Applied Soft Computing 105 (2021) 107275

Contents lists available at ScienceDirect

Applied Soft Computing

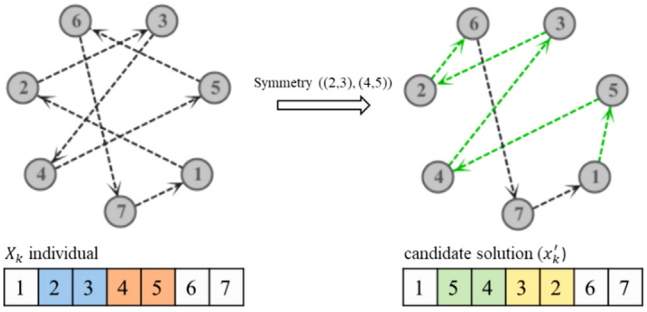
journal homepage: www.elsevier.com/locate/asoc

DJAYA: A discrete Jaya algorithm for solving traveling salesman problem

Mesut Gunduz [a], Murat Aslan [b,*]

[a] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Konya Technical University, 42075 Konya, Turkey
[b] Department of Computer Engineering, Faculty of Engineering, Şırnak University, 73000 Şırnak, Turkey

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 8 December 2020
Received in revised form 7 February 2021
Accepted 26 February 2021
Available online 9 March 2021

Keywords:
Discrete optimization
JAYA
Transformation operator
Traveling salesman problem

ABSTRACT

Jaya algorithm is a newly proposed stochastic population-based metaheuristic optimization algorithm to solve constrained and unconstrained continuous optimization problems. The main difference of this algorithm from the similar approaches, it uses best and worst solution in the population in order improve the intensification and diversification of the population, and this provides discovering potential solutions on the search space of the optimization problem. In this study, we propose discrete versions of the Jaya by using two major modifications in the algorithm. First is to generate initial solutions by using random permutations and nearest neighborhood approach to create population. Second is the update rule of the basic Jaya algorithm rearranged to solve discrete optimization problems. Due to characteristics of the discrete optimization problem, eight transformation operators are used for the discrete variants of the proposed algorithm. Based on these modifications, the discrete Jaya algorithm, called DJAYA, has been applied to solve fourteen different symmetric traveling salesman problem, which is one of the famous discrete problems in the discrete optimization. In order to improve the obtained best solution from DJAYA, 2-opt heuristic is also applied to the best solution of DJAYA. Once population size, search tendency and the other parameters of the proposed algorithm have been analyzed, it has been compared with the state-of-art algorithms and their variants, such as Simulated Annealing (SA), Tree-Seed Algorithm (TSA), State Transition Algorithm (STA) Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Genetic Algorithm (GA) and Black Hole (BH). The experimental results and comparisons show that the proposed DJAYA is highly competitive and robust optimizer for the problem dealt with the study.

1. Introduction

Optimization problems can be element of single objective or multi-objective models [1]. In addition, it is hard to solve many seemingly simple optimization problems in a reasonable time because if the dimensionality of the problem increases, the number of potential solutions exponentially increases and seeking for every possible combination is so complicated for large dimensional problems [2]. But, in the light of heuristic algorithms, it is possible to reach optimal or near optimal solution in a reasonable time for many optimization problems [1].

* Corresponding author.
E-mail address: murataslan@sirnak.edu.tr (M. Aslan).

