Neural Computing and Applications (2022) 34:7627–7652
https://doi.org/10.1007/s00521-021-06883-x

ORIGINAL ARTICLE

# A list-based simulated annealing algorithm with crossover operator for the traveling salesman problem

İlhan İlhan[1] · Gazi Gökmen[2]

Received: 15 June 2021 / Accepted: 19 December 2021 / Published online: 9 January 2022


**Abstract**
The traveling salesman problem (TSP) is one of the most popular combinatorial optimization problems today. It is a problem that is easy to identify but hard to solve. Therefore, it belongs to the class of NP-hard optimization problems, and it is a problem of high time complexity. The TSP can be used to solve various real-world problems. Therefore, researchers use it as a standard test bench for performance evaluation of new algorithms. In this study, a new simulated annealing algorithm with crossover operator was proposed, and it was called LBSA-CO. The LBSA-CO is a population-based metaheuristic method. In this method, a list-based temperature cooling schedule, which can adapt to the topology of the solution space of the problem, was used. The solutions in the population were improved with the inversion, insertion and 2-opt local search operators. The order crossover (OX1) and genetic edge recombination crossover (ER) operators were applied to the improved solutions to accelerate the convergence. In addition, the Taguchi method was used to tune the parameters of the LBSA-CO. The proposed method was tested on 65 well-known TSP instances. The results indicated that this method performs better than the state-of-the-art methods on many instances.

**Keywords** Cooling schedule · Genetic edge recombination crossover · Order crossover · Simulated annealing · The Taguchi method · The traveling salesman problem

## 1 Introduction

The traveling salesman problem (TSP) is one of the most frequently studied combinatorial optimization problems. It was introduced by Leonhard Euler in 1759 and was defined mathematically by Karl Menger in 1932. It is a problem that is easy to identify but hard to solve. Therefore, it belongs to the class of NP-hard optimization problems [1]. The TSP can be used to solve various real-world problems, such as scheduling, route planning, placement of goods in warehouses, facility layout design in factories and printed circuit design. Researchers who work on optimization techniques use it as an ideal testing environment [2].

The approaches used to solve the TSP are divided into three categories: exact, heuristic and metaheuristic methods. Exact methods include such methods as dynamic programming [3], branch-and-bound [4], branch-and-cut [5], cutting plane [6] and LP relaxation [7]. These methods can solve small-size TSPs in a reasonable time. However, as the size of the problem increases, the execution times of exact methods also increase. As a result, it becomes impossible for these methods to achieve optimum results. Heuristic approaches include such methods as k-opt, Lin-Kernighan (LK) [8], chained LK (CLK) [9], Helsgaun's LK (LKH) [10], LKH-2 [11], Dong's approach [12] and tour merging [13]. These methods can provide quality solutions for well-known TSP benchmarks in a reasonable time [14]. However, there is no guarantee that they can provide optimal solutions. Heuristic algorithms seek solutions through trial and error. These algorithms usually work fine, but there are instances in which they fail to work.

✉ İlhan İlhan
ilhan@erbakan.edu.tr

Gazi Gökmen
gazigokmen@gmail.com

[1] Department of Mechatronic Engineering, Faculty of Engineering, Necmettin Erbakan University, 42090 Konya, Turkey

[2] Department of Educational Sciences, Institute of Educational Sciences, Necmettin Erbakan University, 42090 Konya, Turkey

They can be used in cases where easily accessible solutions, rather than the best ones, would suffice [15]. Metaheuristic methods are used to develop heuristic algorithms that combine local search and randomization procedures. These methods generally outperform simple heuristic methods, and they can produce satisfactory results within a short running time [15].

Recently, many studies have been conducted using metaheuristic algorithms to solve the TSP [14, 16–24]. For instance, Nagata and Kobayashi [14] proposed a powerful genetic algorithm using the edge assembly crossover (EAX) operator. They used the local and global versions of the operator EAX in the algorithm. The initial solutions were created using the 2-opt algorithm and the diversity in the population was evaluated using the edge entropy. The suggested algorithm was tested on 57 well-known benchmark instances. Hussain et al. [16] proposed a new crossover operator, called improved cycle crossover (ICX). The proposed operator was used in the genetic algorithm (GA). The operator ICX was tested on 10 well-known instances. Saji and Riffi [17] suggested a new discrete bat algorithm (DBA). They used 2-exchange crossover heuristics to generate the new solution. Additionally, they applied the 2-opt local search algorithm to improve it. The DBA was tested on 41 well-known instances. Ezugwu and Adewumi [18] proposed the discrete symbiotic organisms search (DSOS) algorithm. They used swap, insertion and inversion as the local search operators. A roulette wheel selection strategy was used to evaluate the probability of individual organisms to be selected. The DSOS was tested on 23 well-known instances. Akhand et al. [19] suggested the velocity tentative particle swarm optimization (VTPSO) algorithm, which considers the swap sequence for the velocity operation of the particles. A velocity swap sequence was represented as a collection of several swap operators. The VTPSO was tested on 45 well-known instances. Zhan et al. [20] proposed a list-based simulated annealing (LBSA) algorithm. A temperature list was created in the LBSA, and this list was used to decide whether the candidate solution would be accepted or not. The temperature list was updated using different update structures. Swap, insertion and inversion were used as the local search operators. The LBSA was tested on 40 well-known instances. Osaba et al. [21] proposed an improved discrete bat algorithm (IBA) to solve the symmetric and asymmetric TSPs. In the IBA, the 2-opt and 3-opt algorithms were used as the local search operators. The distances between bats in the population were measured using the Hamming distance. The IBA was tested on 37 well-known instances. Hatamlou [22] proposed the black hole (BH) algorithm to solve the TSP. The BH was tested on 10 well-known instances. Khan and Maiti [23] proposed a swap sequence based artificial bee colony (ABCSS) algorithm. The swap sequence and swap operators were used in the ABCSS. In the employed and onlooker bee phases, a solution was developed using one of eight rules. The rule to be used was determined using roulette wheel selection. The 3-opt operator was used to improve solutions in the scout bee phase. The ABCSS was tested on nine well-known instances. Akhand et al. [24] proposed an algorithm called discrete spider monkey optimization (DSMO) to solve the TSP. The swap sequence and swap operators were employed to enable interaction among the monkeys in the DSMO. The spider monkeys were updated based on the experiences of local and global leaders, in line with the two main steps of the DSMO. The DSMO was tested on 45 well-known instances.

Besides the studies outlined above, hybrid methods including several metaheuristic algorithms have been proposed to solve the TSP [25–28]. For instance, Gündüz et al. [25] proposed a hierarchical approach based on swarm intelligence. This approach aimed to overcome the stagnation behavior of the ant colony optimization (ACO) algorithm. To do this, the artificial bee colony (ABC) algorithm was used in the second half of the iterations. The proposed approach was tested on six well-known benchmark instances. Mahi et al. [26] proposed a hybrid method called PSO-ACO-3Opt. The alpha and beta parameters of the ACO were optimized by the particle swarm optimization (PSO), and the obtained solutions were improved using the 3-opt operator. The PSO-ACO-3Opt was tested on 10 well-known instances. Ezugwu et al. [27] proposed a simulated annealing based symbiotic organisms search (SOS-SA) algorithm. The swap operator was used as the local search operator in the SOS-SA. The simulated annealing (SA) was used to ensure that the symbiotic organisms search (SOS) was not trapped in the local minimum. Simultaneously, the SA increased the level of diversity in the population while searching for the optimum solution in the problem search space. The SOS-SA was tested on 40 well-known instances. Wang et al. [28] proposed a multiagent SA algorithm with instance-based sampling (MSA-IBS). The MSA-IBS effectively used the learning ability of the instance-based search algorithm. In addition, it managed to avoid premature stagnation, thanks to the probabilistic accepting criterion of the SA. The MSA-IBS used swap, insertion and inversion as the local search operators. It was tested on 60 well-known instances.

In this study, a new simulated annealing algorithm with crossover operator was proposed, which is called the LBSA-CO. The LBSA-CO is a population-based metaheuristic method. In this method, a list-based temperature cooling schedule, which can adapt to the topology of the solution space of the problem, was used. The solutions in the population were improved through the inversion, insertion and 2-opt local search operators. Moreover, the

order crossover (OX1) and genetic edge recombination crossover (ER) operators were applied to the improved solutions to accelerate the convergence. Finally, the Taguchi method was used to tune the parameters of the LBSA-CO.

The aim of this study is to show that the LBSA-CO can be an alternative and competitive method to solve the TSP. For this, the LBSA-CO was tested on 65 well-known TSP instances. The obtained results were compared with those of seven methods such as the SA, VTPSO [19], LBSA [20], IBA [21], ABCSS [23], DSMO [24] and GSAACS [29]. Furthermore, its performance was compared with those of other methods using the Friedman and Wilcoxon signed-rank tests.

## 2 Traveling salesman problem

The TSP can be described as an undirected weighted graph denoted as $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is a set of vertices or nodes, and $E = \{(i, j): i, j \in V\}$ is a set of edges. Here, each node represents a city, and each edge represents the path between two cities. Each edge has a cost $d_{ij} > 0$, where the $d_{ij}$ is the travel distance from City $i$ to $j$. In the TSP, a salesman or vehicle is required to visit every city only once by starting from an initial city and to return to the starting point. Meanwhile, the aim is to minimize the total travel distance. The cost is represented by the total travel distance. The TSP is mathematically formulated as follows [7]:

The decision variable:

$$x_{ij} = \begin{cases} 1, & \text{if the path goes from the city } i \text{ to } j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The objective function:

$$Minimize \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} d_{ij} x_{ij} \quad (2)$$

Subject to:

$$\sum_{i=1, j \neq i}^{n} x_{ij} = 1, j \in V \quad (3)$$

$$\sum_{j=1, i \neq j}^{n} x_{ij} = 1, i \in V \quad (4)$$

$$\sum_{i,j \in S, i \neq j}^{n} x_{ij} \leq |S| - 1, \forall S \subset V \quad (5)$$

$$x_{ij} \in \{0, 1\}, i \neq j; i, j \in V \quad (6)$$

The binary decision variable $x_{ij}$, which shows whether the salesman travels from City $i$ to $j$, is provided in (1). The objective function is presented in (2). Each city needs to be visited only once. This constraint is specified in (3) and (4). Any possible sub-tours should be avoided. This constraint is formulated in (5). The variable $x_{ij}$ equals to integer 0 or 1. This constraint is provided in (6).

If the complete tour of a salesman is $(\pi_1, \pi_2, \ldots. \pi_n)$, where $\pi_j \in V$, $\forall j \in V$ and all $\pi_j$'s are distinct, then the TSP can mathematically be formulated as follows [30]:

The complete tour:

$$\pi = (\pi_1, \pi_2, \ldots. \pi_n) \quad (7)$$

The objective function:

$$Minimize \sum_{j=1}^{n-1} d_{\pi_j \pi_{j+1}} + d_{\pi_n \pi_1} \quad (8)$$

## 3 The proposed method

The simulated annealing (SA) algorithm is a trajectory-based metaheuristic method, which was introduced by Kirkpatrick et al. [31]. The SA begins searching with a single randomly generated solution. It iteratively improves this solution and forms a search trajectory in the solution space. It mimics the metallurgical annealing process. However, the SA suffers from the following downsides: (1) Its performance significantly depends on the quality of the starting point. (2) The solution space may not be fully explored through a single solution [32]. (3) It takes a long computation time to find a reasonable solution. These drawbacks are apparent, especially when the problem has multiple dimensions and many local optima [32]. In the present study, the SA was transformed into a population-based metaheuristic method to eliminate the aforementioned drawbacks. A list-based temperature cooling schedule which can adapt to the topology of the solution space of the problem was used [33]. Furthermore, the population of the solutions was processed by crossover operators so that the convergence could be accelerated. The proposed algorithm was called the LBSA-CO.

The pseudocode of the LBSA-CO is presented in Algorithm 1. As seen in the algorithm, procedure InitializeParameters() was used to initialize the parameters I, P, T, Len, CR and $TSP_{INS}$, whose meanings are provided in the algorithm. The $TSP_{INS}$ is a file with a ".tsp" extension. The file contains information such as the number of cities, their coordinates and edge weight type. This information was loaded using procedure LoadTSPInstance(TSPINS) and the distance matrix was calculated using the coordinates of the cities and edge weight type. The following

subsections elaborate on the other procedures in the algorithm.

**Algorithm 1.** The pseudocode of the LBSA-CO.

**Input:** The parameter values;
I: The number of iterations, P: The population size, T: The number of trials
Len: The length of the temperature list, CR: The crossover rate, $TSP_{INS}$: The TSP instance

**Output:** The best solution and its fitness value;
$p_{best}$: The best solution, $f(p_{best})$: The fitness value of the best solution

1 **begin**
*// The initialization of the parameters I, P, T, Len, CR and $TSP_{INS}$*
2 InitializeParameters()
*// The loading of the test instance $TSP_{INS}$*
*// M includes the number of cities and the distance matrix*
3 M = LoadTSPInstance($TSP_{INS}$)
*// The generation of the initial population and temperature list*
4 (p, Lst) = GenerateInitialPopulationAndTemperatureList(M, P, Len)
*// The start of iterations*
5 **for** k = 1 **to** I
6 **for** i = 1 **to** P
*// The finding of the maximum value from the temperature list of the $i^{th}$ solution*
7 (tMax, tIdx) = max($Lst_i$(:))
8 s = 0, c = 0
9 **for** j = 1 **to** T
*// The selection of the local search operator with a probability of 0.5*
10 **if** random(0,1) < 0.5 **then**
11 $p_{new}$ = ApplyInversionOperator($p_i$)
12 **else**
13 $p_{new}$ = ApplyInsertionOperator($p_i$)
14 **end if**
15 $\Delta f = f(p_{new}) - f(p_i)$
16 **if** $\Delta f < 0$ **then**
17 $p_i = p_{new}$
18 **else**
19 r = random(0,1)
20 **if** r < e^(-$\Delta$f / tMax) **then**
*// The accepting of the worse solution*
21 $p_i = p_{new}$
22 s = s + (-$\Delta$f / ln(r))
23 c = c + 1
24 **end if**
25 **end if**
26 **end for**
*// The updating of the maximum value from the temperature list of the $i^{th}$ solution*
27 **if** c > 0 **then**
28 $Lst_i$(tIdx) = s / c
29 **end if**
*// The applying of 2-opt operator on the $i^{th}$ solution*
30 $p_i$ = Apply2-OptOperator(M, $p_i$)
31 **end for**
*// The applying of the crossover operator on the population p*
32 p = ApplyCrossoverOperator(CR, p, $p_{best}$)
*// The updating of the best solution*
33 (fMin, fIdx) = min(f(:))
34 **if** fMin < $f(p_{best})$ **then**
35 $p_{best} = p_{fIdx}$
36 **end if**
*// The end of iterations*
37 **end for**
*// The applying of 3-opt operator on the best solution $p_{best}$*
38 $p_{best}$ = Apply3-OptOperator(M, $p_{best}$)
39 **return**($p_{best}$, $f(p_{best})$)
40 **end**

## 3.1 The initial population and temperature list

The pseudocode used for generating the initial population and temperature list is presented in Algorithm 2. As seen in the algorithm, each solution was randomly generated as a permutation of the set of cities. Then, each solution was developed using the 2-opt algorithm. In Steps 5 through 17, they were further developed with the inversion and insertion local search operators. Simultaneously, the initial temperature list of each solution was also created in these steps.

In the SA, the initial temperature and the cooling coefficient are two key parameters that directly affect its performance [20]. Even in simple problems, it proves hard to find optimal values for these parameters. One of the ways of coping with this task is to use an adaptive cooling strategy [34–36]. The adaptive cooling strategy is an effective method that makes the SA less sensitive to user-defined parameters [20]. However, it also causes the loss of the simplicity of the SA. Another way is to use a different cooling schedule that has fewer or more robust parameters. In the present study, a list-based cooling schedule was used to perform this task. The initial temperature list was generated using the method proposed by Zhong et al. [37]. Zhong et al. proposed that the absolute value of the difference between the two fitness values can be used directly as the initial temperature value (as in Step 11 in Algorithm 2). In the present study, the method proposed by Wang et al. [33] was used to improve the robustness of the initial temperature values and reduce the effect of noise. Wang et al.'s study showed that the top Len/2 temperatures and the bottom Len/2 ones can be deleted from the temperature list which has the length of 2*Len (as in Steps 16 and 17 in Algorithm 2).

