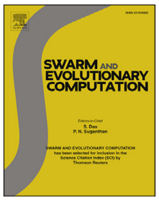
ARTICLE IN PRESS

Swarm and Evolutionary Computation xxx (2018) 1–11

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem

Indadul Khan [a,*], Manas Kumar Maiti [b]

[a] Department of Computer Science, Chandrakona Vidyasagar Mahavidyalaya, Paschim-Medinipur, West Bengal, 721201, India
[b] Department of Mathematics, Mahishadal Raj College, Mahishadal, Purba-Medinipur, West Bengal, 721628, India

ARTICLE INFO

Keywords:
Traveling Salesmen Problem
Artificial Bee Colony Algorithm
Swap sequence
Swap operation
K-opt

ABSTRACT

In this research paper, Artificial Bee Colony algorithm is modified with multiple update rules and K-opt operation to solve the Traveling Salesman Problem. Here the features of swap sequences and swap operations on the sequence of cities (solution/path) of the problem are used to create different solution (path) update rules of the algorithm. Eight different rules are proposed to update solutions in the algorithm. Updation of a solution by an employed bee or by an onlooker bee is done by a randomly selected rule from the rule set using Roulette Wheel selection process. In the scout bee phase of the algorithm, the perturbation technique, K-opt operation is applied on any stagnant solution for a fixed number of times for the possible improvement of it. The K-opt operation is again used at the end of the search process to improve the quality of the final solution (if possible). Proposed method is tested with a set of benchmark test problems from TSPLIB and it is observed that the efficiency of the algorithm is adequate with respect to the accuracy and the consistency for solving standard TSPs (Symmetric as well as Asymmetric) compared to the existing algorithms in the literature.

1. Introduction

The Traveling Salesmen Problem (TSP) is one of the standard combinatorial discrete optimization problem. The problem consists of a set of N vertices (nodes/cities) where the distance between any two vertices is known. A salesman starts from a vertex, visits all the vertices exactly once and returned to the starting vertex in such a way that the total distance travelled is a minimum. So the goal of the problem is to find a shortest possible tour through the set of vertices such that each vertex is visited exactly once except for the starting vertex. It is a well known NP-hard problem, can't be solved exactly using any polynomial time algorithm [1,2]. In a TSP, when the distance between any two vertices x_i and x_j is equal to the distance between x_j and x_i then the problem is called Symmetric TSP (STSP) [3,4]. On the other hand, if the distance between the vertices x_i and x_j is not equal to the distance between x_j and x_i, for at least one pair of vertices then the problem belongs to Asymmetric TSP (ATSP) [5]. Generally, there are two approaches to solve a TSP: exact methods and heuristic methods. The exact methods require enormous time for larger N, thus the heuristic methods are typically used to solve a TSP. The exact methods include cutting plane [6], LP relaxation [7], branch and bound [8], branch and cut [9], etc. Only small size TSPs can be solved by exact methods in a reasonable time window. On the other hand, several TSPs have been solved using heuristics or soft computing based techniques such as Ant Colony Optimization (ACO) [3], local search [10], hybrid algorithm [11], Genetic Algorithm (GA) [12], Particle Swarm Optimization (PSO) [4], Artificial Bee Colony (ABC) [13], etc. There are several well established heuristics for STSP. Wang et al. [4] used concepts of swap operator and swap sequence, and redefined some operators of PSO on the basis of them to solve TSP. Combing the features of PSO, ACO and 3-opt a hybrid algorithm PSO-ACO-3-opt is presented by Mahi et al. [14] to solve standard TSPs. Akhand et al. [15] proposed PSO with partial search algorithm for solving TSPs. Akhand et al. [16] improved this algorithm to find solution of the TSPs and named it velocity tentative PSO. Geng et al. [17] proposed an effective local search algorithm based on Simulated Annealing (SA) and greedy search techniques to solve the TSPs. Jolai and Ghanbari [18] presented an improved Artificial Neural Network (ANN) approach for solving the TSPs. Dorigo et al. [3] proposed an Ant System to solve the TSPs. Dorigo and Gambardella [19] described an ACO capable of solving the TSPs. Bontoux and Feillet [20] proposed a hybrid algorithm to solve the TSPs. Beam-ACO algorithm, which is a hybrid method combining ACO with beam search was used to solve

* Corresponding author.
E-mail addresses: indadulkhan@gmail.com (I. Khan), manasmaiti@yahoo.co.in (M.K. Maiti).

https://doi.org/10.1016/j.swevo.2018.05.006
Received 22 November 2016; Received in revised form 19 May 2018; Accepted 19 May 2018
Available online XXX

the TSPs [21]. Gunduz et al. [22] presented a new hierarchic method based swarm intelligence algorithm for solving TSPs. Khan et al. [23] presented a hybrid algorithm based on ACO- PSO- K-otp for solving TSP. Huang et al. [24] proposed a PSO based algorithm for solving TSPs. Moreover different heuristic approaches are established their efficiency in solving other NP-hard combinatorial optimization problems like vehicle routing problem [25], flow shop schedule problem [26], etc. Recently, a detailed survey on the different bee colony algorithms developed so far for different class of optimization problems has been made by Rajasekhar et al. [27]. Depending upon the biological perspectives and motivations, they mentioned the prospects of different algorithms in the respective class of problems and compare the algorithms with the others.

From the above discussion, it can be concluded that heuristic approaches are more powerful for solving TSP in a feasible time window. Since 2005, ABC has been proven to succeed in continuous optimization problems, much work has been done effectively in this area [28–30]. Some discrete versions of ABC have already been introduced in the literature [26,31,32]. Karaboga and Gorkemli [13] proposed a new ABC algorithm called Combinatorial ABC for the TSPs. Applying features of swap sequence and swap operation, Pan et al. [26] proposed a discrete ABC for the lot-streaming flow shop scheduling problem. Iqbal et al. [33] proposed a new bee algorithm for solving multi-objective vehicle routing problem with soft time windows. When used for TSP, any heuristic algorithm sometimes converges to the local optimal path(tour). The K-opt is a perturbation technique which can be applied on a tour(path) to overcome this convergence. In fact, K-opt is the most widely used heuristic approach for searching better neighbour of any potential solution of a TSP. It works like a mutation function of a GA. The K-opt is a tour improvement algorithm, where in each step K links of the current tour are replaced by K links in such a manner that a shorter tour is achieved [34].

In this paper, a new variant of ABC algorithm is proposed to solve a STSP as well as an ATSP. In this algorithm, eight different rules are proposed to improve a solution by an employed bee or by an onlooker bee. The rules are defined using swap sequence and swap operations on a potential path (solution) of the salesman [16,35]. To improve a solution an employed bee or an onlooker bee selects a rule from the rule set randomly using Roulette Wheel(RW) selection process [36]. For a solution assigned to a scout bee, at first K-opt operation is applied for a fixed number of iterations for its possible improvement. If it is not capable of improving the solution, the solution is randomly regenerated. K-opt operation is again used at the end of the algorithm to improve the quality of the final solution (if possible). Proposed method is tested with a set of benchmark test problems from TSPLIB [37] and it is observed that the efficiency of the algorithm is adequate with respect to the accuracy and the consistency for solving standard TSPs (ATSP as well as STSP) compared to the existing algorithms in the literature.

The rest of the paper is organized as follows: in section 2, mathematical formulation of the problem is presented. In section 3, swap sequence and swap operations are discussed. In Section 4, features of ABC are discussed. K-opt operation is presented in section 5. Proposed algorithm is presented in section 6. Experimental results are discussed in section 7. A brief conclusion is drawn in section 8.

