A self-tuning firefly algorithm to tune the parameters of ant colony system

M.K.A. Ariyaratne* and T.G.I. Fernando

Department of Computer Science,
Faculty of Applied Sciences,
University of Sri Jayewardenepura, Sri Lanka
Email: anuradhaariyaratne@gmail.com
Email: gishantha@dscs.sjp.ac.lk
*Corresponding author

Sunethra Weerakoon

Department of Mathematics,
Faculty of Applied Sciences,
University of Sri Jayewardenepura, Sri Lanka
Email: sunethra.weerakoon@gmail.com

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 optimisation algorithms combined with the firefly algorithm (FA) to tune the parameters of the ACS solving symmetric TSP problems. The FA optimises 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.

Keywords: self-tuning framework; ant colony system; ACS; travelling salesman problem; TSP; firefly algorithm.

Reference to this paper should be made as follows: Ariyaratne, M.K.A., Fernando, T.G.I. and Weerakoon, S. (2018) 'A self-tuning firefly algorithm to tune the parameters of ant colony system', *Int. J. Swarm Intelligence*, Vol. 3, No. 4, pp.309–331.

Biographical notes: M.K.A. Ariyaratne completed her BSc in Computer Science from the University of Sri Jayewardnepura, Sri Lanka in 2012. Currently, she is pursuing her PhD in the Department of Computer Science, Faculty of Graduate Studies, University of Sri Jayewardnepura, Sri Lanka. Her major areas of interests are nature inspired optimisation techniques and machine learning algorithms.

T.G.I. Fernando is a Senior Lecturer/Researcher in Computer Science at the Department of Computer Science, University of Sri Jayewardenepura, Sri Lanka. He has completed his BSc in Mathematics in 1993 in the University of Jayewardenepura Sri Lanka. Then he completed his MSc on Industrial Mathematics in 1998 from the University of Sri Jayewardenepura Sri Lanka. Then in 2002, he completed his MSc in Computer Science at the Asian Institute of Technology Thailand in 2002. Finally, he completed his PhD under Intelligent Systems in the Brunel University in London. His research interests lie in intelligent systems, evolutionary computing, swarm intelligence, neural networks (including deep learning neural networks), machine learning and multi-objective combinatorial optimisation.

Sunethra Weerakoon retired in 2017 as a Senior Professor of Mathematics from the University of Sri Jayawardenepura, Sri Lanka where she served since 1976. She finished her Bachelor's in Mathematics at the University of Peradeniya, Sri Lanka (1972/76). She pursued her Master's and Doctoral studies as a Fulbright Scholar at the Pennsylvania State University, University Park, USA (1979/84). She was a Visiting Teaching Fellow at the Curtin University of Technology, Western Australia (1991/93). She was also a Visiting Professor at the Departments of Mathematics, Cornell University (2004/05) and Texas A & M University, College Park (2005/06). She has written a few books: *Elementary Numerical Methods*, *Numerical Solution of Ordinary Differential Equations* and *Partial Differential Equations* and edited a few others. Her research interests are in numerical analysis and partial differential equations. She has received a number of awards for her academic achievements and research.

1 Introduction

The collective behaviour of natural insects – ants, bees, fireflies and termites mimic the problem solving capabilities of the swarms (Gordon, 2016; Morse, 1963; de Cock and Matthysen, 2005). These capabilities were adopted in various heuristics and meta-heuristics to solve difficult optimisation problems. Each meta-heuristic has a real world inspiration of optimisation. As such, the main inspiration for the ant colony system (ACS) 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 optimised behaviour. Following this real world strategy, the ACS algorithm was developed by Dorigo and Gambardella (1997) to suit for path optimisation 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 ACS 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 optimisation where the optimal parameter set will give the optimum performance.

Though the initial study relied on trial and error, Dorigo and Gambardella (1997) had stated some important features of parameters such as pheromone behaviour, the number of ants and how they affect the performance of the algorithm. 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 optimisation 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 ACS by Hao et al. (2006) is a study carried out to enhance the performance of the ant system solving the TSP. Although it has been mentioned as the ACS, they have used the ant system for the study. The particle swarm optimisation (PSO) has been used to optimise 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.

A review done by Stützle et al. (2011) provides a good background study on investigating the parameter adaption in ACS algorithms. The review has done under several classifications as pre-scheduled, adaptive and search based of parameter tuning techniques. The study has shown that there is a good interest on automatic parameter tuning of ant colony optimisation algorithms. The problem raised by the conclusion is that the contributions of studies do not address or understand the effect of individual parameters and the adaption have done in arbitrary ways. Research on self-tuning/self-adapting approaches were discussed on behalf the ant colony optimisation where an ant represents a tour and a set of algorithm parameters (Randall, 2004). However the paper reviews that the comparison carried with default parameters settings is inconclusive.

In evolving ant colony optimisation, another research by Botee and Bonabeau (1998), they have made the use of genetic algorithm (GA) to evolve the best set of parameters. Here also, one parameter set represents an individual of the GA (a chromosome). 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 an example, Erol et al. (2012) in their research, have used artificial neural networks (ANNs) to find the best parameter set for ACS solving a given TSP. The research focuses on only two parameters, α: pheromone decay parameter and β: the relative importance of the pheromone vs. distance. Initially the ACSANN hybrid algorithm runs with different α and β 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 (2008). 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 studies which have been conducted to find better parameter sets for the ACS in solving TSP (Gomez-Cabrero and Ranasinghe, 2005; Zitar and Hiyassat, 2005). 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 (FA) and the self-tuning framework for optimisation algorithms proposed by Yang, to optimise the parameters of the ACS algorithm solving symmetric TSP instances. In the implementation of the self-tuning framework have used the FA to apply the framework to tune FA's parameters (Yang et al., 2013). In their framework, the problem solved by the optimisation algorithm and the parameter set of the algorithm both were considered as a single problem. The framework was initially tested for the FA and proved its capability of tuning parameters of itself (FA). In a research done by Ariyaratne et al. (2016), they use this self-tuning framework combined with the FA to approximate roots of nonlinear equations. They have solved univariate nonlinear equations having complex roots with the help of a modified FA. To find the best parameter values, the self-tuning framework was implemented on the FA. 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 optimising parameters.

