A self-tuning Firefly algorithm to tune the parameters of Ant Colony System (ACSFA)

M. K. A. Ariyarathne¹, T. G. I. Fernando² and S. Weerakoon³

¹Department of Computer Science, Faculty of Computing, General Sir John Kotelawala Defence University, Sri Lanka.
²Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura, Sri Lanka.
³Department of Mathematics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Sri Lanka.

Abstract

Ant colony system (ACS) is a promising approach which has been widely used in problems such as Travelling Salesman Problems (TSP), Job shop scheduling problems (JSP) and Quadratic Assignment problems (QAP). In its original implementation, parameters of the algorithm were selected by trial and error approach. Over the last few years, novel approaches have been proposed on adapting the parameters of ACS in improving its performance. The aim of this paper is to use a framework introduced for self-tuning optimization algorithms combined with the firefly algorithm (FA) to tune the parameters of the ACS solving symmetric TSP problems. The FA optimizes the problem specific parameters of ACS while the parameters of the FA are tuned by the selected framework itself. With this approach, the user neither has to work with the parameters of ACS nor the parameters of FA. Using common symmetric TSP problems we demonstrate that the framework fits well for the ACS. A detailed statistical analysis further verifies the goodness of the new ACS over the existing ACS and also of the other techniques used to tune the parameters of ACS.

1 Introduction

The collective behavior of natural insects - ants, bees, fireflies and termites mimic the problem solving capabilities of the swarms [10, 13, 4]. These capabilities were adopted in various heuristics and meta-heuristics to solve difficult optimization problems. Each meta-heuristic has a real world inspiration of optimization. As such, the main inspiration for the ant colony system algorithm is the natural food finding strategy of ants. Ants are capable of finding the shortest path from the food source to their nests. A chemical inside them call pheromone is the reason for this optimized behavior. Following this real world strategy, the ant colony system algorithm was developed by Marco Dorigo et al.[6] to suit for path optimization problems. Initially the algorithm was developed to solve the TSP. This original implementation supports the hypothesis that the ant colony algorithm is successful in finding the shortest path for the TSP.

Similar to any meta-heuristic, the ant colony system also has algorithm specific parameters. The initial implementation used the trial and error method to find the best collection of parameters. The values of the parameters depend on the problem at hand and hence at each instance, the most suitable parameter set for the problem should be evaluated. The
task of parameter tuning again is an optimization where the optimal parameter set will give the optimum performance.

Though the initial study relied on trial and error, Dorigo had stated some important features of parameters such as pheromone behavior, the number of ants and how they affect the performance of the algorithm [6]. The original paper discusses the ranges for different parameters so that an idea about the distribution of each parameter is given. The results were in an encouraging position ascertaining that the algorithm is successful in solving the TSP. Instances from TSPLIB and randomly generated TSPs’ were used to evaluate the algorithm.

Since the original ACS works well with pre-tested parameter values, finding approaches to set better parameters without trial and error can improve the performance of the algorithm. The best way is to consider setting of parameters as another optimization problem. The same matter of tuning parameters of an ACS is considered by many other researchers. Yet none was successful in maintaining fully parameterless environment. An Adaptive Parameter Control Strategy for Ant Colony Systems by Zhi-Feng Hao et al. is a study carried out to enhance the performance of the ant system solving the TSP [8]. Although it has been mentioned as the ant colony system, they have used the ant system for the study. The particle swarm optimization (PSO) has been used to optimize the parameters of the ant system (PSOACS). The parameters of the ant system were considered as a whole where one particle represents approximation for the set of parameters. The number of particles is the same as the number of ants, and once ants complete a tour the PSO will update the parameters. The conclusions are based on the performance of the new algorithm and the existing ACS. The results support the conclusions, but the following matters were not addressed. Although the study focused on a parameter tuning technique, the effects of the method towards the parameters were not mentioned. Basically the study was focused on improving the TSP results of the existing approach. For both PSOACS and original ACS, the results obtained conflict with the optimal results given in the TSPLIB. The reason may be the differences in the implementations. Also in PSOACS, the equations used are not clear enough to get an idea about the new algorithm.

In Evolving Ant Colony Optimization, another research by Hozefa M. Botee and Eric Bonabeauy, they have made the use of genetic algorithm (GA) to evolve the best set of parameters [3]. Here also, one parameter set represents an individual of the genetic algorithm. However the implementation was tested only over two TSP instances. The results were encouraging, but the parameters of the ACS depend on the selection of the parameters of GA such as crossover and mutation probability.