2. Problem formulation

A TSP can be represented by a graph $G = (V, E)$, where $V = \{1, 2, \dots N\}$ is the set of vertices or nodes and E is the set of edges. Here, each node represents a city and each edge represents the path between two cities. Each edge is associated with a distance which represents the distance between the cities associated with it. A salesman travels distances to visit N number of cities (or nodes) cyclically. In one tour he visits each city exactly once and finishes up where he started with a minimum travel distance. Let d_{jk} be the distance between j-th city and k-th city. Then the model can be mathematically formulated as [7],

$$\left.\begin{array}{ll}\text{Determine} & x_{jk},\ \forall j,k \in V \,\&\, j \neq k \text{ to}\\ \text{Minimize} & Z = \sum_{j=1}^{N}\sum_{k=1}^{N} x_{jk} d_{jk}\\ \text{subject to} & \sum_{j=1}^{N} x_{jk} = 1,\ \forall k \in V\\ & \sum_{k=1}^{N} x_{jk} = 1,\ \forall j \in V\end{array}\right\} \tag{1}$$

where $x_{jk} = 1$ if the salesman travels from city-j to city-k, otherwise $x_{jk} = 0$.

Let $(x_1, x_2 \dots, x_N, x_1)$ be a complete tour of a salesman, where $x_j \in V$, $\forall j \in V$and all x_j's are distinct, i.e., $(x_1, x_2 \dots, x_N, x_1)$ is the sequence of cities in which the salesman travels the cities. Then the above problem reduces to [38],

$$\left.\begin{array}{l}\text{Determine a complete tour } (x_1, x_2, \dots\dots\dots .x_N, x_1)\\ \text{to minimize } Z = \sum_{j=1}^{N-1} d_{x_j x_{j+1}} + d_{x_N x_1}\end{array}\right\} \tag{2}$$

3. Swap sequence and swap operators for TSP

The concept of swap operator and swap sequence was introduced by Wang et al. [35] in 2003 for solving TSPs using PSO. Pan et al. [26] used these operations in DABC to solve flow shop scheduling problem. Brief descriptions of these operations are given below.

Swap Operator: Consider a normal solution sequence of nodes $X = (x_1, x_2, \dots x_N, x_1)$ of a TSP with N nodes, having node set $V = \{1, 2, \dots N\}$, where $x_i \in V$and $x_i \neq x_j$, $\forall i \neq j$. Here, swap operator, $SO(i, j)$ is defined as the exchange of the node x_i and the node x_j in the solution sequence X. Then $X' = X \diamond SO(i, j)$ is defined as a new solution sequence after operating the operator $SO(i, j)$ on X. The symbol $\diamond$ represents the binary operation-swap operation.

It can be given a concrete example: suppose there is a TSP with six nodes, and $X = (x_1, x_2, x_3, x_4, x_5, x_6) = (1, 3, 5, 2, 4, 6)$ be a solution sequence. Let $SO(2, 4)$, be a swap operator then $X' = X \diamond SO(2, 4) = (1, 3, 5, 2, 4, 6) \diamond SO(2, 4) = (1, 2, 5, 3, 4, 6)$, i.e, nodes of position 2 and position 4 of X are exchanged to create the new solution sequence X'.

Swap Sequence: A collection of different swap operators in a particular order is called a swap sequence SS. Let $SS = (SO_1, SO_2, \dots, SO_N)$, where $SO_1, SO_2, \dots, SO_N$ are swap operators. Swap sequence acting on a solution means all the swap operators of the swap sequence act on the solution in order. This can be described by the following formula:

$$X' = X \diamond SS = X \diamond (SO_1, SO_2, \dots, SO_N) = (\dots((X \diamond SO_1) \diamond SO_2) \dots \diamond SO_N)$$

Different swap sequences acting on the same solution may produce the same new solution. All these swap sequences are named the equivalent set of swap sequences. In the equivalent set, the sequence which has the least number of swap operators is called Basic swap sequence of the set or Basic swap sequence (BSS) in short.

Several swap sequences can be merged into a new swap sequence. Here, the operator $\oplus$ is defined as the merging of two swap sequences into a new swap sequence. Suppose there are two swap sequences, $SS1$ and $SS2$ act on one solution X in order, namely $SS1$ first, $SS2$ second and a new solution X' is obtained. Let there are another swap sequence SS' acting on the same solution X and get the solution X', then SS' is called merging of $SS1$ and $SS2$ and it is represented as:

$SS' = SS1 \oplus SS2$

Here, SS' and $SS1 \oplus SS2$ are in the same equivalent set.

The construction of basic swap sequence: Suppose there are two solutions, A and B and our task is to construct a BSS, SS which can act on B to get solution A. Here, SS is defined as $SS = A \ominus B$. So the operator $\ominus$ represents the difference between two sequences of cities and obviously this difference is represented by some consecutive swap operations, i.e., swap the nodes in B according to A from left to right to get SS. So there must be an equation $A = B \diamond SS$. For example, consider two solutions:

$A = (1, 2, 3, 4, 5), B = (2, 3, 1, 5, 4)$

Here, $A(1) = B(3) = 1$. So the first swap operator is $SO(1, 3)$. Then operating $SO(1, 3)$ on B, the new sequence $B1 = B \diamond SO(1, 3)$ is obtained as below:

$B1 = B \diamond SO(1, 3) = (1, 3, 2, 5, 4)$

Again $A(2) = B1(3) = 2$, so the second operator is $SO(2, 3)$ and operating $SO(2, 3)$ on $B1$, new sequence $B2$ is obtained as

$B2 = B1 \diamond SO(2, 3) = (1, 2, 3, 5, 4)$

Similarly the third operator is $SO(4, 5)$, and

$B3 = B2 \diamond SO(4, 5) = (1, 2, 3, 4, 5) = A$

Hence, the BSS, $SS = A \ominus B = (SO(1, 3), SO(2, 3), SO(4, 5))$.

4. Artificial Bee Colony for TSP

Mimicking the foraging and waggle dance behaviour of honey bee in their colonies, the heuristic, ABC was introduced by Karaboga in 2005 for solving numerical optimization problems [39]. An ABC algorithm normally starts searching for optimal solution of the optimisation problem under consideration with a set of randomly generated potential solutions in the search space. In the algorithm potential solutions are analogous to the positions of different food sources. The goal of the bees is to search the best food source which is analogous to the optimal solution of the problem. Depending upon different activities, the bees can be classified into three categories: employed bee, onlooker bee and scout bee. Accordingly, each iteration of the algorithm consists of three phases, namely employed bee phase, onlooker bee phase and scout bee phase. First phase is employed bee phase. Each solution of the randomly generated solution set is associated with an employed bee. In employed bee phase each employed bee tries to improve its position (i.e., corresponding solution) for searching better food source (i.e., improved solution) using a specified rule. The algorithm consists of a set of onlooker bee. After completion of employed bee phase onlooker bee phase starts. In this phase each onlooker bee selects one solution from the solution set according to the fitness of the solutions and tries to improve it. Each solution is associated with a counter and its value is set to zero at the time of generation. If a solution is not improved by its employed bee or by any onlooker bee then its counter is increased by one. In the last phase, i.e., in scout bee phase, all the solutions are checked. If it is found that a solution is not improved by either its employed bee or by any onlooker bee in a predefined number of iterations then the solution is treated as a stagnant solution and is assigned to a scout bee. The scout bee regenerates it randomly and its corresponding counter value is set to zero.

In the proposed variant of ABC, it is considered that in the scout bee phase, a scout bee initially tries to improve its solution using K-opt operation for a predefined number of times. If any improvement is found then the solution goes to the next iteration with its counter value zero otherwise the scout bee regenerates it randomly and its counter value is set to zero. The K-opt operation is discussed in section §5. Moreover, similar to Kiran et al. [28], here a set of solution update rules (applicable for TSPs) is proposed for the updation of a solution by an employed bee or by an onlooker bee. At the time of their operation an employed bee or an onlooker bee selects a rule from the rule set using a proper selection process (which is discussed later) and tries to improve its solution using the selected rule.