**Algorithm 2.** The pseudocode for the generation of the initial population and temperature list.

**Input:** The parameter values;
M: The number of cities and the distance matrix, P: The population size
Len: The length of temperature list

**Output:** The initial population and temperature list;
p: The initial population, Lst: The temperature list

1 **begin**
2 **for** i = 1 **to** P
*// The random generation of each solution as the permutation of the set of the cities*
3 $p_i$ = randperm(M)
*// The applying of 2-opt operator on the $i^{th}$ solution*
4 $p_i$ = Apply2-OptOperator(M, $p_i$)
*// The generating the temperature list (2*Len) with local search operators*
5 **for** j = 1 **to** 2 * Len
*// The selection of the local search operator with a probability of 0.5*
6 **if** random(0,1) < 0.5 **then**
7 $p_{new}$ = ApplyInversionOperator($p_i$)
8 **else**
9 $p_{new}$ = ApplyInsertionOperator($p_i$)
10 **end if**
11 $tempLst_j = |f(p_{new}) - f(p_i)|$
12 **if** $tempLst_j$ < 0 **then**
13 $p_i = p_{new}$
14 **end if**
15 **end for**
*// The ranking of the temperature list*
16 tempLst = rank(tempLst)
*// Deleting the top Len/2 and bottom Len/2 temperatures from the list*
17 $Lst_i$(:) = $tempLst_{(Len/2)+1:3*(Len/2)}$
18 **end for**
19 **return**(p, Lst)
20 **end**

## 3.2 The fitness value

The fitness value of a solution is the total distance of the route. This distance was calculated by summing the distances between the cities (nodes). The type of distances between the cities is specified in the section reserved for the Edge Weight Type in the TSP instance file. The distances can be expressed as the Euclidean distance or the geographical distance. They can also be expressed as the special distance functions. Most of the test instances used in this paper used the Euclidean distance. However, some used the geographical distance (e.g., gr96, ali535) and the special distance functions (e.g., att48, att532).

### 3.3 The local search operators

The local search (neighborhood search) operators are frequently used to iteratively improve a solution [17]. Also, these operators are applied to increase the convergence rate of the algorithm. In the present study, the inversion and insertion operators, which are among the most popular local search operators, were used. The mathematical formula for the local search process is as follows: $S' = (0.5) \odot$ (inversion(S, i, j), insertion(S, i, j)), where S is the current solution; i and j are randomly selected nodes and $S'$ is the new solution. The operator $\odot$ is a probabilistic selection operator used to select one of the local search operators with a probability of 0.5. The local search operators can be described as follows:

**The inversion operator:** The inversion operator simply inverses the cities between positions i and j. In other words, it randomly selects two cities (positions) on the route and then reverses the sub-tour between these two cities. inversion($S$, $i$, $j$) generates a new solution $S'$, which is $S_i^{'} = S_j, S_{i+1}^{'} = S_{j-1}, \ldots, S_{i+k}^{'} = S_{j-k}, \ldots, S_j^{'} = S_i,$ where $1 \leq$ i, $j \leq$ n and $i < j$. This operator replaces two edges for the TSP problems.

**The insertion operator:** The insertion operator randomly selects the cities i and j in the route, where i $\neq$ j. Then it moves the city from Position j to Position i. insertion($S$, $i$, $j$) generates a new solution $S'$, which is $S_i^{'} = S_j, S_{i+1}^{'} = S_i, S_{i+2}^{'} = S_{i+1}, \ldots, S_j^{'} = S_{j-1},$ where $1 \leq$ i, $j \leq$ n and $i < j$. This operator usually replaces three edges for the TSP problems.

The examples of how these operators work on a sample solution are presented in Fig. 1.

### 3.4 The temperature list updating procedure

In the present study, the temperature list was updated using the method proposed by Zhan et al. [20]. The maximum temperature tMax in the temperature list of each solution in the population was used as the current temperature of the related solution. In each iteration, the temperatures tMax of the solutions were updated with the equation $t$Max = s/c, where c is the number of instances in which the worse solutions were accepted as the current solution. s is the sum of the values ($-$ $\Delta$f/ln(r)) calculated for the worse solutions. $\Delta f$ is the difference between the fitness values of the two solutions (the current and new). r is the random number that allows the worse solutions to be accepted. This updating procedure ensures that the tMax value calculated for any solution is lower than the one used in the previous iteration [20]. Thus, the temperature value used for each solution was iteratively reduced. The Steps 7 through 29 in Algorithm 1 show the updating of the temperature list.

### 3.5 The crossover operators

In the present study, the crossover operators were applied after the local search operators. The local search operators handled and improved the solutions in the population one by one. Then, the crossover operators exchanged information between the improved solutions individually. Thus, these operators contributed to the convergence of the algorithm. In this study, the OX1 [38] and ER [39] operators were used. The operators to be used were selected with a probability of 0.5. The crossover procedure is presented in Algorithm 3. As seen in the algorithm, one of the parents to be crossed was selected based on the predefined crossover possibility. The other is the solution with the best fitness value in the population. The crossover operators can be described as follows:

**The OX1 operator:** First, two cut points are randomly selected, and the sub-tours between these cut points are copied into the offsprings. Then, starting from the second cut point of the first parent, the rest of the cities are copied in the order in which they appear in the other parent. Meanwhile, the cities that already exist in the first parent are discarded without copying. When the end of the parent string is reached, the operation continues from the first position. Thus, the first offspring is created. This procedure is also applied to create the second offspring. Figure 2a presents the implementation of the OX1 operator on the two parents.

**The ER operator**: The ER is an operator which is suitable for the symmetric TSPs. It makes the assumption that only the values of the edges, rather than their direction, are important. In accordance with this assumption, the edges of a tour can be seen as the carriers of the hereditary information. It attempts to preserve the edges of the parents to pass the largest amount of information to the offspring. The breaking of edges is seen as unwanted mutation.

The ER operator uses an adjacency table called the edge table. This table lists the links into and out of a city found in the two parent sequences. The ER works based on the following algorithm: (1) Choose an initial city from one of the two parent sequences. This city can be chosen at random or according to the criteria outlined in Step 4. It is the current city. (2) Remove all the occurrences of the current city from the right side of the edge table. These can be found by referring to the edge list of the current city. (3) If the current city has entries in its edge list, continue from Step 4. Otherwise, go to Step 5. (4) Determine which of the cities in the edge list of the current city has the fewest entries in its own edge list. The city with the fewest entries

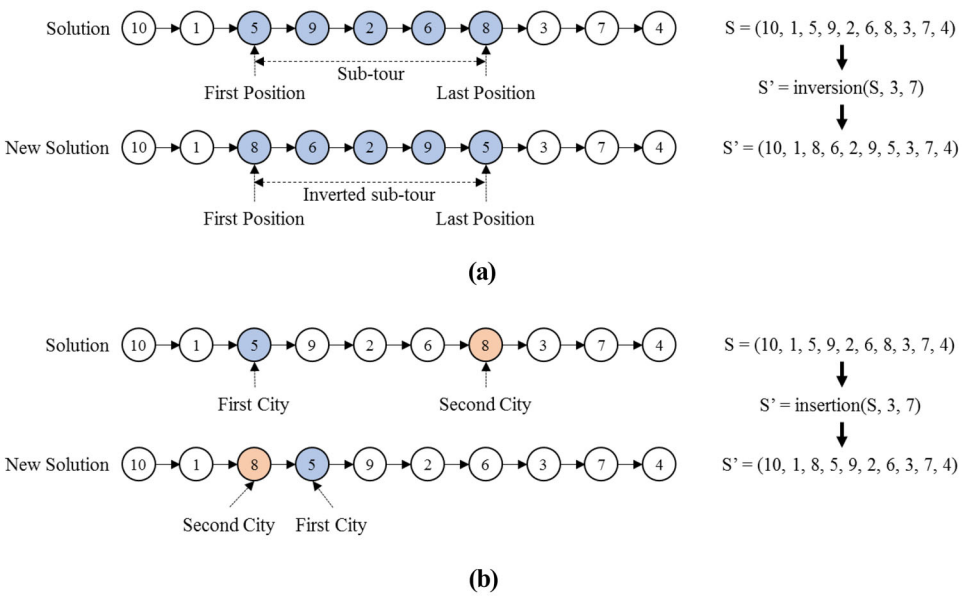


**Fig. 1** The new solutions S′ generated by the **a**) inversion(S, 3, 7) and **b**) insertion(S, 3, 7) operators

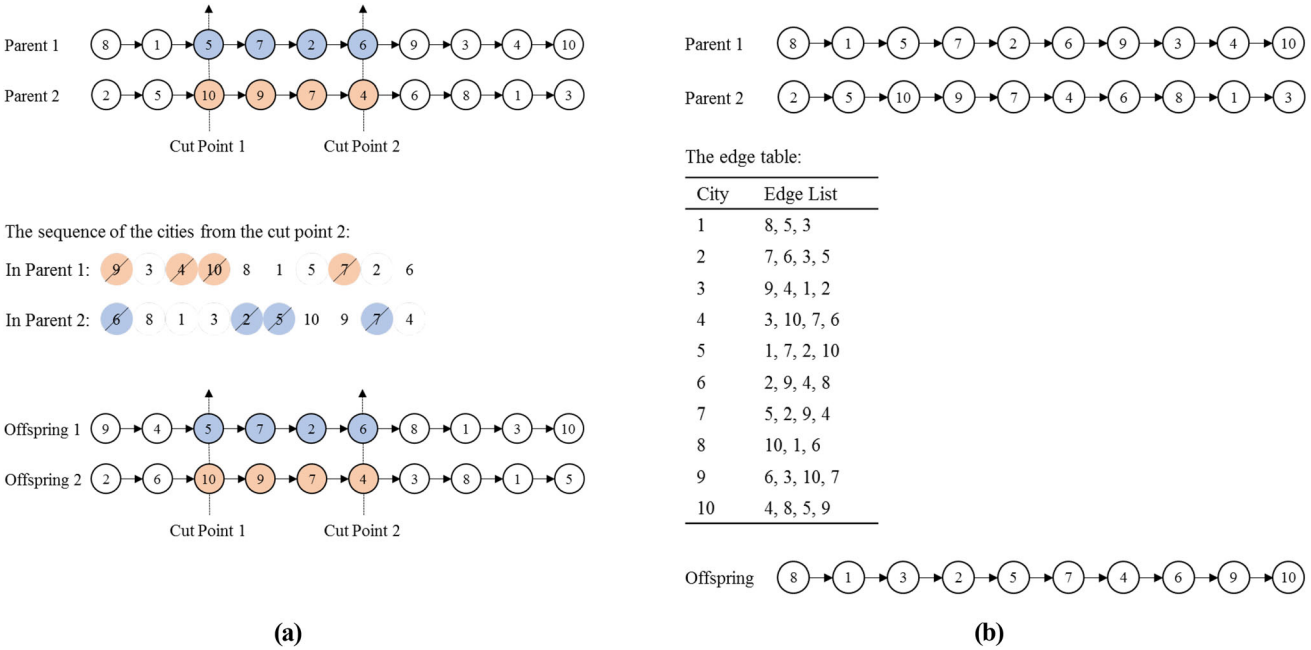


**Fig. 2** The implementation of the **a**) OX1 operator and **b**) ER operator on two parents

becomes the current city. If all the cities have the same number of entries, choose one of them randomly. (5) If there are no remaining unvisited cities, then stop. Otherwise, randomly choose an unvisited city and go to Step 2.

Figure 2b presents the implementation of the ER operator on two parents. As seen in the figure, the new offspring tour is initialized with one of the two initial cities from its parents. The initial cities 8 and 2 have three and four edges, respectively. City 8 is chosen because it has fewer edges than City 2. Then, all occurrences of City 8 are removed from the right side of the edge table. The edge list for City 8 indicates that the candidates for the next city are Cities 10, 1 and 6. Cities 10 and 6 have three edges, while City 1 has two. City 1 becomes the current city as it has fewer edges than the others. Then, all occurrences of City 1 are removed from the right side of the edge table. City 1 has edges to Cities 5 and 3. Cities 5 and 3 both have three edges. City 3 is randomly chosen and all occurrences of City 3 are removed from the right side of the edge table. City 3 has edges to Cities 9, 4 and 2. All of them have three edges. City 2 is randomly chosen and all its occurrences are removed from the right side of the edge table. This

procedure continues until all cities are visited. As a result, a new offspring sequence is obtained.

**Algorithm 3.** The pseudocode of the crossover operator.

```
Input: The parameter values;
    CR: The crossover rate, p: The population, p_best: The best solution

Output: The population;
    p: The population

1  begin
2      for i = 1 to P
           // The selection of the solutions to be crossed based on the crossing rate
3          if random(0,1) < CR then
               // The selection of the crossover operator with a probability of 0.5
4              if random(0,1) < 0.5 then
5                  (o1, o2) = ApplyOX1Operator(p_best, p_i)
6                  if f(o2) < f(o1) then
7                      o1 = o2
8                  end if
9                  if f(o1) < f(p_i) then
10                     p_i = o1
11                 end if
12             else
13                 o = ApplyEROperator(p_best, p_i)
14                 if f(o) < f(p_i) then
15                     p_i = o
16                 end if
17             end if
18         end if
19     end for
20     return(p)
21 end
```

### 3.6 The 2-opt and 3-opt operators

The 2-opt operator was introduced by Lin [40]. It is a simple local search operator used to solve routing problems. In the 2-opt operator, two edges of a solution are removed. Then, two new edges are created, avoiding the creation of the sub-tours. This process goes on until a better solution is found. In the LBSA-CO, the 2-opt was used during the generation of the initial population (as in Step 4 of Algorithm 2). Additionally, this operator was applied to them after the solutions in the population were improved with local search operators (as in Step 30 of Algorithm 1).

The 3-opt operator was also introduced by Lin [40]. In the 2-opt, two edges of a solution are removed, while three edges are removed in this operator. In place of the removed edges, three new edges are added, and they are evaluated. This process is repeated for a different set of three new edges. There are eight sets in total. The optimum one among these sets is accepted. The 3-opt was used to further improve the best solution in the LBSA-CO (as in Step 38 of Algorithm 1).

### 3.7 The differences between the LBSA and LBSA-CO

The flowcharts are given in Figs. 3 and 4 to illustrate the differences between the LBSA and LBSA-CO methods. Considering these flowcharts, along with the information given in the previous subsections for the LBSA-CO and the original paper of the LBSA [20], the differences between the two methods can be listed as follows:

1. The LBSA randomly generates a single initial solution and iteratively improves it. On the other hand, the LBSA-CO randomly generates an initial population and applies the 2-opt operator to the solutions in the population. Then, it iteratively improves them.
2. The LBSA generates an initial temperature list using the inversion, insertion, and swap local search operators with equal probability. The LBSA-CO generates an initial temperature matrix, each row of which corresponds to a solution in the population. The inversion and insertion local search operators are used with equal probability for this operation.
3. The neighboring solutions are generated by using the inversion, insertion and swap operators with equal probability in the LBSA. In the LBSA-CO, the neighboring solutions are obtained separately for each of the solutions in the population by using the insertion and inversion operators with equal probability.
4. The temperature list updating procedure is applied for a single solution in the LBSA. In the LBSA-CO, it is applied separately for each of the solutions in the population.
5. In the LBSA-CO, after the solutions are developed as much as the number of trials, they are improved with the 2-opt operator.
6. In the LBSA-CO, the OX1 and ER operators are applied with equal probability to the solutions in the population at each iteration by taking into account the crossover rate.
7. Unlike the LBSA, the 3-opt operator is applied to the best solution found at the end of the iterations in the LBSA-CO.

## 4 Experimental studies

The experiments were carried out in Matlab 2017a environment in a desktop computer with Intel Core i5-3470 3.20 GHz CPU, 6 GB RAM and Windows 7 Professional 64 Bit Operating System. The performance of the LBSA-CO was tested on 65 well-known TSP instances obtained from the TSPLIB [41], which is the standard library for the TSP problems. The number of cities in these instances is between 14 and 783.

