Simulated Annealing based Symbiotic Organisms Search Optimization Algorithm for Traveling Salesman Problem

Absalom El-Shamir Ezugwu[1]*, Aderemi Oluyinka Adewumi[1], Marc Eduard Frîncu[2]

*[1]School of Mathematics, Statistics and Computer Science, University of Kwazulu-Natal, Westville Campus, Private Bag X54001, Durban 4000, South Africa
[2]Faculties of Mathematics and Computer Science, West University of Timisoara, Timisoara, Romania

ezugwua@ukzn.ac.za, adewumia@ukzn.ac.za, marc.frincu@e-uvt.ro

Abstract. Symbiotic Organisms Search (SOS) algorithm is an effective new metaheuristic search algorithm, which has recently recorded wider application in solving complex optimization problems. SOS mimics the symbiotic relationship strategies adopted by organisms in the ecosystem for survival. This paper, presents a study on the application of SOS with Simulated Annealing (SA) to solve the well-known traveling salesman problems (TSPs). The TSP is known to be NP-hard, which consist of a set of $(n-1)!/2$ feasible solutions. The intent of the proposed hybrid method is to evaluate the convergence behaviour and scalability of the symbiotic organism's search with simulated annealing to solve both small and large-scale travelling salesman problems. The implementation of the SA based SOS (SOS-SA) algorithm was done in the MATLAB environment. To inspect the performance of the proposed hybrid optimization method, experiments on the solution convergence, average execution time, and percentage deviations of both the best and average solutions to the best known solution were conducted. Similarly, in order to obtain unbiased and comprehensive comparisons, descriptive statistics such as mean, standard deviation, minimum, maximum and range were used to describe each of the algorithms, in the analysis section. The oneway ANOVA and Kruskal-Wallis test were further used to compare the significant difference in performance between SOS-SA and the other selected state-of-the-art algorithms. The performances of SOS-SA and SOS are evaluated on different sets of TSP benchmarks obtained from TSPLIB (a library containing samples of TSP instances). The empirical analysis' results show that the quality of the final results as well as the convergence rate of the new algorithm in some cases produced even more superior solutions than the best known TSP benchmarked results.

Keywords: Symbiotic organisms search (SOS); simulated annealing (SA); traveling salesman problem (TSP); simulated annealing based symbiotic organisms search (SOS-SA).

1. Introduction

The traveling salesman problem (TSP) is an NP-hard problem, which has remained an interesting problem for a long time in the field of discrete or combinatorial optimization techniques, which are based on linear and non-linear programming. The TSP presents the task of finding an optimum path through a set of given locations (cities), such that each location is passed through only once, and the salesman returns to the start location (Durbin, 1987; Durbin *et al.*, 1989). In operational research, TSPs still remain one of the most challenging problems, which cannot be solved easily by using traditional optimization techniques such as enumeration methods and mathematical programming (Çunkaş and Özsağlam, 2009). Solving TSP optimally takes huge computational time and therefore the need for the development of fast heuristics that gives near optimal solution in a reasonable computational effort (Matai *et al.*, 2010). While on small graphs the execution time may not be significant, on large datasets containing millions of vertices and edges the limitations (e.g., traceroutes, social graphs) of existing approaches become obvious. With the emergence of the Big Data era when we deal with huge graphs with different properties (e.g., sparse, power law, dense) there is a crucial need to develop novel techniques based on new paradigms and scalable algorithms. Among possible approaches are those inspired from metaheuristics which allow for a better exploration of the solution space and faster convergence to suboptimal solutions.

In the past decades, many metaheuristic based algorithmic strategies were proposed in the quest for finding near-optimum solutions to the TSPs, among which include Tabu Search (TS) (Knox, 1994), SA (Kirkpatrick *et al.*, 1983), Genetic Algorithm (GA) (Johnson and McGeoch, 1997), Ant Colony Optimization (ACO) (Dorigo and Gambardella, 1997), Particle Swarm Optimization (PSO) (Shi *et al.*, 2007), Artificial Immune System (AIS) (Farmer *et al.*, 1986), Artificial Neural Network (ANN) (Jolai and Ghanbari, 2010), Elastic Net (EN) (Durbin *et al.*, 1989), SOS (Cheng and Prayogo, 2014).

* Corresponding author:
Email: EzugwuA@ukzn.ac.za (A. E. Ezugwu)

In this paper we **focus** on the SOS algorithm for reasons explained next. The algorithm draws inspiration from nature through the symbiotic relationships strategies, which exist among organisms in the ecosystem. The SOS algorithm was initially proposed to solve continuous engineering optimization problems. Several results (Tran *et al.*, 2016; Cheng *et al.*, 2015; Aulady, 2013; Verma *et al.*, 2015), which have used the SOS algorithm as an optimization tool to find global optimum solutions, indicate that the algorithm shows a considerable robustness in its performance when tested on complex mathematical benchmark problems. Therefore, the potential of SOS in finding global solution to the aforementioned optimization problems makes it attractive for further investigation. Furthermore, since SOS has not gained wide recognition in solving discrete problems, such as, routing and assignment problems, we believe that demonstrating its effectiveness in solving TSP could pave the way for wide scale applicability in solving complex discrete problems.

The TSP optimization problem is considered to be a large-scale optimization problem, which makes it difficult to obtain satisfactory results by just using classical metaheuristic optimization algorithms such as SA, TS, GA and ACO. Recent researches have shifted focus to employing different hybridization techniques to solve all kinds of complex large scale optimization problems. The essence of the hybridization process is mainly to utilize the complimentary advantages and value-added information found in several algorithms and insufficient in single algorithm based approaches to enhance the efficiency of solving the large-scale problem like the TSP. Two recent researches on the application of SOS to solve related discrete optimization problems for instance, have shown that the classical version of the SOS algorithm still required some level of improvement for it to achieve better solution quality, as evident in the work presented in (Yu *et al.*, 2016; Eki *et al.*, 2015). The implementation results from these researches, shows that by combining basic SOS with some other algorithms, like the local search methods or solution representation, this significantly improves the computation efficiency and quality of the solutions. Another impact of the hybrid features is that it allows the SOS algorithm to easily escape from falling into local optimum.

While the specific **objective** of this paper is to show that the hybrid SOS is a promising candidate optimization solution for the TSP, the result also emphasizes the future applicability of the SOS algorithm augmented with SA in efficiently solving a wider range of complex discrete problems. The proposed SOS-SA method was implemented in Matlab and tested using TSP data sets (*http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/*) against other state-of-the-art algorithms such as Genetic Algorithm Particle Swarm Optimization Ant Colony Optimization (GA-PSO-ACO) (Deng *et al.*, 2012), Adaptive Simulated Annealing Algorithm with Greedy Search (ASA-GS) (Geng *et al.*, 2011), Multi-agent Simulated Annealing Algorithm with Instance-Based Sampling (MSA-IBS) (Wang *et al.*, 2015), List-Based Simulated Annealing (LBSA) (Zhan *et al.*, 2016), and Improved Discrete Bat algorithm (IBA) (Osaba *et al.*, 2016). These sets of algorithms have been selected because of their common similarities in implementation techniques with the SOS-SA algorithm. Results show that the hybridized SOS-SA algorithm is able to achieve better results in solving most of the TSP benchmark problems with graphs ranging from 42 up to 33,810 cities.

The technical contributions of this paper are as follows:

i. Proposal of a new TSP optimization method, called simulated annealing based symbiotic organisms search optimization algorithm.
ii. Implementation of the proposed method using different scale of TSP benchmark instances.
iii. Performance comparison of the proposed hybrid method with other state-of-the-art algorithms (GA-PSO-ACO, ASA-GS, MSA-IBS, LBSA, and IBA).
iv. Descriptive statistical validation of the SOS-SA results against other selected methods using different statistical analysis tests.

The remainder of this paper is organised as follows: Section 2 presents the related work; Section 3 provides a short description of the TSP problem; Section 4 presents the proposed SOS-SA method of solving TSP; while Section 5 describes and discusses the simulation results carried out on some benchmarked TSP instances; finally, conclusions and directions for future research are given in Section 6.

2. Related works

The TSP being a hard combinatorial optimization problem with high social interest, has over the past decades drawn the attention of the scientific communities, with a record number of optimization algorithms being proposed to address the minimization problem (Chen and Chien, 2011;). Interested readers may refer to (Wang *et al.*, 2016; Kanda *et al.*, 2016; Sundar and Rathinam, 2016; Cornu *et al.*, 2016; Zhang and Zhou, 2016; Delgadillo *et al.*, 2016; Zhang *et al.*, 2016; Barbato *et al.*, 2016; Mohan *et al.*, 2017) for more recent results on the TSP cases. In this section we present some of the most representative results.

The TSP being an NP-hard problem, which in most cases does not admit any constant factor approximation (Garey and Johnson, 1979, except in some exceptional cases: Bender and Chekuri, 2000 and Mohan *et al.*, 2017), has resulted in the proposition of different optimization approaches, which are intended to provide solutions to the complex problem. In (Matai *et al.*, 2010) for instance, two approaches of solving TSP were identified. The first approach uses the exact methods, of which guaranty of achieving optimal solution is greatly disadvantaged by the exponential cost of execution time that scale with the problem dimension. Thus, this approach is considered unsuitable for solving large TSPs. Two common examples of this approach are the dynamic programming (Bellman, 1962), and branch and bound (Lawler and Wood, 1966; Volgenant and Jonker, 1982). The second approach is known as the approximation algorithm. This approach only gives near optimal solution, but does not guarantee optimal solution. One main advantage of this approach is that it requires minimal computational effort regardless of the problem dimension. The approximation algorithms can further be classified into local search and heuristic optimization algorithms. The local search or improvement heuristics are usually applied to improve the quality of TSP solution generated. Examples of these heuristics are the 2-Opt (Johnson, 1990), and 3-Opt (Lourenço *et al.*, 2003) exchange heuristics.

In recent years, most of the new proposed methods for solving TSP indicate shift towards improving the solution quality of the traditional based heuristics, through the development of hybrid algorithms that overcome the disadvantages of the individual algorithms. Recent studies also show that the combined efforts of two or more algorithms are usually more effective than the effort of each individual algorithm (Lin *et al.*, 2016; Tsai *et al.*, 2004; Katayama *et al.*, 2000; Talbi, 2002). This generally implies that often at times the capabilities of most hybrid algorithms are more effective and efficient than that of the individual algorithms.

Some of the existing hybrid heuristic optimization based approaches used for searching near optimal solution for TSPS outside those aforementioned in the previous section include: a hybrid of genetic algorithm particle swarm optimization ant colony optimization (GA-PSO-ACO) (Deng *et al.*, 2012), adaptive simulated annealing algorithm with greedy search (ASA-GA) (Geng *et al.*, 2011), multi-agent simulated annealing algorithm with instance-based sampling (MSA-IBS) (Wang *et al.*, 2015), list-based simulated annealing (LBSA) (Zhan *et al.*, 2016), invasive weed colony optimization (IWO) (Zhou *et al.*, 2015), mosquito host-seeking algorithm (MHSA) (Feng *et al.*, 2009), an improved discrete bat (IBA) algorithm (Osaba *et al.*, 2016), Discrete Cuckoo Search (DCS) algorithm (Ouaarab *et al.*, 2014), genetic simulated annealing ant colony system with particle swarm optimization (Chen and Chien, 2011) and the symbiotic organisms search (SOS) algorithm (Yu *et al*., 2016; Eki *et al.*, 2015). These algorithms are problem independent and have strong global search capability, while the hybrid features allows them to easily escape from falling into local optimum. Subsequently, brief reviews of the related literature are discussed.

GA-PSO-ACO (Deng et al., 2012) is an algorithm which combines the evolution ideas of the genetic algorithm, particle swarm optimization and ant colony optimization algorithm to solve the travelling salesman problem. The implementation entails applying the combination of randomicity, rapidity and wholeness of the genetic algorithm and particle swarm optimization methods to achieve a series of sub-optimal solutions. The resulting solution is later exploited by the ant colony optimization procedure, by taking the advantage of the parallel, positive feedback and high accuracy of solution to implement solving of whole problem. Osaba et al. (2016) proposed an improved discrete version of bat algorithm (IBA) for solving both symmetric and asymmetric TSP. The algorithm which was tested on 37 TSP instances produced an interesting result, which outperformed the other alternative benchmarked algorithms in most of the cases.

Geng *et al.* (2011) proposed an adaptive hybrid algorithm that combines the problem solving efforts of simulated annealing and greedy search technique (ASA-GS) to solve the TSP. The greed search technique assist in speeding up the solution convergence rate, while the hybrid algorithm achieves better trade-off between computation time and solution quality. The algorithm evaluation shows that it has good scalability and performs better even with large-scale TSP instance. A combination of multi-agent and simulated annealing with instance based sampling (MSA-IBS) was proposed by Wang *et al.* (2015) and used to solve the TSP. The hybrid process exploited the learning ability of the instance-based search algorithm to improve the sampling efficiency of the simulated annealing, the algorithm competed favourably in terms of solution quality and utilization of system resource (like cpu time) as compared to the ASA-GS algorithm. In the work of Zhan *et al.* (2016), a list-based simulated annealing algorithm was proposed also to solve the TSP. The algorithm uses the effectiveness and parameter sensitivity of the list-based cooling schedule to control temperature reduction in SA, which is used as acceptance criteria for choosing candidate solution. The simulation result of the LBSA shows that it is robust and performs fairly well compared to some other state-of-the-art algorithms.

Similar works that uses simulated annealing can be found in (Chen and Chien, 2011), where the authors proposed a new hybrid optimization method for solving the TSP. This paper consists of a hybrid of genetic simulated annealing, ant colony system, and particle swarm optimization technique. In the implementation, the ant colony system is used to generate the initial solution for the genetic algorithm's procedure, after which the initial solution is fine-tuned with the simulated annealing, which generates better solutions than the previous one. The role of the particle swarm optimization technique is to facilitate the exchange of pheromone information among the populations in the ant colony system after a predefined number of cycles. The simulation results showed that the hybrid algorithm performed better compared to the other algorithms. In (Malek *et al.*, 1998) parallel and serial version of simulated annealing and tabu search algorithms was implemented and used to solve the TSP. In (Fang *et al.*, 2007), particle swarm optimization with simulated annealing was implemented to solve TSP. The simulated annealing was applied to slowdown the degeneration of PSO swarm and to also increase the swarm's diversity. In addition, the choice of selecting the benchmarked algorithms that were compared with the proposed SOS-SA was made considering two significant characteristics; (i) population based algorithm implementations namely, GA-PSO-ACO, IBA and SOS, (ii) SA-based hybrid algorithm implementations namely, ASA-GS, MSA-IBS and LBSA.

Due to the wider interest of this area of study, summarising all the related materials available in the literatures can be a daunting task to embark on. Therefore, the interested readers are referred to the following materials for further information on TSP and its computational solution (Applegate et al., 2011, Reinelt, 1998; Jünger et al., 1995).

In this work we will show that existing algorithms such as GA-PSO-ACO, MSA-IBS, LBSA, SOS, and IBA underperform our hybrid algorithm SOS-SA which enables SOS to escape local minimums and improves in some cases some of the best results known so far.