Apart from meta-heuristics, machine learning techniques have also been used to tune parameters of ACS. For example Ayse Hande Erol et al. in their research, have used artificial neural networks (ANN) to find the best parameter set for ACS solving a given TSP [7]. The research focuses on only two parameters, $\alpha$: pheromone decay parameter and $\beta$: the relative importance of the pheromone vs. distance. Initially the ACS-ANN hybrid algorithm runs with different $\alpha$ and $\beta$ values for 50 times. These parameters work as the inputs to the ANN. Then ANN predicts the best parameter values for a given TSP instance. The hybrid algorithm was tested with several TSP instances from the TSPLIB [21]. The results support the hypothesis that the hybrid algorithm performs well in finding better parameter values. However the study focused on tuning only two parameters. The information about the ANN
such as the training methods, weights and their effects are not mentioned in the research.

As such there are other researches which have been conducted to find better parameter sets for the ACS in solving TSP \cite{9, 20}. But as a whole, all these methods rely on another algorithm or technique, where it again contains its own parameters. Therefore the performance of the ACS again depends on the parameters of the used algorithm. To overcome this issue we have designed an algorithm with the help of the firefly algorithm and the self-tuning framework for optimization algorithms proposed by Xin-She Yang, to optimize the parameters of the ACS algorithm solving symmetric TSP instances. Xin-She Yang et al., in the implementation of the self-tuning framework have used the firefly algorithm to apply the framework to tune FA’s parameters \cite{19}. In their framework, the problem solved by the optimization algorithm and the parameter set of the algorithm both were considered as a single problem. The framework was initially tested for the firefly algorithm and proved its capability of tuning parameters of itself (FA). In a research done by M.K.A. Ariyaratne et al., use this self tuning framework combined with the firefly algorithm to solve nonlinear equations \cite{2}. They have solved univariate nonlinear equations having complex roots with the help of a modified firefly algorithm. To find the best parameters values, the self tuning framework was implemented on the firefly algorithm. In their research, a firefly carries an approximation for a root as well as approximations to the parameter values. Finally as the output, they receive best approximations for the roots in a given range as well as best approximations for the parameter values. The research again confirms the powerfulness of the self tuning framework in optimizing parameters.

The significance of this research lies in the potential of the developed ant colony optimization algorithm for the TSP with the self tuning framework to optimize the parameters. Despite the recent advancements in the field of route optimization and parameter optimization, the following issues have not been addressed by other researchers (See Table ?? for details of previously applied approaches) where our research has accomplished.

- None of the existing systems are capable of providing virtually parameter-free environments for the ant colony systems to solve TSPs.
- In most of the researches, parameter optimization is done using another meta-heuristic algorithm whose parameters should be manually selected.
- None of the approaches have used a designed framework for parameter optimization.
| Author                                      | Year | Parameter Optimization method | # of TSP problems tested | Advantages                                      | Limitations                                      |
|---------------------------------------------|------|-------------------------------|--------------------------|-------------------------------------------------|--------------------------------------------------|
| Hozefa M. Botee and Eric Bonabeauy          | 1998 | GA                            | 2                        | GA optimizes the parameters of ACS              | Only some parameters of ACS were optimized. The parameters of GA should be manually updated. |
| Raed Abu Zaitar and Hussein Hiyassat        | 2005 | GA                            | 8                        | GA optimizes the parameters of ACS              | Only two parameters of ACS (α and β) were updated in the first phase. |
| D. GÃşmez-Cabrero and D. N. Ranasinghe      | 2005 | PSO                           | 24                       | PSO optimizes the parameters of ACS             | The parameters of PSO should be manually updated. High computational overhead. |
| Zhi-Feng Hao et al.                         | 2006 | PSO                           | 10                       | PSO optimizes the parameters of ACS             | The parameters of PSO should be manually updated. |
| Ayse Hande Erol et al.                      | 2012 | Artificial Neural Networks (ANN) | 16                       | ANN optimizes the parameters of ACO             | Only some parameters of ACO were optimized.      |

Table 1: Previous work on parameter optimization of Ant Systems

GA-Genetic Algorithms, PSO-Particle Swarm Optimization, ACS-Ant Colony Systems, ACO-Ant Colony Optimization

In this paper, Yang’s self-tuning framework is studied, and a parameter selection strategy based on the firefly algorithm is developed. To deliver a better idea about the present work, the remainder of this paper is structured as follows. Section 2 provides some preliminaries related to the Ant colony system and the firefly algorithm. In section 3, we briefly describe the self-tuning framework for the firefly algorithm and how we adopt it to tune the parameters of the ACS when solving symmetric TSPs. There, we also present the hybrid ACS-FA. Section 4 points out the TSP examples, the parameters used and the results obtained from the new approach. To emphasize the goodness of the new algorithm, we compare the results with the original ACS and with some other parameter tuning approaches. Finally, we draw conclusions briefly in Section 5.