### 4.1 Parameter tuning

The Taguchi method is a statistical experimental design tool, which was introduced by Genichi Taguchi [42]. This method divides the parameters into two groups: controllable and uncontrollable (noisy) parameters. It tries to minimize the effect of noise, as it is often impossible to

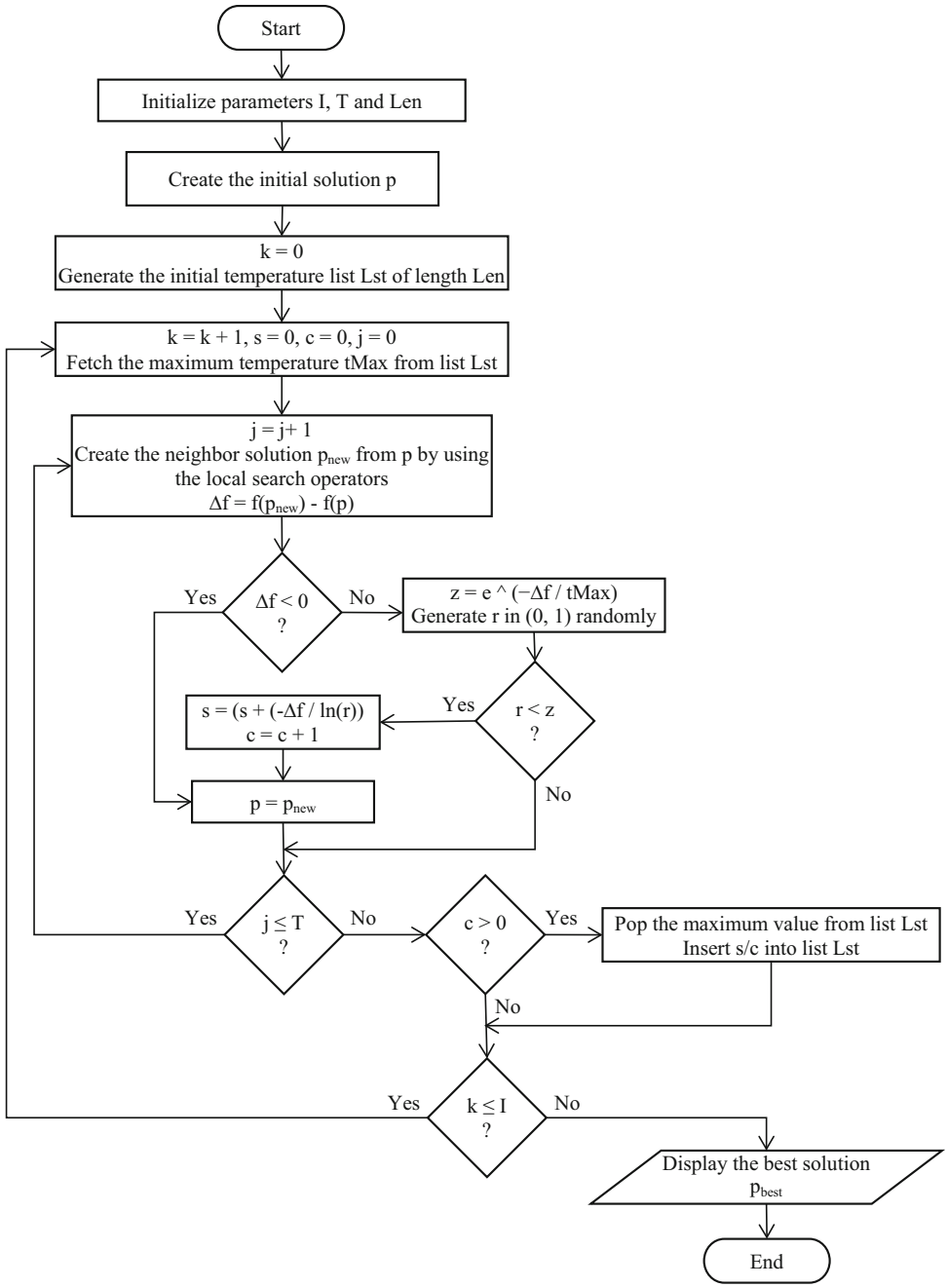


**Fig. 3** The flowchart of the LBSA algorithm

remove uncontrollable parameters. In addition, it tries to find the best value of the controllable parameters in terms of stability. It looks at how the parameters affect the performance of the algorithm and determines their relative values [43].

In the present study, the Taguchi method was used to tune the parameters of the LBSA-CO. The LBSA-CO has five parameters. The first parameter is the number of iterations, and it was designated as 100 in all Taguchi experiments. Each of the rest has four levels. Table 1 presents the levels of these parameters. Based on these levels, the Taguchi method generated an orthogonal array ($L_{16}$), which included 16 Taguchi experiments. Twelve instances of different sizes were used in these experiments. Each experiment was repeated 10 times on each instance to increase reliability. The average values of the 10 runs are provided in Table 2.

The average values in Table 2 were processed by Minitab 19 statistical analysis software and were converted into the signal-to-noise (S/N) ratios. The S/N ratio indicates the amount of variation in the desirable value. Therefore, the aim is to maximize this ratio [43]. The S/N ratios

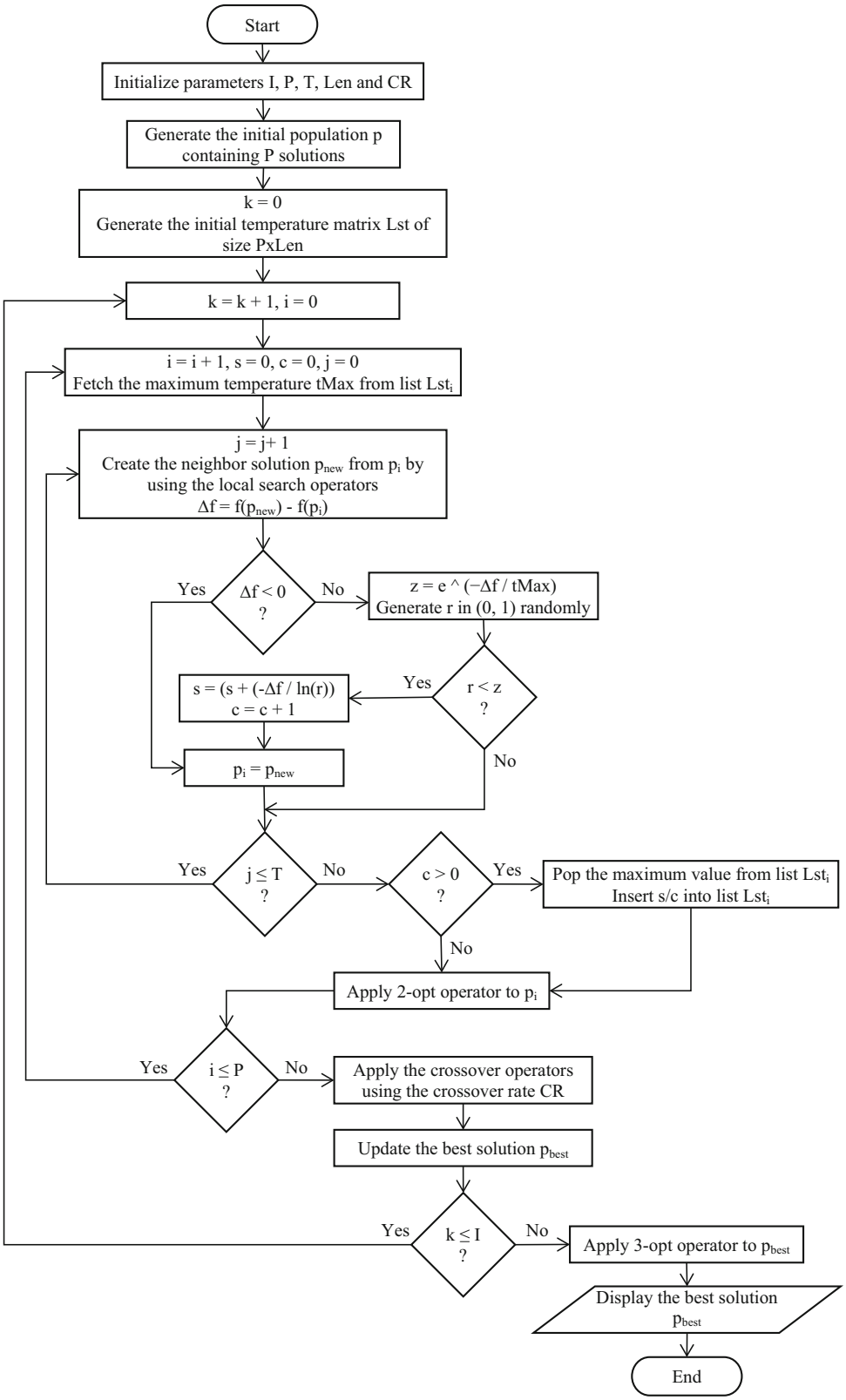


**Fig. 4** The flowchart of the LBSA-CO algorithm

**Table 1** The parameters of the LBSA-CO and their levels

| Parameter | | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|---|
| Name | Symbol | | | | |
| The population size | P | 20 | 30 | 40 | 50 |
| The number of trials | T | 40 | 60 | 80 | 100 |
| The length of the temperature list | Len | 4 | 6 | 8 | 10 |
| The crossover rate | CR | 0.60 | 0.70 | 0.80 | 0.90 |

calculated for four parameters and the four levels are presented in Table 3. The maximum values of the parameters are written in bold. As seen in the table, the maximum values for all the parameters are at the fourth level. This indicates that the fourth level values of the parameters are the best values. As can be understood from Table 1, the parameters of the LBSA-CO should be as follows: P = 50, T = 100, Len = 10 and CR = 0.90. The row labeled "Rank" in Table 3 provides the order of importance of the parameters and is determined based on the values in the row labeled "Delta." Based on its rank, the parameter that most strongly affects the performance of the LBSA-CO is the population size (P), while the parameter that least strongly affects it is the number of trials (T).

In the present study, the Taguchi method was also used to tune the parameters of the SA. The SA has four parameters. The first parameter is the number of iterations and was designated as 1000 in all Taguchi experiments. Each of the rest has four levels. Table 4 presents the levels of these parameters. The Taguchi method generated an orthogonal array $L_{16,}$ which included 16 Taguchi experiments. The same instances used for the LBSA-CO were also used in these experiments. Each experiment was repeated 10 times on each instance. The average values are provided in Table 5.

After the average values were processed by Minitab 19, the S/N ratios in Table 6 were obtained. The maximum values of these ratios are shown in bold. As can be understood from Table 4, the parameters of the SA should be as follows: T = 500, $t_{start}$ = 70 and $\alpha$ = 0.85. According to the variable "Rank" in Table 6, the parameter that most strongly affects the performance of the SA is the number of trials (T).

## 4.2 Performance evaluation of the operators

In the LBSA-CO, the inversion and insertion operators were used together to search for neighborhoods. To evaluate the performances of these operators, a number of experiments were carried out on 12 instances of different sizes. The results which were obtained for 10 runs on each instance are presented in Table 7. In this table, "Instance" and "BKS" denote the name of the TSP problems and the best known solution, respectively. "Inversion" and "Insertion" show the results obtained using these operators alone. "Both" denotes the results obtained using both operators with equal probability. "Best" and "Worst" refer to the percentage errors of the best and worst solutions in all runs from BKS, respectively. "Mean" is the percentage error of the average value of all runs from BKS. These values were calculated using the formulae in (9–11). As seen in the row labeled "Average" in Table 7, the insertion operator has lower values than the inversion operator. Therefore, the effect of this operator on the performance of the LBSA-CO is greater than that of the other.

$$Best = \frac{(BestSol - BKS)}{BKS} * 100, BestSol \quad : \textit{The best solution in all runs} \tag{9}$$

$$Mean = \frac{(Avg - BKS)}{BKS} * 100, Avg \quad : \textit{The average value of all runs} \tag{10}$$

$$Worst = \frac{(WorstSol - BKS)}{BKS} * 100, WorstSol \quad : \textit{The worst solution in all runs} \tag{11}$$

In the LBSA-CO, the OX1 and ER operators were used together to cross the solutions. To evaluate the performances of these operators, a number of experiments were carried out on 12 instances, which were also used to evaluate the performances of the local search operators. The results which were obtained for 10 runs on each instance are presented in Table 8. In this table, "OX1" and "ER" show the results obtained using these operators alone. "Both" denotes the results obtained using both operators with equal probability. As seen in the row labeled "Average" in Table 8, the OX1 operator has lower values than the ER operator. Therefore, the effect of this operator on the performance of the LBSA-CO is greater than that of the other.

**Table 2** The average values obtained for the orthogonal array $L_{16}$

| Orthogonal array | | | | | Instances | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | P | T | Len | CR | eil76 | kroA100 | ch130 | u159 | d198 | pr226 | gil262 | a280 | lin318 | rd400 | d493 | u574 |
| 1 | 20 | 40 | 4 | 0.60 | 542.40 | 21,317.60 | 6207.40 | 42,399.20 | 15,909.20 | 80,771.90 | 2446.70 | 2636.80 | 43,331.60 | 15,921.00 | 36,203.70 | 38,612.60 |
| 2 | 20 | 60 | 6 | 0.70 | 540.00 | 21,300.90 | 6171.30 | 42,257.00 | 15,886.80 | 80,597.30 | 2449.60 | 2637.20 | 43,146.50 | 15,759.40 | 36,273.00 | 38,413.50 |
| 3 | 20 | 80 | 8 | 0.80 | 540.90 | 21,294.40 | 6189.70 | 42,345.60 | 15,892.00 | 80,620.30 | 2436.10 | 2624.20 | 43,145.00 | 15,785.40 | 36,042.50 | 38,403.00 |
| 4 | 20 | 100 | 10 | 0.90 | 540.90 | 21,298.20 | 6170.80 | 42,295.90 | 15,879.00 | 80,589.40 | 2429.70 | 2611.80 | 42,935.70 | 15,714.90 | 35,999.60 | 38,391.60 |
| 5 | 30 | 40 | 6 | 0.80 | 541.00 | 21,323.50 | 6165.60 | 42,119.50 | 15,859.10 | 80,620.60 | 2424.90 | 2631.40 | 42,930.50 | 15,766.30 | 36,087.80 | 38,454.60 |
| 6 | 30 | 60 | 4 | 0.90 | 540.00 | 21,288.60 | 6194.00 | 42,313.50 | 15,867.80 | 80,572.10 | 2437.00 | 2614.40 | 43,017.00 | 15,805.70 | 36,036.70 | 38,351.00 |
| 7 | 30 | 80 | 10 | 0.60 | 539.90 | 21,292.60 | 6147.40 | 42,178.30 | 15,879.10 | 80,497.00 | 2430.80 | 2617.40 | 43,023.60 | 15,773.80 | 35,981.10 | 38,334.60 |
| 8 | 30 | 100 | 8 | 0.70 | 539.20 | 21,294.90 | 6164.00 | 42,241.70 | 15,831.30 | 80,596.80 | 2425.50 | 2621.20 | 42,803.40 | 15,692.50 | 36,074.00 | 38,561.90 |
| 9 | 40 | 40 | 8 | 0.90 | 539.60 | 21,288.60 | 6164.20 | 42,313.40 | 15,829.30 | 80,423.00 | 2430.00 | 2600.80 | 42,731.80 | 15,705.00 | 35,948.10 | 38,231.00 |
| 10 | 40 | 60 | 10 | 0.80 | 540.60 | 21,288.60 | 6151.10 | 42,143.20 | 15,837.60 | 80,482.00 | 2418.00 | 2610.20 | 42,795.20 | 15,697.20 | 36,029.00 | 38,142.50 |
| 11 | 40 | 80 | 4 | 0.70 | 538.90 | 21,296.90 | 6181.60 | 42,154.50 | 15,847.20 | 80,538.50 | 2431.60 | 2613.60 | 43,055.00 | 15,775.20 | 35,908.10 | 38,485.10 |
| 12 | 40 | 100 | 6 | 0.60 | 539.10 | 21,284.00 | 6193.20 | 42,174.00 | 15,833.20 | 80,407.00 | 2428.00 | 2620.90 | 42,997.50 | 15,685.20 | 36,024.30 | 38,334.70 |
| 13 | 50 | 40 | 10 | 0.70 | 539.90 | 21,286.30 | 6157.20 | 42,257.70 | 15,816.20 | 80,371.00 | 2427.40 | 2600.80 | 42,783.60 | 15,691.00 | 35,937.10 | 38,161.80 |
| 14 | 50 | 60 | 8 | 0.60 | 539.90 | 21,285.30 | 6150.10 | 42,182.70 | 15,847.40 | 80,544.60 | 2418.10 | 2605.60 | 42,929.20 | 15,677.40 | 36,033.30 | 38,113.50 |
| 15 | 50 | 80 | 6 | 0.90 | 539.40 | 21,283.00 | 6153.50 | 42,178.80 | 15,846.30 | 80,483.40 | 2424.30 | 2611.40 | 42,766.60 | 15,701.30 | 35,954.40 | 38,210.50 |
| 16 | 50 | 100 | 4 | 0.80 | 541.50 | 21,290.90 | 6172.50 | 42,167.00 | 15,846.30 | 80,417.30 | 2416.30 | 2613.20 | 42,781.40 | 15,716.30 | 35,944.30 | 38,267.40 |

**Table 3** The S/N ratios for the parameters of the LBSA-CO

| Level | P | T | Len | CR |
|---|---|---|---|---|
| 1 | − 90.653 | − 90.637 | − 90.642 | − 90.640 |
| 2 | − 90.638 | − 90.634 | − 90.634 | − 90.635 |
| 3 | − 90.625 | − 90.634 | − 90.633 | − 90.631 |
| 4 | **− 90.619** | **− 90.630** | **− 90.626** | **− 90.630** |
| Delta | 0.034 | 0.006 | 0.016 | 0.010 |
| Rank | 1 | 4 | 2 | 3 |

### 4.3 The performance evaluation of the temperature list updating procedure

The procedure, which was explained in detail in Sect. 3.4, was used to update the temperature list. To illustrate the effectiveness of this procedure, a number of experiments were carried out using the geometric updating strategy. The experiments were conducted for the values of {0.6, 0.7, 0.8, 0.9} of the cooling coefficient α. However, since the results obtained for α = 0.9 are better than those of others, these results are provided in this section. Other parameter values were equal to those given in Sect. 4.1. The results which were obtained for the 20 runs on each of 65 instances are presented in Table 9. In this table, "StdDev" stands for standard deviation. As seen in the row labeled "Average" in Table 9, the better results were obtained with the temperature list updating strategy used in the LBSA-CO, compared to the geometric updating strategy.