The significance of this research lies in the potential of the developed ant colony optimisation algorithm for the TSP with the self-tuning framework to optimise the parameters. Despite the recent advancements in the field of route optimisation and parameter optimisation, the following issues have not been addressed by other researchers (see Table 1 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 ACS to solve TSPs
- in most of the studies, parameter optimisation is done using another meta-heuristic algorithm whose parameters should be manually selected
- none of the approaches have used a designed framework for parameter optimisation.

In this paper, Yang's self-tuning framework is studied, and a parameter selection strategy based on the FA 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 ACS and the FA. In Section 3, we briefly describe the self-tuning framework for the FA 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 and information about the parameters used in the study. Section 5

emphasises the results obtained from the new approach, goodness of the new algorithm comparing the results with the original ACS and with some other parameter tuning approaches. Finally, we draw conclusions briefly in Section 6.

Table 1 Previous work on parameter optimisation of ant systems

Author	*Year*	*Parameter optimisation method*	*# of TSP problems tested*	*Limitations*
Botee and Bonabeau	1998	GA	2	Some parameters of ACS were optimised (m: number of Ants, q_0: exploitation probability, α: weights of the pheromone trail, β: distance between two nodes, ρ_l and ρ_g: local and global trail decay, τ_0: amount of local reinforcement).
				The parameters of GA should be manually updated.
Pilat and White	2002	GA	4	Some parameters of ACS were optimised (β: the relative importance of the pheromone vs. distance, ρ: pheromone evaporation coefficient, q_0: exploitation probability).
				The parameters of GA should be manually updated.
Randall	2004	ACO (self-adaptive)	4	The comparison of the results to the default parameter settings is inconclusive.
Zitar and Hiyassat	2005	GA	8	Two parameters of ACS (α and β) were updated in the first phase.
Gaertner and Clark	2005	ACO (self-adaptive)	6	Three parameters of ACO (ρ, q_0 and β) were adapted. Used TSP instances to test were small.
Gomez-Cabrero and Ranasinghe	2005	PSO	24	The parameters of PSO should be manually updated. High computational overhead.
Hao et al.	2006	PSO	10	The parameters of PSO should be manually updated.
Erol et al.	2012	Artificial neural networks (ANN)	16	Only some parameters of ACO were optimised (α: pheromone decay parameter, β: the relative importance of the pheromone vs. distance).

Notes: GA – genetic algorithms; PSO – particle swarm optimisation; ACS – ant colony systems; ACO – ant colony optimisation.

2 Preliminaries

2.1 Travelling salesman problem

The TSP is one of the most intensively studied problems in computational mathematics which is simple to state but very difficult to solve (Applegate et al., 2007). The problem

is NP hard, making it not computable in polynomial time (van Leeuwen, 1990). 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

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 (Hölldobler and Wilson, 1990). 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 optimised behaviours of real ants are based on implementing artificial ant colonies.

The ACS is an example of an ant colony optimisation method from the field of swarm intelligence, meta-heuristics and computational intelligence. Around 1990s, Dorigo introduced the idea of the ant system and later Dorigo and Gambardella introduce the ACS. In the original implementation, ACS was applied to solve the TSP (Dorigo and Gambardella, 1997). 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. Its next city s is determined by equation (1).

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ \left[\tau(r,u)^{\theta} \right] \cdot [\eta(r,u)]^{\beta} \right\} & q \leq q_0 \\ S & \text{Otherwise} \end{cases} \tag{1}$$

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 θ and β (θ, $\beta \geq 0$). 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 equation (2).

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

ACS local and global updating happens according to equations (3) and (4), respectively.

$$\tau(r,s)\leftarrow(1-\rho)\cdot\tau(r,s)+\rho\cdot\Delta\tau(r,s) \tag{3}$$

where $0<\rho<1$ is a parameter.

$$\tau(r,s)\leftarrow(1-\alpha)\cdot\tau(r,s)+\alpha\cdot\Delta\tau(r,s) \tag{4}$$

where

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

$0<\alpha<1$ 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 optimisation and hence many studies have been carried out to improve the algorithm as well as to check its ability over solving other optimisation problems.

2.3 Firefly algorithm

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 (South et al., 2011). Focusing on their flashing behaviour, the FA was developed by Yang (2009). 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.

Many meta-heuristic algorithms generate the initial population of a given problem in a random manner. Similarly, if FA, the initial population for a particular problem is generated randomly. Also, the parameter set of the algorithm should be specified properly. After these initial steps, the fireflies in the population start moving towards brighter fireflies according to the equation (6).

$$x_i=x_i+\beta\left(x_j-x_i\right)+\alpha(rand-0.5) \tag{6}$$

where

$$\beta = \beta_0 \cdot e^{-\gamma r^2} \tag{7}$$

Here x_i and x_j refer to two fireflies. β is the attraction between two fireflies and α is the parameter controlling the step size. β_0 is the attraction at $r = 0$, where r is the distance between two fireflies. The three terms in equation (6) represent the contribution from the current firefly, attraction between two fireflies and a randomisation term respectively. The equation supports both exploitation and exploration. α plays an important role in the randomisation process, which is from uniform or Gaussian distribution. To control the randomness, after each iteration, Yang has used δ, the randomness reduction factor which reduces α according to the equation (8).

$$\alpha = \alpha \cdot \delta \quad \text{where } \delta \in [0, 1] \tag{8}$$

FA, as a well-established performer in the world of meta-heuristics, marked its remarkable capability of handling optimisation problems (Fister et al., 2013; Gandomi et al., 2011; Yang et al., 2012). Yang et al. (2013) introduced a framework for self-tuning algorithms and it was implemented with the FA successfully. The framework allows a meta-heuristic algorithm to solve a problem while optimising its own algorithm specific parameters.