Following the above discussion, to solve a TSP using proposed ABC, its different steps and procedures are briefly discussed here. As in ABC algorithm, each food source represents a potential solution of the problem, here a sequence of nodes is taken as a solution (food source) of the considered TSP. So a solution X_i of the initial solution set is randomly generated in the solution space using (3).

$$X_i = (x_{i1}, x_{i2}, \dots x_{iN}, x_{i1}),\ i = 1, 2, \dots f_s \tag{3}$$

where, $x_{ij} \in V, j = 1, 2, \dots, N,\ V = \{1, 2, \dots N\}$ is the node set, $x_{ij} \neq x_{ik}$ for $j \neq k$, f_s is the number of solutions (food sources) in the solution set. After initialization of all the solutions by equation (3), each solution is assigned to an employed bee, i.e., number of solutions equals to the number of employed bees. After generation of the initial solution set, the fitness of the i-th solution is calculated using (4).

$$fit_i = 1/(M_c - f(X_i) + 1) \tag{4}$$

where fit_i is the fitness of X_i, $f(X_i)$ is the total tour length of X_i and M_c = maximum of the set $\{f(X_1), f(X_2), \dots, f(X_{f_s})\}$.

In the employed bee phase of ABC algorithm, each employed bee tries to improve its solution by randomly selecting one of the following equations (5)–(12). The equation selection technique is discussed later.

$$Y_i = X_j \diamond (r \odot (X_i \ominus X_k)), \qquad i = 1, 2, \dots f_s, i \neq k \tag{5}$$

$$Y_i = X_i \diamond (r \odot (X_j \ominus X_k)), \qquad i = 1, 2, \dots f_s, j \neq k \tag{6}$$

$$Y_i = X_{best} \diamond (r \odot (X_i \ominus X_k)), \quad i = 1, 2, \dots f_s, i \neq j \neq k \tag{7}$$

$$Y_i = X_i \diamond (r \odot (X_i \ominus X_{best})), \quad i = 1, 2, \dots f_s, \tag{8}$$

$$Y_i = X_{best} \diamond (r \odot (X_{best} \ominus X_k)), \quad i = 1, 2, \dots f_s, i \neq k \tag{9}$$

$$Y_i = X_i \diamond (r \odot (X_{best} \ominus X_{worst})), \quad i = 1, 2, \dots f_s, i \neq k \tag{10}$$

$$Y_i = X_i \diamond (r \odot (X_{best} \ominus X_k) \oplus r_1 \odot (X_k \ominus X_i)) \tag{11}$$

$$Y_i = X_j \diamond (r \odot (X_{best} \ominus X_i)), \quad i = 1, 2, \dots f_s, i \neq j \tag{12}$$

where Y_i is a new solution derived from X_i by an employed bee. X_j, X_k are randomly selected solution from the solution set, $k \neq j$ and $k, j \in \{1, 2, \dots f_s\}$. X_{best} is the best solution obtained so far and X_{worst} is the worst (i.e low fitness) solution in the solution set, r is the random number in the range $[0, 1]$. Here, $r \odot (X_i \ominus X_k)$ means all swap operators in BSS $(X_i \ominus X_k)$ should be maintained with the probability of r, i.e., each swap operator in BSS $(X_i \ominus X_k)$ should be selected with probability r. Here, the operator $\oplus$ is defined as the merging of two swap sequences into a new swap sequence. The meaning of these operators are same for all other expressions in (5)–(12). After the generation of the candidate (new) solution Y_i, it is compared with X_i. If Y_i is better than X_i then X_i is replaced by Y_i and its counter is set to zero otherwise X_i remains unchanged and it's counter is increased by 1.

Next step of ABC algorithm is onlooker bee phase. Initially, the chance p_i of selecting i-th food source x_i by an onlooker bee is calculated by the following equation (13).

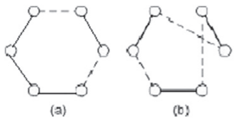


Fig. 1. All combinations of sub_toures for k = 2.

$$p_i = \frac{fit_i}{\sum_{j=1}^{E_b} fit_j}, \quad i = 1, 2, \dots O_b \tag{13}$$

Here, E_b is the number of employed bee and O_b is the number of onlooker bee in the proposed ABC algorithm (presented later). After selection, the Onlooker bee tries to improve the solution by using one of equations (5)–(12). The selection process of a rule is presented later. The solution is updated using the same way as employed bee phase. Remaining steps of RW-selection of a solution are listed below:

(i) Calculate the cumulative probability cp_i of i-th solution by the formula $cp_i = \sum_{j=1}^{i} p_j$.
(ii) Generate a random number r in the interval $(0, 1)$.
(iii) If $r < cp_1$ then select X_1. Otherwise, select X_i if $cp_{i-1} \le r < cp_i$.

The last phase of ABC algorithm is scout bee phase, which is already discussed. So newly proposed features of the algorithm can be summarised as below:

- Using swap sequence and swap operation, a set of solution update rules (Eqs. (5)–(12)) appropriate for TSP is proposed and used.
- A proper fitness evaluation rule of different solutions (food sources) appropriate for TSP is proposed (Eq. (4)). Using the fitness values RW-selection process is properly applied to select solutions for the onlooker bees.
- In scout bee phase of any ABC algorithm, normally stagnant solutions are regenerated. Here 3-opt operation is appropriately used for the possible improvement of the stagnant solutions. This approach prevents searching from newly generated solutions and helps to search solution using less number of iterations.
- Here 3-opt operation is again used for the possible improvement of the best found solution at the end of the algorithm.

Rule Selection Technique: Each of the first two phases of ABC algorithm selects a rule from the set of rules (Eqns. (5)–(12)) to improve a solution. Here, each rule is associated with a counter having initial value 1. Depending upon this counter value, RW-selection process is used to select a rule. After selection of a rule, it is used to improve a solution. If improvement takes place then counter of the rule is increased by one and this process continues throughout the algorithm. So the probability of selecting i-th rule ρ_i is given by the following equation (14).

$$\rho_i = \frac{v_i}{\sum_{j=1}^{N_{eq}} v_i}, \quad i = 1, 2, \dots N_{eq} \tag{14}$$

where ρ_i is the probability of selection of i-th rule by an employed bee or an onlooker bee, N_{eq} is the cardinality of the rule set and v_i is the value of the counter corresponding to i-th rule. Remaining steps of RW-selection of a rule are listed below:

(i) Calculate the cumulative probability $c\rho_i$ of i-th rule by the formula $c\rho_i = \sum_{j=1}^{i} \rho_j$.
(ii) Generate a random number r in the interval $(0, 1)$.
(iii) If $r < c\rho_1$ then select rule-1. Otherwise select rule-i if $c\rho_{i-1} \le r < c\rho_i$.

5. K-opt operation for TSP

K-opt is a perturbation technique [40] mainly used in TSP which is based on the exchange of K parts (sub-tours) and their reverses (reverse sub-tours) of a tour(path) of the TSP under consideration to find a better tour. Here, K-opt operation is used in scout bee phase of ABC algorithm to improve a solution. If it is assumed that nodes/cites are connected by edges, then, while breaking (removes) K edges of a solution in a tour, there are $(K - 1)!2^{K-1}$ ways to reconnect it (including the initial sequence) to form a valid sequence. Each new combination gives a new solution. Among these solutions one may produce better tour than the tour of the original solution and can be taken as an improvement. In the case of 2-opt algorithm, two edges of a solution (sequence of cities) are removed, and all possible combinations of sub-sequences and their reverses are reconnected to search better solution (Fig. 1). This process is continued until no improvement (by 2-opt) is found. Similarly in the case of 3-opt, breaking 3 edges in a sequence there are in total 8 cases of reconnection (Fig. 2) [41]. If a sequence is 3-optimal, it is also 2-optimal [40]. Continuing this process of breaking (removing) more edges form a sequence, i.e., $K = 1, 2, 3 \dots, n$, one can get new algorithms, like 2-opt, 3-opt, 4-opt and so on. But increase of K increases time complexity. Due to this, here 3-opt operation is used and it is found that it acts better than 2-opt operation for large size TSPs.