2 Preliminaries

2.1 Traveling Salesman Problem (TSP)

The Traveling Salesman Problem is one of the most intensively studied problems in computational mathematics which is simple to state but very difficult to solve [1]. The problem is NP hard, making it not computable in polynomial time [15]. The problem is about finding the shortest possible tour through a set of \( n \) cities/nodes so that each city/node is visited exactly once. A weighted graph \( G(N, E) \) can represent a TSP, where \( N \) represents the cities and \( E \) represents the set of edges connecting cities. There is a specific distance \( d \) for each \((i, j) \in E\). If \( d(i, j) = d(j, i) \), it is known as a symmetric TSP where in an asymmetric TSP, \( d(i, j) \neq d(j, i) \) can occur. In our study we consider only the symmetric situation.

2.2 Ant Colony Systems (ACS)

Ants, the popular social insects normally live in colonies represent a highly structured distributed system. There are many ant species, some of which are blind. All ant species are known to be deaf [11]. Despite of these incapabilities, ants are inbred with a strong indirect communication using a chemical produced within them. While foraging, ants lay the chemical; pheromone on the ground and follow the pheromone placed by other ants. The pheromones tend to decay over time and hence the ants in the colony will choose the path with high pheromone density at the moment. This pheromone communication allows the ants to find the shortest path from the food source to their nest. The optimized behaviors of real ants are based on implementing artificial ant colonies.

The ant colony systems is an example of an ant colony optimization method from the field of swarm intelligence, meta-heuristics and computational intelligence. Around 1990’s Marko Dorigo introduced the idea of the ant system and later Dorigo and Gambardella introduce the ant colony system. In the original implementation, ACS was applied to solve the TSP [6]. Initially, ants in the artificial colony are positioned on random nodes/cities. They travel from one node to another keeping the travel history in a data structure. The likelihood of selecting a node is based on the pheromone density of the cities laid by other ants, which in the algorithm known as the state transition rule. Once visiting a city, an ant lays an amount of pheromone using the local pheromone updating. Upon completing the tours by all ants, the cities belong to the globally best path again get updated with pheromones using global pheromone updating rule.

2.2.1 State Transition Rule, Local and Global updating

State transition rule is responsible for an ant to find its next visiting city. Assume the ant is in the node \( r \). It’s next city \( s \) is determined by the equation [1]

\[
S = \left\{ \begin{array}{ll}
\arg \max_{u \in J_k(r)} \{ [\tau(r, u)^\theta] \cdot \eta(r, u)^\beta \} & q \leq q_0 \\
\text{Otherwise} & 
\end{array} \right.
\]

\[ (1) \]

where \( \tau(r, u) \) is the pheromone density of an edge \((r, u)\), \( \eta(r, u) \) is \([1/distance(r, u)]\) for TSP. \( J_k(r) \) is the set of cities that remain to be visited by ant \( k \) positioned on city \( r \). The relative
importance of the pheromone trail and the heuristic information are represented by the parameters \( \theta \) and \( \beta \) \((\theta, \beta \geq 0)\). \( q \) is a random number uniformly distributed in \([0, 1]\), \( q_0 \) is a parameter \((0 \leq q_0 \leq 1)\), and \( S \) is a random variable from the probability distribution given by the equation (2).

\[
P_k(r,s) = \begin{cases} 
\left[ \frac{\tau (r,u)^\theta \cdot \eta (r,u)^\beta}{\sum_{u \in J_k(r)} [\tau (r,u)^\theta \cdot \eta (r,u)^\beta]} \right] & \text{if } s \in J_k(r) \\
0 & \text{Otherwise}
\end{cases}
\] (2)

ACS local and global updating happens according to the equation (3) and equation (4) respectively.

\[
\tau (r,s) \leftarrow (1 - \rho).\tau (r,s) + \rho.\Delta \tau (r,s)
\] (3)

where \(0 < \rho < 1\) is a parameter.

\[
\tau (r,s) \leftarrow (1 - \alpha).\tau (r,s) + \alpha.\Delta \tau (r,s)
\] (4)

where

\[
\Delta \tau (r,s) = \begin{cases} 
(L_{gb})^{-1} & \text{if } (r,s) \in \text{global best tour} \\
0 & \text{Otherwise}
\end{cases}
\] (5)

0 < \alpha < 1 is the pheromone decay parameter and \( L_{gb} \) is the length of the globally best tour. In the original implementation, Dorigo et al. have given the set of parameter values obtained from the trial and error approach to suit the selected TSP instances.

The ACS grasped the attention of the world of optimization and hence many researches have been carried out to improve the algorithm as well as to check its ability over solving other optimization problems.