3 Self-tuning FA optimising ACS's parameters

In this research, the main aim is to tune the parameters of the ACS algorithm while obtaining the shortest path for the selected symmetric TSP. To tune the parameters of ACS, FA has been used here. The reason is that the self-tuning framework can be easily implemented on FA rather than directly implementing it on ACS, since the ACS solves a discrete problem (TSP) and the parameters are continuous in nature.

Regarding the ACS, β, θ, ρ and q_0 parameters have to be tuned. The FA also has several parameters such as α, γ, β and the randomness reduction factor δ. The main aim of the self-tuning concept is to find the best parameter settings that minimise the computational cost. When applying the self-tuning framework to the FA solving a given optimisation 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 which in this case is having the minimum route distance.

In this research the ACS along with FA has created a hybrid algorithm ACSFA. The FA is used to tune both parameters of ACS and FA while ACS optimises a given TSP. The objective of the algorithm is to find the optimal solution of a given TSP instance while finding the most suitable parameters for both ACS and FA algorithms. The pseudo code of the proposed ACSFA algorithm is presented in Algorithm 1.

Algorithm 1 Pseudo code of the ACSFA

1: Begin;
2: Initialise parameter values and ranges of parameter values to be tuned
3: Initialise `antffs` in the population. (see section 3.1)
4: Define the objective function ($F(X)$ – *Inverse of the distance*)

```
5:  while End condition do
6:      Begin Tour
7:          Build the tour while local pheromone update by using equations (1), (2), (3);
8:      End Tour
9:      Calculate fitness I using the objective function (here I = F(X))
10:     Do global pheromone update by using equation (4)
11:     for i = 1: n (all n antffs) do
12:         for j = 2: n (n antffs) do
13:             if I_j > I_i then
14:                 Move antff i towards antff j by using equation (6);
15:             end if
16:             Attractiveness varies with distance r via e−γr2 using equation (7);
17:             Evaluate new solutions and update parameter values;
18:         end for
19:     end for
20:     Rank the antffs and find the current best. (best tour + best parameter values);
21: end while
22: Post process results and visualisation;
23: End
```

3.1 Representation of an ant and a firefly

An ant and a firefly are joined to create a hybrid entity. We shall call it as `antff`. An `antff` consists of visited cities and parameter values of ACO and FF algorithms. For example, the i^{th} hybrid entity is represented as follows:

```
antff[i] = [(visited-cities), (ACO & FF parameters)]
```

```
antff[i] = [(visited-cities), (β, ρ, q0, FF_γ, FF_δ)]
```

Initially, the list of visited cities (`visited-cities`) contains only the starting city and random values that are assigned (within the given range of each parameter) for each ACS and FF parameters. For an example, if the algorithm uses 10 hybrid `antffs`, then each `antff` carries a starting city and different set of parameter approximations at the beginning. An `antff[i]` builds a tour following the ACS algorithm using the randomly assigned ACS parameter values for it. Inverse of a tour distance has taken as the objective function value of the TSP. The problem is set as a maximisation problem. The ACS algorithm will build tours as follows until a predetermined end condition (number of iterations) is completed.

- Each `antff` completes a tour using the state transition rule stated in equation (1).
- While completing the tour `antffs` will lay pheromones on the visited cities according to the equation (3).

- Upon completing tours by all, 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).
- After completing a tour, all `antffs` will update the parameters of FA and ACS using the self-tuning FA. The `antffs` will move in the direction of better `antffs` according to the equation (6).
- When all `antffs` complete one round moving towards better `antffs`, the algorithm will check the parameter values of all `antffs` to make sure the parameters are within the specified range. Otherwise their values will be updated according to the defined ranges.

Here the fitness of an `antff` is determined by the inverse of tour lengths obtained by the ACS. At the end of the iterations, the `antff` having the best tour distance and best parameter set will be detected. The flow chart of the algorithm is shown in Figure 1.

For example, at the beginning of the algorithm the i^{th} hybrid `antff` is represented as follows (assume a TSP with five cities):

```
antff[i] = [(5),(2 0.6 0.8 5 0.9)]
```

After completing an iteration it will build a tour according to the ACS as well as adjust its parameters according to the FA.

```
antff[i] = [(5 3 4 2 1),(1.2 0.7 0.85 3.4 0.83)]
```

When reaching the stopping criteria, the globally best antff (`gBest_antff`) carries the best tour given by the ACS algorithm and the best parameter values of ACS and FA given by the self-tuning FA.

```
gBest_antff = [(43 5 1 2),(1.3 0.75 0.9 2.1 0.93)]
```