5.1. K-opt operation on a complete tour X_k

Detailed algorithm of K-opt operation for $K = 3$ is presented below. In the algorithm a one-dimensional array of size N, X_{tem_k}, is used to represent a temporary solution corresponding to X_k in iteration t. X_{ki} and X^r_{ki}, $i = 1, 2, 3$ are one-dimensional arrays used to represent sub-tours and reverse_sub-tours of the original tour of X_k. $Maxit3$ is the maximum number of iterations.

6. Proposed ABC Algorithm for TSP

It is assumed that problem involves N nodes, d_{ij} represents the distance between node i and node j. X_i is a one-dimensional array of size N, used to represent i-th solution. $f(X_i)$ represents total tour length of the solution X_i. Here, E_b represents the number of employed bees and O_b is the number of onlooker bees in ABC algorithm. Here, Ec_i is a one-dimensional array of size N_{eq}(i.e number of rules), used to represent counters of different rules. Initially, equation counters are set to 1 corresponding to all the rules. Another one-dimensional array Fc_i of size f_s(i.e numbers of food sources) is used to represent food source counters of different solutions. Initially, all food source counters are set to zero. lim is a positive integer which is used as the upper limit of the food source counter. $MaxGen$ is a positive integer that represents the maximum number of iterations and t is iteration counter. Other notations in the algorithm are same as previously discussed.

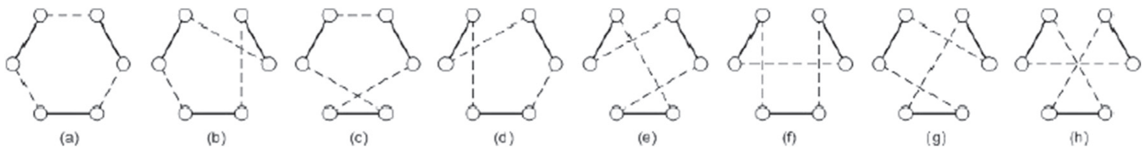


Fig. 2. All combinations of sub_toures for k = 3.

```
1:  Start Algorithm.
2:  for i=1: $Maxit3$ do
3:      Remove 3 edges(randomly selected) from complete tour $X_k$, it makes 3 sub-tour $X_{ki}$, $i = 1, 2, 3$.
4:      Reverses of the contents of these sub-tour are called reverse_sub-tour, represented as $X^r_{ki}$, $i = 1, 2, 3$, i.e., $X^r_{k1}$ = reverse_sub-tour of $X_{k1}$, $X^r_{k2}$ = reverse_sub-tour of $X_{k2}$, $X^r_{k3}$ = reverse_sub-tour of $X_{k3}$.
5:      Now combing the sub-tours $\{X_{k1}, X_{k2}, X_{k3}\}$, $\{X^r_{k1}, X^r_{k2}, X^r_{k3}\}$ new sequence of tour can be formed in following 8 combinations:
6:      $\{X_{k1}, X_{k2}, X_{k3}\}$.
7:      $\{X_{k1}, X^r_{k2}, X_{k3}\}$.
8:      $\{X_{k1}, X_{k2}, X^r_{k3}\}$.
9:      $\{X_{k1}, X^r_{k3}, X^r_{k2}\}$.
10:     $\{X_{k1}, X_{k3}, X^r_{k2}\}$.
11:     $\{X_{k1}, X^r_{k3}, X_{k2}\}$.
12:     $\{X_{k1}, X^r_{k2}, X^r_{k3}\}$.
13:     $\{X_{k1}, X^r_{k3}, X_{k2}\}$.
14:     for each combination do
15:         Create a new sequence of solution from the combination and let it be $X_{tem_k}$.
16:         Determine $f(X_{tem_k})$.
17:         if $f(X_{tem_k}) < f(X_k)$ then
18:             $X_k = X_{tem_k}$
19:         end if
20:     end for
21: end for
22: return $X_k$.
23: end Algorithm
```

6.1. Implementation of the algorithm

The algorithm is implemented in Dev C++ 5.8.3 programming language in a computer with core $i3$ CPU @ 2.10 GH$_z$, Windows 8.1 Operating System and 4 GB RAM. For implementation of the algorithm, values of different parameters of the algorithm are considered as-number of employed bee (food source), $E_b = 20$, maximum number of iterations, $MaxGen = 500$, number of onlooker bee, O_b, is considered as a function of E_b and is of the form $O_b = [\frac{E_b}{2}] + 1$, upper limit of a food source counter, lim $= 5$, $Maxit3 = 10$. Here $[\frac{E_b}{2}]$ means integral part of $\frac{E_b}{2}$.

7. Experimental results

The performance of the proposed algorithm is tested using different size standard TSPs from TSPLIB. From each problem algorithm is tested by running the program 10 times for different seeds of random number generator and the best solution obtained, the average value of the solutions, standard deviation (SD) value of the solutions and percentage of relative error (Error (%)) according to the optimal solution are calculated. The percentage of relative **Error (%)** is calculated using the following equation.

$$\textbf{Error}(\%) = \frac{\textit{average solution} - \textit{optimal solution}}{\textit{optimal solution}} \times 100 \tag{15}$$

The results obtained by the proposed algorithm for eighteen different test problems from TSPLIB are presented in Table 1.

In Table 1, results of STSPs and ATSPs are displayed separately. In the table **Best** column represents the best solution obtained by the proposed method and optimal solutions are taken from TSPLIB. The problems whose optimal solutions (according to TSPLIB) are obtained by the proposed approach are presented in bold face. It is found from Table 1 that the proposed algorithm produces optimal solution for most of the problems considered for the test and for others it gives solutions very near to the optimal solutions. For problems like, *eli*101, *kroA*200, *fiv*56, the algorithm does not provide optimal solution but other parameters like **average**, **Sd** and **Error (%)** are better compared to some existing algorithms (cf.Table 3). Also small values of **SD** and **Error (%)** of the solutions of the problems ensure that obtained solutions of the problems are very close to the corresponding optimal solutions as well as the algorithm is efficient and consistent to solve such problems. It can be concluded from Table 1 that the algorithm needs very small amount of time to find solutions of the test problems (see Table 4).

Table 2 represents the results obtained by the proposed method due to different test problems using 2-opt and 3-opt operations in the algorithm. In the case of small size problems like *gr*17 and *br*17 both the approaches provide the same solution as the optimal solution. But for large size problems 2-opt and 3-opt produces different solutions. Problems for which optimal solutions are obtained by the algorithm are presented in bold face in Table 2. In some problems like *eli*51, *kroA*200, *fiv*56, using 3-opt, the algorithm does not provide an optimal solution but it produces a better solutions than that obtained by the algorithm using 2-opt. It is also clear from Table 2 that for all the problems algorithm with 3-opt provide better result than that using 2-opt. So in the proposed algorithm 3-opt operation is used.

Table 3 represents a comparison of some computational results of the proposed algorithm with some other existing algorithms in the literature. In the table results of other algorithms are taken from Refs. [14,46]. From Table 3, it is clear that for most of the considered problems the performance of the proposed approach with respect to accuracy is better compared to the other existing approaches in the literature. For the test problems like *eli*51, *st*70, *eli*76, *rat*99, *eli*101, *kroA*200 the algorithm produces better values of *Avg*, *SD*, *Error* compared to another algorithms. For the test problem *kroA*100, proposed method provides optimal solution but other parameters like *Avg*, *SD*, *Error* are not better compared to WFA with 3-opt [45]. In some test problems, the proposed method does not provide optimal solutions but the obtained solutions are very near to the corresponding optimal solutions and also the results are better compared to other existing algorithms as the proposed algorithm produces minimum standard deviation (SD) compared to the other algorithms under consideration.