3. Problem Formulation for TSP

The TSP is a well-known combinatorial optimization problem that has for the past decades attracted the interest of research communities. There are different solution approaches proposed in the literatures, which are currently being used to solve the three classes of the TSPs namely, the symmetric, asymmetric and multi traveling salesman problems. The TSP problems are said to be NP-hard optimization problems, which mean that there is no known polynomial time algorithm that can specifically guarantee the attainment of its optimal solution and that is why heuristic or approximation approaches remain the preferred methods often recommended for solving the TSP problems. The TSP has numerous application areas which were highlighted in (Matai *et al.*, 2010), some of which include: drilling of printed circuit boards, overhauling gas turbine engines, x-ray crystallography, computer wiring, crew scheduling, interview scheduling, mission planning, vehicle routing, mask plotting in PCB production, and design of global navigation satellite system surveying networks. In this paper, of interest is the symmetric travelling salesman problem.

The symmetric TSP can also be defined in terms of a complete undirected graph $G = (V, E)$, where the set $V = \{1, 2, \dots, n\}$ is the vertex set, $E = \{(i, j): i, j \in V, i < j\}$ is an edge set (Matai, Singh, and Mittal, 2010). A cost matrix $X = (x_{i,j})_{n\times n}$ is defined on E. The cost matrix satisfies the triangle inequality whenever $x_{i,k} + x_{j,l} \leq x_{i,l} + x_{j,k}$for all $1 \leq i < j \leq n, 1 \leq k < l \leq n$, or $x_{i,j} \leq x_{i,k} + x_{k,j}$, for all i, j, k. In particular, this is the case of planer problems for which the vertices are points $d_i = (q_i, p_i)$ in the plane, and $x_{i,j} = \sqrt{(q_i - q_j)^2 + (p_i - p_j)^2}$ is the Euclidean distance. The triangle inequality is also satisfied if x_{ij} is the length of a shortest path from i to j on G. Also, in the classical problem in combinatorial optimization (Ozcan, and Erenturk, 2004), the TSP can be defined as follows: given n cities and the distance x_{ij} between them, the shortest distance φ through all the cities can be computed by minimizing the function expressed in *Eq. 1*.

$$f(\varphi) = \sum_{i=1}^{n} x_{\varphi(i),\varphi(i+1)} + x_{\varphi(n),\varphi(1)} \tag{1}$$

where, φ a set of permutations $\varphi \rightarrow \{1,2, \dots, n\}$ with n being all the possible number of tours of the problem, and $f(\varphi)$ representing the cost of the permutation φ.

4. Simulated Annealing based Symbiotic Organisms Search (SOS-SA)

In this section, the two basic search algorithms that make up the hybrid algorithm proposed for solving the TSP problem are discussed.

4.1 Symbiotic Organisms Search Algorithm

In the real world, the close association between two or more different organisms of different species living together in an ecosystem, often but not necessarily benefits each member. When the relationship is beneficial to both organisms, it is called mutualism and symbiosis. When it is beneficial to one without effect on the other it is called commensalism, and when it is beneficial to one and detrimental to the other it is called parasitism. Almost all the metaheuristics optimization algorithms are bio-inspired from natural biological phenomena, which follow in the same trend with the symbiotic relationship explained in this section. The SOS algorithm which applies the same symbiotic relationship principles seen among organisms in nature in solving optimization problems differs greatly from other similar metaheuristic algorithms, in the sense that it does not require any algorithm-specific parameters (Cheng *et al.*, 2015). One major advantage of this is that an improper tuning related to algorithm-specific parameters would lead to an increase in computational time and premature convergence.

The algorithm is implemented by first creating a random ecosystem or population matrix, with each row (known as organism) representing a candidate solution to the corresponding problem. The size of the population often referred to as the ecosystem size (eco_size) defines the number of organisms that make up the ecosystem, a parameter usually set by the user. The search process starts immediately after the initial ecosystem has been created and it comprises of continuous interactions among the ecosystem member organisms. The interactions follow the three phases of symbioses interaction namely, mutualism, commensalism, and parasitism, which the organisms adopt to increase their survival and fitness advantage for a prolonged period of time. In the course of the interaction process, an organism would either receive a benefit or harmed, in which case the one that benefits evolve to a fitter organism whereas the one that is harmed is eliminated. Iteratively the best organism is modified and updated until the stopping criterion is reached. The SOS is implemented using the pseudocode shown in algorithm listing 1.

The classical SOS algorithm was designed to operate on real-value variables, and this would probably limits it application to discrete optimization problems, a conversion function, which converts the variables from real values to integer values is given in equation 4. This idea follows similar concept proposed in (Tran *et al.*, 2016) to make SOS suitable for application to solve the TSP. Consider a distance or cost matrix where $x_{i,j}$ is the distance of i^{th} city to j^{th} city, which is optimized by the SOS algorithm during the search process. However, the ecosystem population is created before the start of the search process and the population consist of all the feasible solutions or possible associated tour costs defined by the distance matrix expressed in *Eq. 2*.

$$X_{m\times n} = \begin{bmatrix} x_{1,1} & x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & x_{i,2} & x_{i,j} & x_{i,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix} \tag{2}$$

where m represent the ecosystem size or the problem size and n represent the number of elements in a vector of decision variables in the problem under consideration. The decision variables for the TSP, which consists of the cities and their associated costs are represented as a vector, which is expressed in *Eq. 3*.

$$X \; = \left[x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,n},\right] \tag{3}$$

Therefore, to convert the real-value variables to integer values, the function expressed in *Eq. 4* is implemented.

$$X_{i,j} = round\left\{x_{i,n} \times swap(\varphi, i+1, j)\right\} \tag{4}$$

where $x_{i,n}$ is the swapped state value, that is, the value for a particular tour through the set of given cities or points. Usually, a neighbour state is obtained by randomly swapping the order of two cities. The $swap$ function represents the total number of swap action for each tour, while n represents the number of tours. The function rounds in Matlab is used to round each point of $X_{i,j}$ to the nearest integer less than or equal to that point.

The SOS optimization strategy is performed by following three search and update phases (i.e., mutualism, commensalism, and parasitism) as presented subsequently.

Mutualism phase: In the mutualism phase, two organisms X_i and $X_j | i \neq j$ (X_j is selected randomly from the population) are considered on the bases of mutual interest. The association between X_i and X_j is to increase mutual survival of the two organisms in the ecosystem. The resulting solution $X_i'$and $X_j'$are computed as shown in *Eqs.5* and *6*:

$$X_i' = X_i + rand(0,1) \times (X_{best} - Mutual_{vect} \times K_1) \quad (5)$$

$$X_j' = X_j + rand(0,1) \times (X_{best} - Mutual_{vect} \times K_2) \quad (6)$$

The mutual vector denoted by $Mutual_{vect}$ is expressed as shown in *Eq. 7.*

$$Mutual_{vect} = \frac{X_i + X_j}{2} \quad (7)$$

The $rand$ (0,1) function is a vector of uniformly distributed random numbers between 0 and 1. The values of the benefit factors K_1 and K_2 are determined randomly as either 1 or 2, and represents the level of benefit to each of the two organisms $X_i$and X_j (where 1 and 2 denotes adequate and huge benefit that can be received by both X_i and X_j in their current mutual symbiotic states). The organism with the best objective or fitness function value in terms of the degree of adaptation in the ecosystem is represented by X_{best}. The $Mutual_{vect}$, signifies mutualistic characteristics exhibited between the two organism to increase their survival advantage. It should be noted that any update for any one of the two organisms is computed only if its new fitness function value denoted by $f(X_i')$ or $f(X_j')$ is better than the previous solutions, $f(X_i)$ and $f(X_j)$. Given the above *Eqs. 5* and *6* become:

$$X_i' = X_i + rand(0,1) \times (X_{best} - Mutual_{vect} \times K_1), \quad if \quad f(X_i') > f(X_i) \quad (7)$$

$$X_j' = X_j + rand(0,1) \times (X_{best} - Mutual_{vect} \times K_1), \quad if \quad f\left(X_j'\right) > f\left(X_j\right) \quad (8)$$

Commensalism phase: In this phase, the organism X_i selected randomly from the ecosystem strives to increase its benefits from its association with X_j. This kind of symbiotic association only places X_i at an advantage position, over X_j, even though, X_j is not harmed in the process. The new solution emanating from the symbiotic relationship is calculated as shown in *Eq. 9*:

$$X_i' = X_i + rand(-1,1) \times \left(X_{best} - X_j\right) \; if \quad f(X_i') > f(X_i) \quad (9)$$

Parasitism phase: Also in Cheng and Prayogo (2014), an example of parasitic symbiotic relationship was illustrated by using the association that exists among three organisms, the plasmodium parasite, anopheles mosquito and the human host. In this kind of association, the human host is harmed, the anopheles mosquito, which is the parasite carrier, is left unharmed, while the plasmodium parasite thrives and reproduces inside the human body. In the SOS model, by mimicking the aforementioned parasitic symbiotic behaviours, X_i is assigned a role akin to the anopheles mosquito through the creation of an artificial vector (or parasite vector) P_{vec} in the search space, by fine-tuning the randomly selected dimension of organism X_i. Then, the organism X_j is selected randomly from the ecosystem and serve as host to P_{vec}. Then, P_{vec}will try to replace X_j in the ecosystem. If P_{vec} has a better fitness value than X_j, then X_j is replaced by P_{vec}, otherwise, $X_j$develops an immunity from P_{vec}, which will invariably cease to exist in the ecosystem. The procedure for the classical SOS algorithm proposed by Cheng and Prayogo (2014) is presented in the algorithm listing 1 below.

Algorithm 1: SOS pseudocode

```
Input: Initial ecosystem $X$, ecosystem size $eco\_size$, maximum iteration $maxitr$
Output: best solution $X_{best}$
1: For counter = 1 to maxitr
2: For each organism in the ecosystem $X_i, i = 1,2,\ldots,eco\_size$
3: Search of the best organism $X_{best}$
4: Update organism by
      a) Mutualism phase
      b) Commensalism phase and
      c) Parasitism phase
5: End for
6: End for
```

The SOS algorithm though efficient in solving complex optimization and discrete engineering problems, still has high probability of plunging into local optimum (Vincent, *et al.*, 2016). Therefore, the SOS-SA algorithm has been proposed to overcome this shortcoming.

4.2 Simulated Annealing Algorithm

The application of SA to solve TSP was first introduced by Kirkpatrick *et al.* (1983). The process begins by considering a solution space S of a particular tour through the set of given cities or points $X_i | i = 1,2,\dots,n$, with an update solutions X_i' created by randomly switching the orders of two cities. The energy function or fitness function, which represents the length of route X_i, is denoted by $f(X_i)$. The relative change in cost Δf between X_i and X_i' is expressed as $\Delta f = \frac{f(X_i') - f(X_i)}{f(X_i)}$. Beginning with the initial solution, only the solution which results in smaller energy value than the previous solution is accepted by the algorithm, in other words, a solution is only accepted when the fitness value of $f(X_i') < f(X_i)$. However, accepting or rejecting a new solution with higher fitness values for X'can be based on the acceptance probability function given as follows (*Eq. 10*):

$$P(\Delta f, T_k) = \begin{cases} e^{\left(\frac{-\Delta f}{T_k}\right)}, & \Delta f > 0 \qquad for\ T_k > 0 \\ 1, & \Delta f \le 0 \end{cases} \tag{10}$$

where T_k is the parameter temperature at the k^{th} instance of accepting a new solution route, and for any given T, for $\Delta f > 0$, P is greater for smaller values of Δf, which means that for the new solution X_i' that is only slightly more costly than the current solution X_i is more likely to be accepted than the new solution X_i' that is much more costly than the current solution X_i. The value of T, which is an important control parameter, decreases proportionally with P, that is as the $\lim_{T \to 0^+} e^{\left(\frac{-\Delta f}{T_k}\right)} = 0, \Delta f > 0$. Therefore, as the value of T decreases, the probability of accepting a degraded route also decreases. In this paper the following cooling schedule is adopted (*Eq. 11*):

$$T_{k+1} = \alpha T_k \tag{11}$$

Where, α denotes the cooling coefficient, which is some random constant values between 0 and 1, it is also the rate at which the temperature is lowered each time a new solution X_i' is discovered. The SA procedure is as presented in the algorithm listing 2 below:

Algorithm 2: Pseudocode for SA

```
Input: Initial temperature $T_0$, final temperature $T_k$, cooling rate $\alpha$, maximum iteration $maxiter$
Output: Best cost
1: Chose a random route $X_i$ and initialize $T_0$ and $\alpha$
2: For counter=1 to $maxiter$
3: Create a new solution $X_i'$ by randomly swapping two cities in neighbourhood of $X_i$
3: Compute $\Delta f = \frac{f(X_i') - f(X_i)}{f(X_i)}$ and use the acceptance probability function to either accept or reject the new
solution, based on the following conditions:
    a) if $\Delta f \le 0$, then $X_i \leftarrow X_i'$
    b) if $\Delta f > 0$, then $X_i \leftarrow X_i'$ depending on Eq. (10)
4: Reduce the temperature using Eq. (11) and increment $k$
5: Update the best solution
6: End for
```

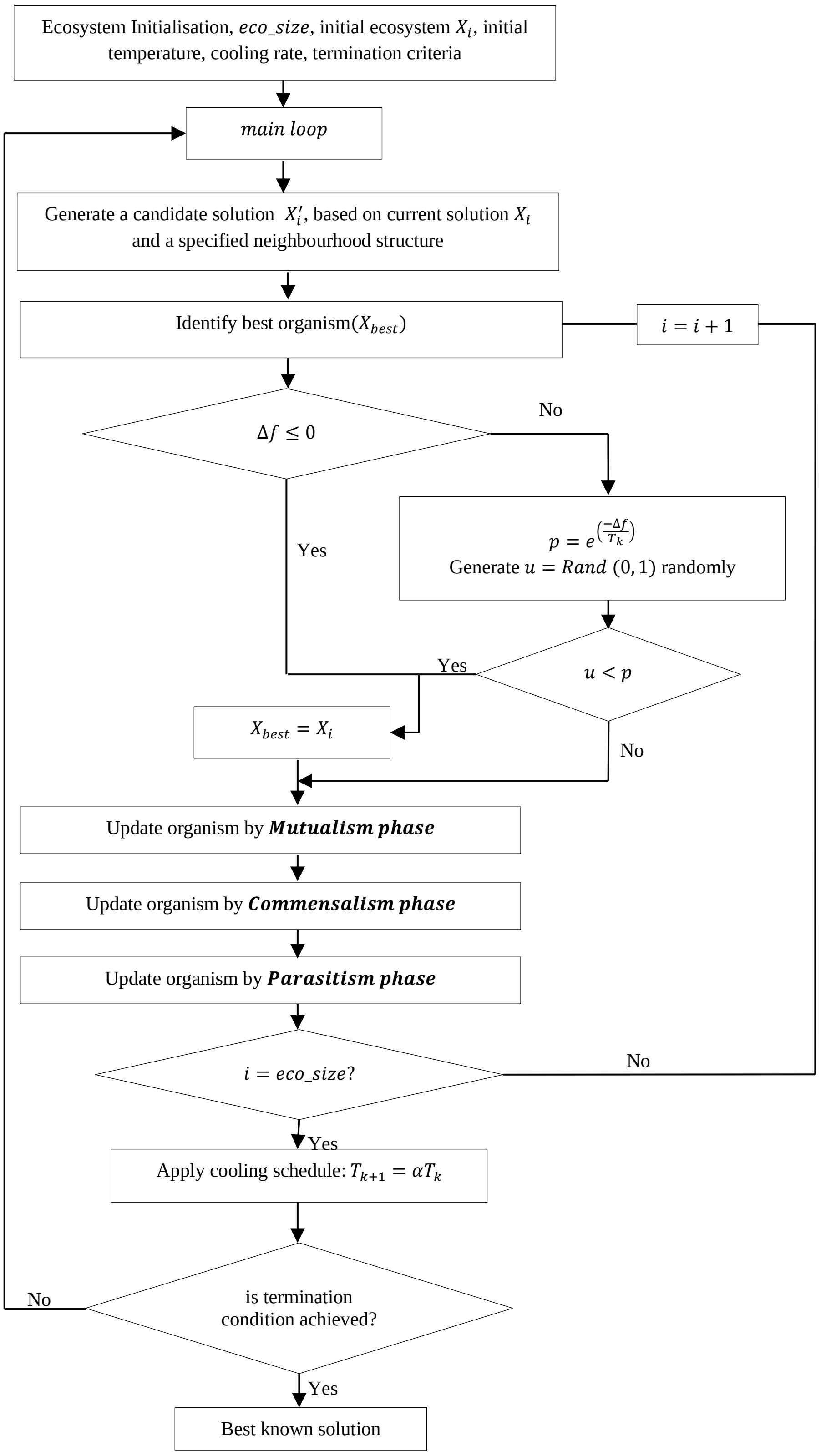
4.3 SOS-SA Framework for Solving TSP

The SOS-SA algorithm is a hybrid of symbiotic organisms search and simulated annealing algorithm. The SA is a local search metaheuristic algorithm widely used for solving both discrete and continuous optimization problems (Kirkpatrick *et al.*, 1983). One of the main benefits of SA lies in its ability to escape the problem of getting stuck in a local minimum by allowing hill-climbing moves to search for a global solution. Therefore, a hybrid approach is proposed by introducing SA is to assist the SOS in avoiding being trapped into local minimum and to also increase its level of diversity while searching for optimum solution in the problem search space. Exploiting the fast optimal search capability of the SOS algorithm with the hill-climbing probability jump property of the SA, as described in algorithm listing 1 and 2 above, a new hybrid algorithm (SOS-SA) is proposed to solve the TSP problem. The steps of the hybrid SOS-SA algorithm are then described in algorithm listing 3.