### 2.3 Firefly Algorithm (FA)

Firefly is a winged beetle commonly known as the lightning bug due to the charming light it emits. The light is used to attract mates or preys. Biological studies reveal many factors about fireflies’ life style that are interesting [14]. Focusing on their flashing behavior, the firefly algorithm was developed by Xin-She-Yang in 2009 [18]. The algorithm basically assumes the following.

- Fireflies’ attraction to each other is gender independent.
- Attractiveness is proportional to the brightness of the fireflies, for any two fireflies, the less brighter one is attracted by (and thus moves toward) the brighter one; however, the brightness can decrease as the distance increases; If there is no brighter one than a particular firefly, it moves randomly.
- The brightness of a firefly is determined by the value of the problem specific objective function.
As many meta-heuristics, the initial population for the particular problem is generated randomly. In FA also, the parameter set should be specified properly. After these initial steps, the fireflies in the population start moving towards brighter fireflies according to the following equation.

$$x_i = x_i + \beta(x_j - x_i) + \alpha(rand - 0.5)$$  \hspace{1cm} (6)

where

$$\beta = \beta_0 e^{-\gamma r^2}$$  \hspace{1cm} (7)

$\beta_0$ is the attraction at $r = 0$. The three terms in equation (6) represent the contribution from the current firefly, attraction between two fireflies and a randomization term respectively. The equation supports both exploitation and exploration. $\alpha$ plays an important role in the randomization process, which is from Uniform or Gaussian distribution. To control the randomness, Yang has used $\delta$, the randomness reduction factor which reduces $\alpha$ according to the equation (8).

$$\alpha = \alpha_0 \cdot \delta \quad \text{where} \quad \delta \in [0, 1]$$  \hspace{1cm} (8)

FA, as a newcomer in the world of meta-heuristics marked its remarkable capability of handling optimization problems. Yang et al. in 2013 introduced a framework for self-tuning algorithms and it was implemented with the firefly algorithm successfully [19]. The framework allows a meta-heuristic algorithm to solve a problem while optimizing its own algorithm specific parameters.

### 3 Self-tuning firefly algorithm optimizing Ant colony system’s parameters (ACSFA)

In this research, the main aim is to tune the parameters of the ant colony system algorithm solving symmetric TSP problems while obtaining the shortest path for the selected TSP. FA as an outstanding performer, has used here in tuning the parameters of the ACS. Another reason is that the self-tuning framework can be easily implemented with firefly algorithm rather than directly on ACS, since the ACS solves a discrete problem (TSP) and the parameters are continuous in nature.

Regarding the ACS, $\beta, \theta, \rho$ and $q_0$ parameters have to be tuned. The FA also has several parameters such as $\alpha, \gamma, \beta$ and the randomness reduction factor $\delta$. The main aim of the self-tuning concept is to find the best parameter settings that minimize the computational cost. When applying the self-tuning framework to the firefly algorithm solving a given optimization problem, both the problem domain and the parameter domain are considered as a single domain in solving the problem. The objective could be the objective of the problem.
In our case the FA is used to tune the parameters of ACS and the ACS solves a given TSP. The objective of the algorithms is to find the optimal solution of a given TSP instance. The problem of this study contains both parameters of ACS and FA. The pseudo code of the proposed ACSFA algorithm is presented in algorithm 1.

After initializing parameters and parameter ranges, inverse of a tour distance is set as the objective of the TSP. The problem is set as a minimization problem. The algorithm will build tours until a predetermined end condition is completed. As in the original ACS, ants are positioned on random starting nodes where each ant completes a tour using the state transition rule stated in equation 1. While completing the tour ants will lay pheromones on the visited cities according to the equation 3. Upon completing tours by all ants, a globally best tour will be identified and the cities belong to the globally best tour will be awarded with extra pheromone values according to the equation 4. Apart from building tours, each ant also carries approximations of the parameters of both ACS and FA.

After completing a tour, all ants will work as fireflies. They now represent approximations of the parameters of FA and ACS. The parameters of ACS and FA will get updated using the self tuning firefly algorithm. The fireflies will move in the direction of better parameters/fireflies according to the equation 6. At the end of each tour, the best parameter set will be detected.