7.1. Statistical Test on the Algorithms (The Friedman Test)

Here a non-parametric statistical procedure Friedman test [47] is considered to compare the performance of the proposed algorithm with some other existing algorithms in the literature. The other considered algorithms are RABNET-TSP [42], Modified RABNET -TSP [43], SA ACO PSO [44], WFA with 2-opt [45], WFA with 3-opt [45], GA-PSO-ACO [46], PSO-ACO-3opt [14]. The main aim of the use of Friedman

```
1:  **Start Algorithm.**
2:  Set values of $N, f_s, E_b, O_b$, lim, $MaxGen, Maxit3$, and set $t = 0$.
3:  **for** $i = 1 : f_s$ **do**
4:    Randomly generate $X_i$.
5:    Determine $f(X_i)$.
6:  **end for**
7:  Set initial best solution $X_{best} = X_k$, where $f(X_k) = \min\{f(X_1), f(X_2), \dots, f(X_{E_b})\}$.
8:  Set initial worst solution $X_{worst} = X_k$, where $f(X_k) = \max\{f(X_1), f(X_2), \dots, f(X_{E_b})\}$.
9:  **repeat until** $(t > MaxGen)$
10:   $t = t + 1$.          //Employed bee phase
11:   **for** $i = 1 : E_b$ **do**
12:     Determine $Y_i$ from $X_i$ following a selected rule (Rule is selected using eqn. (14)).
13:     Determine $f(Y_i)$.
14:     **If** $f(Y_i) < f(X_i)$ then
15:       $X_i = Y_i$.
16:       $Fc_i = 0$.
17:     **else**
18:       $Fc_i = Fc_i + 1$.
19:     **end if**
20:   **end for**
21:   **for** $i = 1 : O_b$ **do**          //Onlooker bee phase
22:     Select a solution $X_k$ according to fitness (Using eqn. (13)).
23:     Determine $Y_k$ from $X_k$ following a selected rule (Rule is selected using eqn. (14)).
24:     Determine $f(Y_k)$.
25:     **If** $f(Y_k) < f(X_k)$ then
26:       $X_k = Y_k$.
27:       $Fc_k = 0$.
28:     **else**
29:       $Fc_k = Fc_k + 1$.
30:     **end if**
31:   **end for**
32:   **for** $i = 1 : f_s$ **do**          //Scout bee phase
32:     **If** $Fc_i >$ limthen
33:       Apply **K-opt operation**(§5.1) on $X_i$ and get the neighbour solution $Y_i$.
34:       $Fc_i = 0$.
35:         **If** $f(X_i) > f(Y_i)$ then
36:           $X_i = Y_i$.
37:         **else**
38:           Randomly generate $X_i$.
39:         **end if**
40:       Determine $f(X_i)$.
41:     **end if**
42:   **end for**
43:   **for** $i = 1 : f_s$ **do**
44:     **If** $f(X_{best}) > f(X_i)$ then
45:       $X_{best} = X_i$.
46:     **end if**
47:   **end for**
48:   $X_{worst} = X_1$.
49:   **for** $i = 2 : f_s$ **do**
50:     **If** $f(X_{worst}) < f(X_i)$ then
51:       $X_{worst} = X_i$.
52:     **end if**
53:   **end for**
54: **End loop**
55: Apply **K-opt operation** on $X_{best}$ a predefined number of iterations to improve its quality (if possible).
56: **Output** $X_{best}$.
57: **End of Algorithm.**
```

Table 1
The Result obtained by the proposed algorithm for some STSP and ATSP from TSPLIB.

					Statistical information of test problems				
	Problem from TSPLIB	Optimal cost of the problem	Best cost obtained	Worst cost obtained	Mean cost of the solutions	Median value of the solution	Standard deviation (SD) of the solutions	Percentage of Error (Error (%))	Computational time in second
STSP	gr17	2085	**2085**	2085	2085.00	2085	0.00	0.00	1.35
	bays29	2020	**2020**	2020	2020.00	2020	0.00	0.00	1.24
	swiss42	1273	**1273**	1273	21273.00	1273	0.00	0.00	1.37
	eli51	426	427	428	427.01	427	0.46	0.07	4.2
	berlin52	7542	**7542**	7542	7542.00	7542	0.00	0.00	3.15
	st70	675	**675**	678	675.77	675	1.17	0.18	100.24
	eli76	538	**538**	540	538.15	538	0.60	0.02	64.34
	rat99	1211	**1211**	1213	1211.50	1211	0.67	0.041	200.21
	kroA100	21282	**21282**	21305	21287.19	21284	8.10	0.02	51.32
	eli101	629	**629**	635	630.59	629	2.37	0.25	400.32
	lin105	14379	**14379**	14385	14379.10	14379	1.30	0.00	28.42
	pr124	59030	**59030**	59090	59054.64	59042	27.30	0.04	130.41
	pr152	73682	**73682**	73705	73691.64	73686	28.26	0.12	1031.32
	kroA200	29368	29450	29500	29469.00	29464	20.03	0.146	900.12
ATSP	br17	39	**39**	39	39.00	39	0.00	0.00	1.13
	ftv33	1286	**1286**	1286	1286.00	1286	0.00	0.00	5.56
	ry48	14422	**14422**	14642	14452.79	14448	64.79	0.21	15.12
	ftv56	1608	1629	1689	1642.19	1636	18.87	0.810	25.12

Table 2
Performance of the algorithm using 2-opt and 3-opt in different test problems from TSPLIB.

	Problem from TSPLIB	Optimal cost of the solution	Cost of the best path obtained by the hybrid algorithm	
			ABC + 2-opt	ABC + 3-opt
TSP	gr17	2085	**2085**	**2085**
	bays29	2020	2028	**2020**
	swiss42	1273	1284	**1273**
	eli51	426	447	427
	berlin52	7542	7800	**7542**
	st70	675	699	**675**
	eli76	538	550	**538**
	rat99	1211	1270	**1211**
	kroA100	21282	21910	**21282**
	eli101	629	795	**629**
	lin105	14379	15500	**14379**
	pr124	59030	62040	**59030**
	pr152	73628	73925	**73682**
	kroA200	29368	30290	29450
ATSP	br17	39	**39**	**39**
	ftv33	1286	1340	**1286**
	ry48	14422	14020	**14422**
	ftv56	1608	1648	1629

test here is to detect the existence of any significant difference between the behaviours of two or more algorithms. The following two assumptions are made for the Friedman Test:

- The results over instances are mutually independent (i.e. the result within one instance do not influence the result within other instances.
- Within each instance, the observations can be ranked.

Also, the following two hypothesis H_0 and H_1 are considered for the Friedman Test, where H_0 is the null hypothesis and H_1 is the alternative hypothesis.

H_0. Each ranking of the algorithms within each problem is equally likely.

H_1. At least one of the algorithms tends to yield larger objective function than at least one of the algorithms.

Here number of algorithms$(m) = 8$, number of test problems$(b) = 7$. Here, only seven test problems are considered for the statistical test because the mean cost obtained by different algorithms for other problems are not provided in the literature. The Friedman ranking table is given below which is prepared according to the average result of Table 3.

Let $A_2 = \sum_{i=1}^{m}\sum_{j=1}^{b}[R(ALi_j)]^2$, $R_i = \sum_{j=1}^{b} R(ALi_j)$ for $i = 1, 2, 3\ldots, m$ and $B_2 = \frac{1}{b}\sum_{i=1}^{m} R_i^2$. Then the test statistic is given is by $T_2 = \frac{(b-1)[B_2 - bm(m+1)^2/4]}{A_2 - B_2}$. Here from Table 3, the values of A_2 and B_2 are calculated as, $A_2 = (448 + 330 + 132 + 100 + 72 + 223 + 78 + 45) = 1428$, $B_2 = \frac{1}{7}[56^2 + 48^2 + 30^2 + 24^2 + 20^2 + 39^2 + 20^2 + 15^2] = 1351.71$. The test statistic, T_2 is calculated as $T_2 = \frac{(7-1)[1351.71 - 7\times 8(8+1)^2/4]}{1428 - 1351.71} = 17.12$. Using the table of the F distribution with a significance level $\alpha = 0.01$, it is found that $F_{(1-\alpha),(m-1),(b-1)(m-1)} = F_{0.99,7,42} \simeq 3.12$. Since $T_2 > F_{0.99,7,42}$, the null hypothesis H_0 is rejected and so H_1 is accepted. Hence, there exist some algorithms whose performances are significantly different from others.