Algorithm 3: SOS-SA pseudocode

```
Input: Initial ecosystem $X$, ecosystem size $eco\_size$, Initial temperature $T_0$, final temperature $T_k$, cooling rate $\alpha$,
          maximum iteration $maxiter$,
Output: best known solution $X_{best}$
1: Create and evaluate new solutions
     a)  Generate $X_i, i = 1,2, \dots, eco\_size$
         $\boldsymbol{For}\ i = 1\ to\ maxiter$
     b)  Compute cost / fitness function of $X_i, f(X_i)$
     c)  Determine the best solution $X_{best}$
     d)  Compute $\Delta f = \frac{f(X_i') - f(X_i)}{f(X_i)}$
         If $\Delta f \leq 0$ or $p > u$, where $p$ is the acceptance probability (Eq, 10) and $u$ is a random number
         between 0 and 1
     e)  then update solution by assigning $X_{best} \leftarrow X_i$
     f)  End if
         $\boldsymbol{For}\ i = 1\ to\ eco\_size$
2: Update organism (route) with SA (Algorithm 1) on the three SOS phases in Algorithm 2
     $\boldsymbol{For}\ i = 1\ to\ eco\_size$
        $a)Mutualism\ phase$
        $b)Commensalism\ phase$   } The three SOS phases are applied to optimize the search process
        $c)Parasitism\ phase$
3: Update the best solution $X_{best}$ ever found
4: Update temperature using the cooling schedule given in Eq. (11)
5: End for
6: End for
7: End for
```

The SOS-SA algorithm follows through all the steps highlighted in algorithm 3, starting by first initializing the ecosystem X_i of size eco_size. Then creating and evaluating each new organism's positions by computing and comparing their respective tour cost functions, such that the organism with the least tour cost is selected as X_{best}. Iteratively, the process is repeated by updating the current solution with the best solution ever found until the organism with the global best solution is discovered. The SOS-SA algorithm uses the three SOS relationships' phases to update the organism. The algorithm finishes when maximum iterations criterion is attained. Otherwise, the algorithm continues to calculate new positions. However, stopping condition is quite an important factor that can determine the final result of the simulation. For example, if the algorithm is stopped too early, the approximation of the solution might not be even close to the targeted global optimum, and prolonging the simulation incurs unnecessary huge amount of computational effort and time. A fixed generation number of 1,000 was set as the stopping condition for the simulation and this setting was adequate, as it limits unproductive work. Fig. 1 illustrates the SOS-SA algorithm procedures.



Ecosystem Initialisation, eco_size, initial ecosystem X_i, initial temperature, cooling rate, termination criteria
$main\ loop$
Generate a candidate solution X_i', based on current solution X_i and a specified neighbourhood structure
Identify best organism(X_{best})
$i = i + 1$
$\Delta f \leq 0$
No
Yes
$p = e^{\left(\frac{-\Delta f}{T_k}\right)}$
Generate $u = Rand\ (0, 1)$ randomly
Yes
$u < p$
No
$X_{best} = X_i$
Update organism by $\boldsymbol{Mutualism\ phase}$
Update organism by $\boldsymbol{Commensalism\ phase}$
Update organism by $\boldsymbol{Parasitism\ phase}$
$i = eco\_size$?
No
Yes
Apply cooling schedule: $T_{k+1} = \alpha T_k$
No
is termination condition achieved?
Yes
Best known solution

Fig.1. Flowchart for the SOS-SA algorithm

5. Simulation and Result

The TSP experimental data sets used in this paper were obtained from the MP-TESTDATA, which covers: The TSPLIB Symmetric Traveling Salesman Problem Instances[2] and Best known solutions for symmetric TSPs[3]. In order to evaluate the performance of the SOS-SA algorithm, two sets of simulation experiments were conducted.

The first experiment was carried out to evaluate the performance of the proposed SOS-SA algorithm, GA-PSO-ACO, MSA-IBS, and LBSA, against the best known solution from the TSPLIB, on six (6) benchmarked TSP results and their results as presented in Table 2.Similar comparisons were also carried out for SOS-SA, SOS, GA-PSO-ACO, MSA-IBS, and LBSA, to compare the quality of solutions for each algorithm. The comparison results are as presented in Tables 3, 4, 5, 6, 7, 8 and 9.

The second experiment was carried out to evaluate the computational effort of the SOS-SA algorithm based on execution time, on 40 benchmarked TSP instances and the results are presented in Table 10. In Table 2, the boldly highlighted column indicate those areas where the SOS-SA competed and performed favourably with the best known solution of the optimal TSPLIB instance and the other three state-of-the-art algorithms. Several statistical tests were also conducted to validate the obtained results, as depicted in Tables 11 (on 40 benchmarked TSP instance), 12 (on 20 benchmarked TSP instance), and 13 (on 16 benchmarked TSP instance). The average of 20 trials was taken and the number of the outer iteration times was set to 1,000 and 2,000 respectively.

5.1 Experimental configurations

The experimental results presented in this section demonstrate the scalability, effectiveness and efficiency of the SOS-SA in solving different TSP instances ranging from 42 up to 33,810 cities. The simulation time and number of iterations used to solve the TSP instances on a single machine are similarly presented here. The experimental testing platform for the SOS-SA and SOS algorithms were conducted on a 2.83GHz CPU Desktop with 2GB RAM, while the implementation software is Matlab R2014b. For the implementation of MSA-IBS and LBSA algorithms, the experiments were run on a 2.8GHz PC with 2GB RAM, ASA-GS was run on 2.83 GHz PC, and GA-PSO-ACO was run on Intel Core i52410 Laptop with 2.30 GHz and 4GB RAM. For the entire TSP instance tested, the maximum iterations for the outer loop were set to 1,000, which correspond to the iteration times set for the other compared algorithms. In the case of other algorithms, the selection population size is dependent on the scale of problem instance, which is the case with SOS-SA and SOS algorithms. As stated in (Zhan *et al.*, 2016), the algorithm execution stopping condition is either when an optimal solution is found or when the iteration times of the outer loop reaches 1,000.

Since parameter selection may significantly influence the solution's quality of each algorithm performance, the parameter settings for all the simulations conducted are presented in Table 1.

Table 1: Experimental parameter configuration

SOS-SA Parameters	SOS Parameter
Population size= 50, 100 Maximum iteration= 1000, 2000 Initial temperature = 0.025 Cooling rate = 0.99 Number of cities to swap = 2	Population size= 100 Maximum iteration= 2000 Initial temperature = 0.025 Cooling rate = NA Number of cities to swap = NA
LBSA Parameters	
Population size=100 Maximum iteration = 1000 Inertia temperature = produced according to initial acceptance probability $\boldsymbol{p_0}$ in the range of $\mathbf{10^{-20}}$ to 0.9 Cooling rate is adaptively selected as follows: $\boldsymbol{t_i = \frac{-d_i}{\ln(r_i)}}$, where $\boldsymbol{r}$ is a random number and $\boldsymbol{d}$ is the difference of objective function values. Number of cities to swap = 2	
ASA-GS Parameter	**MSA-IBS**
Population size = NA Maximum iteration= 1000 Initial temperature = 1000 Cooling rate $= ((\boldsymbol{\alpha \times N^\wedge 0.5 - 1})) / (\boldsymbol{\alpha \times N^\wedge 0.5})$, N=no. of cities & $\boldsymbol{\alpha = 1}$ Number of cities to swap = NA	Population size = NA Maximum iteration= NA Initial temperature = NA Cooling rate = NA Number of cities to swap = NA
IBA	**GA-PSO-ACO**

[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/

[3] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html

Population size= 50 Maximum iteration= 1000 Initial temperature = NA Cooling rate = NA Number of cities to swap = 2-opt & 3-opt	Population size= 100 Maximum iteration= 1000 Initial temperature = NA Cooling rate = NA Number of cities to swap = NA

5.2 Evaluation

Table 2, describes the summary of the best known results so far obtained using the SOS-SA algorithm. Where the second column represents the name of the TSP instance, the third column represents the best known solution length taken from the TSPLIB, the fourth column represents the length of the best known solution found by SOS-SA algorithm, the fifth column represents the average length of the solution found by SOS-SA algorithm, and the sixth column represents the percentage deviation of the SOS-SA best solution. The SOS-SA best solution percentage deviation (PDbest), which determines the closeness of the solution to the best known solution (BKS), is calculated as shown in *Eq. 12*:

$$PDbest = \frac{(Best - BKS)}{BKS} \times 100 \tag{12}$$

where $Best$ denotes the best length value for each algorithm for the total number of runs under each problem instance.

The percentage deviation of the SOS-SA mean solution was also computed and used to compare it performance with the best known solution and other algorithms. The percentage deviation of the mean solution is subsequently defined as follows:

$$PDmean = \frac{(Mean - BKS)}{BKS} \times 100 \tag{13}$$

where $Mean$ denotes the average length value for each algorithm for the total number of runs under each problem instance.

5.3 Discussion of results

Table 2 demonstrates the extreme performance capability of the SOS-SA algorithm in comparison with other state-of-the-art algorithms. The new algorithm outperformed all the three algorithms and this include the best known solution so far "BKS" in the six TSP instance examined with percentage deviation $(PDbest) < 0$ and performance accuracy that is above 100%. Though the SOS-SA share some common characteristics with GA-PSO-ACO based on the SOS component, and also due to the fact that each algorithm has a strong global search capability, the GA-PSO-ACO still has very strong tendency of falling into a local minimum, for which the SA component in the SOS-SA is able to prevent. As can be seen from the results shown in Table 2, MSA-IBS and LBSA competed relatively and favourably with over 99% performance accuracy against the best known solution in all the instances. The possible challenge with the two algorithms can be traced to few factors, some of which include the use of several parameters, high computation and communication cost incurred during the iterative execution of the algorithm, especially for the multi-agent based MSA-IBS algorithm. Fig. 2 illustrates the performance evaluation of the SOS-SA algorithm among other three algorithms including the BKS based on computed percentage deviation.

Table 2: Best-so-far solutions found by SOS-SA algorithm compared with the beset know solution from TSPLIB and other algorithms, the best results are highlighted in bold.

S/N	Instance	BKS[a]	GA-PSO-ACO [**Deng *et al.*, 2012**]				MSA-IBS [**Wang *et al.*, 2015]**				LBSA [**Zhan et al., 2016**]				SOS-SA			
			Best[b]	Mean	Diff.	PDbest (%)[c]	Best	Mean	Diff.	PDbest (%)	Best	Mean	Diff.	PDbest (%)	Best	Mean	Diff.	PDbest (%)
1	dantzig42	699	NA		NA	NA	NA		NA	NA	NA		NA	NA	**679**	699.4823	-20	-2.86
2	Berlin52	7542	7544.37	7544.37	2.37	0.03	7542	7542	0	0	7542	7542	0	0	**7540**	7541.107	-2	-0.03
3	Pr76	108159	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	**107899**	107899	-260	-0.24
4	Rat99	1211	1218	1275	7	0.578	1211	1211.04	0	0	1211	1211.1	0	0	**1210**	1210.108	-1	-0.08
5	Pr107	44303	44316	44589	13	0.029	44303	44379.88	0	0	44303	44392.25	0	0	**44301**	44302.83	-2	-0.01
6	Pr124	59030	59051	60157	21	0.036	59030	59032.88	0	0	59030	59031.8	0	0	**58985**	59010.65	-45	-0.08

[a]Best know solution so far or the theoretical value [http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html]
[b]Best known solution for each of the algorithms
[c]Relative percentage error for the results obtained by 10 runs.

The negative signs against the difference (Diff.) values and percentage deviations ($PDmean$ and $PDbest$) in Table 2 and other Tables are left as indication to show that the SOS-SA solution outperformed in some cases the best known solution, although percentage deviation is supposed to be an absolute value.

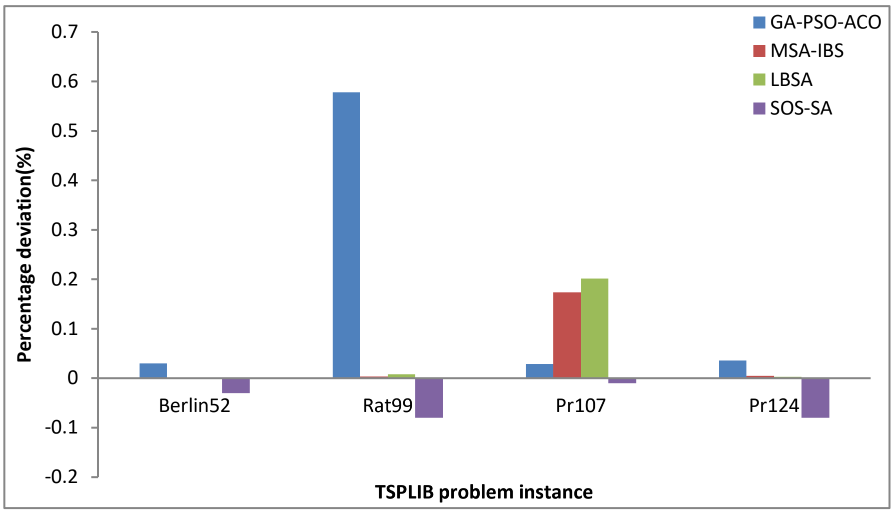


Fig.2. SOS-SA best solution percentage deviations compared with GA-PSO-ACO, MSA-IBS, and LBSA.