Algorithm 1 : Pseudo code of the ACSFA

1: Begin;
2: Initialize parameter values and ranges of parameter values to be tuned
3: Assign approximations of the parameters to be tuned to each ant (Firefly).
4: Define the objective function \((I - \text{Inverse of the distance})\)
5: while End condition do
6: Begin Tour
7: Position ants on starting nodes
8: Build the tour while local pheromone update
9: End Tour
10: Do global pheromone update
11: for \(i = 1 : n\) (all \(n\) ants) do
12: for \(j = 2 : n\) (\(n\) ants) do
13: if \(I_j < I_i\) then
14: Move firefly \(i\) towards firefly \(j\) by using equation \(6\);
15: end if
16: Attractiveness varies with distance \(r\) via \(e^{-\gamma r^2}\) using equation \(7\);
17: Evaluate new solutions and update parameter values;
18: end for
19: end for
20: Rank the fireflies and find the current best (best parameter values);
21: end while
22: Post process results and visualization;
23: End
4 Experimentation

The goal of this experiment is to analyze the performance of the ACSFA algorithm solving symmetric TSPs’ while selecting the most suitable parameter values for ACS and FA. We aim to formulate the new algorithm with minimum number of user input parameters where all other parameters of the two algorithms are to be tuned while optimizing the TSP instances.

4.1 Parameter values and problem instances

The new algorithm is implemented to solve symmetric Traveling Salesman problems. 12 symmetric TSPs are selected for testing [21]. The parameters of ACS include: $\beta, \alpha, \rho, q_0, m, \tau_0$ and $\tau$. $\tau_0$ is based on the nearest neighbor heuristic. Pheromone density $\tau$ depends on $\alpha$ and $\rho$ parameters. Since $\alpha$ works only with the globally best ant, we initialize $\alpha$ to be 0.1. Here, we consider $\beta$, $\rho$ and $q_0$ to be tuned by the self-tuning firefly algorithm. The range for these parameters are $\beta \in [0, 8], \rho \in [0.5, 1]$ and $q_0 \in [0.5, 1]$ which are large enough to select the best parameter values for many symmetric TSP instances. Apart from that, we tune the parameters of the FA. The parameters of firefly algorithm include $\alpha, \beta, \gamma, \delta$ and number of fireflies. In equation [?] the brightness $\beta$ depends on three main factors; $\beta_0$ which we initiate to be 1, $\gamma$; the light absorption coefficient and $r$; the Cartesian distance between two fireflies which should be calculated during iterations. in equation [?] the value of $\alpha$ get reduced with $\alpha_0$ and $\delta$. $\alpha_0$ is initialized to be 2.3. Therefore the value of $\alpha$ depends on $\delta$. Hence the only factors to be tuned in FA are $\gamma$ and $\delta$. For convenience, we will use $FF_\alpha$, $FF_\beta_0$, $FF_\gamma$ and $FF_\delta$ to indicate the parameters of FA. These parameters varies in the ranges $FF_\gamma \in [0, 10]$ and $FF_\delta \in [0.8, 1]$. For further clarification a detailed list of the parameters used for the algorithms are presented in Table (2).

| ACSFA       | Value/Range | ACS          | Parameter | Value/Range | PSOACS       | Value/Range |
|-------------|-------------|--------------|-----------|-------------|--------------|-------------|
| $\alpha$    | 0.1         | $\alpha$    | 0.1       | $\beta$    | [0, 8]       |
| $\beta$     | [0, 8]      | $\beta$     | 2         | $\rho$     | [0.5, 1]     |
| $\rho$      | [0.5, 1]    | $\rho$      | 0.1       | $q_0$      | [0.5, 1]     |
| $q_0$       | [0.5, 1]    | $PSO_\gamma$| 2         | $PSO_\delta$| [0.8, 1]     |
| $FF_\alpha$ | 2.3         | $PSO_Q_1$   | 2         |             |              |
| $FF_\beta$  | 1           | $PSO_Q_2$   | 2         |             |              |
| $FF_\gamma$ | [0, 10]     |             |           |             |              |
| $FF_\delta$ | [0.8, 1]    |             |           |             |              |

Table 2: Parameter values and ranges used for ACSFA, ACS and PSOACS

The new algorithm is implemented using MATLAB [12] and the experiment is conducted on a laptop with an Intel (R) Core (TM) i5-5200U CPU @ 2.20 GHz processor and 8GB memory. Since the speed relies on the programming language, structure and the type of the machine, the comparing algorithms were also implemented using the same environment. The
new algorithm is compared with the original ant colony algorithm (ACS) [6] and an adaptive parameter control strategy for ACO implemented using the PSO algorithm (PSOACS) [8]. The TSP instances and their best known solutions are indicated in Table (3).

| TSP Instance | Optimal Tour Length |
|--------------|---------------------|
| ulysses16    | 6859                |
| bays29       | 2020                |
| Oliver30     | 420                 |
| eil51        | 426                 |
| pr76         | 108159              |
| kroA100      | 21282               |
| lin105       | 14379               |
| tsp225       | 3916                |
| gil262       | 2378                |
| lin318       | 42029               |
| rat575       | 6773                |
| rat783       | 8806                |