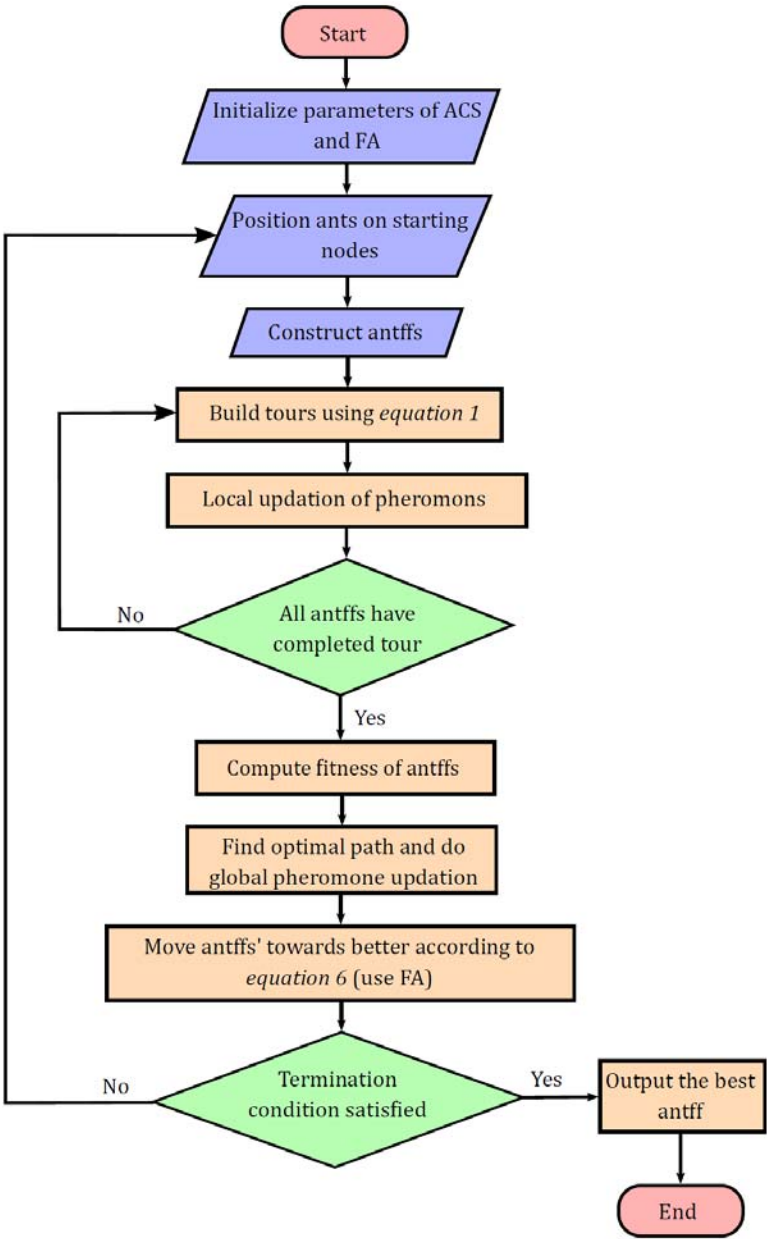
4 Experimentation

The goal of this experiment is to analyse 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 optimising the TSP instances.

4.1 Parameter values and problem instances

The new algorithm is implemented to solve symmetric TSPs. Twelve symmetric TSPs are selected for testing (TSPLIB, 2008). The parameters of ACS include: β, α, ρ, q_0, m, τ_0 and $\tau \cdot \tau_0$ is based on the nearest neighbour heuristic. Pheromone density τ depends on α and ρ parameters. Since α works only with the globally best ant, we initialise α to be 0.1. Therefore, we consider β, ρ and q_0 in ACS to be tuned by the self-tuning FA. The ranges 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.

Figure 1 Flowchart of ACSFA algorithm (see online version for colours)



Apart from that, we tune the parameters of the FA. The parameters of FA include α, β, β_0, γ, δ and number of fireflies. For the convenience let us rename the parameters of FA as *FF_α*, *FF_β*, *FF_β*$_0$, *FF_γ* and *FF_δ*. In equation (7), the brightness *FF_β* depends on three main factors; *FF_β*$_0$ which we initialise to be 1, *FF_γ*; the light absorption coefficient and *r*; the Cartesian distance between two fireflies which should be calculated during iterations. In equation (8), the value of *FF_α* get reduced with *FF_δ*. *FF_α* is

initialised to be 2.3. Therefore the value of FF_α depends on FF_δ. Hence the only factors to be tuned in FA are FF_γ and FF_δ. The parameters to be tuned vary 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. These parameter values and the ranges were selected according to the information given in the original implementations of the used algorithms.

Table 2 gives the information about the parameters used in the present study (ACSFA), the original ACS solving the TSP (Dorigo and Gambardella, 1997) and an adaptive parameter control strategy for ACO implemented using the PSO algorithm (PSOACS) (Hao et al., 2006). In ACSFA, we have both parameters of ACS and FA. In first two columns, those parameters and values (value is given as a range, when the parameter is going to be tuned) are given. When it is the original ACS (Dorigo and Gambardella, 1997), no tuning has been done, therefore constant values for parameters of ACS were used. In PSOACS (Hao et al., 2006), only the parameters of ACS were tuned (therefore ranges for the parameters to be tuned are given), and parameters of PSO (Q_1 and Q_2) contain constant values.

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

ACSFA		*ACS*		*PSOACS*	
Parameter	*Value/range*	*Parameter*	*Value/range*	*Parameter*	*Value/range*
α	0.1	α	0.1	β	[0 8]
β	[0 8]	β	2	ρ	[0.5 1]
ρ	[0.5 1]	ρ	0.1	q_0	[0.5 1]
q_0	[0.5 1]			PSO_Q_1	2
FF_α	2.3			PSO_Q_2	2
FF_β_0	1				
FF_γ	[0 10]				
FF_δ	[0.8 1]				

Table 3 TSP instances and the optimal tour lengths found so far

TSP instance	*Optimal tour length*
ulysses16	6,859
bays29	2,020
Oliver30	420
eil51	426
pr76	108,159
kroA100	21,282
lin105	14,379
tsp225	3,916
gil262	2,378
lin318	42,029
rat575	6,773
rat783	8,806

The new algorithm is implemented using MATLAB (2010) and the experiments are conducted on a laptop with an Intel (R) Core (TM) i5-5200U CPU @ 2.20 GHz processor and 8 GB 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) (Dorigo and Gambardella, 1997) and an adaptive parameter control strategy for ACO implemented using the PSO algorithm (PSOACS) (Hao et al., 2006). The TSP instances and their best-known solutions are indicated in Table 3.

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 (TSPLIB, 2008). Table 5 presents the results. Each algorithm executed ten 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 FA.

However, to further strengthen the results of the new algorithm, a statistical analysis is also conducted over the obtained results.