Tables 3, 4, 5, 6, 7, 8, and 9, demonstrate the comparisons among GA-PSO-ACO, MSA-IBS, LBSA, SOS, and SOS-SA algorithms on seven (7) different TSP instances. The comparisons are based on the quality of results produced by each of the algorithms. Also, as earlier stated, negative value means that the SOS-SA solution is better than the BKS solution.

Table 3: Algorithms comparison for Berlin52

Algorithm	**Scale**	**BKS**	**Best**	**Mean**	**Diff.**	**PDbest (%)**	**PDmean(%)**
GA-PSO-ACO [Deng et al., 2012]	52	7542	7544.37	7544.37	2.37	0.0314	0.0314
MSA-IBS [Wang et al., 2015]	52	7542	7542	7542	0	0	0
LBSA [Zhan et al., 2016]	52	7542	7542	7542	0	0	0
SOS	52	7542	7647	7659.48	104.68	1.3880	1.5577
SOS-SA	52	7542	**7540**	7541.12	-2	-0.0265	-0.0117

Table 4: Algorithms comparison for Rat99

Algorithm	**Scale**	**BKS**	**Best**	**Mean**	**Diff.**	**PDbest (%)**	**PDmean(%)**
GA-PSO-ACO [Deng et al., 2012]	99	1211	1218	1275	7	0.5780	5.2849
MSA-IBS [Wang et al., 2015]	99	1211	1211	1211.04	0	0	0.0033
LBSA [Zhan et al., 2016]	99	1211	1211	1211.1	0	0	0.0083
SOS	99	1211	1284	1297.381	73	6.0281	7.1330
SOS-SA	99	1211	**1210**	1210.108	-1	-0.0826	-0.0737

Table 5: Algorithms comparison for Pr107

Algorithm	**Scale**	**BKS**	**Best**	**Mean**	**Diff.**	**PDbest (%)**	**PDmean(%)**
GA-PSO-ACO [Deng et al., 2012]	107	44303	44316	44589	13	0.0293	0.6456
MSA-IBS [Wang et al., 2015]	107	44303	44303	44379.88	0	0	0.1735
LBSA [Zhan et al., 2016]	107	44303	44303	44392.25	0	0	0.2015
SOS	107	44303	46097	46112.22	1794	4.0494	4.0837
SOS-SA	107	44303	**44301**	44302.83	-2	-0.0045	-0.0004

Table 6: Algorithms comparison for Pr124

Algorithm	**Scale**	**BKS**	**Best**	**Mean**	**Diff.**	**PDbest (%)**	**PDmean(%)**
GA-PSO-ACO [Deng et al., 2012]	124	59030	59051	60157	21	0.0004	1.9092
MSA-IBS [Wang et al., 2015]	124	59030	59030	59032.88	0	0	0.0049
LBSA [Zhan et al., 2016]	124	59030	59030	59031.80	0	0	0.0030
SOS	124	59030	68942	69211.12	9912	0.1679	17.2474
SOS-SA	124	59030	**58985**	59010.65	-45	-0.0008	-0.0328

Table 7: Algorithms comparison for Rat575

Algorithm	Scale	BKS	Best	Mean	Diff.	PDbest (%)	PDmean(%)
GA-PSO-ACO [Deng et al., 2012]	575	6773	6912	6952	139	2.0523	2.6428
MSA-IBS [Wang et al., 2015]	575	6773	6819	6854.64	46	0.6792	1.2054
LBSA [Zhan et al., 2016]	575	6773	6829	6847.95	56	0.8268	1.1066
SOS	575	6773	7018.22	6991.92	245.22	3.6206	3.2322
SOS-SA	575	6773	6773	6802.04	0	0	0.4288

Table 8: Algorithms comparison for Rat783

Algorithm	Scale	BKS	Best	Mean	Diff.	PDbest (%)	PDmean(%)
GA-PSO-ACO [Deng et al., 2012]	783	8806	9030	9126	224	2.5437	3.6339
MSA-IBS [Wang et al., 2015]	783	8806	8897	8918.28	91	1.0334	1.2750
LBSA [Zhan et al., 2016]	783	8806	8887	8913.25	81	0.9198	1.2179
SOS	783	8806	9884.37	9906.88	1078.37	12.2459	12.501
SOS-SA	783	8806	8806	8817.91	0	0	0.1353

Table 9: Algorithms comparison for Pr1002

Algorithm	Scale	BKS	Best	Mean	Diff.	PDbest (%)	PDmean(%)
GA-PSO-ACO [Deng et al., 2012]	1002	259045	265987	266774	6942	2.6798	2.9837
MSA-IBS [Wang et al., 2015]	1002	259045	261463	262211.7	2418	0.9334	1.2225
LBSA [Zhan et al., 2016]	1002	259045	261490	262202.5	2445	0.9439	1.2189
SOS	1002	259045	280169.68	281140.08	21124.68	8.1548	8.5294
SOS-SA	1002	259045	261491	262301.8	2446	0.9442	1.2572

In Table 10, the computational cost for the four algorithms are given in the last column under the title 'Time', and it's clear that the SOS-SA has the least convergence time frame compared to the other three algorithms. Generally, in terms of the convergence time, it can be argued that the SOS-SA is more successful than the remaining three algorithms, considering for example: the TSP instance 'Rd400' where it takes SOS-SA 2.71 seconds to converge, while in the case of the other three, it took 30.4 seconds for ASA-GS, 3.2 seconds for MSA-IBS, and 3.46 seconds for LBSA respectively. However, there are instances where the SOS-SA is outperformed by both LBSA and MSA-IBS, and this could be attributed to the deep explorative and exploitative capability of the SOS component in SOS-SA algorithm that would at times incur additional cost in finding global best tour routes. One example of this case can be seen in the problem instance 'D1291' where MSA-IBS with convergence time of 10.59 outperformed both LBSA and SOS-SA algorithms, with each having convergence time of 11.77 and 12.08 respectively. This exceptional behaviour exhibited at times by MSA-IBS can be traced to some level of intelligence acquired from the learning-based sampling process, which can effectively improve the performance of the SA's sampling efficiency. Figs. 3-5 show the different convergence rates for the four algorithms with regards to varying TSP problem lengths.

Table 10: ASA-GS, MSA-IBS, LBSA, and SOS-SA convergence times and speed comparison, the results are the average of 1000 executions. The best results are highlighted in bold.

S/N	Instance	BKS	**ASA-GS** [Geng *et al.,* 2011]		**MSA-IBS** [Wang *et al.*, 2015]		**LBSA** [Zhan *et al.,* 2016]		**SOS-SA**	
			Mean	**Time**	**Mean**	**Time**	**Mean**	**Time**	**Mean**	**Time**
1	Ch150	6528	6539.8	10.91	6529	0.86	6529.8	1.29	6529.8384	1.0296
2	Kroa150	26524	26538.6	10.9	26524	0.82	26524	0.98	26524.0176	**0.7020**
3	Krob150	26130	26178.1	10.9	26135	1.51	26137	1.65	26131.8331	**1.1700**
4	Pr152	73682	73694.7	10.85	73682	0.84	73682	0.87	73682.1801	**0.8268**
5	U159	42080	42398.9	11.49	42080	0.79	42080	0.91	42080.9819	0.8736
6	Rat195	2323	2348.05	14.37	2330.2	1.86	2328	1.93	2326.5979	**1.1856**
7	D198	15780	15845.4	14.6	15780	1.39	15780	1.53	15782.1061	**1.0452**
8	Kroa200	29368	29438.4	14.26	29378	1.74	29373.8	1.67	29370.7811	**1.2792**
9	Krob200	29437	29513.1	14.24	29439.8	1.95	29442.2	2.1	29449.8182	**1.5132**
10	Ts225	126643	126646	16.05	126643	1.3	126643	1.54	126701.0841	1.4196
11	Pr226	80369	80687.4	16.7	80369	1.93	80369.8	2.16	80369.3077	**1.5444**
12	Gil262	2378	2398.61	19.43	2378.8	2.39	2379.2	2.72	2381.9145	**2.0387**
13	Pr264	49135	49138.9	19.09	49135	1.43	49135	1.49	49135.7188	1.5056
14	Pr299	48191	48326.4	21.94	48226.4	2.67	48221.2	2.93	48227.9301	**2.3444**
15	Lin318	42029	42383.7	23.35	42184.4	2.4	42195.6	2.58	42179.3111	2.6121
16	Rd400	15281	15429.8	30.4	15347.2	3.2	15350.4	3.46	15451.8108	**2.7114**
17	Fl417	11861	12043.8	32.02	11875.6	3.72	11867.8	4.01	11877.5194	3.9410
18	Pr439	107217	110226	34.92	107407.2	3.6	107465.2	3.95	107561.1441	**3.1801**
19	Pcb442	50778	51269.2	35.75	50970	3.68	50877	4.31	50871.8228	4.4017
20	U574	36905	37369.8	48.47	37155.8	5.21	37164.6	6.13	37164.4871	6.6099
21	Rat575	6773	6904.82	52.1	6839.8	5.27	6837.4	5.99	6839.5194	**5.1184**
22	U724	41910	42470.4	66.83	42212.2	8.11	42252	8.34	42262.1108	**7.9999**
23	Rat783	8806	8982.19	78.9	8893.4	8.99	8888.2	8.9	8899.5507	**8.6130**
24	Pr1002	259045	264274	164.42	261481.8	12.71	261805.2	12.96	261802.4892	12.8141
25	Pcb1173	56892	57820.5	193.08	57561.6	8.9	57431.8	9.61	57569.9388	**8.7301**
26	D1291	50801	52252.3	214.64	51343.8	10.59	51198.8	11.77	51291.0871	12.0816
27	Rl1323	270199	273444	210.16	271818.4	11.53	271714.4	12.64	271710.6288	**11.0188**
28	Fl1400	20127	20782.2	232.02	20374.8	17.72	20249.4	15.43	20231.0177	**14.7381**
29	D1655	62128	64155.9	281.88	62893	16.18	63001.4	16.45	64111.9201	16.1902
30	Vm1748	336556	343911	276.98	339617.8	19.7	339710.8	19.05	336719.3891	**18.2714**
31	U2319	234256	236744	410.97	235236	17.02	235975	18.94	235338.0944	18.1111
32	Pcb3038	137694	141242	554.28	139706.2	27.64	139635.2	29.05	139701.8133	**25.6712**

33	Fnl4461	182566	187409	830.9	185535.4	30.43	185509.4	29.67	185546.0411	32.7422
34	Rl5934	556045	575437	1043.95	566166.8	50.76	566053	52.5	566211.7184	**49.9871**
35	Pla7397	23260728	24166453	1245.22	2.38E+07	100.69	2.38E+07	89.61	2.38E+07	**98.7222**
36	Usa13509	19982859	20811106	2016.05	2.04E+07	365.12	2.04E+07	326.76	2.14E+07	**313.1080**
37	Brd14051	469385	486197	2080.5	478609.6	375.28	478010	369.86	478098.9076	370.8801
38	D18512	645238	669445	2593.97	658149.2	654.85	657457.2	629.14	659457.4512	**601.8544**
39	Pla33810	66048945	69533166	4199.88	68075607	1959.68	68029226.4	1998.19	68076220.2281	**1899.9919**
40	Pla85900	142382641	156083025	8855.13	146495515.6	7596.18	145526542.6	7586.6	146429581.1412	7591.1833
		Average		**650.31**		**283.52**		**282.49**		**278.99403**

Overall, the SOS-SA performed better than the remaining three algorithms in 62.5% of the computed graphs ranging from 150 up to 85,900 cities (i.e., 25 out of 40 instances).

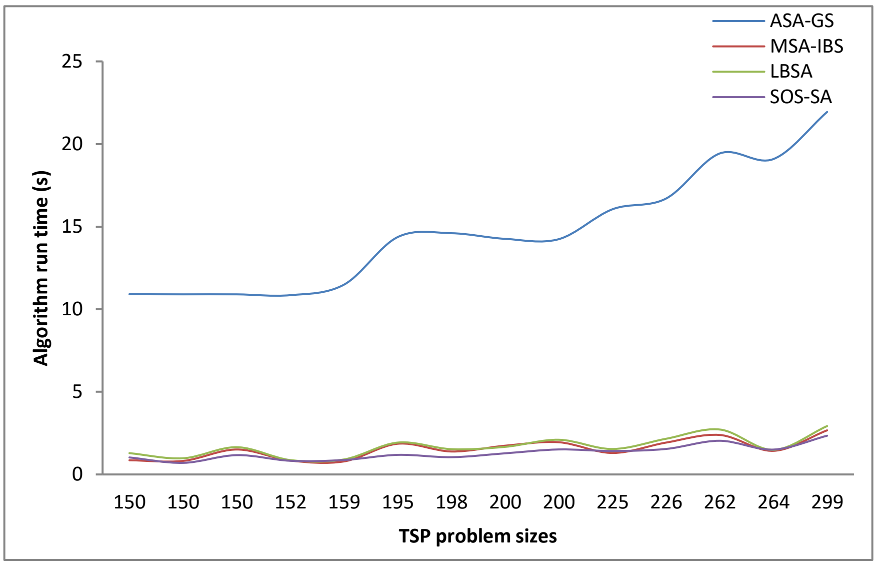


Fig.3. ASA-GS, MSA-IBS, LBSA, and SOS-SA time comparison, for TSP with graphs ranging from 150 up to 299 cities

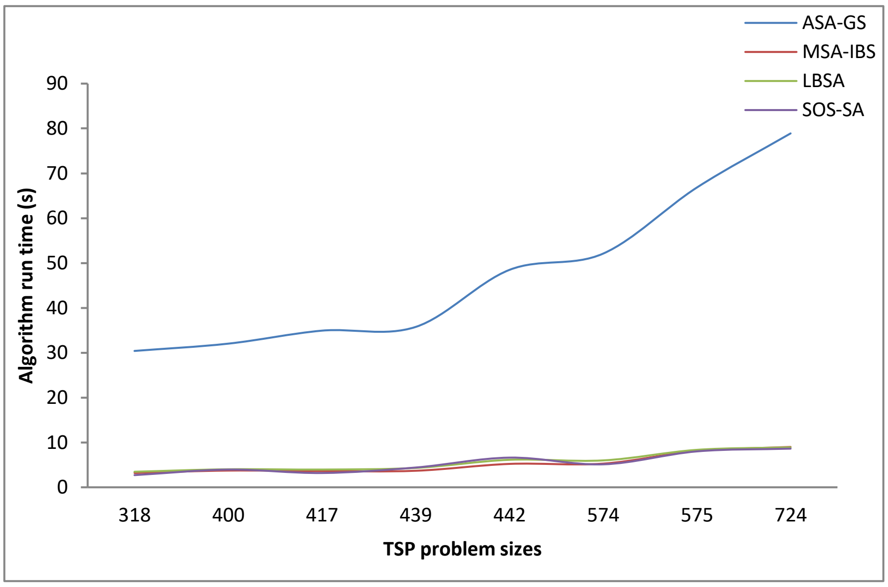


Fig.4. ASA-GS, MSA-IBS, LBSA, and SOS-SA time comparison, for TSP with graphs ranging from 318 up to 724 cities.