Table 3: Problem instances and the Optimal tour lengths found so far

5 Results

To accomplish the experimental comparison, we considered randomly selected 12 TSP instances from the TSPLIB, that have been presented in the Table (3) [21]. Table (4) presents the results. Each algorithm executed 10 times with each TSP instance to get the results. Results are formatted as the best TSP distance, the average, the worst and the average time taken by each algorithm. The obtained results illustrate the strength of the ACSFA over other two algorithms. The interesting factor is that, in ACSFA, results are better and most of the parameters are handled by the self tuning firefly algorithm.
| TSP Instance | Algorithm | Best     | Average | Worst   | $t_{avg}$ (s) |
|------------|----------|---------|---------|--------|---------------|
| ulysses16  | ACSFA    | 6859    | 6891.2  | 6909   | 11.02         |
|            | PSOACS   | 6909    | 6909    | 6909   | 11.47         |
|            | ACS      | 6875    | 6891.2  | 6909   | 11.02         |
| bays29     | ACSFA    | 2026    | 2065    | 2085   | 36.58         |
|            | PSOACS   | 2028    | 2033.6  | 2036   | 36.87         |
|            | ACS      | 2038    | 2041.25 | 2042   | 33.30         |
| Oliver30   | ACSFA    | 421     | 425     | 426    | 22.27         |
|            | PSOACS   | 425     | 425.5   | 426    | 23.85         |
|            | ACS      | 426     | 428.83  | 434    | 19.77         |
| eil51      | ACSFA    | 428     | 432.6   | 438    | 67.30         |
|            | PSOACS   | 429     | 429.8   | 431    | 70.40         |
|            | ACS      | 430     | 434     | 439    | 60.16         |
| pr76       | ACSFA    | 108358  | 108474  | 108644 | 116.80        |
|            | PSOACS   | 108358  | 103755.8| 110488 | 157.92        |
|            | ACS      | 110281  | 111342.6| 112714 | 91.16         |
| kroA100    | ACSFA    | 21396   | 21390.71| 21537  | 163.67        |
|            | PSOACS   | 21835   | 21874   | 21914  | 180.48        |
|            | ACS      | 22011   | 22703.16| 23385  | 131.25        |
| lin105     | ACSFA    | 14412   | 14706.25| 14860  | 211.09        |
|            | PSOACS   | 14492   | 14503.33| 14571  | 194.43        |
|            | ACS      | 14844   | 15051   | 15353  | 153.10        |
| TSP225     | ACSFA    | 3978    | 4107.8  | 4232   | 519.57        |
|            | PSOACS   | 4009    | 4039.25 | 4059   | 611.89        |
|            | ACS      | 4077    | 4220.2  | 4308   | 395.20        |
| gil262     | ACSFA    | 2435    | 2448.16 | 2471   | 686.22        |
|            | PSOACS   | 2442    | 2472    | 2488   | 788.59        |
|            | ACS      | 2722    | 2872    | 2859   | 489.55        |
| lin318     | ACSFA    | 43061   | 43718.6 | 44192  | 1031.35       |
|            | PSOACS   | 43191   | 43737.6 | 44211  | 1156.92       |
|            | ACS      | 47960   | 48445   | 49427  | 595.66        |
| rat575     | ACSFA    | 7097    | 7420.33 | 7674   | 2052.31       |
|            | PSOACS   | 7189    | 7242.33 | 7346   | 3373.18       |
|            | ACS      | 7819    | 7877.5  | 7965   | 1609.39       |
| rat783     | ACSFA    | 10067   | 10226.2 | 10382  | 5998.23       |
|            | PSOACS   | 10540   | 10540   | 10540  | 6216.225      |
|            | ACS      | 10165   | 10491.25| 10642  | 5143.26       |

Table 4: The best, average, worst performance and the average time of the algorithms.
However, to prove the results in a convenient way, a statistical analysis is also conducted over the obtained results.