https://doi.org/10.1016/j.asoc.2021.107275
1568-4946/

From another perspective, optimization problems can be divided into continuous and discrete optimization groups [3]. While the decision variables in continuous optimization problems are real-valued, in discrete optimization, they can be element of a set. Discrete optimization is a popular research field for researchers as well as continuous optimization. Many real word optimization problems are studied in discrete optimization area such as traveling salesman problem (TSP) [4–7], manufacturing cell formation problem [8], graph coloring problem [9,10], water pump switching problem [11] and so on. TSP is one of the famous problem in combinatorial discrete optimization and it is classified as Np-hard optimization problem [2,12] because the number of solutions is exponentially increased by depending on the decision variables and we have no polynomial time algorithm to solve it yet. In addition, there are several important applications in TSP area, such as computer wiring, vehicle routing, clustering a data array, job-shop scheduling [13], drilling of printed circuit boards, X-ray crystallography [14,15] and so on. TSP can be described as follow: a salesperson in order to visit n cities with minimum total cost should find shortest Hamilton cycle through which the person can visit all the cities just once, and at last return to start point [16]. Because of TSP is one of the Np-hard problem, it cannot be solved optimally by any known method in polynomial time, and there are many approaches proposed in literature for working-out on TSP. There are some deterministic and approximate algorithms in literature such as dynamic programming [17,18], branch-and-bound [19], branch and cut [20,21], branch and price [22], cutting planes [23,24], and Lagrangian dual [25]. These exact algorithms generally find acceptable solutions for small and medium-size problems. But, by depending on the increasing the dimensionality of the problem, the performance of these algorithms scales down [2,4]. Therefore, for solving the problems classified in Np-hard category, linear and dynamic programming based algorithms do not get optimal solution in generally [26]. Conversely, metaheuristic algorithms achieve effective solution in reasonable time, and they are problem-free, easy adaptable and have a simple structure [2,27]. There were a lot of heuristic methods proposed in literature for solving TSP and some of them are such as cuckoo search (CS) [2], tabu search (TS) [28], simulated annealing (SA) [29,30], genetic algorithm (GA) [31–38], ant colony optimization (ACO) [5,6,39,40], artificial bee colony (ABC) [6,41–43], particle swarm optimization (PSO) [5,7,44], black hole (BH) [45], state transition algorithm (STA) [4,46] and tree seed algorithm (TSA) [47] etc.

The position update rule of continuous optimization algorithms can be modified with path improvement methods for discretization [6]. Jaya algorithm is a newly proposed stochastic population-based metaheuristic optimization algorithm to solve constrained and unconstrained continuous optimization problems [48]. The peculiar feature of Jaya is to using the current, best and worst individuals for producing the candidate solution, and this provides a wide search area in search space [27]. Therefore, our proposed discrete Jaya algorithm (for short DJAYA) also uses current, best and worst individuals for position update rule. Discrete tree seed algorithm (DTSA) [47], discrete state transition algorithm (DSTA) [4,46] and state transition algorithm (STA) [4,46] were used transformation operators (swap, shift and symmetry) for create candidate solutions. In the light of these algorithms, these transformation operators are used in DJAYA for produce the candidate solutions. In our algorithms, the initial population is generated with a random permutation, except the first individual is created by nearest neighbor tour like DTSA [47]. Eight different combination of transformation operator's; swap, shift, symmetry, *swap+shift* with equal selection, *swap+symmetry* with equal selection, *shift+symmetry* with equal selection, combined 1 (swap, shift and symmetry with equal selection), combined 2 (swap, shift and symmetry according to roulette wheel selection), respectively are used in update mechanism of the proposed algorithm. After the termination condition is satisfied, 2-opt [49] local search method has been implemented on the best individual to enhanced the solution quality. The main superiorities of proposed algorithm from the other algorithms in the literature, and the main contributions and motivations of this study are given as follows:

- DJAYA is a novel and alternative discrete optimization approach for solving permutation-coded based discrete problems.
- There is no permutation-coded discrete version of Jaya in the literature, so in this study a discrete form of Jaya called as DJAYA presented for solving TSPs.
- The DJAYA has been applied to solve 14 different symmetric traveling salesman problems taken from TSPLIB [50].
- Before the setting the peculiar parameters of the proposed algorithm for comparing the algorithms, the peculiar parameters of the DJAYA such as population size and search tendency parameters have been analyzed.
- Rank mechanism is applied to the experimental results of proposed algorithm and the other compared algorithms in order to a fair comparison.
- According to mean, standard deviation, relative error and rank criteria, the obtained results on symmetric TSPs demonstrate that the proposed algorithm is an alternative and competitive optimizer.

The remainder of the paper is organized as follows: In Section 2, a literature review of discrete optimization problem is given. In Section 3, the basic Jaya algorithm and the proposed algorithm are given. In Section 4, the proposed algorithm is tested on TSP benchmark dataset and the experimental results of DJAYA compared with state-of-art population-based algorithms. In Section 5, the result and discussion are given and the conclusions and future works are given in Section 6.