Table 4 Results of ANOVA for the 'error'

Analysis for error					
Method					
Factor coding (–1, 0, +1)					
Factor information					
Factor	*Type*		*Levels*	*Values*	
Algorithm	Fixed		3	ACS, ACSFA, PSOACS	
TSP	Fixed		12	Bays29, eil51, gil262, kroA100, lin105, lin318, Oliver30, pr76, rat575, rat783, TSP225, ulysses16	
Analysis of variance					
Source	*DF*	*Adj SS*	*Adj MS*	*F-value*	*P-value*
Algorithm	2	4,043,366	2,021,683	3.00	0.070
TSP	11	21,614,129	1,964,921	2.92	0.016
Error	22	14,808,697	673,123		
Total	35	40,466,192			
Model summary					
S	*R-sq*		*R-sq (adj)*	*R-sq (pred)*	
820.440	63.40%		41.78%	2.01%	

Table 5 The best, average, worst performance and the average time of the algorithms

TSP instance	*Algorithm*	*Best*	*Average*	*Worst*	*tavg (s)*
ulysses16	ACSFA	6,859	6,891.2	6,909	11.02
	PSOACS	6,909	6,909	6,909	11.47
	ACS	6,875	6,891.2	6,909	11.02
bays29	ACSFA	2,026	2,065	2,085	36.58
	PSOACS	2,028	2,033.6	2,036	36.87
	ACS	2,038	2,041.25	2,042	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	108,358	108,474	108,644	116.80
	PSOACS	108,358	107,755.8	110,488	157.92
	ACS	110,281	111,342.6	112,714	91.16
kroA100	ACSFA	21,396	21,390.71	21,537	163.67
	PSOACS	21,835	21,874	21,914	180.48
	ACS	22,011	22,703.16	23,385	131.25
lin105	ACSFA	14,412	14,706.25	14,860	211.09
	PSOACS	14,492	14,503.33	14,571	194.43
	ACS	14,844	15,051	15,353	153.10
TSP225	ACSFA	3,978	4,107.8	4,283	519.57
	PSOACS	4,009	4,039.25	4,059	611.89
	ACS	4,077	4,220.2	4,308	395.20
gil262	ACSFA	2,435	2,448.16	2,471	686.22
	PSOACS	2,442	2,472	2,488	788.59
	ACS	2,722	2,872	2,859	489.55
lin318	ACSFA	43,061	43,718.6	44,192	1,031.35
	PSOACS	43,191	43,737.6	44,211	1,156.92
	ACS	47,960	48,445	49,427	595.66
rat575	ACSFA	7,097	7,420.33	7,674	2,952.31
	PSOACS	7,189	7,242.33	7,346	3,373.18
	ACS	7,819	7,877.5	7,965	1,609.39
rat783	ACSFA	10,067	10,226.2	10,382	5,998.23
	PSOACS	10,540	10,540	10,540	6,216.225
	ACS	10,165	10,491.25	10,642	5,143.26

5.1 Statistical analysis

A convenient statistical analysis was conducted to prove the validity of the results. The guidelines provided by Derrac et al. (2011) 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 analysis of variance (ANOVA) technique (Winer, 1991). ANOVA is useful when we need to do an experiment to conduct a comparison over more than two samples. Therefore, to compare the three algorithms in terms of error, ANOVA with randomised complete block designs (RCBD) has been applied. The hypothesis tested here is:

H_0 There is no any difference between three algorithms.

H_1 At least one algorithm is different from others.

The results of the performed statistical test are shown in Table 4.

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 (Tukey, 1949). The results are shown in Figure 2.

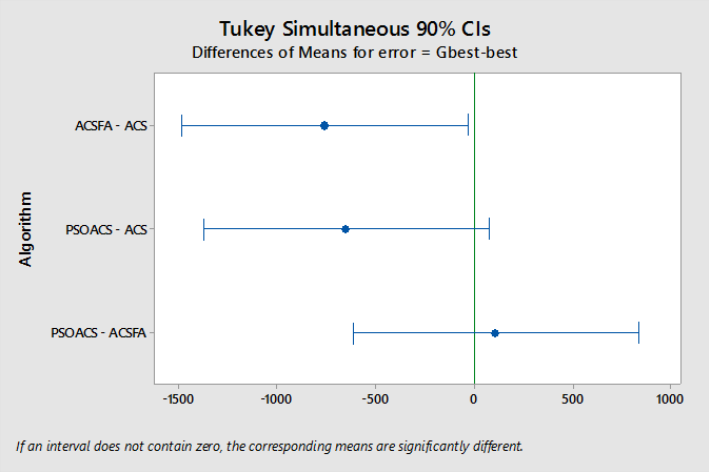
Figure 2 Tukey pairwise comparisons for 'error' (see online version for colours)

Tukey pairwise comparisons: response = error, term = algorithm

Grouping information using the Tukey method and 90% confidence

Algorithm	*N*	*Mean*	*Grouping*
ACS	12	1,016.75	A
PSOACS	12	366.67	A B
ACSFA	12	257.58	B

Means that do not share a letter are significantly different.



The hypothesis used this time is as follows:

H_0 There is no any difference between two algorithms.

H_1 There is a difference between two 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 shown in Figures 3 and 4.

Figure 3 Tukey pairwise comparisons for 'average' (see online version for colours)

Tukey pairwise comparisons: response = average, term = algorithm

Grouping information using the Tukey method and 90% confidence

Algorithm	*N*	*Mean*	*Grouping*
ACS	12	19,399.8	A
PSOACS	12	18,525.5	A B
ACSFA	12	18,163.5	B

Means that do not share a letter are significantly different.