### 5.1 Statistical Analysis

A convenient statistical analysis was conducted to prove the validity of the results. The guidelines provided by Derrac et al. [5] were followed to perform the statistical analysis. However, here we have used parametric methods to conduct the statistical comparison. To test the hypothesis related to the study, we have used ANOVA (Analysis of variance) technique [17]. ANOVA is useful when we need to do an experiment to conduct a comparison over more than 2 samples. Therefore, to compare the 3 Algorithms in terms of error, ANOVA with RCBD (Randomized Complete Block Designs) has been applied. The hypothesis tested here is:

- **$H_0$:** There is no any significant difference between 3 algorithms
- **$H_1$:** At least one algorithm is different from others

The results of the performed statistical test is as follows:

**Analysis for error**

| Method | Factor coding (-1, 0, +1) |
|--------|--------------------------|
| Factor Information | |

| Algorithm | Type  | Levels | Values                  |
|-----------|-------|--------|-------------------------|
| ACS, ACSFA, PSOACS |

| TSP | Fixed | 12 | Bays29, eil51, gil262, kroA100, lin105, lin318, Oliver30, pr76, rat575, rat783, TSP225, ulysses16 |

| Source | DF | Adj SS | Adj MS | F - Value | P - Value |
|--------|----|--------|--------|-----------|-----------|
| Algorithm | 2 | 4043366 | 2021683 | 3.00 | 0.070 |
| TSP | 11 | 21614129 | 1964921 | 2.92 | 0.016 |
| Error | 22 | 14808697 | 673123 |  |  |
| Total | 35 | 40466192 |  |  |  |

**Model Summary**

| S | R-sq | R-sq(adj) | R-sq(pred) |
|---|------|-----------|------------|
| 820.440 | 63.40% | 41.78% | 2.01% |

Table 5: Results of ANOVA for the ‘error’

P value of the algorithm field (0.07) is less than 0.1. Based on that, we reject $H_0$ and the conclusion can be made as that there exist at least one algorithm which is significantly different from others at a 0.1 significance level. To find which algorithms are different from which, we have conducted a pairwise comparison over the algorithms. To conduct a pairwise comparison over the error, Tukey’s method was applied [16]. The results were as follows.
Tukey Pairwise Comparisons: Response = error, Term = Algorithm

Grouping Information Using the Tukey Method and 90% Confidence

| Algorithm | N | Mean   | Grouping |
|-----------|---|--------|----------|
| ACS       | 12| 1016.75| A        |
| PSOACS    | 12| 366.67 | A B      |
| ACSFA     | 12| 257.58 | B        |

Means that do not share a letter are significantly different.

Figure 1: Tukey Pairwise Comparisons for ‘error’

The hypothesis used this time is as follows.

\[ H_0: \text{There is no any difference between 2 algorithms} \]
\[ H_1: \text{There is a difference between 2 algorithms} \]

The pairwise comparison on the error concluded that there is no any difference between ACSFA and PSOACS at 90% confidence level and there is a difference between ACSFA and ACS at 90% confidence level.

The same procedure was conducted considering the average and the best results of the algorithms for 12 TSP instances. Having P-values as 0.072 and 0.070, Analysis of variance in both cases supported to reject \( H_0 \) concluding that at least one algorithm is significantly different from others at a 0.1 significance level. Pairwise comparisons were also conducted in both cases. The results are as follows.
Tukey Pairwise Comparisons: Response = Average, Term = Algorithm

Grouping Information Using the Tukey Method and 90% Confidence

| Algorithm | N | Mean  | Grouping |
|-----------|---|-------|----------|
| ACS       | 12| 19399.8 | A        |
| PSOACS    | 12| 18525.5 | A, B     |
| ACSFA     | 12| 18163.5 | B        |

Means that do not share a letter are significantly different.

Figure 2: Tukey Pairwise Comparisons for ‘average’

Tukey Pairwise Comparisons: Response = Best, Term = Algorithm

Grouping Information Using the Tukey Method and 90% Confidence

| Algorithm | N | Mean  | Grouping |
|-----------|---|-------|----------|
| ACS       | 12| 19137.3 | A        |
| PSOACS    | 12| 18487.2 | A, B     |
| ACSFA     | 12| 18378.2 | B        |

Means that do not share a letter are significantly different.
Figure 3: Tukey Pairwise Comparisons for ‘best results’

Both states support the conclusion that there is no any difference between ACSFA and PSOACS at 90% confidence level. Considering the best case, ACSFA and ACS appeared to be different at 90% confidence level.

Finally the analysis support the facts that the performance of ACSFA and PSOACS is equally strong where the performance of ACS is not as strong as ACSFA and PSOACS. But the results considering the best case, demonstrate that ACSFA outperforms other two algorithms. However although there is no significant difference between ACSFA and PSOACS from the statistical viewpoint, there exists a strong advantage of ACSFA over PSOACS: the ability of performing well without considering the selection of suitable parameter values for both ACS and FA.

5.2 Parameter Optimization

The statistical study emphasizes that there is no significant difference between ACSFA and PSOACS. But still ACSFA is better since it does provide a parameter-free environment to the user. The firefly algorithm with the self-tuning framework tunes the necessary parameters of ACS as well as FA. The figure 4 represents the evolution of parameter values of ACS over the iterations for the eil51 TSP instance. Here the mean value of each parameter for each iteration is calculated.
It is necessary to see the behavior of the parameters of the FA as well. Figure 5 shows the evolution of the parameters of FA during iterations for the eil51 TSP instance. Here also, the mean value of each parameter for each iteration is calculated.