Table 3
Comparison of results obtained by the proposed approach with some other approaches in the literature.

Different Method (Symbolic name)	Test problem	eli51	berlin52	st70	eil76	rat99	kroA100	eil101	lin105	kroA200
RABNET-TSP [42] (AL1)	Avg.	438.70	8073.97	–	556.10	–	21868.47	654.83	14702.17	30257.53
	SD	3.52	270.14	–	8.03	–	245.76	6.57	328.37	342.98
	Error (%)	2.98	7.05	–	3.36	–	2.76	4.11	2.25	3.03
Modified RABNET -TSP [43] (AL2)	Avg.	437.47	7932.50	–	556.33	–	21522.73	648.64	14400.7	30190.27
	SD	4.20	277.25	–	5.30	–	93.34	3.85	44.03	273.38
	Error (%)	2.69	5.18	–	3.41	–	1.13	3.12	0.15	2.80
SA ACO PSO [44] (AL3)	Avg.	427.27	7542.00	–	540.20	–	21370.30	635.23	14406.37	29738
	SD	0.45	0.00	–	2.94	–	123.36	3.59	37.28	356.07
	Error (%)	0.30	0.00	–	0.41	–	0.41	0.99	0.19	1.26
WFA with 2-opt [45] (AL4)	Avg.	426.65	7542.00	–	541.22	–	**21282.00**	639.87	14379.2500	29654.03
	SD	0.66	0.00	–	0.66	–	0.00	2.88	0.00	151.42
	Error (%)	0.15	0.00	–	0.60	–	0.00	1.73	0.00	0.97
WFA with 3-opt [45] (AL5)	Avg.	426.60	7542	–	539.44	–	21282.80	633.50	14459.40	29646.50
	SD	0.52	0.00	–	1.51	–	**0.00**	3.47	1.38	110.91
	Error (%)	0.14	0.00	–	0.27	–	**0.00**	0.72	0.56	0.95
GA-PSO-ACO [46] (AL6)	Avg.	431.84	7544.37	694.6	550.16	1275	21305	637.93	14521	31015
	SD	–	–	–	–	–	–	–	–	–
	Error (%)	1.37	0.03	2.90	2.26	5.28	3.5	1.41	0.98	5.60
PSO-ACO-3opt [14] (AL7)	Avg.	426.45	7543.20	678.20	538.30	1227.40	21445.10	632.70	14379.15	29646.05
	SD	0.61	2.37	1.47	0.47	1.98	78.24	2.12	**0.48**	114.71
	Error (%)	0.11	0.02	0.47	0.06	0.28	0.77	0.59	**0.00**	0.95
Proposed Method (AL8)	**Avg.**	427.01	**7543.00**	**675.77**	**538.15**	**1211.50**	21287.19	**630.59**	**14379.10**	**29469.00**
	SD	**0.46**	**0.00**	**0.00**	**0.60**	**0.67**	8.01	**2.37**	1.30	**20.03**
	Error (%)	**0.07**	**0.00**	**0.18**	**0.02**	**0.04**	0.02	**0.25**	**0.00**	**0.146**

Table 4
Ranking of the Friedman Test.

b	Method(m) rank(R)	RABNET-TSP(AL1) $R(AL1_b)$	Mo.RABNET-TSP(AL2) $R(AL2_b)$	SA-ACO-PSO(AL3) $R(AL3_b)$	WFA with-2-opt(AL4) $R(AL4_b)$	WFA with-3-opt(AL5) $R(AL5_b)$	GA-PSO-ACO(AL6) $R(AL6_b)$	PSO-ACO-3Opt(AL7) $R(AL7_b)$	Proposed-Method(AL8) $R(AL8_b)$
1	eli51	8	7	5	3	2	6	1	4
2	berlin52	8	7	3	2	1	6	5	4
3	eil76	8	7	4	5	3	6	2	1
4	kroA100	8	7	5	1	2	4	6	3
5	eil101	8	7	4	6	3	5	2	1
6	lin105	8	7	4	3	6	5	2	1
7	kroA200	8	6	5	4	3	7	2	1
Average rank		7.0	6.8	4.2	3.4	2.8	5.5	2.8	2.1
$R_i = \sum_{j=1}^{b} R(ALi_j)\, fori\ =\ 1,2,3\ldots,m$		56	48	30	24	20	39	20	15

7.2. (Post Hoc) paired comparison

This method is used to know if algorithms ALi and ALj are considered different after the rejection of the null hypothesis with the Friedman test. Following this technique [47] calculate the absolute difference of the summation of the ranks of the algorithms ALi and ALj and declare ALi and ALj different if

$$|R_i - R_j| > t_{1-\frac{\alpha}{2}}\left[\frac{2b(A_2 - B_2)}{(b-1)(m-1)}\right]^{\frac{1}{2}}$$

where $t_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$ quantile of the $t-$ distribution with $(b\ -\ 1)(m\ -\ 1)$ degrees of freedom. Here, value of $t_{1-\frac{\alpha}{2}}$ for $\alpha\ =\ 0.01$ and 42 degree of freedom is $\simeq$2.7 and the critical value of the difference is $2.7[\frac{2\times7(1428-1351.71)}{6\times7}]^{\frac{1}{2}} = 13.6$. The following table summarizes the paired compression; underlined values indicate that the algorithms are different.

From Table 5, it can be concluded that, proposed algorithm have not outperformed than all other algorithms but it perform $\simeq$60% better compared to another algorithms. All the algorithms have solved

Table 5
Paired comparison of the Friedman Test.

$\|R_i\ -\ R_j\|$	Mo.RABNET-TSP(AL2)	SA-ACO-PSO(AL3)	WFA with-2-opt(AL4)	WFA with-3-opt(AL5)	GA-PSO-ACO(AL6)	PSO-ACO-3-opt(AL7)	Proposed-Method(AL8)
RABNET-TSP(AL1)	8	26	32	36	17	36	41
Mo.RABNET-TSP(AL2)	–	18	24	28	9	28	33
SA-ACO-PSO(AL3)	–	–	6	10	9	10	15
WFA with-2-opt(AL4)	–	–	–	4	15	4	9
WFA with-3-opt(AL5)	–	–	–	–	19	0	5
GA-PSO-ACO(AL6)	–	–	–	–	–	19	24
PSO-ACO-3-opt(AL7)	–	–	–	–	–	–	5

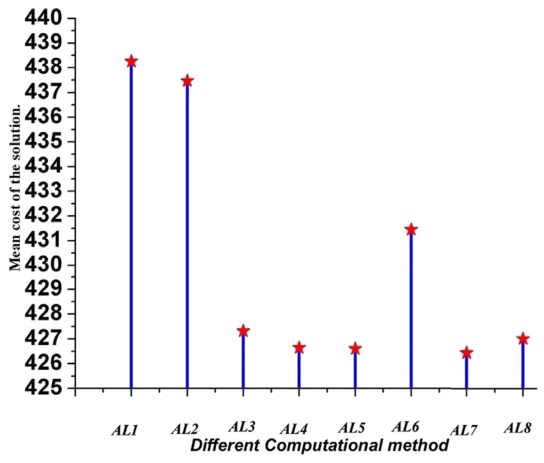


Fig. 3. Comparison of the mean costs of the solutions obtained by different approaches for instances eil51.

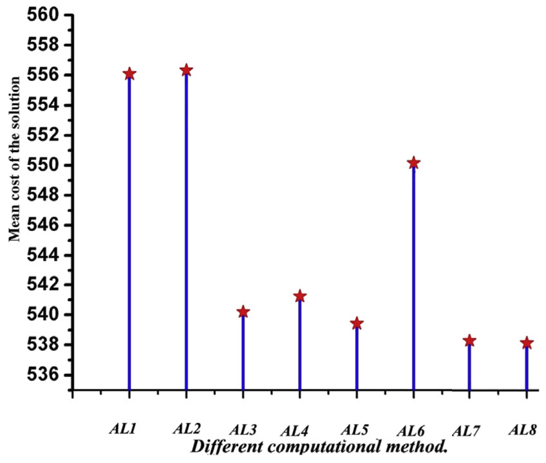


Fig. 6. Comparison of the mean costs of the solutions obtained by different approaches for instance eil76.