### 4.4 Experimental results

Table 10 presents the results of the LBSA-CO, VTPSO [19], ABCSS [23] and DSMO [24] on 32 instances. The results of the LBSA-CO were obtained by running 20 times on each instance. The results of the other methods were taken from the study conducted by Akhand et al. [24], in which it was reported that these results were obtained by running 20 times on each instance. The results were calculated using the floating-point Euclidean distance. In Table 10, "BestSol" refers to the best solution in all runs. "Avg" and "StdDev" are the average value and the standard deviation of all runs, respectively. As seen in the column labeled "BestSol", the LBSA-CO achieved better results than the other methods on 29 of 32 instances. In terms of Avg, it showed better performance compared to the other methods on all instances. The superiority of the LBSA-CO over the other methods can be seen in the row labeled "Average."

The LBSA-CO was compared with the SA and LBSA [20] on 65 instances. The SA and LBSA were re-coded in Matlab. The parameter values of the SA were determined using the Taguchi method (Subsect. 4.1). The parameter values were as follows: The number of iterations was identified as 1000, the number of trials as 500, the start temperature as 70 and the temperature reduction rate as 0.85. As in the LBSA-CO, the inversion and insertion operators were used as the local search operators in the SA and its initial solution randomly generated was improved by the 2-opt operator. The parameter values of the LBSA were taken from the original paper and they were as follows: The number of iterations was determined as 1000, the number of trials as 2*the number of cities in the problem and the length of the temperature list as 120. The SA and LBSA were run 20 times on each instance. The results were calculated using the rounded Euclidean distance. They are presented together with the results of the LBSA-CO in Table 11. In the table, "Time" is the running time in seconds. In terms of Best, the LBSA-CO achieved BKS for 37 instances. For the remaining 28 instances, it obtained the closest values to BKS, compared to the other methods. On the other hand, while the SA achieved BKS for 8 instances, the LBSA obtained BKS for 9 instances. In terms of obtaining BKS, the success rate of the LBSA-CO was 37/65 = 0.57, whereas the success rates of the SA and LBSA were 8/65 = 0.12 and 9/65 = 0.14, respectively.

The LBSA-CO was compared with the IBA [21] on 37 instances. The IBA was re-coded in Matlab. The parameter values of the IBA were taken from the original paper, and they were as follows: The number of iterations was determined as 100, the population size as 30, $A_i^0$ as the

**Table 4** The parameters of the SA and their levels

| Parameter | | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|---|
| Name | Symbol | | | | |
| The number of trials | $T$ | 200 | 300 | 400 | 500 |
| The start temperature | $t_{start}$ | 10 | 40 | 70 | 100 |
| The temperature reduction rate | $\alpha$ | 0.80 | 0.85 | 0.90 | 0.95 |

**Table 5** The average values obtained for the orthogonal array $L_{16}$

| Orthogonal array | | | | Instances | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | $T$ | $t_{start}$ | $\alpha$ | eil76 | kroA100 | ch130 | u159 | d198 | pr226 | gil262 | a280 | lin318 | rd400 | d493 | u574 |
| 1 | 200 | 10 | 0.80 | 558.80 | 22,140.10 | 6531.30 | 44,639.50 | 16,403.20 | 85,399.60 | 2609.10 | 2928.60 | 46,920.10 | 17,567.50 | 41,225.70 | 47,181.30 |
| 2 | 200 | 40 | 0.85 | 563.50 | 22,571.40 | 6514.00 | 44,923.80 | 16,295.20 | 86,500.80 | 2597.70 | 2900.10 | 47,080.80 | 17,520.40 | 41,920.60 | 46,917.90 |
| 3 | 200 | 70 | 0.90 | 560.10 | 22,063.10 | 6490.30 | 45,567.50 | 16,329.10 | 84,192.10 | 2615.40 | 2955.30 | 47,079.20 | 17,504.10 | 41,331.50 | 47,110.40 |
| 4 | 200 | 100 | 0.95 | 560.80 | 22,137.20 | 6479.00 | 45,242.60 | 16,321.30 | 85,637.50 | 2638.10 | 2936.40 | 47,286.20 | 18,059.50 | 41,363.00 | 47,200.40 |
| 5 | 300 | 10 | 0.85 | 563.90 | 22,296.10 | 6483.10 | 45,991.00 | 16,300.70 | 83,929.80 | 2576.40 | 2828.30 | 45,950.60 | 17,124.40 | 39,673.60 | 44,662.80 |
| 6 | 300 | 40 | 0.80 | 558.60 | 22,495.90 | 6499.70 | 45,146.70 | 16,284.00 | 84,125.20 | 2577.00 | 2898.00 | 46,183.20 | 17,047.80 | 39,961.80 | 44,555.00 |
| 7 | 300 | 70 | 0.95 | 560.30 | 22,313.10 | 6472.60 | 45,192.70 | 16,214.80 | 83,921.10 | 2599.40 | 2855.10 | 46,215.20 | 17,106.60 | 40,059.00 | 44,866.70 |
| 8 | 300 | 100 | 0.90 | 564.00 | 22,226.50 | 6456.10 | 45,106.80 | 16,336.00 | 84,952.70 | 2583.30 | 2873.90 | 46,441.90 | 17,149.90 | 39,938.30 | 44,525.90 |
| 9 | 400 | 10 | 0.90 | 561.70 | 22,294.40 | 6447.10 | 45,683.20 | 16,279.00 | 83,839.10 | 2562.90 | 2821.20 | 46,274.20 | 16,760.70 | 39,063.80 | 43,299.10 |
| 10 | 400 | 40 | 0.95 | 557.10 | 22,157.90 | 6450.70 | 44,100.90 | 16,231.00 | 86,640.30 | 2570.10 | 2856.60 | 45,987.70 | 16,868.00 | 39,325.60 | 43,192.80 |
| 11 | 400 | 70 | 0.80 | 560.20 | 22,345.70 | 6432.40 | 45,369.60 | 16,274.80 | 85,049.80 | 2578.60 | 2868.90 | 45,770.80 | 16,753.70 | 39,086.20 | 43,221.80 |
| 12 | 400 | 100 | 0.85 | 556.70 | 21,981.90 | 6398.00 | 44,817.40 | 16,277.00 | 84,207.70 | 2558.70 | 2861.00 | 45,694.90 | 16,854.70 | 39,060.10 | 43,246.10 |
| 13 | 500 | 10 | 0.95 | 555.20 | 22,174.00 | 6441.40 | 44,757.10 | 16,266.00 | 85,335.80 | 2549.30 | 2823.70 | 45,315.00 | 16,661.20 | 38,471.20 | 42,427.10 |
| 14 | 500 | 40 | 0.90 | 559.00 | 22,548.10 | 6524.80 | 45,566.50 | 16,201.90 | 84,812.60 | 2582.70 | 2821.20 | 45,489.90 | 16,705.00 | 38,735.80 | 42,383.00 |
| 15 | 500 | 70 | 0.85 | 558.50 | 22,440.00 | 6475.60 | 43,981.40 | 16,335.80 | 84,240.70 | 2559.50 | 2819.30 | 45,451.10 | 16,814.50 | 38,510.80 | 42,323.80 |
| 16 | 500 | 100 | 0.80 | 561.30 | 22,628.60 | 6494.20 | 45,424.80 | 16,205.80 | 85,050.20 | 2560.30 | 2811.00 | 45,894.80 | 16,717.40 | 38,598.80 | 42,204.50 |

**Table 6** The S/N ratios for the parameters of the SA

| Level | $T$ | $t_{start}$ | $\alpha$ |
|---|---|---|---|
| 1 | − 91.41 | − 91.25 | − 91.26 |
| 2 | − 91.25 | − 91.29 | **− 91.24** |
| 3 | − 91.21 | **− 91.23** | − 91.25 |
| 4 | **− 91.16** | − 91.27 | − 91.28 |
| Delta | 0.25 | 0.06 | 0.04 |
| Rank | 1.00 | 2.00 | 3.00 |

random number in [0.7, 1.0], $r_i^0$ as the random number in [0.0, 0.4], $\alpha$ as 0.98 and $\gamma$ as 0.98. The IBA was run 20 times on each instance. The results were calculated using the rounded Euclidean distance. They are presented together with the results of the LBSA-CO in Table 12. In terms of Best, the LBSA-CO achieved BKS for 33 instances. For the remaining 4 instances, it obtained the closest values to BKS compared to the IBA. On the other hand, the IBA achieved BKS for 11 instances. In terms of obtaining BKS, the success rate of the LBSA-CO was 33/37 = 0.89, whereas that of the IBA was 11/37 = 0.30.

The LBSA-CO was compared with the GSAACS [29] on 21 instances. The results of the GSAACS were taken from the original paper. They are presented together with the results of the LBSA-CO in Table 13. In terms of Best,

**Table 7** Comparison of the performances of local search operators on 12 instances

| Instance | BKS | Inversion | | | Insertion | | | Both | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst |
| eil76 | 538 | 0.00 | 0.15 | 0.93 | 0.00 | 0.22 | 0.93 | 0.00 | 0.15 | 0.74 |
| kroA100 | 21,282 | 0.00 | 0.01 | 0.11 | 0.00 | 0.00 | 0.05 | 0.00 | 0.01 | 0.05 |
| ch130 | 6110 | 0.23 | 0.59 | 1.62 | 0.26 | 1.04 | 1.72 | 0.23 | 0.88 | 1.49 |
| u159 | 42,080 | 0.00 | 0.38 | 0.83 | 0.00 | 0.21 | 0.89 | 0.00 | 0.39 | 0.79 |
| d198 | 15,780 | 0.10 | 0.27 | 0.46 | 0.12 | 0.40 | 0.82 | 0.15 | 0.34 | 0.73 |
| pr226 | 80,369 | 0.00 | 0.03 | 0.29 | 0.00 | 0.15 | 0.47 | 0.00 | 0.05 | 0.53 |
| gil262 | 2378 | 1.30 | 1.88 | 2.99 | 0.42 | 1.56 | 2.78 | 0.21 | 1.55 | 2.57 |
| a280 | 2579 | 0.39 | 1.02 | 1.78 | 0.16 | 0.86 | 2.21 | 0.08 | 0.81 | 2.29 |
| lin318 | 42,029 | 0.51 | 1.94 | 3.16 | 0.62 | 1.92 | 2.97 | 0.65 | 1.43 | 1.96 |
| rd400 | 15,281 | 1.88 | 2.65 | 3.31 | 2.00 | 2.47 | 3.38 | 1.62 | 2.69 | 3.88 |
| d493 | 35,002 | 1.73 | 2.51 | 3.96 | 1.83 | 2.35 | 2.73 | 1.75 | 2.41 | 3.15 |
| u574 | 36,905 | 2.08 | 3.32 | 3.97 | 2.32 | 3.22 | 4.21 | 2.24 | 2.89 | 3.49 |
| Average | | 0.68 | 1.23 | 1.95 | 0.64 | 1.20 | 1.93 | 0.58 | 1.13 | 1.80 |

**Table 8** Comparison of the performances of crossover operators on 12 instances

| Instance | BKS | OX1 | | | ER | | | Both | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst |
| eil76 | 538 | 0.00 | 0.17 | 0.74 | 0.00 | 0.19 | 0.74 | 0.00 | 0.19 | 0.93 |
| kroA100 | 21,282 | 0.00 | 0.02 | 0.11 | 0.00 | 0.00 | 0.05 | 0.00 | 0.00 | 0.05 |
| ch130 | 6110 | 0.00 | 0.65 | 1.19 | 0.00 | 0.74 | 1.72 | 0.23 | 0.76 | 1.64 |
| u159 | 42,080 | 0.00 | 0.59 | 1.45 | 0.00 | 0.17 | 0.93 | 0.00 | 0.00 | 0.00 |
| d198 | 15,780 | 0.01 | 0.35 | 0.93 | 0.37 | 0.72 | 1.28 | 0.13 | 0.32 | 0.59 |
| pr226 | 80,369 | 0.00 | 0.18 | 0.58 | 0.00 | 0.07 | 0.62 | 0.00 | 0.11 | 0.47 |
| gil262 | 2378 | 1.30 | 1.67 | 3.53 | 1.85 | 3.01 | 4.12 | 0.17 | 1.41 | 3.20 |
| a280 | 2579 | 0.54 | 1.64 | 2.83 | 0.97 | 2.49 | 3.64 | 0.19 | 0.86 | 2.21 |
| lin318 | 42,029 | 0.91 | 1.63 | 1.91 | 1.55 | 2.45 | 3.36 | 0.73 | 1.77 | 2.48 |
| rd400 | 15,281 | 1.58 | 2.02 | 2.63 | 2.26 | 3.43 | 4.31 | 0.84 | 2.24 | 3.15 |
| d493 | 35,002 | 1.72 | 2.53 | 3.31 | 2.81 | 3.44 | 4.04 | 1.96 | 2.56 | 3.18 |
| u574 | 36,905 | 2.19 | 3.02 | 3.94 | 3.27 | 4.40 | 5.79 | 2.11 | 2.91 | 3.88 |
| Average | | 0.69 | 1.21 | 1.93 | 1.09 | 1.76 | 2.55 | 0.53 | 1.09 | 1.81 |