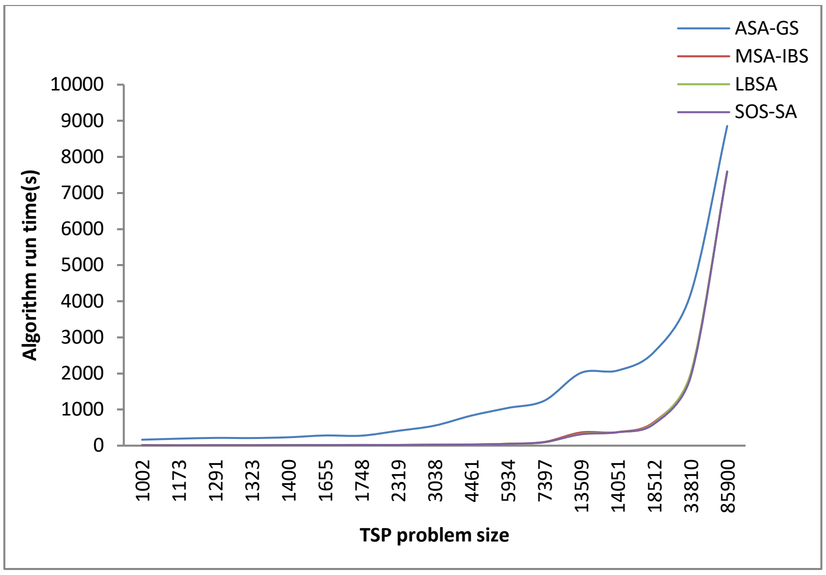


Fig.5. ASA-GS, MSA-IBS, LBSA, and SOS-SA time comparison, for TSP with graphs ranging from 1002 up to 85,900 cities.

In Table 11, the percentage deviation of the SOS-SA for the 35 TSP instance considered was computed to be 0.2645, which is significantly better than the 1.2929 of GA-PSO-ACO, 0.3385 of MSA-IBS, and 0.3229 of LBSA respectively. Therefore, this verifies the fact that the performance of the SOS-SA algorithms competes favourably with the state-of-the-art TSP algorithms and in some instance with the best known solution from the TSPLIB problem instances.

Table 11: GA-PSO-ACO, MSA-IBS, LBSA, and SOS-SA comparison on 35 benchmarked TSPLIB tour instances for 20 trials. The columns shows the best solution found, the average solution and the relative percentage error computed using Eq. (10). The best results so far are highlighted in bold.

S/N	**Instance**	**BKS**	**GA-PSO-ACO** [Deng *et al.*, 2012]			**MSA-IBS** [Wang *et al.*, 2015]			**LBSA** [Zhan *et al.*, 2016]			**SOS-SA**		
			Best	Mean	PDbest (%)	Best	Mean	PDbest (%)	Best	Mean	PDbest (%)	Best	Mean	PDbest (%)
1	Att48	33522	33524	33662	0.0060	33522	33554.64	0	33522	33536.6	0	33523	33539.68	0.0030
2	Eli51	426	426	431.84	0	426	426.48	0	426	426.5	0	426	426	0
3	Berlin52	7542	7544.37	7544.37	0.0314	7542	7542	0	7542	7542	0	**7540**	7541.107	-0.0265
4	St70	675	679.6	694.6	0.6815	675	677.16	0	675	675.55	0	675	675	0
5	Eil76	538	545.39	550.16	1.3736	538	538.2	0	538	538	0	538	538	0
6	Pr76	108159	109206	110023	0.9680	108159	108288	0	108159	108268.3	0	108162	109214.8	0.0028
7	Rat99	1211	1218	1275	0.5780	1211	1211.04	0	1211	1211.1	0	**1210**	1210.108	-0.0826
8	Rad100	7910	7936	8039	0.3287	7910	7914.56	0	7910	7914.7	0	7910	7912.875	0
9	KroD100	21294	21309	21484	0.0704	21294	21340.64	0	21294	21314.2	0	21294	21410.02	0
10	Eil101	629	633.07	637.93	0.6471	629	629.6	0	629	629	0	629	629	0
11	Lin105	14379	14397	14521	0.1252	14379	14380.48	0	14379	14379	0	14379	14380.27	0
12	Pr107	44303	44316	44589	0.0293	44303	44379.88	0	44303	44392.25	0	**44301**	44302.83	-0.0045
13	Pr124	59030	59051	60157	0.0356	59030	59032.88	0	59030	59031.8	0	**58985**	59010.65	-0.0762
14	Bier127	118282	118282	120301	0	118282	118334.6	0	118282	118351.2	0	118282	118331.2	0
15	Ch130	6110	6141	6203.47	0.5074	6110	6121.96	0	6110	6127.95	0	6110	6110	0
16	Pr144	58537	58595	58662	0.0991	58537	58549.72	0	58537	58570.15	0	58537	58610.91	0
17	KroA150	26524	26676	26803	0.5731	26524	26538.2	0	26524	26542.6	0	26524	27032.18	0
18	Pr152	73682	73861	73989	0.2429	73682	73727.96	0	73682	73688.8	0	73683	73689.56	0.0014
19	U159	42080	42395	42506	0.7486	42080	42182.32	0	42080	42198.85	0	42080	42188.47	0
20	Rat195	2323	2341	2362	0.7749	2328	2334.2	0.2152	2328	2331	0.2152	2325	2329.032	0.0861
21	RroA200	29368	29731	31015	1.2360	29368	29422.64	0	29368	29405.35	0	29368	29435.76	0
22	Gil262	2378	2399	2439	0.8831	2379	2383.56	0.0421	2379	2382.45	0.0421	2381	2384.695	0.1262
23	Pr299	48191	48662	48763	0.9774	48192	48263.08	0.0021	48191	48250	0	48191	48197.49	0

24	Lin318	42029	42633	42771	1.4371	42076	42292.04	0.1118	42070	42264.35	0.0976	42029	42291.67	0
25	Rd400	15281	15464	15503	1.19757	15324	15377.56	0.2814	15311	15373.75	0.1963	15310	15318.11	0.1898
26	Pcb442	50778	51414	51494	1.2525	50879	51050.12	0.1989	50832	51041.7	0.1063	50812	51039.21	0.0670
27	Rat575	6773	6912	6952	2.0523	6819	6854.64	0.6792	6829	6847.95	0.8268	6773	6802.04	0
28	U724	41910	42657	42713	1.7824	42150	42302.12	0.5727	42205	42357.8	0.7039	41910	42262.11	0
29	Rat783	8806	9030	9126	2.5437	8897	8918.28	1.0334	8887	8913.25	0.9198	8806	8817.91	0
30	Pr1002	259045	265987	266774	2.6798	261463	262211.7	0.9334	261490	262202.5	0.9439	261491	262301.8	0.9442
31	D1291	50801	52378	52443	3.1043	51098	51340.84	0.5846	51032	51358.7	0.4547	51091	51316.18	0.5709
32	D1655	62128	64401	65241	3.6586	62784	63011.96	1.0559	62779	62994.65	1.0478	62779	63014.18	1.0478
33	Nl4461	182566	189334	192574	3.7072	185377	185631.1	1.5397	185290	185501.7	1.4921	185361	185401.7	1.5310
34	Brd14051	469385	490432	503560	4.4840	478040	478618.8	1.8439	477226	477612.7	1.6705	478385	477817	1.9174
35	Pla33810	66048945	70299195	72420147	6.4350	67868250	68038833	2.7545	67754877	67848535	2.5828	68004101	67100510	2.9602
				Average values:	**1.2929**			**0.3385**			**0.3229**			**0.2645**

In Tables 12 and 13, the comparisons of SOS-SA, SOS, and IBA are presented. The computed average values for the $PDbest$ and $PDmean$ of SOS-SA were obtained as -0.1262 and 0.4230 as shown in Table 12, and 0.0262 and 0.4891 as shown in Table 13. The average values for the SOS were obtained as 2.5019 and 4.3537 respectively, while those of the IBA in Table 13 were obtained as 0.2106 and 1.3911 respectively. Therefore, comparing the three algorithms, the computed average values for $PDbest$ and $PDmean$ show that SOS-SA obtained smaller percentage deviations than the SOS and IBA algorithms in all the instances considered in the two tables.

The percentage deviation test carried out shows the capability of the SOS-SA algorithm to find the best solution in a more effective and efficient manner. This can be attributed to the hybridization characteristics of the individual algorithms, where the systematic reasoning skill of the SA based on its ability to use the acceptance probability criteria to find better solution within the problem local search space is added to the exploration and exploitation capability of the SOS algorithm.

Table12: Comparison of SOS-SA with SOS, where $PDbest$ and $PDmean$ represent the percentage deviations of both the best solution found and the average solution to the best known solution (BKS).

S/N	**Instance**		**SOS-SA**					**SOS**				
	Name	BKS	Mean	Best	Diff.	PDbest (%)	PDmean(%)	Mean	Best	Diff.	PDbest (%)	PDmean(%)
1	dantzig42	699	699.482	679	-20	-2.8612	0.0690	710.502	702	3	0.4292	1.6455
2	Eil51	426	426	426	0	0	0	438.728	429	3	0.7042	2.9878
3	Berlin52	7542	7541.11	7540	-2	-0.0265	-0.0118	7659.49	7647	105	1.3922	1.5578
4	St70	675	675	675	0	0	0	699.129	675	0	0	3.5747
5	Eil76	538	538	538	0	0	0	556.312	542	4	0.7435	3.4037
6	Rat99	1211	1210.11	1210	-1	-0.0826	-0.0735	1297.38	1284	73	6.0281	7.1329
7	KroA100	21282	21424	21282	0	0	0.6672	21633.8	21401	119	0.5592	1.6530
8	KroB100	22140	22331.8	22140	0	0	0.8663	23142.8	22155	15	0.0678	4.5294
9	KroC100	20749	20860.8	20749	0	0	0.5388	21020.2	20811	62	0.2988	1.3071
10	KroD100	21294	21494.1	21294	0	0	0.9397	22044.3	21492	198	0.9298	3.5235
11	KroE100	22068	22205.9	22068	0	0	0.6249	22467.1	22128	60	0.2719	1.8085
12	Eil101	629	629	629	0	0	0	659.713	649	20	3.1797	4.8830
13	Pr107	44303	44302.8	44301	-2	-0.0045	-0.0005	46112.2	46097	1794	4.0494	4.0837
14	Pr124	59030	59010.7	58985	-45	-0.0762	-0.0327	69211.1	68942	9912	16.7915	17.2473
15	Pr136	96772	98636	97129	357	0.3689	1.9262	100461	98018	1246	1.2876	3.8121
16	Pr144	58537	58610.8	58537	0	0	0.1261	60136.9	58587	50	0.0854	2.7331
17	Pr152	73682	73689.6	73683	1	0.0014	0.0103	74699.8	74229	547	0.7424	1.3813
18	Pr264	49135	50201.6	49212	77	0.1567	2.1708	52498.5	51477	2342	4.7665	6.8454
19	Pr299	48191	48197.5	48191	0	0	0.0135	50102.4	49624	1433	2.9736	3.9663
20	Lin318	42029	42291.7	42029	0	0	0.6250	45811.1	44020	1991	4.7372	8.99879
				Average Values:		**-0.1262**	**0.4230**				**2.5019**	**4.3537**

Table 13: Table12: Comparison of SOS-SA with IBA, where $PDbest$ and $PDmean$ represent the percentage deviations of both the best solution found and the average solution to the best known solution (BKS).

S/N	**Instance**		**SOS-SA**					**IBA [Osaba et al., 2016]**				
	Name	BKS	Mean	Best	Diff.	PDbest (%)	PDmean(%)	Mean	Best	Diff.	PDbest(%)	PDmean(%)
1	Eil51	426	426	426	0	0	0	428.1	426	0	0	0.4929
2	Berlin52	7542	7541.11	7540	-2	-0.0265	-0.0118	7542	7542	0	0	0
3	St70	675	675	675	0	0	0	679.1	675	0	0	0.6074
4	Eil76	538	538	538	0	0	0	548.1	539	1	0.1859	1.8773
5	KroA100	21282	21424	21282	0	0	0.6672	21445.3	21282	0	0	0.7673

6	KroB100	22140	22331.8	22140	0	0	0.8663	22506.4	22140	0	0	1.6549
7	KroC100	20749	20860.8	20749	0	0	0.5388	21050	20749	0	0	1.4507
8	KroD100	21294	21494.1	21294	0	0	0.9397	21593.4	21294	0	0	1.4060
9	KroE100	22068	22205.9	22068	0	0	0.6249	22349.6	22068	0	0	1.2761
10	Eil101	629	629	629	0	0	0	646.4	634	5	0.7949	2.7663
11	Pr107	44303	44302.8	44301	-2	-0.0045	-0.0005	44793.8	44303	0	0	1.1078
12	Pr124	59030	59010.7	58985	-45	-0.0762	-0.0327	59412.1	59030	0	0	0.6473
13	Pr136	96772	98636	97129	357	0.3689	1.9262	99351.2	97547	775	0.8009	2.6652
14	Pr144	58537	58610.8	58537	0	0	0.1261	58876.2	58537	0	0	0.5795
15	Pr152	73682	73689.6	73683	1	0.0014	0.0103	74676.9	73921	239	0.3244	1.3503
16	Pr264	49135	50201.6	49212	77	0.1567	2.1708	50908.3	49756	621	1.2639	3.6090
					Average values:	**0.0262**	**0.4891**				**0.2106**	**1.3911**

Figs. 6-9, demonstrate the simulation tests of a number of TSB problem instances of different scales which were benchmarked to test the effectiveness of the new SOS-SA algorithm. The simulation results show route tracing and convergence graphs of each of the cities using SOS-SA algorithm. The algorithm execution was terminated after 2000 runs for each TSP instance. However, we are only able to show convergence graphs for the proposed algorithm because the other compared algorithms' data were taken from literature and this information was unavailable.