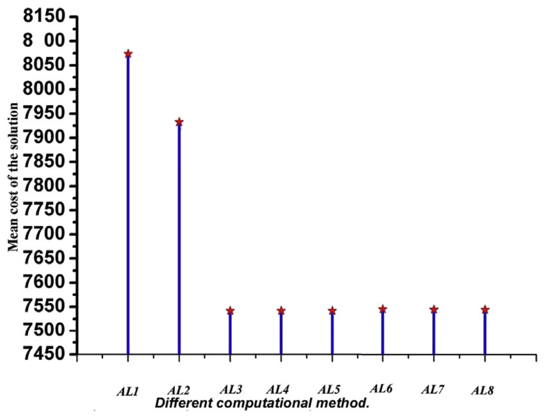


Fig. 4. Comparison of the mean costs of the solutions obtained by different approaches for instance berlin52.

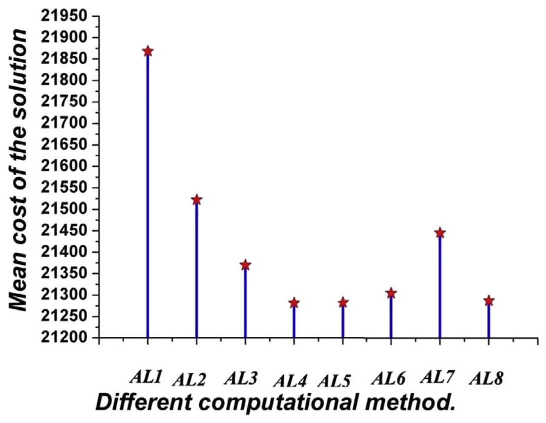


Fig. 7. Comparison of the mean costs of the solutions obtained by different approaches for instance KroA100.

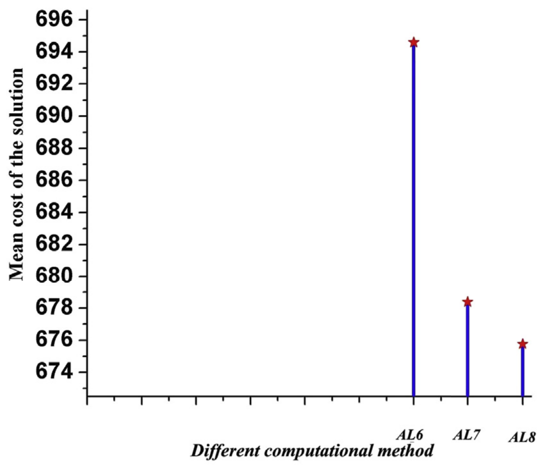


Fig. 5. Comparison of the mean costs of the solutions obtained by different approaches for instance st70.

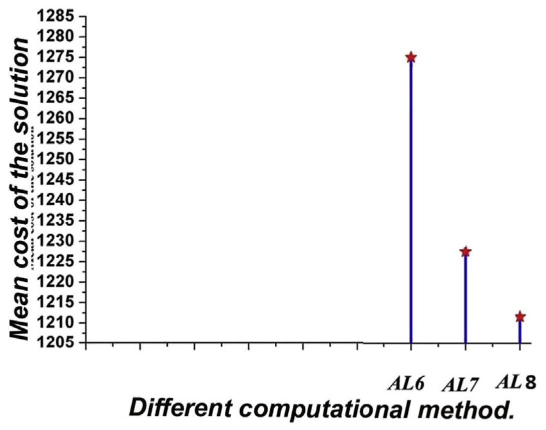


Fig. 8. Comparison of the mean costs of the solutions obtained by different approaches for instance rat99.

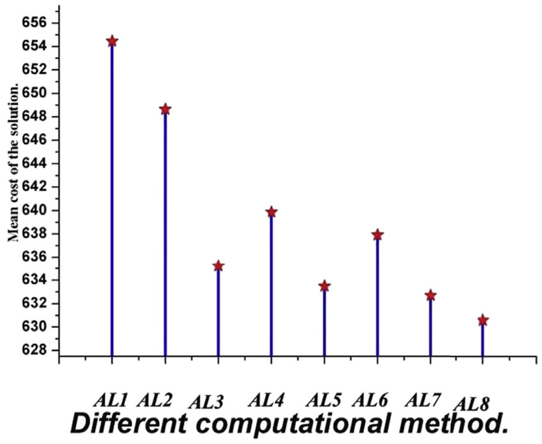


Fig. 9. Comparison of the mean costs of the solutions obtained by different approaches for instance eil101.

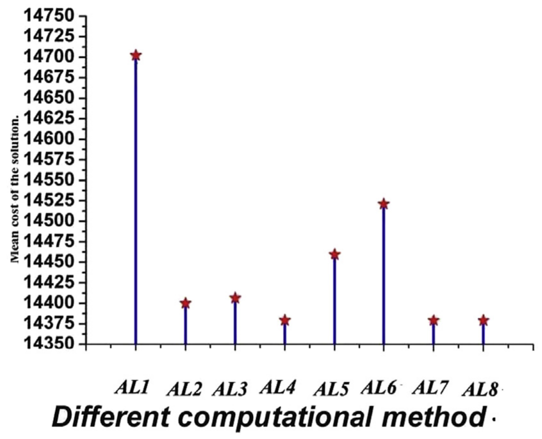


Fig. 10. Comparison of the mean costs of the solutions obtained by different approaches for instance lin105.

the same problem but using difference approaches. Proposed algorithm introduces a new concept of hybridization of ABC with the features of swap sequence, swap operation and 3-opt to solve TSP. ABC generally used for continuous optimization problems; here it is used to solve discrete optimization problems.

Mean costs of the solutions of each problem obtained in different runs by different algorithms are plotted in a separate figure. Figs. 3–11 represent such plottings of nine different problems from TSPLIB. In each figure, the symbols- AL1,AL2, AL3, AL4, AL5,AL6,AL7,AL8 represent symbolic names of different methods (cf., Table 3) used to solve the corresponding TSP. From these figures a comparison of average performances of different algorithms for solving different benchmark test problems of TSPLIB can be easily made. From figures it is clear that performance of the proposed algorithm is better than most of the existing algorithms and for others also it's performance is adequate.

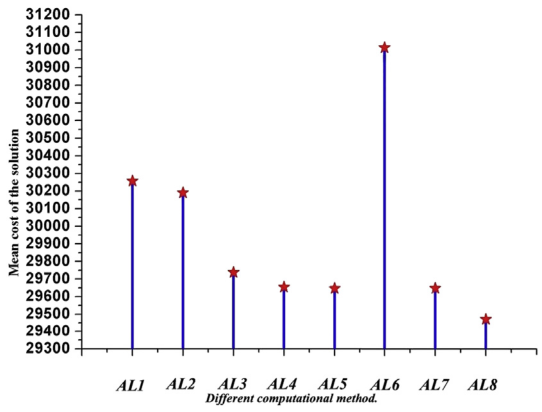


Fig. 11. Comparison of the mean costs of the solutions obtained by different approaches for instance KroA200.

8. Conclusion

Here, for the first time the features of swap sequence and swap operation on a permutation of the set of nodes of a TSP are used to define a set of solution updation rules of ABC algorithm applicable for TSP and using them a new variant of ABC is presented to solve STSP as well as ATSP. In the proposed method number of onlooker bees is a linear function of the number of employed bees. Scout bee phase of any ABC algorithm generally regenerates stagnant solutions. Here, 3-opt is used for the possible improvement of any stagnant solution in this phase. Not only that at the end of the search process by the algorithm here 3-opt is again imposed on the best found solution a predefined number of times for its improvement (if possible). All the experimental results imply that the performance of the proposed approach is adequate compare to the other existing approaches in the literature with respect to accuracy, efficiency and consistency. So, it can be concluded that ABC can be used to solve discrete optimization problems successfully by proper modification. The algorithm can be easily used to solve TSPs in fuzzy environment and solid TSPs with minor modification.