**Table 9** Comparison of the temperature list updating strategies on 65 instances

| Instance | BKS | Geometric updating strategy | | | | Updating strategy used in the LBSA-CO (Described in Sect. 3.4) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | StdDev | Best | Mean | Worst | StdDev |
| burma14 | 3323 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ulysses16 | 6859 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| gr17 | 2085 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ulysses22 | 7013 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| bays29 | 2020 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| dantzig42 | 699 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| swiss42 | 1273 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| att48 | 10,628 | 0.00 | 0.03 | 0.24 | 7.99 | 0.00 | 0.14 | 0.53 | 24.69 |
| eil51 | 426 | 0.00 | 0.16 | 0.47 | 0.66 | 0.00 | 0.19 | 0.47 | 0.52 |
| berlin52 | 7542 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| st70 | 675 | 0.00 | 0.20 | 0.89 | 2.16 | 0.00 | 0.39 | 1.33 | 2.85 |
| eil76 | 538 | 0.00 | 0.25 | 1.49 | 2.01 | 0.00 | 0.44 | 1.86 | 3.53 |
| pr76 | 108,159 | 0.00 | 0.17 | 0.86 | 380.02 | 0.00 | 0.17 | 0.86 | 380.02 |
| gr96 | 55,209 | 0.00 | 0.17 | 0.78 | 116.58 | 0.00 | 0.27 | 0.68 | 122.07 |
| rat99 | 1211 | 0.00 | 0.50 | 1.49 | 5.63 | 0.00 | 0.52 | 1.57 | 5.26 |
| kroA100 | 21,282 | 0.00 | 0.04 | 0.50 | 24.29 | 0.00 | 0.02 | 0.11 | 7.57 |
| kroB100 | 22,141 | 0.00 | 0.18 | 0.59 | 41.03 | 0.00 | 0.18 | 0.98 | 53.70 |
| kroC100 | 20,749 | 0.00 | 0.09 | 0.54 | 38.39 | 0.00 | 0.16 | 1.13 | 65.31 |
| kroD100 | 21,294 | 0.00 | 0.23 | 0.70 | 45.73 | 0.00 | 0.35 | 1.40 | 101.15 |
| kroE100 | 22,068 | 0.00 | 0.44 | 0.94 | 64.65 | 0.15 | 0.42 | 1.05 | 55.37 |
| rd100 | 7910 | 0.00 | 0.21 | 1.73 | 33.54 | 0.00 | 0.39 | 1.67 | 33.79 |
| eil101 | 629 | 0.00 | 0.80 | 1.91 | 3.85 | 0.00 | 0.85 | 2.23 | 4.80 |
| lin105 | 14,379 | 0.00 | 0.02 | 0.15 | 6.77 | 0.00 | 0.00 | 0.00 | 0.00 |
| pr107 | 44,303 | 0.00 | 0.12 | 0.49 | 66.66 | 0.00 | 0.21 | 0.61 | 85.67 |
| pr124 | 59,030 | 0.00 | 0.04 | 0.26 | 37.29 | 0.00 | 0.06 | 0.88 | 116.09 |
| bier127 | 118,282 | 0.00 | 0.32 | 0.77 | 248.66 | 0.00 | 0.48 | 1.36 | 546.97 |
| ch130 | 6110 | 0.23 | 0.83 | 1.54 | 28.49 | 0.00 | 0.77 | 1.67 | 31.47 |
| pr136 | 96,772 | 0.01 | 0.60 | 1.52 | 394.58 | 0.01 | 0.60 | 1.83 | 488.74 |
| gr137 | 69,853 | 0.00 | 0.31 | 0.84 | 270.76 | 0.00 | 0.31 | 1.23 | 300.26 |
| pr144 | 58,537 | 0.00 | 0.00 | 0.09 | 11.85 | 0.00 | 0.05 | 0.09 | 27.19 |
| ch150 | 6528 | 0.00 | 0.89 | 1.62 | 35.08 | 0.00 | 0.66 | 1.75 | 31.93 |
| kroA150 | 26,524 | 0.02 | 0.58 | 1.40 | 96.34 | 0.00 | 0.60 | 1.56 | 114.29 |
| kroB150 | 26,130 | 0.00 | 0.37 | 1.19 | 84.38 | 0.00 | 0.48 | 2.04 | 118.53 |
| pr152 | 73,682 | 0.00 | 0.27 | 0.82 | 174.33 | 0.00 | 0.25 | 1.02 | 178.45 |
| u159 | 42,080 | 0.00 | 0.32 | 1.71 | 252.23 | 0.00 | 0.60 | 1.46 | 164.05 |
| rat195 | 2323 | 0.86 | 2.00 | 3.06 | 15.00 | 0.52 | 1.33 | 2.88 | 15.25 |
| d198 | 15,780 | 0.06 | 0.41 | 0.82 | 36.02 | 0.05 | 0.29 | 0.61 | 24.30 |
| kroA200 | 29,368 | 0.28 | 0.94 | 1.89 | 139.53 | 0.16 | 0.72 | 1.56 | 133.09 |
| kroB200 | 29,437 | 0.05 | 1.19 | 2.59 | 187.19 | 0.04 | 0.61 | 1.51 | 132.24 |
| gr202 | 40,160 | 0.86 | 1.36 | 1.77 | 100.58 | 0.34 | 1.20 | 2.19 | 196.39 |
| tsp225 | 3919 | 1.20 | 1.97 | 3.55 | 21.52 | 0.89 | 1.44 | 2.32 | 16.28 |
| ts225 | 126,643 | 0.00 | 0.06 | 0.32 | 122.86 | 0.00 | 0.00 | 0.00 | 0.00 |
| pr226 | 80,369 | 0.00 | 0.08 | 0.62 | 140.16 | 0.00 | 0.13 | 0.62 | 183.23 |
| gr229 | 134,602 | 0.59 | 1.28 | 2.42 | 634.68 | 0.38 | 1.21 | 2.02 | 583.61 |
| gil262 | 2378 | 0.80 | 2.29 | 3.70 | 19.38 | 0.21 | 1.59 | 3.20 | 16.22 |
| pr264 | 49,135 | 0.00 | 0.41 | 1.55 | 226.93 | 0.00 | 0.19 | 0.63 | 108.38 |

**Table 9** (continued)

| Instance | BKS | Geometric updating strategy | | | | Updating strategy used in the LBSA-CO (Described in Sect. 3.4) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | StdDev | Best | Mean | Worst | StdDev |
| a280 | 2579 | 0.31 | 1.37 | 3.06 | 16.46 | 0.00 | 0.76 | 2.02 | 16.19 |
| pr299 | 48,191 | 0.61 | 1.25 | 2.55 | 246.47 | 0.10 | 0.53 | 1.28 | 164.67 |
| lin318 | 42,029 | 0.70 | 1.99 | 3.15 | 253.63 | 0.17 | 1.27 | 2.12 | 221.16 |
| linhp318 | 41,345 | 2.96 | 3.78 | 5.14 | 202.80 | 2.41 | 3.30 | 4.48 | 199.12 |
| rd400 | 15,281 | 2.11 | 3.00 | 4.06 | 104.39 | 1.38 | 2.23 | 3.13 | 58.81 |
| fl417 | 11,861 | 0.14 | 0.45 | 0.81 | 20.89 | 0.08 | 0.29 | 1.02 | 22.73 |
| gr431 | 171,414 | 1.78 | 2.56 | 3.44 | 735.43 | 0.91 | 1.82 | 2.89 | 746.11 |
| pr439 | 107,217 | 0.48 | 1.04 | 1.99 | 398.60 | 0.32 | 1.07 | 2.50 | 705.03 |
| pcb442 | 50,778 | 1.84 | 3.34 | 5.82 | 460.48 | 1.10 | 1.95 | 2.53 | 243.67 |
| d493 | 35,002 | 1.99 | 2.78 | 4.07 | 187.19 | 1.46 | 2.12 | 2.94 | 155.97 |
| att532 | 27,686 | 1.86 | 2.92 | 4.55 | 183.32 | 1.66 | 2.32 | 3.44 | 137.73 |
| ali535 | 202,310 | 1.26 | 2.50 | 3.66 | 1375.95 | 0.80 | 1.98 | 2.70 | 1060.81 |
| u574 | 36,905 | 2.86 | 3.97 | 4.95 | 214.45 | 1.49 | 2.90 | 3.82 | 211.55 |
| rat575 | 6773 | 3.66 | 4.64 | 5.89 | 42.98 | 2.14 | 3.28 | 4.03 | 33.50 |
| p654 | 34,643 | 0.29 | 0.49 | 0.91 | 58.04 | 0.18 | 0.38 | 0.85 | 64.40 |
| d657 | 48,912 | 2.63 | 3.46 | 4.07 | 190.37 | 1.49 | 2.42 | 3.68 | 289.54 |
| gr666 | 294,358 | 2.95 | 4.20 | 5.30 | 2173.31 | 2.06 | 2.94 | 3.89 | 1422.28 |
| u724 | 41,910 | 2.92 | 4.18 | 5.02 | 275.93 | 2.60 | 3.02 | 3.82 | 160.26 |
| rat783 | 8806 | 3.74 | 4.88 | 6.03 | 47.92 | 2.71 | 3.74 | 4.63 | 37.89 |
| Average | | 0.62 | 1.12 | 1.88 | 173.94 | 0.40 | 0.87 | 1.61 | 160.38 |

the LBSA-CO achieved BKS for 15 instances, while the GSAACS achieved BKS for 12 instances. In terms of obtaining BKS, the success rate of the LBSA-CO was 15/21 = 0.71, whereas the success rate of the GSAACS was 12/21 = 0.57.

In Tables 11 and 12, the running times of the LBSA-CO and those of the other methods are also presented. As seen in these tables, the success rate of the LBSA-CO is higher than that of the other methods. In addition, it requires less time than the IBA (Table 12), while it spends more time than the SA and LBSA (Table 11). At this point, the critical question is whether these methods can achieve better results than the LBSA-CO in the same running times. Some experiments were conducted to answer this question. First, nine instances were selected from Table 11. Then, the SA and LBSA were run 20 times on each instance. Instead of the iteration number, the running times of the LBSA-CO were used as the stopping criterion. The obtained results are presented in Table 14. In the table, "Time" is the running time of the LBSA-CO in seconds. As seen in the table, the SA and LBSA cannot achieve the results obtained by the LBSA-CO in the same running times.

The LBSA-CO and other methods were run on eight instances of different sizes. The convergence curves are provided in Fig. 5. As seen in the figure, the LBSA-CO starts to search with better solutions compared to the SA and LBSA. Furthermore, it converges faster in the initial steps of the iterations. Two reasons can account for this superiority: First, it begins searching with an initial population instead of a single initial solution. Second, it improves the initial solutions with the 2-opt operator. The LBSA-CO continues convergence in the next steps without being trapped in the local minima because it produces the neighboring solutions as much as the number of trials using local search operators and finally improves them with the 2-opt operator. Information should be exchanged between the improved solutions individually. This was provided with the crossover operators in the LBSA-CO, and the obtained child solutions contributed to its convergence rate. The 3-opt operator was applied to the best solution found in the last iteration. This allowed the LBSA-CO to converge the global minimum even after the last iteration.

As seen in Fig. 5, the convergence rate of the LBSA-CO was lower than that of the IBA. The 2-opt and 3-opt

**Table 10** The performance comparison of the LBSA-CO and the other methods on 32 instances

| Instance | VTPSO | | | ABCSS | | | DSMO | | | LBSA-CO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BestSol | Avg | StdDev | BestSol | Avg | StdDev | BestSol | Avg | StdDev | BestSol | Avg | StdDev |
| gr17 | 2332.58 | 2332.58 | 0.00 | 2332.58 | 2332.58 | 0.00 | 2332.58 | 2332.58 | 0.00 | 2085.00 | 2085.00 | 0.00 |
| eil51 | 429.51 | 439.87 | 5.65 | 428.98 | 437.01 | 4.98 | 428.86 | 436.96 | 4.73 | 428.86 | 428.97 | 0.14 |
| berlin52 | 7544.37 | 7728.00 | 161.80 | 7544.37 | 7807.86 | 177.55 | 7544.37 | 7633.60 | 85.40 | 7544.32 | 7544.32 | 0.00 |
| st70 | 682.57 | 711.50 | 14.57 | 682.57 | 690.50 | 5.35 | 677.11 | 702.64 | 15.04 | 677.06 | 677.06 | 0.00 |
| eil76 | 559.25 | 569.10 | 9.12 | 550.24 | 561.48 | 7.21 | 558.68 | 572.70 | 7.56 | 544.33 | 545.52 | 2.29 |
| pr76 | 109,586.10 | 112,549.20 | 1420.00 | 108,879.70 | 109,758.57 | 850.64 | 108,159.40 | 111,299.30 | 2050.48 | 108,159.40 | 108,159.40 | 0.00 |
| rat99 | 1256.25 | 1325.80 | 34.83 | 1242.32 | 1265.93 | 13.54 | 1225.56 | 1291.93 | 21.07 | 1219.25 | 1221.40 | 2.11 |
| kroA100 | 21,307.44 | 22,400.48 | 529.13 | 21,299.00 | 21,878.83 | 455.63 | 21,298.21 | 22,024.27 | 508.89 | 21,285.47 | 21,296.26 | 25.80 |
| kroB100 | 22,475.67 | 23,236.43 | 522.63 | 22,229.71 | 22,707.96 | 259.83 | 22,308.00 | 23,022.37 | 277.32 | 22,139.03 | 22,170.84 | 38.04 |
| rd100 | 8094.75 | 8502.73 | 167.70 | 7944.32 | 8207.80 | 172.52 | 8041.30 | 8377.76 | 209.40 | 7910.41 | 7926.51 | 33.73 |
| eil101 | 653.16 | 679.14 | 13.29 | 646.05 | 662.63 | 7.13 | 648.66 | 674.40 | 10.97 | 640.13 | 644.76 | 3.24 |
| lin105 | 14,581.58 | 15,684.64 | 610.38 | 14,406.12 | 14,766.55 | 263.01 | 14,383.00 | 15,114.00 | 500.76 | 14,382.94 | 14,386.41 | 8.47 |
| pr107 | 44,436.25 | 45,287.90 | 1207.52 | 44,525.68 | 44,927.27 | 319.03 | 44,385.86 | 45,666.99 | 1300.43 | 44,301.64 | 44,331.93 | 60.20 |
| pr124 | 61,076.73 | 63,939.97 | 1739.40 | 59,030.74 | 59,772.68 | 516.56 | 60,285.21 | 62,443.49 | 1644.93 | 59,030.75 | 59,041.77 | 19.57 |
| pr136 | 99,247.01 | 102,945.70 | 1688.20 | 97,853.91 | 101,795.57 | 1916.45 | 97,538.68 | 102,872.00 | 2855.28 | 96,795.51 | 97,221.60 | 230.97 |
| kroA150 | 27,232.10 | 28,262.98 | 455.95 | 26,981.98 | 27,971.36 | 554.50 | 27,591.44 | 28,354.09 | 524.91 | 26,524.85 | 26,586.03 | 74.29 |
| kroB150 | 26,579.73 | 27,987.85 | 620.72 | 26,760.79 | 27,653.49 | 509.24 | 26,601.94 | 27,576.16 | 625.26 | 26,127.37 | 26,218.44 | 107.37 |
| pr152 | 74,414.17 | 76,272.37 | 1199.90 | 74,337.62 | 76,097.48 | 904.45 | 74,243.91 | 76,526.77 | 1663.08 | 73,683.52 | 73,814.66 | 99.80 |
| u159 | 43,579.82 | 45,441.55 | 1178.06 | 42,862.51 | 45,234.92 | 1212.54 | 42,598.30 | 42,598.30 | 0.00 | 42,075.73 | 42,163.52 | 147.06 |
| rat195 | 2452.92 | 2554.10 | 47.56 | 2469.31 | 2579.27 | 44.14 | 2372.89 | 2488.55 | 50.48 | 2342.82 | 2357.47 | 10.31 |
| d198 | 16,066.44 | 16,489.28 | 206.79 | 16,270.22 | 16,483.73 | 162.08 | 15,978.13 | 16,270.47 | 171.20 | 15,818.02 | 15,860.97 | 32.64 |
| kroA200 | 30,602.81 | 31,502.33 | 531.41 | 30,701.86 | 31,938.81 | 656.84 | 30,481.35 | 31,828.64 | 652.32 | 29,369.37 | 29,472.87 | 90.86 |
| kroB200 | 30,767.52 | 31,923.15 | 553.02 | 31,508.85 | 32,208.73 | 526.77 | 30,716.50 | 31,781.62 | 487.39 | 29,450.48 | 29,595.98 | 145.68 |
| tsp225 | 4095.01 | 4210.65 | 66.44 | 4140.24 | 4276.92 | 72.80 | 4013.68 | 4162.79 | 66.08 | 3867.81 | 3922.14 | 28.45 |
| pr226 | 81,050.23 | 88,031.31 | 3461.96 | 82,266.00 | 87,400.60 | 3482.64 | 83,587.98 | 85,935.69 | 2105.13 | 80,370.27 | 80,430.30 | 144.03 |
| gil262 | 2547.16 | 2631.69 | 39.44 | 2713.75 | 2839.11 | 73.01 | 2543.15 | 2627.87 | 42.39 | 2397.34 | 2422.19 | 15.85 |
| pr299 | 50,571.83 | 53,154.34 | 1389.36 | 64,464.76 | 67,620.95 | 2092.63 | 50,579.82 | 51,747.99 | 863.32 | 48,248.90 | 48,525.24 | 200.33 |
| lin318 | 44,724.38 | 46,209.53 | 773.01 | 55,744.52 | 61,902.84 | 3824.65 | 44,118.66 | 45,460.25 | 660.47 | 42,094.04 | 42,710.49 | 219.32 |
| linhp318 | 44,337.02 | 46,329.87 | 948.05 | 56,834.19 | 60,853.66 | 2306.85 | 43,831.44 | 45,730.57 | 929.73 | 42,062.38 | 42,606.31 | 274.77 |
| fl417 | 12,376.53 | 12,980.24 | 311.88 | 22,253.99 | 27,237.13 | 2479.90 | 12,218.98 | 12,950.77 | 360.99 | 11,934.12 | 11,956.66 | 28.96 |
| pr439 | 112,088.00 | 119,442.20 | 2770.58 | 206,233.14 | 244,192.44 | 21,576.50 | 112,105.20 | 116,379.20 | 2462.82 | 107,573.76 | 108,506.44 | 646.96 |
| d493 | 37,132.09 | 38,159.18 | 603.84 | 68,556.68 | 78,968.31 | 5776.85 | 36,844.63 | 37,861.14 | 426.97 | 35,376.00 | 35,781.19 | 198.47 |
| Average | 1,034,880.98 | 1,079,915.66 | 23,282.19 | 1,204,696.70 | 1,293,032.97 | 51,225.82 | 1,030,243.48 | 1,064,745.86 | 21,584.80 | 1,006,460.88 | 1,010,612.62 | 2889.69 |