2. Literature review

Many methods proposed in literature such as exact, approximate, local search and metaheuristics algorithms for solving TSP. Ergun and Orlin [17], proposed a dynamic programming methodology in very large scale neighborhood search for TSP. In another study, Chentsov and Korotayeva [18], developed a procedure of the dynamic programming (DP) for the discrete–continuous problem of a route optimization. Radharamanan and Choi [19], were presented a branch and bound method with penalty tour structure is developed for solving TSP and transportation routing problems. Padberg and Rinaldi [20], proposed a method based on branch-and-cut to solve optimization of a 532-city symmetric traveling salesman problem. In another study, Hernández-Pérez and Salazar-González [21] proposed a branch-and-cut algorithm for a routing problem called one-commodity pickup and delivery traveling salesman problem. Barnhart et al. [22], proposed a branch and price method for column generation to solve huge integer programs. Fleischmann [23], proposed a cutting plane procedure for the TSP on road networks. Irnich et al. [51], proposed a sequential search algorithm that allows neighborhoods within local-search algorithms such as Or-opt, 2-opt and 3-opt for capacitated vehicle-routing problem (CVRP). Knox [28], presented a tabu search (TS) based approach for solving symmetric TSP and compared TS with K-OPT procedure. Geng et al. [30], proposed an effective local search algorithm based on simulated annealing and greedy search techniques for solving the TSP and in order to reach more effective solutions, SA algorithm was used with combination of three kinds of mutations and greedy search method. Genetic algorithm [52] was easy adopted to the

continuous and discrete optimization problems. So, GA update operators such as crossover and mutation operator are used in many heuristic update mechanism. In another study, Grefenstette et al. [31] was proposed a method based on genetic algorithm to solve TSP. Ulder et al. [33] proposed genetic local search for TSP. They presented two approaches based on GA with 2-opt and in-Kernighan neighborhoods [53]. In another study, Potvin [34] presented a survey of genetic algorithms with randomized search techniques such as different crossover and mutation techniques. Üçoluk [36] proposed a genetic algorithm with avoiding special crossover and mutation for TSP. Ahmed [37], proposed GA based approach for solving TSP using sequential constructive crossover operator (SCX). SCX was compared with some existence crossover operators such as, edge recombination crossover (ERX) and generalized N-point crossover (GNX). According to Experimental results of [37], SCX based GA was better than the ERX and GNX. In another study, Hussain et al. [38], proposed a method based on GA to solve TSP with modified cycle crossover operator (CX2). Their approach has been linked with path representation, which is the most natural way to represent a legal tour.