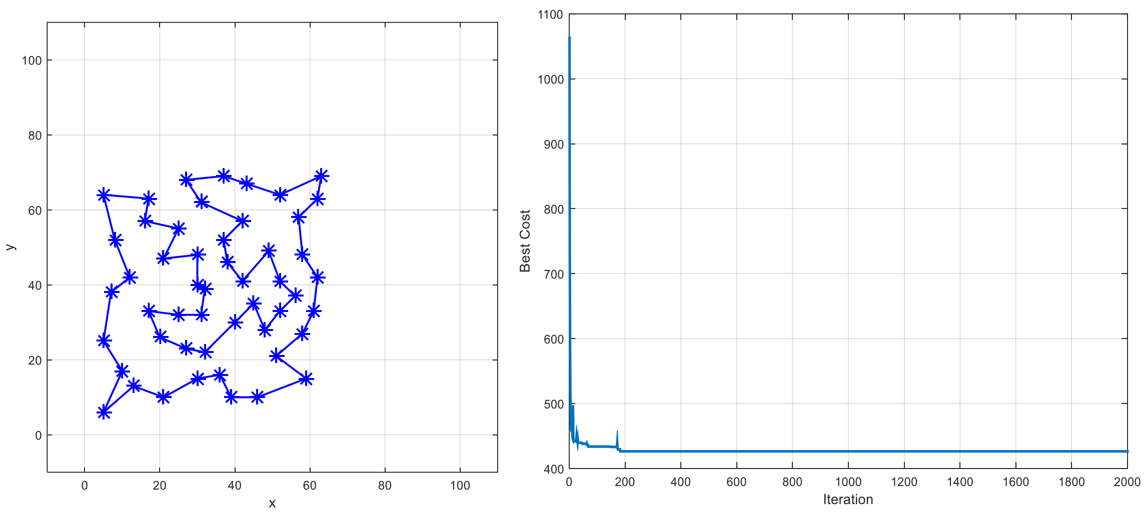


Fig.6. the result of 52-city (Eli52) TSPs using SOS-SA

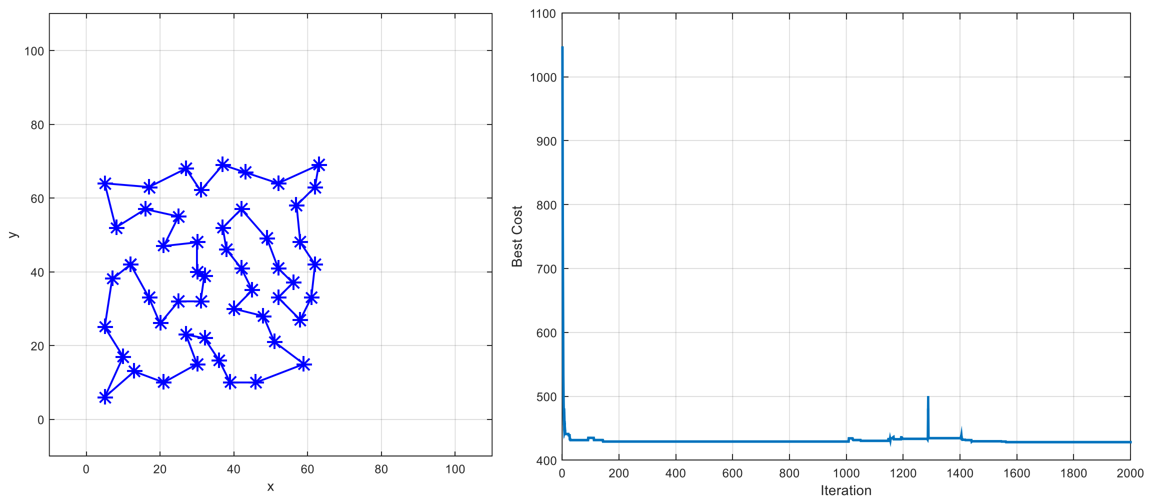


Fig.7. the result of 54-city (Eil54) TSPs using SOS-SA

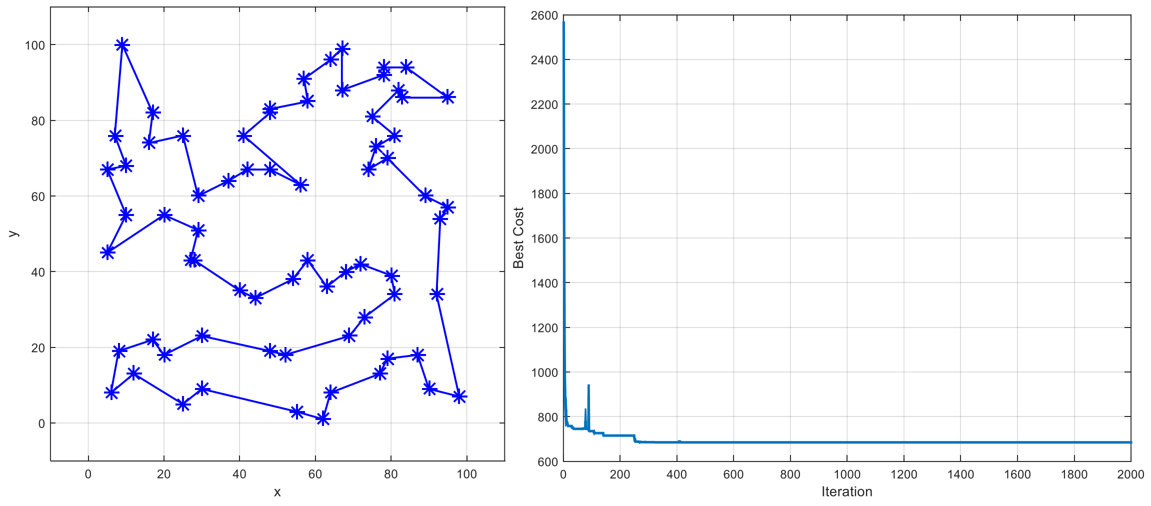


Fig.8. the result of 70-city (St70) TSPs using SOS-SA

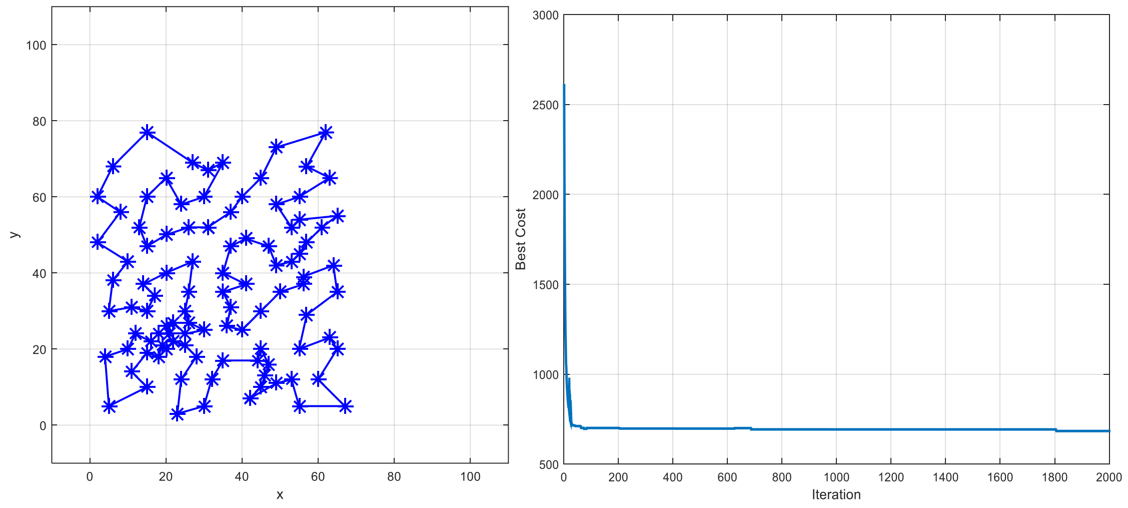


Fig.9. the result of 101-city (Eil101) TSPs using SOS-SA

5.4 Descriptive Statistical analysis

In this section, the Stata statistical package was used to further validate of the algorithm performance. The Shapiro-Will (W) test was used to formerly test whether the data were normally distributed. The Levene's test was used to test whether all the algorithms have the same variance, while both the oneway ANOVA test and Kruskal-Wallis test were used to test for difference in performance among all the algorithms. The oneway ANOVE is used whenever the parametric assumption were met, while the Kruskal-Wallis test is used whenever the parametric assumption were not met. In summary, descriptive statistics such as mean, standard deviation, minimum, maximum and range were used to describe the algorithms. Histograms and Shapiro-Wilk test were used to assess the normality of the algorithms, while the Levene's test was used to test the equality of Variance among the algorithms. Finally, to examine the significant difference in performance among the algorithms, the Friedman Test (with post hoc tests) was used. It is important to note that the difference in performance tests was done not based on the actual data but on transformed data. The main purpose of the difference in performance test was to verify that each of the selected algorithms that were compared with the SOS-SA is of high standard as claimed in the respective literatures. It is important to note that, in statistics, data transformation, which is the application of a deterministic mathematical function to each point in a data set, is utilized to help improve the normality of the experimental data sets and the interpretability or appearance of graphs (Zuur *et al.*, 2010; Jason, 2002).

5.4.1 Descriptive analysis of the SOS-SA and SOS algorithms

Table 14 presents the descriptive statistics of the performance of SOS-SA and SOS with the theoretical value or the best known solution as the control algorithm. The SOS-SA is averagely smaller than the SOS in terms of mean, standard deviation, minimum, maximum and rang compared in terms of the best known solution. Therefore, this suggests that the SOS-SA is better than the SOS. The SOS' data has the widest range and spread of data around its mean value, while SOS-SA algorithm has the smallest range and dispersion of data around its mean value. Moreover, the standard deviations of the two algorithms are quiet high which suggests that there is great variation around the mean value for the two algorithms. The implication is that the data may not be normally distributed and parametric approaches cannot be used directly to test the significance of the difference in performance between SOS-SA and SOS.

Table 14: Descriptive statistics validation of SOS-SA and SOS algorithms compared to the best known solution (BKS)

Algorithm	Mean	Std deviation	Min	Max	Range
BKS	29,546.60	28,177.05	426	96,772	96,346
SOS	30,545.45	29,211.58	429	98,018	97,589
SOS_SA	29,564.85	28,223.54	426	97,129	96,703

To conveniently study whether the normality assumption is violated in the sample data under study, both informal and formal approaches were used. Based on the histograms (Fig. 10) of the algorithms SOS-SA and SOS, it can be readily seen that the algorithms' histograms are flatter and skewer than that of a normal distribution curve (Bell-shape curve) therefore providing more evidence of the non-normality of the three algorithms. However, the algorithms seem to follow the same form of skewness and flatness; meaning that, the spread of the data may not be significantly different within the two algorithms. In other words, the variance of the two algorithms may be equal.

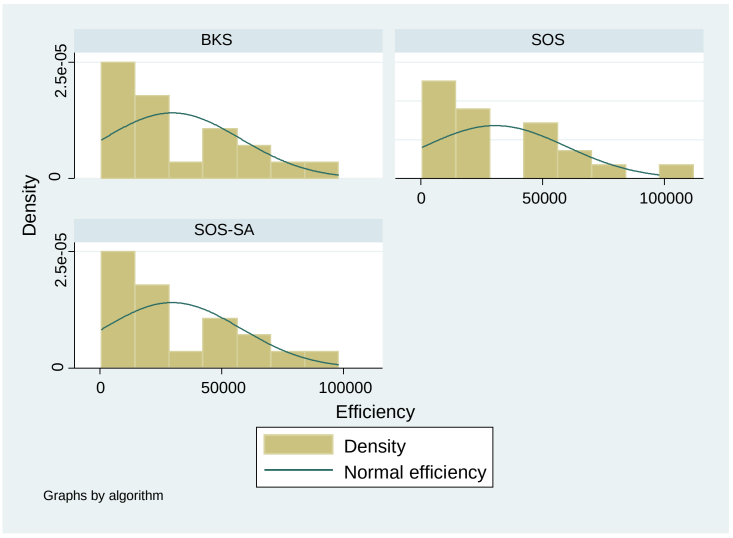


Figure 10: Histogram of SOS-SA and SOS-SA with the BKS as the control algorithm

Formally, the Shapiro-Wilk (W) test statistic was considered in the analysis as being more robust than other tests statistics such as the skewness-Kurtosis, Kolmogorov-Smirnov and the Shapiro-Francia tests statistics. According to Table 15, after checking for the presence of outliers, the normality test was conducted based on the level data and four transformed data to avoid the problem of failing to adequately transform the data. The results show that the algorithms are indeed non-normally distributed at the level form, but were only found so based on the λ-parameter Box-Cox power transformation. On this basis, Box-Cox power transformed forms of the algorithms were further used for the test of equality of variance.

Table 15: Normality test of SOS-SA and SOS algorithms at different forms using the Shapiro-Wilk test statistic

Form	BKS	SOS	SOS-SA
Level	0.889**	0.888**	0.888**
Log	0.813***	0.815***	0.812***
Square root	0.813***	0.815***	0.812***
K-parameter log	0.895***	0.893***	0.895***
Lambda-parameter Box-Cox power	0.907	0.905	0.906

Note: ***<0.01 and **<0.05

The analysis of equal variance across the three algorithms was based on the Levene's test because it is robust even with departure from normality of the data. The result in Table 16 indicates that based on mean, median and 10th percentile, the test was found to be statistically insignificant; implying that, the null hypothesis of equal variance cannot be rejected. In other words, compared with the best known solution (BKS), SOS-SA and SOS have equal variance.

Table 16: Equal variance test of SOS-SA and SOS algorithms against the best known solution using the Levene's test statistic

Statistic	Levene's	P-value
W0	0.001	0.998
W50	0.001	0.998
W10	0.002	0.997

W0=mean, W50=Median, W10=10th percentile.

Based on the findings obtained so far, one way Analysis of Variance (ANOVA) was carried out to assess the difference between the BKS, SOS-SA and SOS. The result of the test presented in Table 17 indicates that the majority of variation in the algorithms' observations is explained in the variation within and not the variation between. In other words, the contribution of the model to explain the difference between the three methods is minimal in relation to that of the residual. Consequently, the ANOVA test was found to be statistically insignificant on the basis of the low F-statistic and high p-value. In other words, there is no statistically significant difference between the best known solution and the proposed SOS-SA algorithm.

Table 17: Oneway ANOVA test of the difference between SOS-SA and SOS compared to the best known solution

Source of variation	SS	df	MS	F	P-value
Between group	250.82	2	125.41	2.30E-03	0.997
Within group	3,106,466.42	57	54,499.41		
Total	3,106,717.25	59	52656.22		

5.4.2 Descriptive analysis of the GA-PSO-ACO, MSA-IBS, LBSA, and SOS-SA algorithms

Table 18 presents the descriptive statistics of the performance of four algorithms namely, GA-PSO-ACO, MSA-IBS, LBSA, and SA- SOS with the best known solution as the control algorithm. The SOS-SA algorithm is averagely the smallest algorithm followed by LBSA, suggesting that SOS-SA is the best algorithm followed by LBSA. The standard deviations of all the algorithms are quite high suggesting that there is great variation around the mean value for all the algorithms. In other words, the data may not be normally distributed and parametric approaches cannot be used directly to test the significance of the difference among the algorithms.

Table 18: Descriptive statistics of GA-PSO-ACO, LBSA, MSA-IBS, and SOS-SA to the best known solution

Algorithm	Mean	Std dev.	Min	Max	Range
GA-PSO-ACO	2,063,992	11,900,000	426	70,300,000	70,299,574
LBSA	1,997,597	11,500,000	426	68,000,000	67,999,574
MSA-IBS	1,993,722	11,500,000	426	67,900,000	67,899,574
SOS-SA	1,990,455	11,400,000	426	67,800,000	67,799,574
BKS	1,941,301	11,200,000	426	66,000,000	65,999,574

Based on the histograms (Fig. 11) of the algorithms GA-PSO-ACO, LBSA, MSA-IBS, and SOS-SA, it can be readily be seen that the algorithms do not follow a normal distribution therefore providing more evidence of the non-normality of the data.

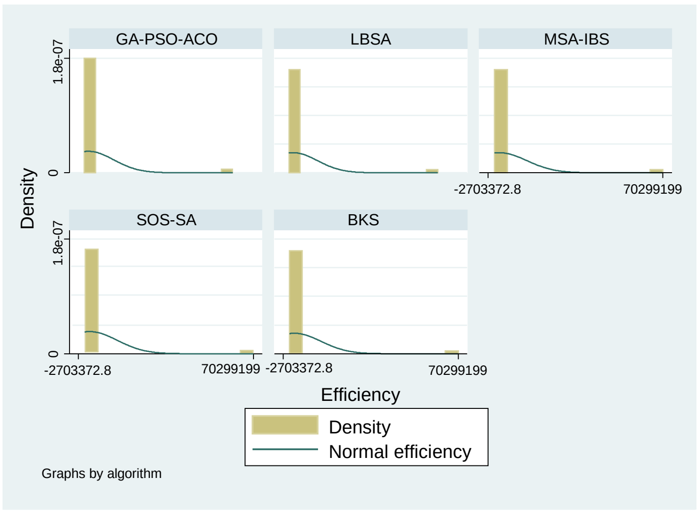


Fig. 11: Histogram of GA-PSO-ACO, LBSA, MSA-IBS with BKS as the control algorithm

According to Table 19, after checking for the presence of outliers, the normality test was conducted based on the level data and four transformed data to avoid the problem of failing to adequately transform the data. The results show that the algorithms are indeed non-normally distributed even after various transformations. The test of the difference among the four (4) algorithms based on their transformed data was carried out using the Kruskal-Wallis test.