**Table 11** The performance comparison of the LBSA-CO, SA and LBSA methods on 65 instances

| Instance | BKS | SA | | | | | LBSA | | | | | LBSA-CO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | StdDev | Time | Best | Mean | Worst | StdDev | Time | Best | Mean | Worst | StdDev | Time |
| burma14 | 3323 | 0.00 | 0.00 | 0.00 | 0.00 | 3.21 | 0.00 | 0.20 | 1.08 | 10.59 | 1.07 | 0.00 | 0.00 | 0.00 | 0.00 | 4.68 |
| ulysses16 | 6859 | 0.00 | 0.07 | 0.16 | 5.61 | 3.18 | 0.00 | 0.22 | 0.77 | 15.09 | 1.15 | 0.00 | 0.00 | 0.00 | 0.00 | 4.72 |
| gr17 | 2085 | 0.00 | 0.10 | 0.24 | 2.48 | 3.10 | 0.00 | 0.27 | 1.73 | 8.97 | 1.19 | 0.00 | 0.00 | 0.00 | 0.00 | 4.75 |
| ulysses22 | 7013 | 0.00 | 0.29 | 2.60 | 48.01 | 3.14 | 0.00 | 0.25 | 1.73 | 35.82 | 1.55 | 0.00 | 0.00 | 0.00 | 0.00 | 4.90 |
| bays29 | 2020 | 0.00 | 0.72 | 2.52 | 15.48 | 3.19 | 0.00 | 0.47 | 0.99 | 5.82 | 2.04 | 0.00 | 0.00 | 0.00 | 0.00 | 5.10 |
| dantzig42 | 699 | 0.00 | 1.99 | 5.87 | 12.88 | 3.27 | 0.00 | 0.87 | 2.86 | 5.37 | 3.00 | 0.00 | 0.00 | 0.00 | 0.00 | 5.55 |
| swiss42 | 1273 | 0.00 | 2.20 | 6.21 | 31.90 | 3.27 | 0.00 | 0.38 | 2.20 | 7.13 | 3.00 | 0.00 | 0.00 | 0.00 | 0.00 | 5.55 |
| att48 | 10,628 | 0.24 | 1.85 | 5.53 | 135.60 | 3.30 | 0.26 | 1.37 | 3.48 | 97.75 | 3.46 | 0.00 | 0.14 | 0.53 | 24.69 | 5.84 |
| eil51 | 426 | 1.41 | 2.89 | 4.46 | 3.53 | 3.31 | 0.00 | 2.10 | 4.46 | 5.46 | 3.68 | 0.00 | 0.19 | 0.47 | 0.52 | 5.97 |
| berlin52 | 7542 | 0.00 | 4.20 | 8.49 | 202.69 | 3.34 | 0.00 | 1.32 | 6.15 | 159.79 | 3.75 | 0.00 | 0.00 | 0.00 | 0.00 | 5.90 |
| st70 | 675 | 0.89 | 3.59 | 6.81 | 8.84 | 3.44 | 0.59 | 2.07 | 4.74 | 7.01 | 5.16 | 0.00 | 0.39 | 1.33 | 2.85 | 6.80 |
| eil76 | 538 | 1.86 | 3.92 | 6.32 | 7.22 | 3.48 | 0.74 | 2.53 | 5.39 | 6.39 | 5.63 | 0.00 | 0.44 | 1.86 | 3.53 | 7.14 |
| pr76 | 108,159 | 2.09 | 5.40 | 9.90 | 1983.71 | 3.51 | 0.12 | 1.53 | 3.26 | 1102.77 | 5.63 | 0.00 | 0.17 | 0.86 | 380.02 | 7.00 |
| gr96 | 55,209 | 2.38 | 5.78 | 10.84 | 1271.63 | 3.61 | 0.17 | 1.89 | 3.02 | 517.43 | 7.25 | 0.00 | 0.27 | 0.68 | 122.07 | 8.03 |
| rat99 | 1211 | 3.22 | 6.38 | 9.83 | 21.97 | 3.63 | 0.50 | 3.00 | 5.37 | 18.83 | 7.55 | 0.00 | 0.52 | 1.57 | 5.26 | 8.24 |
| kroA100 | 21,282 | 0.50 | 4.46 | 10.63 | 478.48 | 3.63 | 0.16 | 1.61 | 5.06 | 235.65 | 7.72 | 0.00 | 0.02 | 0.11 | 7.57 | 8.23 |
| kroB100 | 22,141 | 0.47 | 4.57 | 9.06 | 583.80 | 3.73 | 0.61 | 2.73 | 5.08 | 243.33 | 7.62 | 0.00 | 0.18 | 0.98 | 53.70 | 8.36 |
| kroC100 | 20,749 | 1.04 | 5.45 | 11.93 | 599.87 | 3.63 | 0.71 | 2.37 | 6.82 | 309.60 | 7.58 | 0.00 | 0.16 | 1.13 | 65.31 | 8.24 |
| kroD100 | 21,294 | 1.38 | 5.32 | 9.12 | 408.68 | 3.66 | 0.38 | 2.29 | 4.75 | 257.34 | 7.62 | 0.00 | 0.35 | 1.40 | 101.15 | 8.31 |
| kroE100 | 22,068 | 1.40 | 4.67 | 10.14 | 433.29 | 3.64 | 0.50 | 1.91 | 3.91 | 249.02 | 7.63 | 0.15 | 0.42 | 1.05 | 55.37 | 8.31 |
| rd100 | 7910 | 0.77 | 5.49 | 11.59 | 255.41 | 3.64 | 0.16 | 2.45 | 6.69 | 154.31 | 7.62 | 0.00 | 0.39 | 1.67 | 33.79 | 8.29 |
| eil101 | 629 | 2.54 | 4.40 | 6.68 | 7.24 | 3.65 | 1.43 | 3.47 | 5.88 | 8.69 | 7.71 | 0.00 | 0.85 | 2.23 | 4.80 | 8.54 |
| lin105 | 14,379 | 1.35 | 6.09 | 13.90 | 424.60 | 3.68 | 0.35 | 2.15 | 4.82 | 183.86 | 8.13 | 0.00 | 0.00 | 0.00 | 0.00 | 8.52 |
| pr107 | 44,303 | 1.02 | 3.91 | 8.74 | 911.70 | 3.70 | 0.09 | 1.20 | 3.29 | 347.80 | 8.18 | 0.00 | 0.21 | 0.61 | 85.67 | 8.94 |
| pr124 | 59,030 | 0.96 | 4.56 | 10.14 | 1600.25 | 3.81 | 0.29 | 2.10 | 4.16 | 585.90 | 9.69 | 0.00 | 0.06 | 0.88 | 116.09 | 9.59 |
| bier127 | 118,282 | 2.92 | 6.36 | 10.42 | 2327.27 | 3.83 | 0.38 | 2.09 | 4.03 | 1154.60 | 9.90 | 0.00 | 0.48 | 1.36 | 546.97 | 10.10 |
| ch130 | 6110 | 2.73 | 5.71 | 7.77 | 83.77 | 3.82 | 0.31 | 2.78 | 4.39 | 63.51 | 10.24 | 0.00 | 0.77 | 1.67 | 31.47 | 10.48 |
| pr136 | 96,772 | 2.62 | 6.06 | 10.82 | 2071.78 | 3.86 | 0.92 | 2.98 | 5.22 | 1268.41 | 10.71 | 0.01 | 0.60 | 1.83 | 488.74 | 10.91 |
| gr137 | 69,853 | 1.28 | 5.39 | 9.29 | 1506.83 | 3.89 | 0.51 | 1.93 | 3.02 | 533.22 | 10.89 | 0.00 | 0.31 | 1.23 | 300.26 | 10.85 |
| pr144 | 58,537 | 0.48 | 6.08 | 18.22 | 2916.65 | 3.93 | 0.77 | 3.18 | 9.66 | 1564.78 | 11.47 | 0.00 | 0.05 | 0.09 | 27.19 | 10.93 |
| ch150 | 6528 | 3.23 | 7.05 | 12.03 | 135.69 | 4.11 | 1.52 | 3.28 | 5.55 | 72.78 | 12.06 | 0.00 | 0.66 | 1.75 | 31.93 | 11.92 |
| kroA150 | 26,524 | 3.88 | 6.23 | 9.05 | 386.80 | 4.01 | 1.14 | 2.80 | 5.74 | 340.46 | 12.06 | 0.00 | 0.60 | 1.56 | 114.29 | 12.06 |
| kroB150 | 26,130 | 2.91 | 6.03 | 9.27 | 492.73 | 3.97 | 1.37 | 2.54 | 4.37 | 240.58 | 12.03 | 0.00 | 0.48 | 2.04 | 118.53 | 12.03 |
| pr152 | 73,682 | 1.78 | 3.81 | 6.55 | 1125.50 | 4.02 | 0.76 | 1.74 | 3.17 | 453.20 | 12.20 | 0.00 | 0.25 | 1.02 | 178.45 | 11.82 |
| u159 | 42,080 | 2.96 | 6.45 | 15.37 | 1377.22 | 4.08 | 0.72 | 3.40 | 7.29 | 848.21 | 12.94 | 0.00 | 0.60 | 1.46 | 164.05 | 12.81 |

**Table 11** (continued)

| Instance | BKS | SA | | | | | LBSA | | | | | LBSA-CO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | StdDev | Time | Best | Mean | Worst | StdDev | Time | Best | Mean | Worst | StdDev | Time |
| rat195 | 2323 | 5.77 | 7.58 | 9.86 | 27.22 | 4.25 | 2.97 | 4.92 | 7.36 | 30.29 | 16.37 | 0.52 | 1.33 | 2.88 | 15.25 | 17.53 |
| d198 | 15,780 | 1.81 | 3.11 | 4.76 | 155.16 | 4.32 | 0.78 | 2.03 | 3.43 | 119.55 | 16.72 | 0.05 | 0.29 | 0.61 | 24.30 | 18.52 |
| kroA200 | 29,368 | 3.79 | 6.68 | 9.94 | 491.82 | 4.26 | 1.73 | 4.04 | 8.56 | 465.90 | 16.86 | 0.16 | 0.72 | 1.56 | 133.09 | 18.69 |
| kroB200 | 29,437 | 2.95 | 6.11 | 8.84 | 515.95 | 4.30 | 2.21 | 4.10 | 6.12 | 338.84 | 16.91 | 0.04 | 0.61 | 1.51 | 132.24 | 17.57 |
| gr202 | 40,160 | 3.53 | 5.52 | 8.87 | 489.43 | 4.35 | 1.85 | 3.20 | 4.86 | 317.32 | 17.08 | 0.34 | 1.20 | 2.19 | 196.39 | 17.91 |
| tsp225 | 3919 | 3.96 | 6.77 | 9.62 | 68.61 | 4.45 | 1.43 | 4.46 | 8.34 | 62.43 | 19.36 | 0.89 | 1.44 | 2.32 | 16.28 | 23.30 |
| ts225 | 126,643 | 0.06 | 5.08 | 9.23 | 3361.17 | 4.48 | 0.63 | 5.19 | 8.70 | 2644.68 | 19.43 | 0.00 | 0.00 | 0.00 | 0.00 | 20.11 |
| pr226 | 80,369 | 0.51 | 3.70 | 7.15 | 1232.73 | 4.49 | 1.31 | 3.90 | 14.57 | 2530.48 | 19.60 | 0.00 | 0.13 | 0.62 | 183.23 | 22.97 |
| gr229 | 134,602 | 2.60 | 5.71 | 9.34 | 2300.80 | 4.54 | 1.19 | 3.37 | 5.83 | 1399.88 | 19.79 | 0.38 | 1.21 | 2.02 | 583.61 | 23.63 |
| gil262 | 2378 | 4.71 | 7.51 | 11.14 | 39.40 | 4.79 | 3.03 | 5.14 | 7.57 | 28.00 | 23.56 | 0.21 | 1.59 | 3.20 | 16.22 | 31.64 |
| pr264 | 49,135 | 4.77 | 9.06 | 13.05 | 976.65 | 4.83 | 3.08 | 6.76 | 13.49 | 1310.64 | 23.73 | 0.00 | 0.19 | 0.63 | 108.38 | 27.30 |
| a280 | 2579 | 4.61 | 9.81 | 15.12 | 58.77 | 4.85 | 3.88 | 7.40 | 12.21 | 51.99 | 25.72 | 0.00 | 0.76 | 2.02 | 16.19 | 30.64 |
| pr299 | 48,191 | 5.75 | 8.54 | 10.53 | 716.47 | 5.03 | 3.22 | 5.27 | 9.72 | 709.35 | 27.79 | 0.10 | 0.53 | 1.28 | 164.67 | 44.44 |
| lin318 | 42,029 | 5.48 | 8.05 | 10.70 | 610.05 | 5.14 | 3.76 | 5.84 | 8.85 | 621.33 | 30.11 | 0.17 | 1.27 | 2.12 | 221.16 | 50.47 |
| linhp318 | 41,345 | 7.61 | 10.79 | 17.06 | 920.13 | 5.14 | 4.74 | 7.78 | 11.49 | 657.97 | 30.10 | 2.41 | 3.30 | 4.48 | 199.12 | 48.09 |
| rd400 | 15,281 | 6.82 | 8.80 | 11.32 | 185.81 | 5.80 | 4.17 | 6.44 | 8.45 | 183.50 | 41.12 | 1.38 | 2.23 | 3.13 | 58.81 | 103.36 |
| fl417 | 11,861 | 2.46 | 8.09 | 12.85 | 411.50 | 5.94 | 2.22 | 4.88 | 10.84 | 268.99 | 43.28 | 0.08 | 0.29 | 1.02 | 22.73 | 90.88 |
| gr431 | 171,414 | 5.18 | 8.38 | 9.93 | 2119.70 | 6.03 | 3.10 | 4.89 | 7.54 | 1824.23 | 45.44 | 0.91 | 1.82 | 2.89 | 746.11 | 168.04 |
| pr439 | 107,217 | 8.01 | 12.02 | 16.21 | 2439.16 | 6.11 | 4.40 | 7.44 | 9.96 | 1752.22 | 46.51 | 0.32 | 1.07 | 2.50 | 705.03 | 123.66 |
| pcb442 | 50,778 | 7.88 | 10.40 | 13.54 | 718.59 | 6.11 | 3.87 | 7.70 | 11.82 | 1125.17 | 47.74 | 1.10 | 1.95 | 2.53 | 243.67 | 122.93 |
| d493 | 35,002 | 7.77 | 10.76 | 14.73 | 601.92 | 6.61 | 3.74 | 6.16 | 9.04 | 485.95 | 55.35 | 1.46 | 2.12 | 2.94 | 155.97 | 209.42 |
| att532 | 27,686 | 8.99 | 11.85 | 13.99 | 390.30 | 7.03 | 4.87 | 7.02 | 9.20 | 358.68 | 61.47 | 1.66 | 2.32 | 3.44 | 137.73 | 274.60 |
| ali535 | 202,310 | 12.06 | 15.38 | 18.27 | 3323.14 | 7.01 | 5.88 | 7.85 | 11.28 | 2758.85 | 64.29 | 0.80 | 1.98 | 2.70 | 1060.81 | 287.68 |
| u574 | 36,905 | 11.89 | 13.95 | 15.87 | 418.07 | 7.34 | 6.51 | 8.43 | 13.20 | 662.67 | 70.07 | 1.49 | 2.90 | 3.82 | 211.55 | 363.26 |
| rat575 | 6773 | 13.21 | 16.34 | 19.18 | 93.21 | 7.57 | 6.27 | 9.34 | 11.68 | 109.97 | 71.56 | 2.14 | 3.28 | 4.03 | 33.50 | 351.30 |
| p654 | 34,643 | 8.22 | 13.34 | 17.10 | 666.36 | 7.79 | 6.02 | 8.68 | 16.41 | 821.46 | 86.38 | 0.18 | 0.38 | 0.85 | 64.40 | 396.86 |
| d657 | 48,912 | 13.85 | 17.14 | 20.17 | 751.82 | 7.94 | 5.33 | 10.02 | 14.64 | 1194.06 | 86.56 | 1.49 | 2.42 | 3.68 | 289.54 | 589.85 |
| gr666 | 294,358 | 14.17 | 19.02 | 21.62 | 5187.82 | 7.92 | 6.19 | 8.55 | 10.87 | 4129.35 | 87.58 | 2.06 | 2.94 | 3.89 | 1422.28 | 666.06 |
| u724 | 41,910 | 17.20 | 19.63 | 21.28 | 549.45 | 8.33 | 7.33 | 11.23 | 17.19 | 1153.22 | 95.14 | 2.60 | 3.02 | 3.82 | 160.26 | 690.92 |
| rat783 | 8806 | 18.33 | 26.94 | 29.29 | 201.29 | 8.85 | 8.20 | 12.24 | 18.61 | 234.79 | 106.28 | 2.71 | 3.74 | 4.63 | 37.89 | 997.08 |
| Average | | 4.03 | 7.13 | 10.79 | 798.49 | 4.64 | 1.99 | 4.03 | 7.11 | 607.25 | 24.32 | 0.40 | 0.87 | 1.61 | 160.38 | 94.43 |

