some simple applications of the travelling salesman problem, *Operational Research Quarterly*, 26(4), 717-733, 1975.
16. Mullen, R.J., Monekosso, D., Barman, S., Remagnino, P., A Review of Ant Algorithms, *Expert Systems with Applications*, vol. 36, 9608-9617, 2009.
17. Pilat, M.L., White, T., Using Genetic Algorithms to Optimize ACS-TSP, *Proceedings of the Third International Workshop on Ant Algorithms*, ANTS-2002, Springer-Verlag, London, UK, 282-287, 2002.
18. Sivagaminathan, R.K., Ramakrishnan, S., A hybrid approach for feature subset selection using neural networks and ant colony optimization, Expert Systems with Applications 33, 49–60, 2007.
19. Solomon, M.M., Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, Operations Research, vol. 35, 254-265, 1987.
20. TSPLIB, TSP Library, Accessed from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ in 02.10.2011.
21. White, T., Pagurek, B., Oppacher, F., ASGA: Improving the Ant System by Integration with Genetic Algorithms, *Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 610-617, 1998.
22. Zaitar, R.A., Hiyassat, H., Optimizing the Ant Colony Optimization Using Standard Genetic Algorithm, *Proceedings of the 23rd International Multi-Conference Artificial Intelligence and Applications*, Innsbruck, Austria, 2005.
23. Zhangqi, W., Xiaoguang, Z., Qingyao, H., Mobile Robot Path Planning based on Parameter Optimization Ant Colony Algorithm, *Procedia Engineering* 15, 2738 – 2741, 2011.