Table 19: Normality test for GA-PSO-ACO, LBSA, MSA-IBS, and SOS-SA algorithms at different forms using the Shapiro-Wilk test statistic

Form	GA-PSO-ACO	LBSA	MSA-IBS	SOS-SA	BKS
Level	0.16557***	0.16564***	0.16564***	0.16563***	0.1657***
Log	0.90744***	0.90755***	0.90754***	0.90756***	0.90775***
Square root	0.90744***	0.90755***	0.90754***	0.90756***	0.90775***
K-parameter log	0.9189**	0.91886**	0.91886**	Na	Na
Lambda-parameter Box-Cox power	Na	Na	Na	Na	Na

Note: ***<0.01

The Kruskal-Wallis test is a nonparametric test that does not depend on the normality and homogeneity of the data. The analysis is based not on the original data but on the rank of the data after sorting them in ascending order. In Table 20, the result of the test both with and without ties indicates that based on the transformed data, there is no statistically significant difference among SOS-SA, LBSA, and MSA-IBS and that the difference observe could be due to sampling peculiarity. This is also to verify that SOS-SA is being compared with some of the best state-of-the-art heuristic algorithms, having the most similar techniques of implementation with the SOS-SA.

Table 20: Difference analysis of GA-PSO-ACO, LBSA, MSA-IBS, SOS-SA and BKS using Kruskal-Wallis test

		Chi-squared	
Algorithm	Sum Rank	Without ties	With ties
GA-PSO-ACO	3,162	0.107	0.107
LBSA	3,069		
MSA-IBS	3,079.5		
SOS-SA	3,046		
BKS	3,033.5		

5.4.3 Descriptive analysis of SOS-SA and IBA algorithms

Table 21 presents the descriptive statistics of the performance of SOS-SA and IBA compared with the best known solution. The SOS-SA algorithm is averagely the smallest, therefore, this suggest that SOS-SA performs better than the IBA. The IBA's data has the widest range and spread of data around its mean value while SOS-SA has the smallest range and dispersion of data around its mean value compared to the best known solution. Moreover, the standard deviations of the two algorithms are quiet high which suggests that there is great variation around the mean value for all the algorithms. The implication is that the data may not be normally distributed and parametric approaches cannot be used directly to test the significance of the difference in performance among SOS-SA and IBA.

Table 21: Descriptive statistics of IBA, BKS and SOS_SA

Algorithm	Mean	Std deviation	Min	Max	Range
IBA	31,277.69	29,494.45	426	97,547	97,121
BKS	31,175.13	29,330.33	426	96,772	96,346
SOS-SA	31,199.25	29,384.09	426	97,129	96,703

Based on the histograms (Fig. 12) of the two algorithms, it can be readily seen that the algorithms' histograms are flatter and skewer than that of a normal distribution curve (Bell-shape curve) therefore providing more evidence of the non-normality of the two algorithms. However, both algorithms with the BKS seem to follow the same form of skewness and flatness; meaning that, the spread of the data may not be significantly different within the three algorithms. In other words, the variance of the three algorithms may be equal.

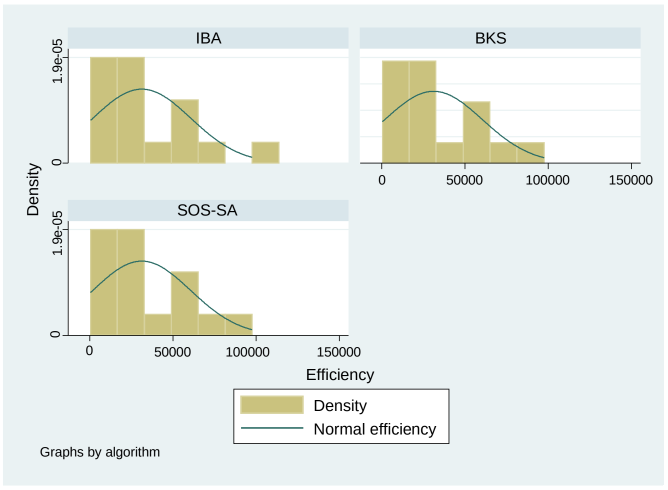


Fig. 12: Histogram of SOS-SA and IBA with BKS as the control algorithm

According to Table 22, the results show that the algorithms are indeed non-normally distributed at the level form but were only found so based on the λ-parameter Box-Cox power transformation. On this basis, Box-Cox power transformed forms of the algorithms were further used for the test of equality of variance.

Table 22: Normality test of BKS, SOS_SA, and IBA algorithms at different forms using the Shapiro-Wilk test statistic

Form	BKS	IBA	SOS_SA
Level	0.8901*	0.8894*	0.8901*
Log	0.8058***	0.8065***	0.806***
Square root	0.8058***	0.8065***	0.806***
K-parameter log	0.9123**	0.9123**	0.9124**
Lambda-parameter Box-Cox power	0.9218	0.9219	0.9219

Note: ***<0.01 and **<0.05

The analysis of equal variance across the two algorithms was based on the Levene's test because it is robust even with departure from normality of the data. The result in Table 23 indicates that based on mean, median and 10th percentile, the test was found to be statistically insignificant; implying that, the null hypothesis of equal variance cannot be rejected. In other words, SOS-SA and IBA, with the best known solution have equal variance.

Table 23: Equal variance test of SOS-SA and IBA algorithms to the best known solution using the Levene's test statistic

Statistic	Levene's	P-value
W0	0.0053	0.9946
W50	0.0048	0.9951
W10	0.0051	0.9949

W0=mean, W50=Median, W10=10th percentile.

Based on the findings obtained so far, one-way Analysis of Variance (ANOVA) was carried out to assess the difference between SOS-SA and IBA algorithms with the best known solution as the control algorithm. The result of the test presented in Table 24 indicates that the majority of variation in the algorithms' observations is explained the variation within and not the variation between. In other words, the contribution of the model to explain the difference between the three algorithms is minimal in relation to that of the residual. Consequently the ANOVA test was found to be statistically insignificant on the basis of the low F-statistic and high p-value. In other words, there is no statistically significant difference between the two algorithms.

Table 24: Oneway ANOVA test of the difference between SOS-SA and IBA

Source of variation	SS	df	MS	F	P-value
Between group	643.66	2	321.83	0.01	0.9943
Within group	2,519,515.64	45	55,989.24		
Total	2,520,159.30	47	53620.41		

Note: SS=Sum of Squares, MS=Mean Sum of Square, df=Degree of freedom

5.4.4 Friedman Test (with post hoc tests) analysis of algorithm performance

In this section, the Friedman's non-parametric test was further used to check for existence of any significant difference in performance among the algorithms whilst running, χ^2 (1) = 19, *p*-value = 0.000 for comparison between SOS-SA and SOS, χ^2 (1) = 8, *p*-value = 0.005 for comparison between SOS-SA and IBA, and $\chi^2(3)$ = 73.189, *p*-value = 0.000 for comparison among SOS-SA with GA-PSO-ACO, MSA-IBS, and LBSA. The median (IQR) perceived effort levels for the SOS and SOS-SA running trial were 21681 (848 to 51013) and 21681 (812 to 48956), while the median (IQR) perceived effort levels for the IBA and SOS-SA running trial were 21681 (2392 to 56341) and 21681 (2391 to 56205), respectively. It can be concluded that there are statistically significant differences among the algorithms based on the mean ranking returned by the Friedman's test presented in table 25. However, since the Friedman's test can only show the existence of significant difference between two algorithms, the test does not pinpoint which groups in particular differ from each other in the case of multiple algorithms comparison (for example, GA-PSO-ACO, MSA-IBS, LBSA, and SOS-SA). Therefore, to adequately evaluate the statistical performance of the SOS-SA, the Friedman test with post hoc tests was further conducted and the result obtained is as reported next.

Table 25: Mean ranking returned by Friedman's non parametric test

Test 1		**Test 2**		**Test 2**	
Algorithms	Ranking	Algorithms	Ranking	Algorithms	Ranking
GA-PSO-ACO	3.91	SOS	2.00	IBA	1.75
MSA-IBS	2.24	SOS-SA	1.00	SOS-SA	1.25
LBSA	1.96				
SOS-SA	1.89				

The Friedman test with post hoc tests indicate that there was a statistically significant difference in the performance of SOS-SA with GA-PSO-ACO, MSA-IBS, and LBSA, this is observed whilst running, $\chi^2(3)$ = 73.189, *p*-value = 0.000. Post hoc analysis with Wilcoxon signed-rank tests was conducted with a Bonferroni correction applied, resulting in a significance level set at computed *p*-value < 0.0125 (i.e. 0.05/4, since we are comparing GA-PSO-ACO, MSA-IBS, LBSA, and SOS-SA). The median (IQR) perceived effort levels for each of the GA-PSO-ACO, MSA-IBS, LBSA, and SOS-SA running trial were 33524 (6912 to 59051), 33522 (6819 to 59030), 33522 (6829 to 59030), and 33523 (6773 to 58985) respectively. There were statistically significant differences between the performance of SOS-SA and GA-PSO-ACO running trials (Z = -5.012, *p*-value = 0.000). However, there were no significant differences between the SOS-SA and MSA-IBS running trials (Z = -1.625, *p*-value = 0.104), or between SOS-SA and LBSA running trials (Z = -0.355, *p*-value = 0.722).

In summary, the statistical analysis has revealed some interesting results with respect to all the algorithms that were compared with the proposed optimization method. First, the analysis result showed that the SOS-SA performed favourable well compared to other state-of-the-art algorithms. This is verified based on the result of the descriptive statistics test using Mean, Standard deviation, Min, Max, and Range described in Tables 14, 18 and 21 respectively. This also corresponds to the analysis of the results presented in Tables 11, 12, and 13, were SOS-SA (with 32%) outperformed the other three algorithms namely, GA-PSO-ACO (with 19%), MSA-IBS (with 20%), and LBSA (with 28%) in terms of convergence. Similar evaluation with SOS and IBA showed the convergence performance of SOS-SA to be 95% (19 out 20 instances) and 50% (8 out of 16 instances) for both SOS and IBA. Second, based on the performance difference among all the algorithms compared with SOS-SA using transformed data and with BKS as control algorithm, the analysis revealed that MSA-IBS, LBSA, and IBA are equally good algorithms as claimed by the respective authors. Third, SOS-SA appears to be next to the control algorithm (or BKS) in most of the performance analysis result presented, for instance in Table 20, the Sum Rank (i.e., SOS-SA = 3,046 and BKS = 3,033.5). Finally, considering the Friedman Test (with post hoc tests) analysis for the individual algorithm performances, since the *p*-values of GA-PSO-ACO, SOS, and IBA are less than 0.05, we can say that SOS-SA is statistically significantly better. In the case of MSA-IBS and LBSA, since their p-values are greater than the computed p-value of 0.0125, therefore, we can say that there are no significant differences in performance between these algorithms and the SOS-SA. Therefore, this consequently verifies the initial claim that the SOS-SA algorithm can compete favorable with even the best known solution, as it tends to perform better in some cases than the BKS, which then verifies the results presented in Table 2. The unique advantage of the SOS-SA over other algorithms can be attributed to its capability to deeply explore and exploit problem search space, during search process. This is made possible by the benefit factor mechanism in the mutualism phase and the artificial vector mechanisms in the parasitism phase of the SOS. Finally, we conclude this analysis section by saying that the proposed SOS-SA optimization method has promising and immense potential for solving the TSP as well as other complex discrete problems.

5.5. Remarks

In the course of the empirical evaluation of the proposed algorithms, some potential Challenges were observed, which are highlighted as follows:

- **Computation Time**. Though the SOS-SA algorithm performed favourably against the best known available solution from TSPLIB and other state-of-the-art algorithms, there is still room for improvement in terms of computational time, and more specifically, the iterative computation of tour cost function. The SOS-SA algorithm spends time recalculating the cost function with every change in iteration and most importantly, it was also observed that the computation cost increases proportionately with increase in the dimension of the TSP problem instance. A more simplified and adaptive method of calculating the cost function is therefore required to speed-up the computation time.
- **Acceptance probability function**. Similarly, the computation of the acceptance probability function consumes a lot of system resources, more specifically, CPU time. This is as a result of the exponential computation required to determine the probability of acceptance or rejection of a new solution. Therefore, approximating the calculation of this function without compromising the decision rule can significantly improve the performance of the framework in terms of cost of execution.
- **SA parameter selection**. SA parameter required some level of experience in selecting a good set of performance parameters, as they would partly affect the performance of the SOS-SA algorithm, in escaping global minimum as quickly as possible. Selecting a good set of SA performance parameters was the main bottleneck experienced during the simulation experiment, as parameter fine-tuning were made more frequently. This challenge also affected the length of time the optimal solutions were attained. A typical example is the selection of an appropriate cooling schedule for the different simulations. Finding an appropriate cooling schedule was a major challenge for the SOS-SA implementation from one problem instance to another. One possible solution is to implement an adaptive method of setting the cooling schedule for different problem instances.
- **Scalability**. In the course of the simulation process specifically for large length TSP problems, the system ran out of memory severally as the dimension of the TSPLIB benchmark increased, as it was observed for Pla33810 and Pla85900. These two instances, during execution, required additional memory. Therefore, considering also the first two aforementioned issues, one possible option would be to identify possible parallelism for the SOS-SA algorithm, which can improve both framework computational time and memory utilization concurrently.

6. Conclusion and Future Direction

In this paper, a novel and hybrid simulated annealing based symbiotic organisms search algorithm is proposed as a new approach for solving symmetric TSP. The SOS algorithm which is inspired by the symbiotic relationships among organisms in the ecosystem was initially proposed to handle engineering optimization problems. The design of a hybrid SOS-SA framework, which incorporates the SA local search capability into the problem search space of SOS algorithm, and the application of the simulation results of the SOS-SA to the TSP were discussed. The simulation results supports the fact that the new SOS-SA framework can realise TSP optimal solutions and compete favourably with other state-of-the-art optimization algorithms being applied to the TSP related problems and complex discrete problems. As future work the authors intend to further improve the algorithm by testing its scalability in a parallel and distributed environments for various Big Data graphs from SNAP (https://snap.stanford.edu/data/) with different properties (e.g., sparse, dense, power law). Scalability will be tested when problem size is fixed and number of cores/machines increases, and when the number of machines is fixed and the problem increases.

Funding

This work was supported by the research grant from University of KwaZulu-Natal, College of Agriculture, Engineering and Science. Durban, South Africa.

Author Contributions

AEE and AOA conceived and designed the experiments, AEE and AOA performed the experiments, AEE and MEF analysed the data, AEE, AOA and MEF wrote the paper.

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Reference

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2011). The traveling salesman problem: a computational study. Princeton university press.

Aulady, M. (2013). A hybrid symbiotic organisms search-quantum neural network for predicting high performance concrete compressive strength.

Barbato, M., Grappe, R., Lacroix, M., & Calvo, R. W. (2016). Polyhedral results and a branch-and-cut algorithm for the double traveling Salesman problem with multiple stacks. Discrete Optimization, 21, 25-41.

Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. Journal of the ACM (JACM), 9(1), pp.61-63.

Bender, M. A., & Chekuri, C. (2000). Performance guarantees for the TSP with a parameterized triangle inequality. Information Processing Letters, 73(1), 17-21.

Chen, S. M., & Chien, C. Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Systems With Applications, 38(12), 14439–14450.

Cheng, M. Y, Prayogo, D. (2014). Symbiotic organism search: A new metaheuristic optimization. Computers and Structures, 139:98-112.

Cheng, M.Y., Prayogo, D. and Tran, D. H. (2015). Optimizing multiple-resources levelling in multiple projects using discrete symbiotic organisms search. Journal of Computing in Civil Engineering, 30(3), p.04015036.

Cornu, M., Cazenave, T., & Vanderpooten, D. (2016). Perturbed Decomposition Algorithm applied to the multi-objective Traveling Salesman Problem. Computers & Operations Research.

Çunkaş, M. and Özsağlam, M. Y. (2009). A comparative study on particle swarm optimization and genetic algorithms for traveling salesman problems. Cybernetics and Systems: An International Journal, 40(6), pp.490-507.

Delgadillo, F. J. D., Montiel, O., & Sepúlveda, R. (2016). Reducing the size of traveling salesman problems using vaccination by fuzzy selector. Expert Systems with Applications, 49(1), 20–30.

Deng, W., Chen, R., He, B., Liu, Y., Yin, L. and Guo, J. (2012). A novel two-stage hybrid swarm intelligence optimization algorithm and application. Soft Computing, 16(10), pp.1707-1722.

Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. BioSystems, 43(2), pp.73-81.

Durbin, R. (1987). An analogue approach to the travelling salesman. Nature, 326, p.16.

Durbin, R., Szeliski, R. and Yuille, A. (1989). An analysis of the elastic net approach to the traveling salesman problem. Neural Computation, 1(3), pp.348-358.

Fang, L., Chen, P., & Liu, S. (2007). Particle swarm optimization with simulated annealing for TSP. In Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'07) (pp. 206-210).

Farmer, J. D., Packard, N. H. and Perelson, A. S. (1986). The immune system, adaptation, and machine learning. Physica D: Nonlinear Phenomena, 22(1), pp.187-204.

Feng, X., Lau, F.C. and Gao, D. (2009). A New Bio-inspired Approach to the Traveling Salesman Problem. Complex Sciences, pp.1310-1321.

Garey, M. R., & Johnson, D. S. (1979). Com-puters and Intractablility: A Guide to the Theory of NP-Completeness.

Geng, X., Chen, Z., Yang, W., Shi, D. and Zhao, K. (2011). Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Applied Soft Computing, 11(4), pp.3680-3689.

Jason, O. (2002). Notes on the use of data transformations. Practical Assessment, Research & Evaluation, 8(6).

Johnson, D. S. (1990). Local optimization and the traveling salesman problem. In International Colloquium on Automata, Languages, and Programming (pp. 446-461). Springer Berlin Heidelberg.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 1, pp.215-310.

Jolai, F., and Ghanbari, A. (2010). Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem. Expert Systems with Applications, 37(7), 5331-5335.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. Handbooks in operations research and management science, 7, 225-330.

Kanda, J., de Carvalho, A., Hruschka, E., Soares, C., & Brazdil, P. (2016). Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features. Neurocomputing.

Katayama, K., Sakamoto, H., & Narihisa, H. (2000). The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem. Mathematical and Computer Modelling, 31(10), 197-203.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), pp.671-680.

Knox, J. (1994). Tabu search performance on the symmetric traveling salesman problem. Computers & Operations Research, 21(8), pp.867-876.

Lawler, E.L. and Wood, D.E. (1966). Branch-and-bound methods: A survey. Operations research, 14(4), pp.699-719.

Lin, Y., Bian, Z., & Liu, X. (2016). Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing–tabu search algorithm to solve the symmetrical traveling salesman problem. Applied Soft Computing, 49, 937-952.

Lourenço, H. R., Martin, O. C. and Stützle, T. (2003). Iterated local search. In Handbook of metaheuristics (pp. 320-353). Springer US.

Malek, M., Guruswamy, M., Pandya, M., & Owens, H. (1989). Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Annals of Operations Research, 21(1), 59-84.

Matai, R., Singh, S. P. and Mittal, M. L. (2010). Traveling salesman problem: An overview of applications, formulations, and solution approaches. Traveling Salesman Problem, Theory and Applications, pp.1-24.

Mohan, U., Ramani, S., & Mishra, S. (2016). Constant factor approximation algorithm for TSP satisfying a biased triangle inequality. Theoretical Computer Science, 657, 111–126.

Osaba, E., Yang, X. S., Diaz, F., Lopez-Garcia, P., & Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Engineering Applications of Artificial Intelligence, 48, 59-71.

Ouaarab, A., Ahiod, B., & Yang, X. S. (2014). Improved and discrete cuckoo search for solving the travelling salesman problem. In Cuckoo Search and Firefly Algorithm (pp. 63-84). Springer International Publishing.

Ozcan, E. and Erenturk, M. (2004). A brief review of memetic algorithms for solving Euclidean 2D traveling salesrep problem. In Proc. of the 13th Turkish Symposium on Artificial Intelligence and Neural Networks (pp. 99-108).

Reinelt, G. (1994). The traveling salesman: computational solutions for TSP applications. Springer-Verlag.

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C. and Wang, Q. X. (2007). Particle swarm optimization-based algorithms for TSP and generalized TSP. Information Processing Letters, 103(5), pp.169-176.

Sundar, K., & Rathinam, S. (2016). Generalized multiple depot traveling salesmen problem—Polyhedral study and exact algorithm. Computers & Operations Research, 70, 39-55.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. Journal of heuristics, 8(5), 541-564.

Tran, D. H., Cheng, M. Y. and Prayogo, D. (2016). A novel Multiple Objective Symbiotic Organisms Search (MOSOS) for time–cost–labor utilization tradeoff problem. Knowledge-Based Systems, 94, pp.132-145.

Tsai, C. F., Tsai, C. W., & Tseng, C. C. (2004). A new hybrid heuristic approach for solving large traveling salesman problem. Information Sciences, 166(1), 67-81.

Verma, S., Saha, S., and Mukherjee, V. (2015). A novel symbiotic organisms search algorithm for congestion management in deregulated environment. Journal of Experimental & Theoretical Artificial Intelligence, 1-21.

Vincent, F. Y., Redi, A. P., Yang, C. L., Ruskartina, E., & Santosa, B. (2016). Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. Applied Soft Computing.

Volgenant, T. and Jonker, R. (1982). A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational Research, 9(1), pp.83-89.

Wang, C., Lin, M., Zhong, Y. and Zhang, H. (2015). Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. International Journal of Computing Science and Mathematics, 6(4), pp.336-353.

Wang, J., Ersoy, O. K., He, M., & Wang, F. (2016). Multi-offspring genetic algorithm and its application to the traveling salesman problem. Applied Soft Computing, 43, 415-423.

Zhan, S. H., Lin, J., Zhang, Z. J. and Zhong, Y. W. (2016). List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. Computational intelligence and neuroscience, 2016, pp. 1-12.

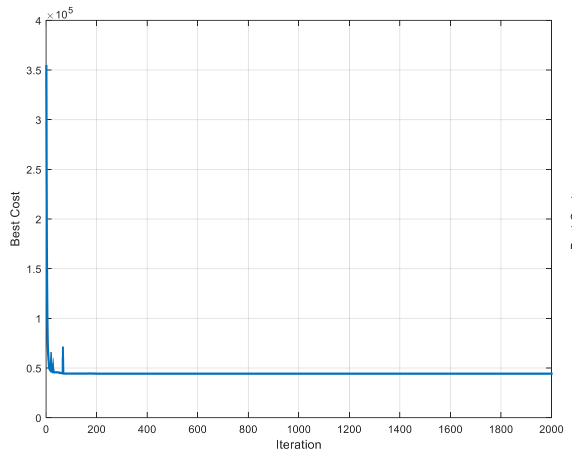
Zhang, H., & Zhou, J. (2016). Dynamic multiscale region search algorithm using vitality selection for traveling salesman problem. Expert Systems with Applications, 60, 81-95.

Zhang, H., Tong, W., Xu, Y., & Lin, G. (2016). The Steiner traveling salesman problem with online advanced edge blockages. Computers & Operations Research, 70, 26-38.

Zhou, Y., Luo, Q., Chen, H., He, A. and Wu, J., 2015. A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing, 151, pp.1227-1236.

Zuur, A. F., Ieno, E. N., & Elphick, C. S. (2010). A protocol for data exploration to avoid common statistical problems. Methods in Ecology and Evolution, 1(1), 3-14.

Appendix 1. Simulation results demonstrating the convergence curves for Pr107, Pr124, U159, Rat195, Gil262, Pr299, Pcb442, and Rat575 TSPLIB instance.



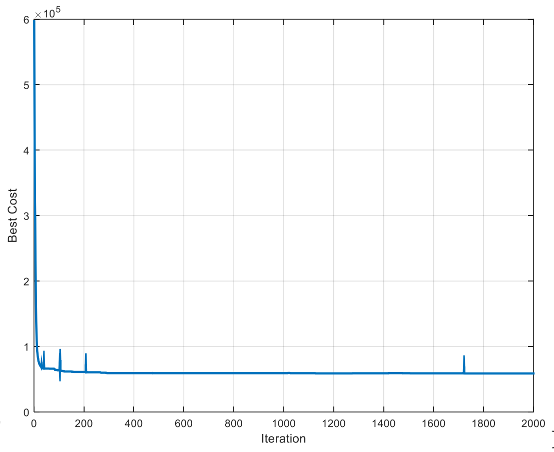
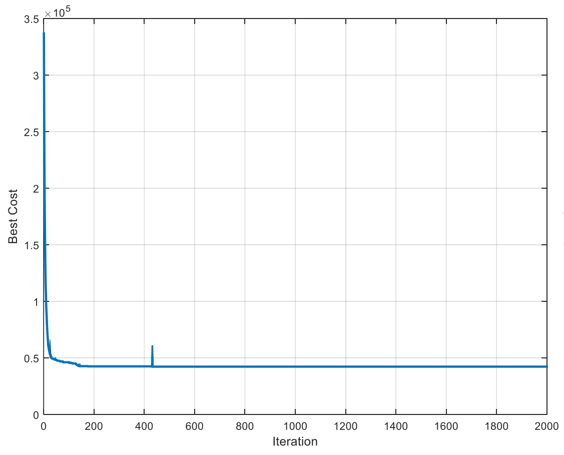


Fig.A1. Convergence curve for Pr107

Fig.A2. Convergence curve for Pr124



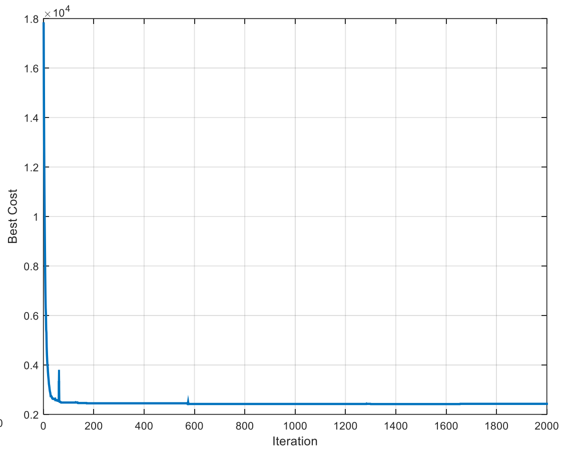


Fig.3A. Convergence curve for U159

Fig.4A. Convergence curve for Rat195

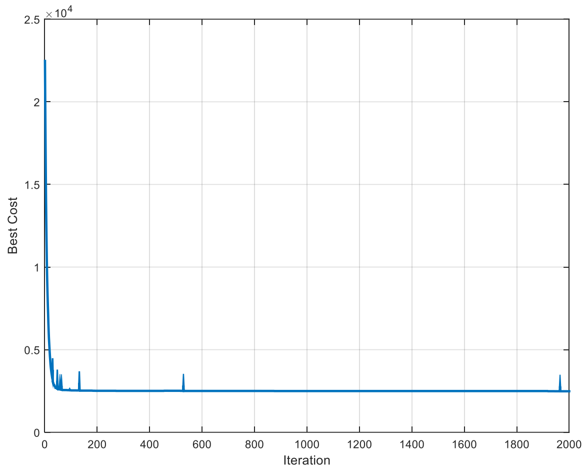


Fig.5A. Convergence curve for Gil262

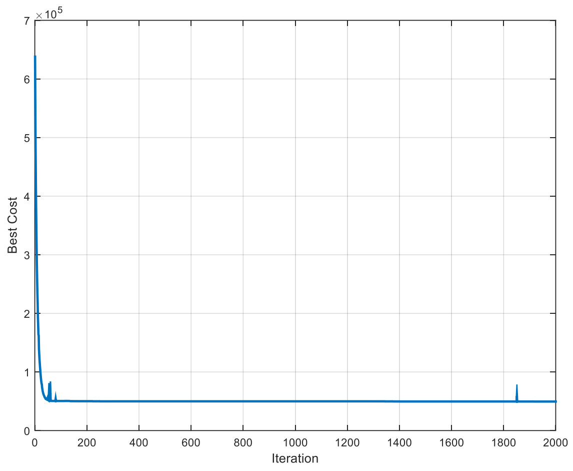


Fig.6A. Convergence curve for Pr299

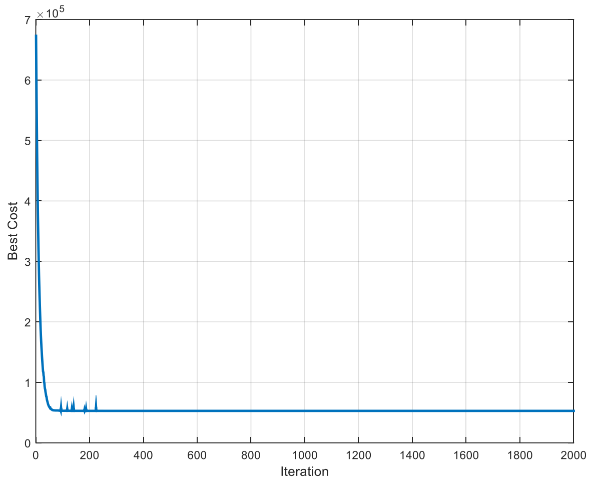


Fig.7A. Convergence curve for Pcb442

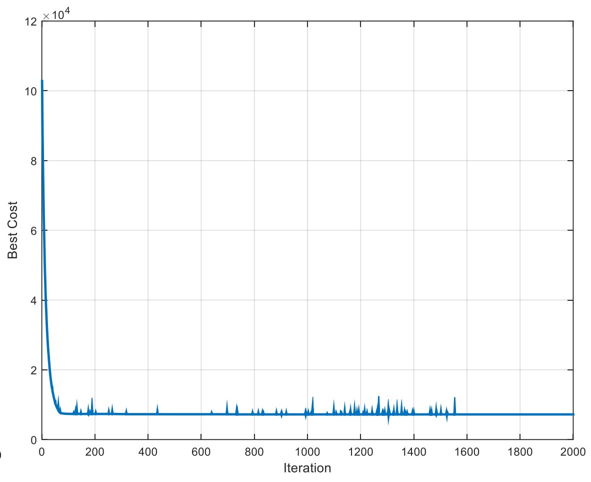


Fig.8A. Convergence curve for Rat575