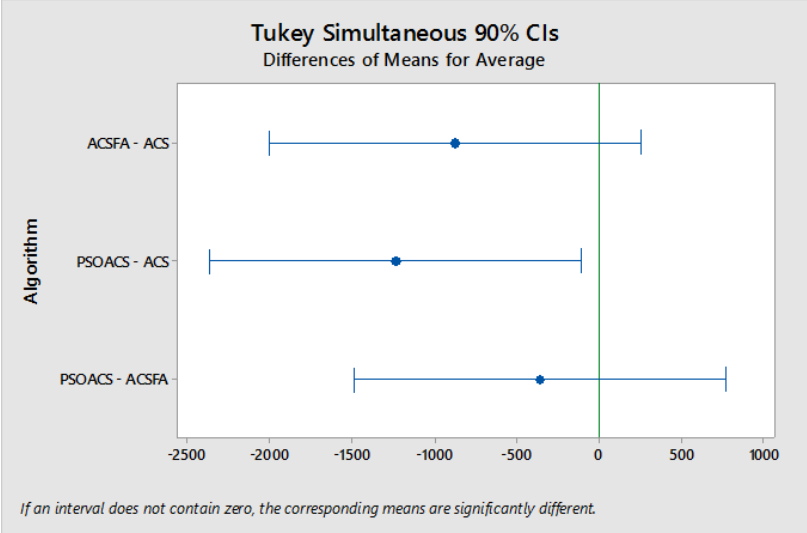
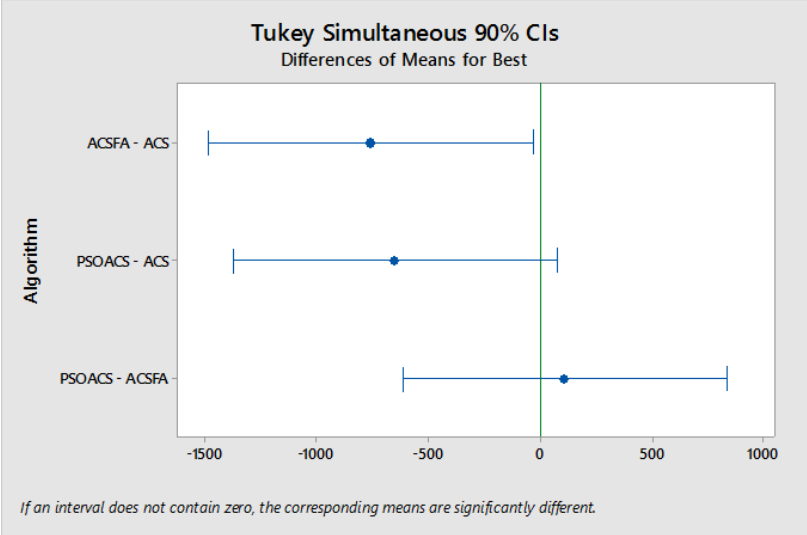


Figure 4 Tukey pairwise comparisons for 'best results' (see online version for colours)

Tukey pairwise comparisons: response = best, term = algorithm			
Grouping information using the Tukey method and 90% confidence			
Algorithm	*N*	*Mean*	*Grouping*
ACS	12	19,137.3	A
PSOACS	12	18,487.2	A B
ACSFA	12	18,378.2	B
Means that do not share a letter are significantly different.			



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 supports 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.

We have conducted a non-parametric analysis to further clarify the results. The Friedman test is a non-parametric alternative to ANOVA with repeated measures (Daniel, 1990). It can be used for situations where sample size is quite small (here it is 12). No normality assumption is required. It is used to test for differences between groups when the dependent variable being measured is at least in ordinal scale (can use with interval and ration data as well). The following hypothesis was tested under a significance level (α) of 0.05.

H_0 There is no any difference between three algorithms.

H_1 At least two algorithms are different from each other.

The test has been conducted over 12 samples (TSP instances) for three algorithms, ACSFA, PSOACS and ACS for the best performance at each problem instance (Table 5). The results were shown in Table 6. All data analyses were performed using Minitab statistical software version 17.0 (Minitab, 2010).

Table 6 Friedman test for the TSP instances over three algorithms

Friedman test: Best distance versus algorithm blocked by block			
S = 19.54	DF = 2	P = 0.000	
S = 19.96	DF = 2	P = 0.000 (adjusted for ties)	
Algorithm	*N*	*Est. median*	*Sum of ranks*
ACSFA	12	6,998.3	12.5
PSOACS	12	7,049.0	25.5
ACS	12	7,181.2	34.0
Grand median = 7,076.2			

Since P-value < α, we reject the null hypothesis and conclude that at least two algorithms are different from each other at 0.05 level of significance.

To find out which algorithm is different from which, we have calculated the critical difference value (CD) for the mean comparison, using the following equation.

$$CD = Z_\alpha / k(k-1)\sqrt{\frac{nk(k+1)}{6}} \tag{9}$$

Here α =0.05, k is the number of difference groups which is 3 and n is the sample size, which is 12.

The CD value calculated according to the equation (9) is 11.7287. When we consider ACSFA and PSOACS, the difference between their sum of ranks is (25.5 – 12.5), 13 which is greater than the CD value (11.7287), which means the performances two algorithms are different. Similarly, for ACSFA and ACS, the difference between sum of ranks is (34 – 12.5), 21.5, which is also greater than the CD value (11.7287), indicating that the performances of those two algorithms are also different (The complete calculations of differences of sum of ranks can be seen in Table 7, The * indicates algorithms which are different from each other).

Table 7 Absolute difference between sum of ranks of the three algorithms

	PSOACS	*ACS*
ACSFA	13*	21.5*
PSOACS	-	8.5
ACS	8.5	-

Therefore, we can conclude that, ACSFA performs differently from both PSOACS and ACS algorithms and since this is a minimisation problem and minimum sum of ranks value belongs to the ACSFA algorithm, conclusions can be made as ACSFA algorithm outperforms other two algorithms.

5.2 Parameter optimisation

The statistical study emphasises that there is a significant difference between ACSFA and PSOACS. ACSFA is better not only because it provides good answers but also it does provide a parameter-free environment to the user. The FA with the self-tuning framework tunes the necessary parameters of ACS as well as FA. Figure 5 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.