**Table 12** The performance comparison of the LBSA-CO and IBA methods on 37 instances

| Instance | BKS | IBA | | | | | LBSA-CO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Mean | Worst | StdDev | Time | Best | Mean | Worst | StdDev | Time |
| burma14 | 3323 | 0.00 | 0.02 | 0.39 | 2.91 | 1.39 | 0.00 | 0.00 | 0.00 | 0.00 | 4.68 |
| ulysses16 | 6859 | 0.00 | 0.03 | 0.16 | 3.77 | 2.49 | 0.00 | 0.00 | 0.00 | 0.00 | 4.72 |
| gr17 | 2085 | 0.00 | 0.15 | 0.24 | 2.01 | 2.69 | 0.00 | 0.00 | 0.00 | 0.00 | 4.75 |
| ulysses22 | 7013 | 0.00 | 0.24 | 1.06 | 30.50 | 7.97 | 0.00 | 0.00 | 0.00 | 0.00 | 4.90 |
| bays29 | 2020 | 0.00 | 0.96 | 2.38 | 14.68 | 16.31 | 0.00 | 0.00 | 0.00 | 0.00 | 5.10 |
| dantzig42 | 699 | 0.00 | 0.36 | 1.57 | 3.47 | 59.26 | 0.00 | 0.00 | 0.00 | 0.00 | 5.55 |
| swiss42 | 1273 | 0.00 | 1.03 | 3.38 | 13.50 | 62.99 | 0.00 | 0.00 | 0.00 | 0.00 | 5.55 |
| att48 | 10,628 | 0.19 | 0.66 | 1.98 | 55.25 | 92.19 | 0.00 | 0.14 | 0.53 | 24.69 | 5.84 |
| eil51 | 426 | 0.23 | 1.91 | 3.05 | 3.23 | 108.90 | 0.00 | 0.19 | 0.47 | 0.52 | 5.97 |
| berlin52 | 7542 | 0.00 | 2.69 | 6.39 | 142.33 | 126.33 | 0.00 | 0.00 | 0.00 | 0.00 | 5.90 |
| st70 | 675 | 0.15 | 1.40 | 2.52 | 4.55 | 343.91 | 0.00 | 0.39 | 1.33 | 2.85 | 6.80 |
| eil76 | 538 | 1.86 | 3.03 | 4.28 | 3.76 | 443.85 | 0.00 | 0.44 | 1.86 | 3.53 | 7.14 |
| pr76 | 108,159 | 0.15 | 1.08 | 1.97 | 515.38 | 403.38 | 0.00 | 0.17 | 0.86 | 380.02 | 7.00 |
| gr96 | 55,209 | 0.81 | 1.89 | 3.36 | 424.22 | 858.74 | 0.00 | 0.27 | 0.68 | 122.07 | 8.03 |
| rat99 | 1211 | 1.82 | 4.22 | 7.35 | 15.83 | 1016.14 | 0.00 | 0.52 | 1.57 | 5.26 | 8.24 |
| kroA100 | 21,282 | 0.16 | 0.87 | 2.30 | 121.11 | 1035.40 | 0.00 | 0.02 | 0.11 | 7.57 | 8.23 |
| kroB100 | 22,141 | 0.41 | 1.81 | 3.20 | 186.56 | 1004.49 | 0.00 | 0.18 | 0.98 | 53.70 | 8.36 |
| kroC100 | 20,749 | 0.17 | 1.73 | 3.81 | 225.12 | 1068.89 | 0.00 | 0.16 | 1.13 | 65.31 | 8.24 |
| kroD100 | 21,294 | 0.45 | 1.53 | 4.60 | 188.78 | 973.96 | 0.00 | 0.35 | 1.40 | 101.15 | 8.31 |
| kroE100 | 22,068 | 1.06 | 1.79 | 2.58 | 105.57 | 1026.48 | 0.15 | 0.42 | 1.05 | 55.37 | 8.31 |
| rd100 | 7910 | 0.04 | 2.36 | 4.78 | 109.32 | 979.71 | 0.00 | 0.39 | 1.67 | 33.79 | 8.29 |
| eil101 | 629 | 2.86 | 4.00 | 6.36 | 5.93 | 1023.94 | 0.00 | 0.85 | 2.23 | 4.80 | 8.54 |
| lin105 | 14,379 | 0.00 | 1.25 | 2.65 | 116.57 | 1164.05 | 0.00 | 0.00 | 0.00 | 0.00 | 8.52 |
| pr107 | 44,303 | 0.10 | 1.03 | 1.69 | 168.27 | 1298.96 | 0.00 | 0.21 | 0.61 | 85.67 | 8.94 |
| pr124 | 59,030 | 0.00 | 0.49 | 1.34 | 245.42 | 1973.23 | 0.00 | 0.06 | 0.88 | 116.09 | 9.59 |
| bier127 | 118,282 | 0.50 | 2.04 | 4.00 | 1012.46 | 2195.48 | 0.00 | 0.48 | 1.36 | 546.97 | 10.10 |
| ch130 | 6110 | 1.49 | 2.73 | 4.44 | 42.21 | 2269.06 | 0.00 | 0.77 | 1.67 | 31.47 | 10.48 |
| pr136 | 96,772 | 1.56 | 3.14 | 4.85 | 906.11 | 2645.00 | 0.01 | 0.60 | 1.83 | 488.74 | 10.91 |
| gr137 | 69,853 | 1.45 | 2.69 | 4.12 | 552.08 | 2802.05 | 0.00 | 0.31 | 1.23 | 300.26 | 10.85 |
| pr144 | 58,537 | 0.00 | 0.12 | 0.39 | 86.08 | 3203.26 | 0.00 | 0.05 | 0.09 | 27.19 | 10.93 |
| ch150 | 6528 | 1.53 | 3.21 | 5.41 | 57.47 | 3823.77 | 0.00 | 0.66 | 1.75 | 31.93 | 11.92 |
| kroA150 | 26,524 | 1.52 | 2.54 | 3.88 | 146.33 | 3770.33 | 0.00 | 0.60 | 1.56 | 114.29 | 12.06 |
| kroB150 | 26,130 | 1.04 | 2.30 | 4.23 | 216.11 | 3668.50 | 0.00 | 0.48 | 2.04 | 118.53 | 12.03 |
| pr152 | 73,682 | 0.27 | 1.18 | 2.18 | 381.15 | 3689.78 | 0.00 | 0.25 | 1.02 | 178.45 | 11.82 |
| u159 | 42,080 | 1.11 | 2.90 | 5.12 | 476.98 | 4363.51 | 0.00 | 0.60 | 1.46 | 164.05 | 12.81 |
| rat195 | 2323 | 3.31 | 4.78 | 6.97 | 19.88 | 8201.29 | 0.52 | 1.33 | 2.88 | 15.25 | 17.53 |
| d198 | 15,780 | 1.29 | 1.69 | 2.28 | 35.64 | 8690.04 | 0.05 | 0.29 | 0.61 | 24.30 | 18.52 |
| Average | | 0.69 | 1.78 | 3.28 | 179.47 | 1741.02 | 0.02 | 0.30 | 0.94 | 83.89 | 8.80 |

operators used in the IBA helped it converge quickly. However, these operators also cause the IBA to get trapped in the local minima (Table 12). The LBSA-CO effectively adjusts the balance between the faster convergence and the escaping from the local minima. It outperforms the IBA for six of the eight instances. Also, the LBSA-CO achieves the best solution in the first 50 iterations for burma14, dantzig42, eil51, pr76 and kroA100.

## 4.5 Statistical analysis

A series of statistical tests were performed to compare the performance of the LBSA-CO with those of the other methods. IBM SPSS Statistics 26 was used for these tests. Table 15 provides the results of Kolmogorov–Smirnov and Shapiro–Wilk normality tests for the LBSA-CO, VTPSO, ABCSS and DSMO. These results were calculated based

**Table 13** The performance comparison of the LBSA-CO and GSAACS methods on 21 instances

| Instance | BKS | GSAACS | | LBSA-CO | |
|---|---|---|---|---|---|
| | | Best | Mean | Best | Mean |
| eil51 | 426 | 0.23 | 0.30 | 0.00 | 0.19 |
| berlin52 | 7542 | 0.00 | 0.00 | 0.00 | 0.00 |
| eil76 | 538 | 0.00 | 0.41 | 0.00 | 0.44 |
| kroA100 | 21,282 | 0.00 | 0.42 | 0.00 | 0.02 |
| kroB100 | 22,141 | 0.00 | 0.64 | 0.00 | 0.18 |
| kroC100 | 20,749 | 0.00 | 0.63 | 0.00 | 0.16 |
| kroD100 | 21,294 | 0.07 | 1.53 | 0.00 | 0.35 |
| kroE100 | 22,068 | 0.00 | 0.52 | 0.15 | 0.42 |
| rd100 | 7910 | 0.00 | 0.98 | 0.00 | 0.39 |
| eil101 | 629 | 0.16 | 0.99 | 0.00 | 0.85 |
| lin105 | 14,379 | 0.00 | 0.19 | 0.00 | 0.00 |
| bier127 | 118,282 | 0.00 | 0.96 | 0.00 | 0.48 |
| ch130 | 6110 | 0.51 | 1.57 | 0.00 | 0.77 |
| ch150 | 6528 | 0.00 | 0.55 | 0.00 | 0.66 |
| kroA150 | 26,524 | 0.00 | 1.41 | 0.00 | 0.60 |
| kroB150 | 26,130 | 0.00 | 1.22 | 0.00 | 0.48 |
| kroA200 | 29,368 | 0.05 | 1.26 | 0.16 | 0.72 |
| kroB200 | 29,437 | 0.35 | 2.03 | 0.04 | 0.61 |
| lin318 | 42,029 | 1.09 | 2.32 | 0.17 | 1.27 |
| rat575 | 6773 | 1.74 | 2.38 | 2.14 | 3.28 |
| rat783 | 8806 | 2.07 | 3.10 | 2.71 | 3.74 |
| Average | | 0.30 | 1.11 | 0.26 | 0.74 |

on the column labeled "BestSol" in Table 10. As can be understood from the p-values (Sig.) in Table 15, the data were not distributed normally. For this reason, the Friedman test, which is a nonparametric test, was used to determine whether there are statistically significant differences between the methods. Table 16 presents the results of this test. The p-value (Asymp. Sig.) indicates that there are significant differences between the methods. To identify which methods differed from each other, the Wilcoxon signed-rank test, a nonparametric test, was used for the pairwise comparisons. Table 17 provides the results of this test. The p-values in the table indicated that there were statistically significant differences between the LBSA-CO and the other methods.

Table 18 provides the results of Kolmogorov–Smirnov and Shapiro–Wilk normality tests for the LBSA-CO, SA and LBSA. These results were calculated based on the column labeled "Best" in Table 11. The p-values (Sig.) in Table 18 indicated that the data were not distributed normally. For this reason, the Friedman test was conducted. The results are presented in Table 19. The p-value showed that that there were significant differences between these methods. Then the Wilcoxon signed-rank test was used for the pairwise comparisons, and the results are provided in Table 20. The p-values showed that there were statistically significant differences between the LBSA-CO and other methods.

Table 21 provides the results of Kolmogorov–Smirnov and Shapiro–Wilk normality tests for the LBSA-CO and IBA. These results were calculated based on the column

**Table 14** The performance comparison of the LBSA-CO, SA and LBSA methods in the same running times on nine instances

| Instance | BKS | Time | SA | | | | LBSA | | | | LBSA-CO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Best | Mean | Worst | StdDev | Best | Mean | Worst | StdDev | Best | Mean | Worst | StdDev |
| eil101 | 629 | 8.54 | 2.54 | 4.59 | 6.84 | 7.96 | 0.32 | 2.95 | 5.09 | 7.76 | 0.00 | 0.85 | 2.23 | 4.80 |
| kroA150 | 26,524 | 12.06 | 3.12 | 5.46 | 8.51 | 400.09 | 0.81 | 3.34 | 6.61 | 403.96 | 0.00 | 0.60 | 1.56 | 114.29 |
| d198 | 15,780 | 18.52 | 1.16 | 2.82 | 3.69 | 112.51 | 0.48 | 1.40 | 3.33 | 124.16 | 0.05 | 0.29 | 0.61 | 24.30 |
| gil262 | 2378 | 31.64 | 5.05 | 7.29 | 11.02 | 39.23 | 1.72 | 3.62 | 6.01 | 29.16 | 0.21 | 1.59 | 3.20 | 16.22 |
| a280 | 2579 | 30.64 | 4.81 | 8.65 | 12.18 | 55.77 | 2.91 | 5.28 | 8.72 | 40.85 | 0.00 | 0.76 | 2.02 | 16.19 |
| gr431 | 171,414 | 168.04 | 3.83 | 6.53 | 11.36 | 3336.93 | 2.91 | 4.73 | 7.56 | 2564.49 | 0.91 | 1.82 | 2.89 | 746.11 |
| d493 | 35,002 | 209.42 | 4.95 | 6.71 | 9.54 | 477.63 | 2.96 | 4.38 | 5.38 | 205.37 | 1.46 | 2.12 | 2.94 | 155.97 |
| p654 | 34,643 | 396.86 | 0.87 | 6.27 | 9.73 | 834.24 | 2.31 | 4.69 | 7.95 | 534.30 | 0.18 | 0.38 | 0.85 | 64.40 |
| u724 | 41,910 | 690.92 | 6.08 | 7.72 | 9.63 | 379.05 | 4.90 | 6.26 | 7.61 | 280.67 | 2.60 | 3.02 | 3.82 | 160.26 |
| Average | | | 3.60 | 6.23 | 9.16 | 627.05 | 2.15 | 4.07 | 6.47 | 465.64 | 0.60 | 1.27 | 2.24 | 144.73 |