Wang et al. [44], proposed a new application of PSO for TSP. They developed some special methods with swap operator and swap sequence, and redefined some operators. Pang et al. [54] proposed a modified discrete PSO algorithm with integrated fuzzy matrices to represent the position and velocity of the particles to solve TSP. In another study, Shi et al. [7] proposed a novel particle swarm optimization (PSO)-based algorithm for TSP. They used an uncertain searching strategy and a crossover eliminated technique for accelerate the convergence speed. Chen and Chien [29] presented a hybrid approach based on GA, SA and ACO with PSO techniques for solving TSP. They implemented their study on 25 instances obtained from the TSPLIB and compared with some evolutionary optimization algorithms. In another study, Mahi et al. [5] proposed a new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt local search algorithms for solving TSP. In their study, they implemented PSO for detecting optimum values of parameters α and β which were used for city selection operations in the ACO algorithm and determines significance of inter-city pheromone and distances. After termination condition was satisfied 3-opt [55] algorithm performed for the reducing the tour lenght. Dorigo and Gambardella [40], proposed an ant colony system (ACS) and ACS-3-opt for solution symmetric and asymmetric TSP. Gülcü et al. [39] presented a parallel cooperative hybrid algorithm (PACO-3opt) including with ACO and 3-opt for solving TSP. Firstly PACO was implemented on TSP and after a predefined number of iterations 3-opt is applied on each colonies to improve the solutions and then shares the best tour with other colonies. According to their study, PACO-3opt showed more efficient and reliable performance. In another study, Gündüz et al. [6] proposed a hierarchic approach based ACO and ABC for solving TSP. In their study, the path construction-based method ACO was used for provide a better initial solution for the path improvement-based technique ABC. Li et al. [42], presented a new discrete artificial bee colony algorithm using the concept of swap operator to produce a better candidate tour by greedy selection. Karaboga and Gorkemli [43] proposed a new artificial bee colony algorithm called Combinatorial ABC (CABC) for solving TSP. In another study Görkemli and Karaboga [56] proposed an improved version of CABC called quick Artificial Bee Colony (qABC) which the onlooker bees behavior is modeled in more detailed way. Kıran et al. [41] proposed the analysis of discrete artificial bee colony algorithm with neighborhood operator to solve TSP. According to their study, update process carried out by random swap (RS), random insertion (RI), random swap of subsequences (RSS), random insertion of subsequence (RIS), random reversing of subsequence (RRS), random reversing swap of subsequences (RRSS) and random reversing insertion of subsequence (RRIS) neighborhood operators. In order to increase quality of the solutions 2-OPT and 3-OPT local approaches integrated with their proposed ABC. Hatamlou [45], presented a method based on black hole algorithm (BH) for solving TSP. Chunhua et al. [46] proposed state transition algorithm (STA) based method to solve TSP. In another study, Zhou et al. [4] proposed a discrete state transition algorithm for unconstrained integer optimization problems and transformation operators were used (swap, shift and symmetry) for producing candidate solutions. Ouaarab et al. [2], proposed discrete version of the Cuckoo Search (CS) algorithm to solve TSP and they improved local search to overcome weakness of CS. In another study, Cinar et al. [47] proposed discrete tree seed algorithm (DTSA) which redesigned the basic TSA by integrating the swap, shift, and symmetry transformation operators in order to solve the permutation-coded optimization problems. In another study, Baş and Ülker [57] proposed a discrete social spider algorithm called as DSSA by adding new explorer spiders and novice spiders. And also they incorporated the 2-opt local search algorithm into the DSSA. The performance of the DSSA was investigated on traveling salesman benchmark problems. According to the experimental results of their study, DSSA can be used as an alternative discrete algorithm for discrete optimization tasks. Akhand et al. [58] proposed a discrete effective variant of sider monkey optimization (SMO) algorithm to solve TSP called as (DSMO). In DSMO, every spider monkey represented a TSP solution where Swap Sequence (SS) and Swap Operator (SO) based operations are employed, which enabled interaction among monkeys in obtaining the optimal TSP solution. The SOs were created using the experience of a specific spider monkey as well as the experience of other members (local leader, global leader, or a randomly selected spider monkey) of the group. For demonstrated the performance and effectiveness of the DSMO, they implemented DSMO on a large set of TSP instances. Huang et al. [59] proposed a discrete shuffled frog leaping algorithm for solving TSP based on nearest neighbor heuristic information, presenting the opposite roulette strategy to select the individuals participating in evolution and using the mutation operator for local search.

There are a lot of different Jaya based algorithms proposed in literature to solve single and multi-objective real world optimization problems [60]. Aslan et al. [27] proposed a Jaya based binary optimization algorithm with 'XOR' Operator (JayaX) for binary optimization. In another study Rao et al. [61], proposed a Jaya based approach for surface grinding process optimization problems. Prakash et al. [62] proposed a binary Jaya algorithm based optimal placement of phasor measurement units for power system observability. Rao and Waghmare [63] presented a Jaya based algorithm for solving complex constrained design optimization problems. In another study, Rao and More [64] proposed a self-adaptive Jaya algorithm for solving design optimization and analysis of selected thermal devices. Wang and Huang [65] proposed a novel elite opposition-based Jaya algorithm (EO-Jaya) for parameter estimation of photovoltaic cell models. Rao et al. [66] developed a new multi-objective optimization algorithm named as MO-Jaya algorithm for modern machining processes. Wang et al. [67] proposed a GPU-accelerated parallel Jaya algorithm for efficiently estimating Li-ion battery model parameters. More and Rao [68] presented a modified Jaya algorithm for design optimization of plate-fin heat exchanger. Dede et al. [69] proposed a Jaya based algorithm for optimization of braced dome structures. In another study, Kumawat et al. [70] presented a Jaya Based approach for optimal allocation of distributed energy resources.

As seen from the literature review on TSP and JAYA, the present study fills a gap in the intersection of JAYA and TSP.