Figure 5 Variation of the average β, ρ and q_0 values of ACS over iterations for eil51 TSP instance (see online version for colours)

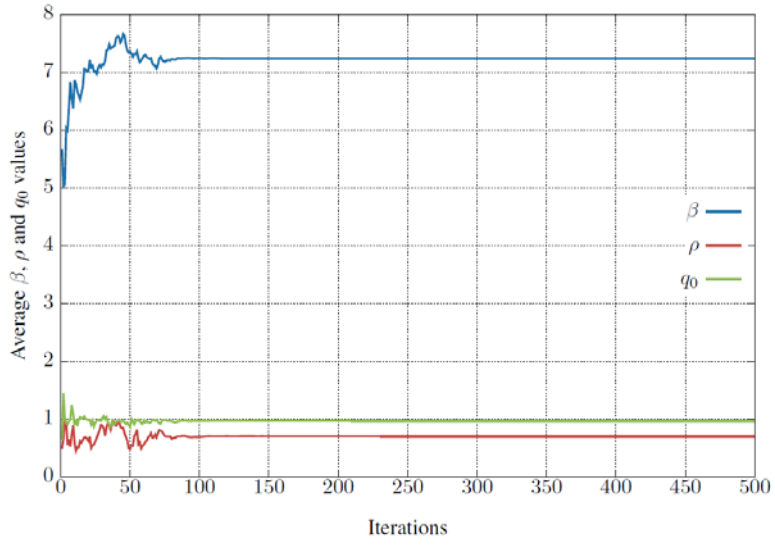
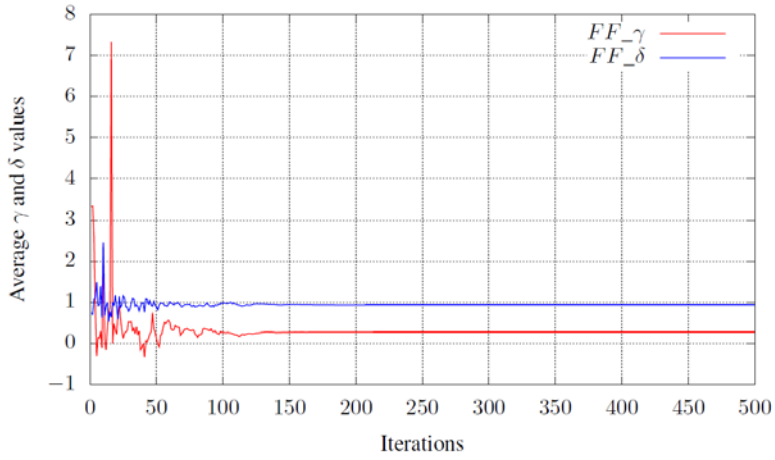


Figure 6 Variation of the average FF_γ and FF_δ values of FA over iterations for eil51 TSP instance (see online version for colours)



It is necessary to see the behaviour of the parameters of the FA as well. Figure 6 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 5 and 6 point out that the parameter values vary up to some number of iterations and then stabilises over an optimum value.

The variation of parameter values may same or different for different problem instances. Table 8 contains the problem instances and best parameter values obtained by ACSFA.

Table 8 Best parameter values obtained by ACSFA for ACS and FA

TSP	*ACS*			*FA*	
	β	ρ	q_0	*FF_γ*	*FF_δ*
ulysses16	0.00014	0.225	0.9536	0.1883	0.8894
bays29	7.7948	0.0994	0.9084	2.02	0.8593
Oliver30	3.9699	0.7997	0.9151	2.6513	0.9002
eil51	2.9374	0.9741	0.9201	8.8669	0.9206
pr76	1.1099	0.4815	0.9927	0.2223	0.9339
kroA100	7.5493	0.8738	0.9437	0.1486	0.926
lin105	3.1221	0.7714	0.9161	0.3917	0.9569
tsp225	0.8258	0.9905	0.9784	0.1405	0.9118
gil262	2.378	0.9202	0.968	0.0183	0.9153
lin318	6.3894	0.8075	0.9567	0.52	0.9978
rat575	0.0464	0.5466	0.9726	1.0255	0.9021
rat783	5.2348	0.9512	0.9764	0.7747	0.9529

It is clear that for some parameters such as β, ρ and *FF_γ*, the best parameter values are highly problem specific. So that the traditional methods like trial and error may take higher time in finding the suitable parameter set for a particular problem. But using our method, without any difficult, able to get a reasonably good solution and a suitable parameter set.

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 FA. 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 optimisation, 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 FA in tuning both parameters of ACS and FA. The graphical representations of the evolution of parameters of both ACS and FA clearly demonstrate the ability of the self-tuning FA. 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 PSO, bee 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 initialisation of parameter ranges as well.

Acknowledgements

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 Department of Export Agriculture, Faculty of Animal Science AND Export Agriculture, Uva Wellassa University, Sri Lanka for the guidance given on the statistical analysis.

References

Applegate, D.L., Bixby, R.E., Chvatal, V. and Cook, W.J. (2007) *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*, Princeton University Press, Princeton, NJ, USA, ISBN 0691129932, 9780691129938.

Ariyaratne, M.K.A., Fernando, T.G.I. and Weerakoon, S. (2016) 'A self-tuning modified firefly algorithm to solve univariate nonlinear equations with complex roots', *2016 IEEE Congress on Evolutionary Computation (CEC)*, July, pp.1477–1484, doi: 10.1109/CEC.2016.7743964.

Botee, H.M. and Bonabeau, E. (1998) 'Evolving ant colony optimization', *Advances in Complex Systems*, Vol. 1, Nos. 2–3, pp.149–159, doi: 10.1142/S0219525998000119 [online] http://www.worldscientific.com/doi/abs/10.1142/S0219525998000119 (accessed 16 May 2017).

Daniel, W.W. (1990) *Applied Nonparametric Statistics*, The Duxbury Advanced Series in Statistics and Decision Sciences, PWS-Kent Publ., ISBN 9780534919764 [online] https://books.google.lk/books?id=0hPvAAAAMAAJ (accessed 20 April 2017).

de Cock, R. and Matthysen, E. (2005) 'Sexual communication by pheromones in a firefly, phosphaenus hemipterus (coleoptera: Lampyridae)', *Animal Behaviour*, Vol. 70, No. 4, pp.807–818, ISSN 0003-3472, doi: http://dx.doi.org/10.1016/j.anbehav.2005.01.011 [online] http://www.sciencedirect.com/science/article/pii/S0003347205002162 (accessed 10 April 2017).