Figures 4 and 5 point out that the parameter values vary up to some number of iterations and then stabilize over an optimum value.
6 Concluding Remarks

The study has focused on implementing an ant colony algorithm to solve symmetric TSP problems whose parameters are handled by a self tuning firefly algorithm. The algorithm was successfully implemented and tested with standard TSP problems. According to the results obtained, some key conclusions can be drawn. In terms of optimization, the results show that the ACSFA performs well in finding the shortest path for a given TSP instance. The comparisons done with ACS and PSOACS shows ACSFA works well. Although the statistical analysis concludes that both ACSFA and PSOACS have same performance, ACSFA outperforms PSOACS by providing a parameter free environment. The self tuning framework worked fine with the firefly algorithm in tuning both parameters of ACS and FA. The graphical representations of the evolution of parameters of both ACS and FA clearly demonstrates the ability of the self tuning firefly algorithm. With these we can consider the new ACSFA as a better performer to solve TSPs using ACS.

For further development, this research encourages us to study the performance of the self tuning framework with other nature inspired algorithms such as particle swarm optimization, bees algorithm etc. Also since the increasing number of cities drops the performance of the algorithm, more experimentation should be done on the population size and the initialization of parameter ranges as well.

Acknowledgment

The authors would like to thank Dr. Xin-She Yang for his valuable suggestions and explanations on implementing the self tuning framework and Ms. W.J. Polegoda, lecturer at the Faculty of Animal Science & Export Agriculture, Uva Wellassa University, Sri Lanka for the guidance given on the statistical analysis.

References

[1] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[2] M. K. A. Ariyaratne, T. G. I. Fernando, and S. Weerakoon. A self-tuning modified firefly algorithm to solve univariate nonlinear equations with complex roots. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, July 2016.

[3] Hozefa M. Botee and Eric Bonabeau. Evolving ant colony optimization. *Advances in Complex Systems*, 01(02n03):149–159, 1998.

[4] R. De Cock and E. Matthysen. Sexual communication by pheromones in a firefly, phosphaenus hemipterus (coleoptera: Lampyridae). *Animal Behaviour*, 70(4):807 – 818, 2005.

[5] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[6] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *Trans. Evol. Comp.*, 1(1):53–66, April 1997.
[7] Ayşe Hande Erol, Merve Er, and Serol Bulkan. Optimizing the ant colony optimization algorithm using neural network for the traveling salesman problem. In *Actas de la Conferencia Internacional de*, 2012.

[8] Z. f. Hao, R. c. Cai, and H. Huang. An adaptive parameter control strategy for aco. In *2006 International Conference on Machine Learning and Cybernetics*, pages 203–206, Aug 2006.

[9] D Gomez-Cabrero and DN Ranasinghe. Fine-tuning the ant colony system algorithm through particle swarm optimization. In *Proceedings of the International Conference on Information and Automation*, 2005.

[10] D. M. Gordon. Collective Wisdom of Ants. *Scientific American*, 314(2):44–47, January 2016.

[11] B. Hölldobler and E.O. Wilson. *The Ants*. Belknap Press of Harvard University Press, 1990.

[12] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[13] Roger A. Morse. Swarm orientation in honeybees. *Science*, 141(3578):357–358, 1963.

[14] Adam South, Kathrin Stanger-Hall, Ming-Luen Jeng, and Sara M. Lewis. Correlated evolution of female neoteny and flightlessness with male spermaphore production in fireflies (coleoptera: Lampyridae). *Evolution*, 65(4):1099–1113, 2011.

[15] Wikipedia. Np-hardness — Wikipedia, the free encyclopedia, 2001.

[16] Wikipedia. Tukey’s range test — Wikipedia, the free encyclopedia, 2007.

[17] B.J. Winer. *Statistical Principles in Experimental Design*. McGraw-Hill, New York, 1991.

[18] Xin-She Yang. Firefly algorithms for multimodal optimization. In *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, SAGA’09*, pages 169–178, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] Xin-She Yang, Suash Deb, Martin Loomes, and Mehmet Karamanoglu. A framework for self-tuning optimization algorithm. *Neural Computing and Applications*, 23(7-8):2051–2057, 2013.

[20] Raed Abu Zitar and Hussein Hiyassat. Optimizing the ant colony optimization using standard genetic algorithm. In *Artificial Intelligence and Applications*, pages 130–133, 2005.

[21] TSPLIB, 2008. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.