References

[1] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization, John Wiley & Sons, New York, 1985.
[2] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21 (2) (1973) 498–516.
[3] M. Dorigo, V. Maniezzo, A. Colorni, Ant system:optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B Cybern. 26 (1) (1996) 29–41.
[4] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, Q.X. Wang, Particle swarm optimization-based algorithms for TSP and generalized TSP, Inf. Process. Lett. 103 (5) (2007) 169–176.
[5] J. Majumdar, A.K. Bhunia, Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times, J. Comput. Appl. Math. 235 (9) (2011) 3063–3078.
[6] P. Miliotis, Using cutting planes to solve the symmetric travelling salesman problem, North-Holland Publishing Company, Math. Program. 15 (1978) 177–188.
[7] G.B. Dantzig, D.R. Fulkerson, S.M. Johnson, Solution of large scale traveling salesman problem, J. Oper. Res. Soc. Am. 2 (4) (1954) 393–410.
[8] M. Padberg, G. Rinaldi, Optimization of a 532-city symmetric traveling salesman problem by branch and cut, Oper. Res. Lett. 6 (1) (1987) 1–7.
[9] M.W. Petberg, S. Homg, On the symmetric traveling salesman problems: a computational study, Math. Program. Study 12 (1980) 61–77.
[10] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, M. Yagiura, Effective local search algorithm for routing and scheduling problems with general time window constraints, Transport. Sci. 39 (2) (2005) 206–232.
[11] F. Focacci, A. Lodi, M. Milano, A hybrid exact algorithm for the TSPTW, INFORMS, J. Comput. (2002) 403–417.
[12] H.D. Nguyen, I. Yoshihara, K. Yamamori, M. Yasunaga, Implementation of an effective hybrid GA for large scale traveling salesman problem, IEEE Trans. Syst. Man Cybern. B Cybern. 37 (1) (2007) 92–99.
[13] D. Karaboga, B. Gorkemli, A combinatorial artificial bee colony algorithm for traveling salesman problem, in: INISTA 2011: International Symposium on Innovations in Intelligent Systems and Applications, 2011. Istanbul, Turkey.
[14] M. Mahi, O.K. Baykan, H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, Appl. Soft Comput. 30 (2015) 484–490.
[15] M.A.H. Akhand, S. Akter, S.S. Rahman, M.M.H. Rahman, Particle swarm optimization with partial search to solve traveling salesman problem, in: Computer and Communication Engineering (ICCCE), 2012 International Conference on, IEEE, 2012.
[16] M.A.H. Akhand, Shahina Akter, M.A. Rashid, Velocity tentative particle swarm optimization to solve TSP, in: 2013 International Conference on Electrical Information and Communication Technology (EICT), IEEE Conference Publications, 2014, pp. 1–6.

[17] X.T. Geng, Z.H. Chen, W. Yang, D.Q. Shi, K. Zhao, Solving the traveling sales-man problem based on an adaptive simulated annealing algorithm with greedy search, Appl. Soft Comput. 11 (4) (2011) 3680–3689.

[18] F. Jolai, A. Ghanbari, Integrating data transformation techniques with Hopfield neural networks for solving traveling salesman problem, Expert Syst. Appl. 37 (2010) 5331–5335.

[19] M. Dorigo, L.M. Gambardella, Ant colonies for the traveling salesman problem, Biosystems 43 (1997) 73–81.

[20] B. Bontoux, D. Feillet, Ant colony optimization for the traveling purchaser problem, Comput. Oper. Res. 35 (2) (2008) 628–637.

[21] M. Lopez-Ibanez, C. Blum, Beam-ACO for the traveling salesman problem with time windows, Comput. Oper. Res. 37 (9) (2010) 1570–1583.

[22] M. Gunduz, M.S. Kiran, E. Ozceylan, A hierarchic approach based on swarm intelligence to solve traveling salesman problem, Turk. J. Electr. Eng. Comput. Sci. 23 (2015) 103–117.

[23] I. Khan, M.K. Maiti, M. Maiti, Coordinating particle swarm optimization ant colony optimization and K-opt algorithm for travelling salesman problem, in: International Conference on Mathematics and Computing, Communication in Computer and Information Science, vol. 655, Springer, 2017, pp. 103–119.

[24] L. Huang, K. Wang, C. Zhou, W. Pang, L. Dong, L. Peng, Particle swarm optimization for traveling salesman problems, Acta Sci. Nat. Univ. Jilinebsis 4 (2003).

[25] Sumaiya Iqbal, M. Sohel Rahman, Vehicle routing problems with soft time windows, in: Electrical & Computer Engineering (ICECE), 2012 7th International Conference on, IEEE, 2012.

[26] Q.K. Pan, M. Fatih Tasgetiren, P.N. Suganthan, T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Inf. Sci. 181 (2011) 2455–2468.

[27] A. Rajasekhar, N. Lynnb, S. Das, P.N. Suganthan, Computing with the collective intelligence of honey Bees – a survey, Swarm Evolut. Comput. 32 (2017) 25–48.

[28] M.S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Inf. Sci. 300 (2015) 140–157.

[29] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. Sci. 214 (2009) 108–132.

[30] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.

[31] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Appl. Soft Comput. 9 (2) (March 2009) 625–631.

[32] Li-Pei Wong, M.Y.H. Low, C.S. Chong, Bee colony optimization with local search for traveling salesman problem, Int. J. Artif. Intell. Tool. 19 (03) (2010) 305–334.

[33] Sumaiya Iqbal, M. Kaykobad, M. Sohel Rahman, Solving the multi-objective vehicle routing problem with soft time windows with the help of bees, Swarm Evolut. Comput. 24 (2015) 50–64.

[34] K. Helsgaun, General k-opt submoves for the Lin Kernighan TSP heuristic, Mathem. Program. Comput. 1 (2009), 199–163.

[35] K.P. Wang, L. Huang, C.G. Zhou, W. Pang, Particle swarm optimization for traveling salesman problem, in: International Conference on Machine Learning and Cybernetics (2003), IEEE Xplore, 2004, ISBN: 0-7803-7865-2, pp. 1583–1585.

[36] Z. Michalewicz, Genetic Algorithms+Data Structure = Evolution Programs, 27. Z. Michalewicz, Springer-Verlag, Berlin Heidelderg, 1992.

[37] G. Reinelt, TSPLIB -a traveling salesman problem library, Oper. Res. Soc. Am. J. Comput. 3 (1991) 376–384.

[38] A. Khanra, M.K. Maiti, M. Maiti, Profit maximization of TSP through a hybrid algorithm, Comput. Ind. Eng. 88 (2015) 229–236.

[39] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Kayseri, Turkey, 2005.

[40] G. Sierksma, Hamiltonicty and the 3-Opt Procedure for the traveling salesman problem, Appl. Math. 22 (2) (2014) 351–358.

[41] I. Khan, M.K. Maiti, A novel hybrid algorithm for generalized traveling salesman problems in different environments, Vietnam J. Comput. Sci. (2018), https://doi.org/10.1007/s40595-017-0099-z, Springer.

[42] R. Pasti, L.N. De Castro, A neuro-immune network for solving the traveling sales-man problem, in: International Joint Conference on Neural Networks, 2006. IJCNN'06, IEEE, 2006, pp. 3760–3766.

[43] T.A.S. Masutti, L.N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Inf. Sci. 179 (10) (2009) 1454–1468.

[44] S.M. Chen, C.Y. Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Syst. Appl. 38 (2011) 14439–14450.

[45] Z.A. Othman, A.I. Srour, A.R. Hamdan, P.Y. Ling, Performance water flow-like algorithm for TSP by improving its local search, Int. J. Adv. Comput. Technol. 5 (14) (2013) 126.

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

[47] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonpara-metric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (2011) 3–18.