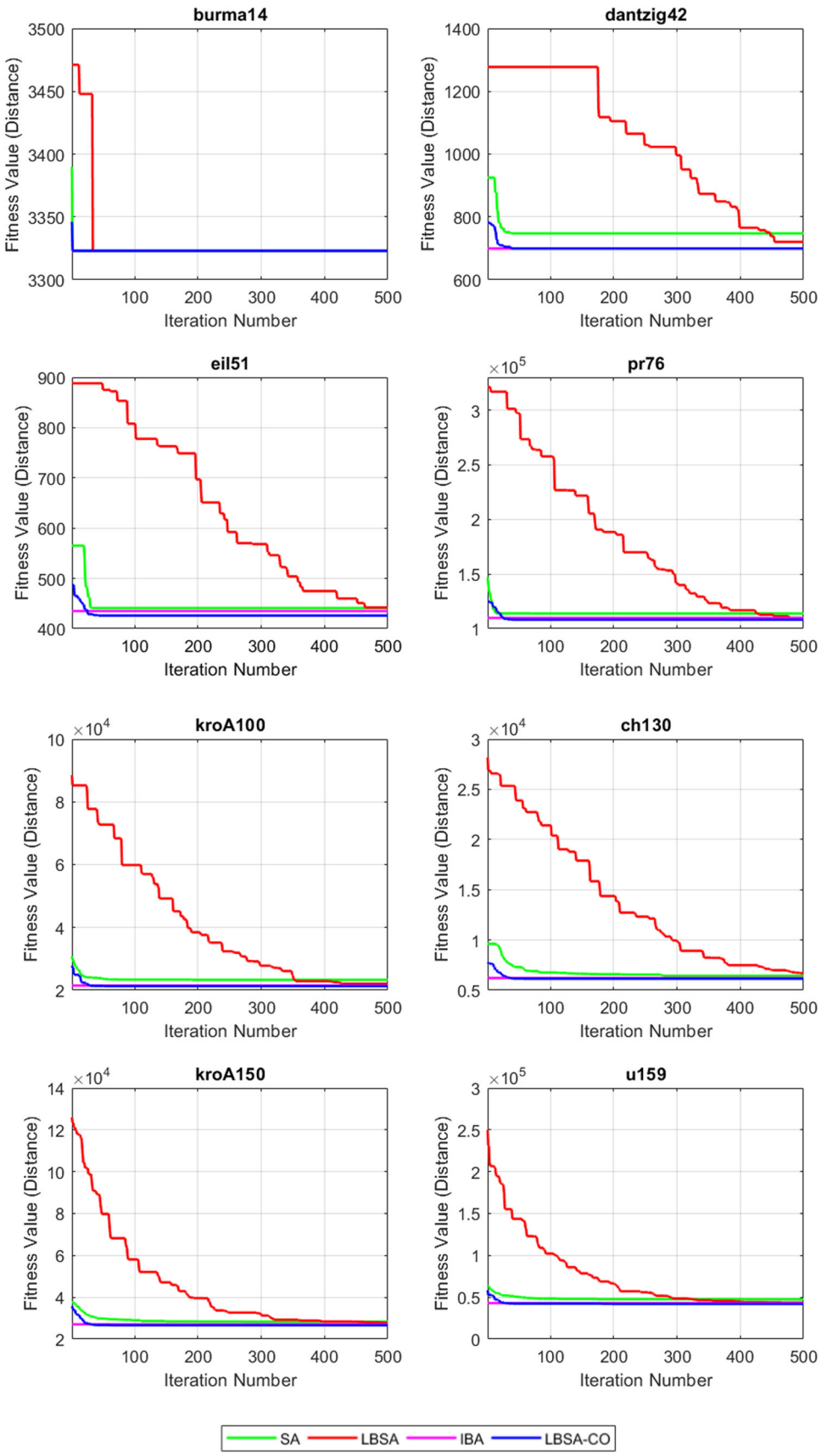


**Fig. 5** Comparison of the convergence curves of the LBSA-CO and other methods on eight instances

**Table 15** The results of normality tests of the LBSA-CO, VTPSO, ABCSS and DSMO methods on the values "BestSol" in Table 10

| Method | Kolmogorov–Smirnov | | | Shapiro–Wilk | | |
|---|---|---|---|---|---|---|
| | Statistic | df | Sig | Statistic | df | Sig |
| VTPSO | 0.166 | 32 | 0.025 | 0.855 | 32 | < 0.001 |
| ABCSS | 0.197 | 32 | 0.003 | 0.788 | 32 | < 0.001 |
| DSMO | 0.166 | 32 | 0.024 | 0.856 | 32 | < 0.001 |
| LBSA-CO | 0.167 | 32 | 0.023 | 0.852 | 32 | < 0.001 |

**Table 16** The results of the Friedman test on the values "BestSol" in Table 10

| N | 32 |
|---|---|
| Chi-Square | 64.612 |
| df | 3 |
| Asymp. Sig | < 0.001 |

**Table 18** The results of normality tests of the LBSA-CO, SA and LBSA methods on the values "Best" in Table 11

| Method | Kolmogorov–Smirnov | | | Shapiro–Wilk | | |
|---|---|---|---|---|---|---|
| | Statistic | df | Sig | Statistic | df | Sig |
| SA | 0.188 | 65 | < 0.001 | 0.815 | 65 | < 0.001 |
| LBSA | 0.202 | 65 | < 0.001 | 0.832 | 65 | < 0.001 |
| LBSA-CO | 0.325 | 65 | < 0.001 | 0.615 | 65 | < 0.001 |

**Table 19** The results of the Friedman test on the values "Best" in Table 11

| N | 65 |
|---|---|
| Chi-Square | 104.256 |
| df | 2 |
| Asymp. Sig | < 0.001 |

labeled "Best" in Table 12. Since this table contains the results of only two methods, the Wilcoxon signed-rank test was applied directly without the Friedman test. The results of the pairwise comparison are presented in Table 22. The p-value indicated that there were statistically significant differences between the LBSA-CO and IBA.

Table 23 provides the results of Kolmogorov–Smirnov and Shapiro–Wilk normality tests for the LBSA-CO and GSAACS. These results were calculated based on the column labeled "Best" in Table 13. Since this table contains the results of only two methods, the Wilcoxon signed-rank test was conducted directly without the Friedman test. The results of the pairwise comparisons are presented in Table 24. The p-value in this table is greater than 0.5, unlike those found in the previous tests. This indicates that there are no statistically significant differences between the LBSA-CO and GSAACS.

## 5 Conclusion

In this study, a new simulated annealing algorithm with crossover operator which is called LBSA-CO was proposed. The LBSA-CO is a population-based metaheuristic method. In this method, a list-based temperature cooling schedule, which can adapt to the topology of the solution space of the problem, was used. The solutions in the population were improved with the inversion, insertion and

**Table 17** The results of the Wilcoxon signed-rank test of the LBSA-CO with respect to the VTPSO, ABCSS and DSMO on the values "BestSol" in Table 10

| Comparison | | N | Mean Rank | Sum of Ranks | Z | *p* value |
|---|---|---|---|---|---|---|
| LBSA-CO—VTPSO | Negative Ranks | 32 | 16.50 | 528.00 | − 4.937 | < 0.001 |
| | Positive Ranks | 0 | 0.00 | 0.00 | | |
| | Ties | 0 | | | | |
| LBSA-CO—ABCSS | Negative Ranks | 31 | 17.00 | 527.00 | − 4.918 | < 0.001 |
| | Positive Ranks | 1 | 1.00 | 1.00 | | |
| | Ties | 0 | | | | |
| LBSA-CO—DSMO | Negative Ranks | 30 | 15.50 | 465.00 | − 4.782 | < 0.001 |
| | Positive Ranks | 0 | 0.00 | 0.00 | | |
| | Ties | 2 | | | | |

**Table 20** The results of the Wilcoxon signed-rank test of the LBSA-CO with respect to the SA and LBSA on the values "Best" in Table 11

| Comparison | | N | Mean Rank | Sum of Ranks | Z | *p* value |
|---|---|---|---|---|---|---|
| LBSA-CO—SA | Negative Ranks | 57 | 29 | 1653.00 | − 6.567 | < 0.001 |
| | Positive Ranks | 0 | 0.00 | 0.00 | | |
| | Ties | 8 | | | | |
| LBSA-CO—LBSA | Negative Ranks | 56 | 28.50 | 1596.00 | − 6.509 | < 0.001 |
| | Positive Ranks | 0 | 0.00 | 0.00 | | |
| | Ties | 9 | | | | |

**Table 21** The results of normality tests for the LBSA-CO and IBA methods on the values "Best" in Table 12

| Method | Kolmogorov–Smirnov | | | Shapiro–Wilk | | |
|---|---|---|---|---|---|---|
| | Statistic | df | Sig | Statistic | df | Sig |
| IBA | 0.229 | 37 | < 0.001 | 0.797 | 37 | < 0.001 |
| LBSA-CO | 0.480 | 37 | < 0.001 | 0.240 | 37 | < 0.001 |

**Table 22** The results of the Wilcoxon signed-rank test for the LBSA-CO with respect to the IBA on the values "Best" in Table 12

| Comparison | | N | Mean Rank | Sum of Ranks | Z | *p* value |
|---|---|---|---|---|---|---|
| LBSA-CO—IBA | Negative Ranks | 26 | 13.50 | 351.00 | − 4.458 | < 0.001 |
| | Positive Ranks | 0 | 0.00 | 0.00 | | |
| | Ties | 11 | | | | |

**Table 23** The results of the normality tests for the LBSA-CO and GSAACS methods on the values "Best" in Table 13

| Method | Kolmogorov–Smirnov | | | Shapiro–Wilk | | |
|---|---|---|---|---|---|---|
| | Statistic | df | Sig | Statistic | df | Sig |
| GSAACS | 0.316 | 21 | < 0.001 | 0.578 | 21 | < 0.001 |
| LBSA-CO | 0.452 | 21 | < 0.001 | 0.398 | 21 | < 0.001 |

**Table 24** The results of the Wilcoxon signed-rank test for the LBSA-CO with respect to the GSAACS on the values "Best" in Table 13

| Comparison | | N | Mean Rank | Sum of Ranks | Z | *p* value |
|---|---|---|---|---|---|---|
| LBSA-CO—GSAACS | Negative Ranks | 6 | 5.67 | 34.00 | − 6.663 | 0.508 |
| | Positive Ranks | 4 | 5.25 | 21.00 | | |
| | Ties | 11 | | | | |

2-opt local search operators. The OX1 and ER operators were also applied to the improved solutions to accelerate the convergence. Furthermore, the Taguchi method was used to tune the parameters of the LBSA-CO. The proposed method was tested on 65 well-known TSP instances. It achieved BKS for 37 instances. It showed a higher performance compared to those of the other state-of-the-art methods on many instances.

## Declarations

**Conflict of interest** The authors declare that there is no conflict of interest.

## References


1. Gary MR, Johnson DS (1979). Computers and intractability: a guide to the theory of np completeness WH Freeman and Co. New York
2. Johnson DS (1990) Local optimization and the traveling salesman problem. In: Paterson MS (ed) Automata, languages and programming. Springer-Verlag, Berlin/Heidelberg, pp 446–461. https://doi.org/10.1007/BFb0032050
3. Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. J Soc Ind Appl Math 10(1):196–210
4. Lawler EL, Wood DE (1966) Branch-and-bound methods: A survey. Oper Res 14(4):699–719
5. Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33(1):60–100
6. Miliotis P (1978) Using cutting planes to solve the symmetric travelling salesman problem. Math Program 15(1):177–188
7. Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. J Oper Res Soc Am 2(4):393–410
8. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21(2):498–516
9. Martin O, Otto SW, Felten EW (1991) Large-step markov chains for the traveling salesman problem. oregon graduate institute of science and technology, department of computer science and engineering
10. Helsgaun K (2006) An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic (Doctoral dissertation. Roskilde University, Department of Computer Science)
11. Helsgaun K (2009) General k-opt submoves for the Lin-Kernighan TSP heuristic. Math Program Comput 1(2–3):119–163
12. Dong C, Jäger G, Richter D, Molitor P (2009) Effective tour searching for TSP by contraction of pseudo backbone edges. In: Goldberg AV, Zhou Y (eds) Algorithmic Aspects in Information and Management: 5th International Conference, AAIM 2009, San Francisco, CA, USA, June 15-17, 2009. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 175–187. https://doi.org/10.1007/978-3-642-02158-9_16
13. Cook W, Seymour P (2003) Tour merging via branch-decomposition. INFORMS J Comput 15(3):233–248
14. Nagata Y, Kobayashi S (2013) A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. INFORMS J Comput 25(2):346–363
15. Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver press
16. Hussain A, Muhammad YS, Sajid MN (2018) An improved genetic algorithm crossover operator for traveling salesman problem. Turkish J Math Comput Sci 9:1–13
17. Saji Y, Riffi ME (2016) A novel discrete bat algorithm for solving the travelling salesman problem. Neural Comput Appl 27(7):1853–1866
18. Ezugwu AES, Adewumi AO (2017) Discrete symbiotic organisms search algorithm for travelling salesman problem. Expert Syst Appl 87:70–78
19. Akhand MA, Akter S, Rashid MA, Yaakob SB. (2015) Velocity tentative PSO: an optimal velocity implementation based particle swarm optimization to solve traveling salesman problem. IAENG Int J Comput Sci 42(3)
20. Zhan SH, Lin J, Zhang ZJ, Zhong YW (2016). List-based simulated annealing algorithm for traveling salesman problem. Computational intelligence and neuroscience
21. Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng Appl Artif Intell 48:59–71
22. Hatamlou A (2018) Solving travelling salesman problem using black hole algorithm. Soft Comput 22(24):8167–8175
23. Khan I, Maiti MK (2019) A swap sequence based artificial bee colony algorithm for traveling salesman problem. Swarm Evol Comput 44:428–438
24. Akhand MAH, Ayon SI, Shahriyar SA, Siddique N, Adeli H (2020) Discrete spider monkey optimization for travelling salesman problem. Appl Soft Comput 86:105887
25. Gündüz M, Kiran MS, Özceylan E (2015) A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. Turk J Electr Eng Comput Sci 23(1):103–117
26. Mahi M, Baykan ÖK, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Appl Soft Comput 30:484–490
27. Ezugwu AES, Adewumi AO, Frîncu ME (2017) Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. Expert Syst Appl 77:189–210
28. Wang C, Lin M, Zhong Y, Zhang H (2015) Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. Int J Comput Sci Math 6(4):336–353
29. Chen SM, Chien CY (2011) Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Syst Appl 38(12):14439–14450
30. Khanra A, Maiti MK, Maiti M (2015) Profit maximization of TSP through a hybrid algorithm. Comput Ind Eng 88:229–236
31. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
32. Askarzadeh A, dos Santos Coelho L, Klein CE, Mariani VC (2016). A population-based simulated annealing algorithm for global optimization. In 2016 IEEE international conference on systems, man, and cybernetics (SMC) (pp. 004626–004633). IEEE
33. Wang L, Cai R, Lin M, Zhong Y (2019) Enhanced List-Based Simulated Annealing Algorithm for Large-Scale Traveling Salesman Problem. IEEE Access 7:144366–144380
34. Ingber L (1996) Adaptive simulated annealing (ASA): lessons learned. Control Cybern 25(1):32–54
35. Jeong SJ, Kim KS, Lee YH (2009) The efficient search method of simulated annealing using fuzzy logic controller. Expert Syst Appl 36(3):7099–7103
36. Hime A, e Oliveira, Antonio Petraglia (2013) Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. Appl Soft Comput 13(11):4349–4357. https://doi.org/10.1016/j.asoc.2013.06.018
37. Zhong Y, Lin J, Wang L, Zhang H (2018) Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem. Swarm Evol Comput 42:77–88
38. Davis L (1985) Applying adaptive algorithms to epistatic domains. In Proceedings of the 9th International joint conference on artificial intelligence, 85, 162–164.
39. Whitley LD, Starkweather T, D'Ann Fuquay (1989). Scheduling problems and traveling salesmen: The genetic edge recombination operator. In Proceedings of the 3rd international conference on genetic algorithms, 89, 133–40

40. Lin S (1965) Computer solutions of the traveling salesman problem. Bell Syst Tech J 44(10):2245–2269
41. Reinelt G (1991) TSPLIB - A traveling salesman problem library. ORSA J Comput 3(4):376–384
42. Taguchi G (1986). Introduction to quality engineering: designing quality into products and processes (No. 658.562 T3)
43. Mozdgir A, Mahdavi I, Badeleh IS, Solimanpur M (2013) Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. Math Comput Model 57(1–2):137–151