Derrac, J., García, S., Molina, D. and Herrera, F. (2011) 'A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms', *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp.3–18, ISSN 2210-6502, doi: http://dx.doi.org/10.1016/j.swevo.2011.02.002 [online] http://www.sciencedirect.com/science/article/pii/ S2210650211000034 (accessed 10 January 2017).

Dorigo, M. and Gambardella, L.M. (1997) 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *Trans. Evol. Comp.*, April, Vol. 1, No. 1, pp.53–66, ISSN 1089-778X, doi: 10.1109/4235.585892 [online] http://dx.doi.org/10.1109/4235.585892 (accessed 11 February 2017).

Erol, A.H., Er, M. and Bulkan, S. (2012) 'Optimizing the ant colony optimization algorithm using neural network for the traveling salesman problem', *Actas de la Conferencia Internacional de*.

Fister, I., Fister Jr., I., Yang, X-S. and Brest, J. (2013) 'A comprehensive review of firefly algorithms', *Swarm and Evolutionary Computation*, Vol. 13, pp.34–46.

Gaertner, D. and Clark, K. (2005) 'On optimal parameters for ant colony optimization algorithms', *Proceedings of the International Conference on Artificial Intelligence 2005*, pp.83–89, CSREA Press.

Gandomi, A.H., Yang, X-S. and Alavi, A.H. (2011) 'Mixed variable structural optimization using firefly algorithm', *Computers and Structures*, Vol. 89, No. 23, pp.2325–2336, ISSN 0045-7949, doi: http://dx.doi.org/10.1016/j.compstruc.201108.002 [online] http://www.sciencedirect.com/science/article/pii/S0045794911002185 (accessed 15 December 2016).

Gomez-Cabrero, D. and Ranasinghe, D.N. (2005) 'Fine-tuning the ant colony system algorithm through particle swarm optimization', *Proceedings of the International Conference on Information and Automation*.

Gordon, D.M. (2016) 'Collective wisdom of ants', *Scientific American*, January, Vol. 314, No. 2, pp.44–47, doi: 10.1038/scientificamerican0216-44.

Hao, Z.F., Cai, R.C. and Huang, H. (2006) 'An adaptive parameter control strategy for ACO', *2006 International Conference on Machine Learning and Cybernetics*, August, pp.203–206, doi: 10.1109/ICMLC.2006.258954.

Hölldobler, B. and Wilson, E.O. (1990) *The Ants*, Belknap Press of Harvard University Press, ISBN 9780674040755 [online] https://books.google.de/books?id= ljxV4h61vhUC (accessed 10 January 2017).

MATLAB (2010) version 7.10.0 (R2010a), The MathWorks Inc., Natick, Massachusetts.

Minitab Inc (2010) *Minitab 17 Statistical Software* [online] http://www.minitab.com (accessed 10 August 2017).

Morse, R.A. (1963) 'Swarm orientation in honeybees', *Science*, Vol. 141, No. 3578, pp.357–358, ISSN 0036-8075, doi: 10.1126/science.141.3578.357 [online] http://science.sciencemag.org/ content/141/3578/357 (accessed 20 May 2017).

Pilat, M.L. and White, T. (2002) 'Using genetic algorithms to optimize ACS-TSP', *International Workshop on Ant Algorithms*, pp.282–287, Springer.

Randall, M. (2004) *Near Parameter Free Ant Colony Optimisation*, pp.374–381, Springer Berlin Heidelberg, Berlin; Heidelberg, ISBN 978-3-540-28646-2.

South, A., Stanger-Hall, K., Jeng, M-L. and Lewis, S.M. (2011) 'Correlated evolution of female neoteny and flightlessness with male spermatophore production in fireflies (coleoptera: Lampyridae)', *Evolution*, Vol. 65, No. 4, pp.1099–1113, ISSN 1558-5646, doi: 10.1111/j.1558-5646.2010.01199.x [online] http://dx.doi.org/10.1111/j.1558-5646. 2010.01199.x (accessed 21 June 2017).

Stützle, T., López-Ibánez, M., Pellegrini, P., Maur, M., de Oca, M.M., Birattari, M. and Dorigo, M. (2011) 'Parameter adaptation in ant colony optimization', *Autonomous Search*, pp.191–215, Springer, Berlin Heidelberg, ISBN 978-3-642-21434-9, doi: 10.1007/978-3-642-21434-9_8 [online] https://doi.org/10.1007/978-3-642-21434-9_8.

TSPLIB (2008) [online] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed 29 June 2017).

Tukey, J.W. (1949) 'Comparing individual means in the analysis of variance', *Biometrics*, Vol. 5, No. 2, pp.99–114, ISSN 0006341X, 15410420 [online] http://www.jstor.org/ stable/3001913 (accessed 29 April 2017).

van Leeuwen, J. (1990) *Handbook of Theoretical Computer Science*, Elsevier Science Inc., New York, NY, USA, ISBN 0444880755.

Winer, B.J. (1991) *Statistical Principles in Experimental Design*, McGraw-Hill, New York.

Yang, X-S. (2009) 'Firefly algorithms for multimodal optimization', *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, SAGA'09*, pp.169–178, Springer-Verlag, Berlin, Heidelberg, ISBN 3-64204943-5, 978-3-642-04943-9.

Yang, X-S., Deb, S., Loomes, M. and Karamanoglu, M. (2013) 'A framework for self-tuning optimization algorithm', *Neural Computing and Applications*, Vol. 23, Nos. 7–8, pp.2051–2057.

Yang, X-S., Hosseini, S.S.S. and Gandomi, A.H. (2012) 'Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect', *Applied Soft Computing*, Vol. 12, No. 3, pp.1180–1186, ISSN 1568-4946, doi: http://dx.doi.org/10. 1016/j.asoc.2011.09.017 [online] http://www.sciencedirect.com/science/article/pii/S1568494611004170 (accessed 1 June 2017).

Zitar, R.A. and Hiyassat, H. (2005) 'Optimizing the ant colony optimization using standard genetic algorithm', *Proceedings of the 23rd IASTED International Multi Conference Artificial Intelligence and Applications*, Innsbruck, Austria, 14–16 February, pp.130–133.