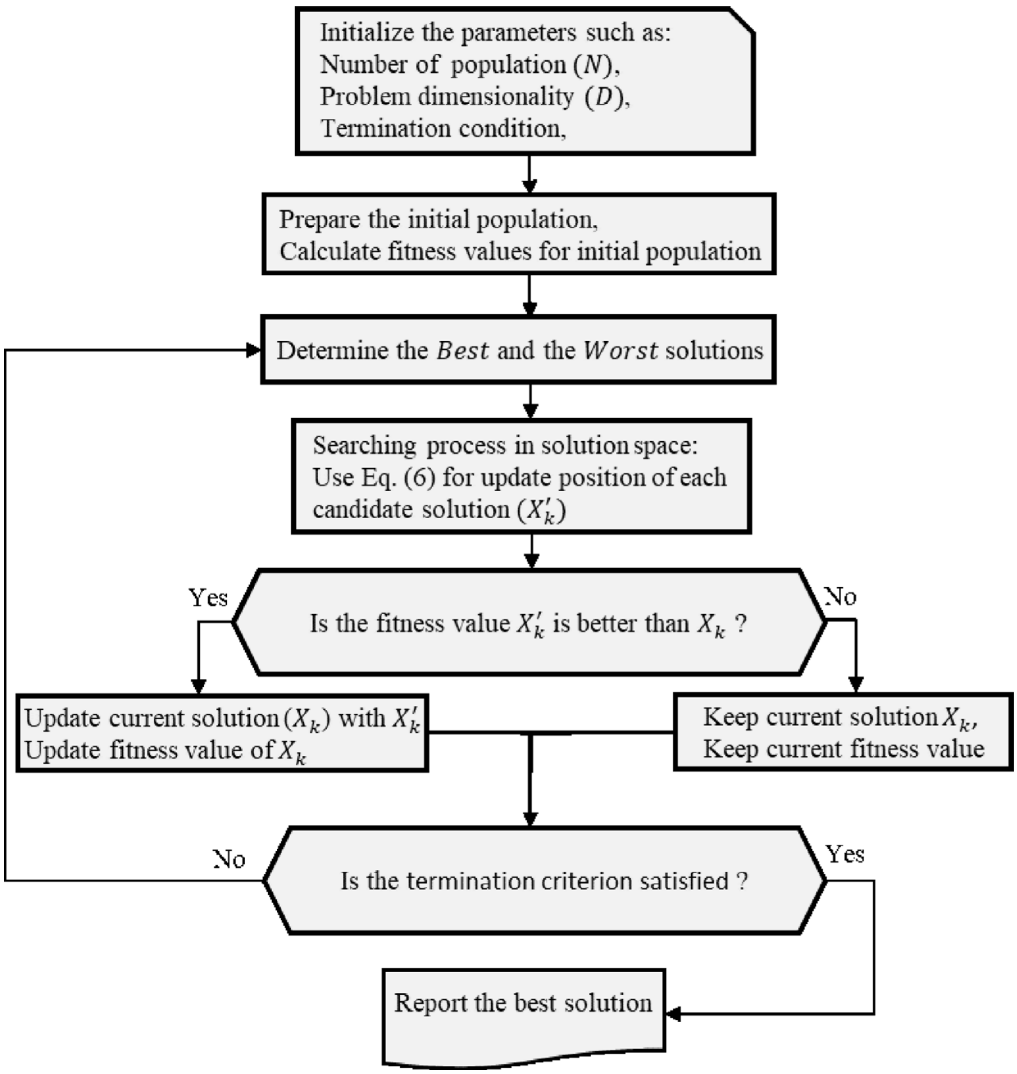


Fig. 1. The flowchart of Jaya algorithm [48].

3. Material and methods

3.1. Traveling salesman problem

TSP can be described as follow: a salesperson in order to visit n cities with minimum total cost should find shortest Hamilton cycle through which the person can visit all the cities just once, and at last return to start point [16]. Traveling salesman problem can be demonstrated with a weighted graph $G = (V, E)$, where V is the set of vertices (cities) and E is the set of edges (curves) fully connecting all cities. Each edge $(t, z) \in E$ and $t \neq z$ is assigned a cost d_{tz}, which is the distance between cities t and z. The objective function of TSP is detailed in the following equations [71,72].

$$Minimize: \sum_{t \neq z}^{D} d_{tz} b_{tz} \tag{1}$$

$$\sum_{z=1}^{D} b_{tz} = 1 \qquad t = 1, 2, \ldots, D \tag{2}$$

$$\sum_{t=1}^{D} b_{tz} = 1 \qquad z = 1, 2, \ldots, D \tag{3}$$

$$\sum_{t,z \in S}^{D} b_{tz} \leq |S| - 1 \qquad S \subset V, 2 \leq S \leq D - 2 \tag{4}$$

$$b_{tz} \in \{0, 1\}, t, z = 1, 2, \ldots, D, t \neq z \tag{5}$$

Where D refers to total number of cities that need to be visited. The b_{tz} matrix holds the information on whether the traveling salesman has traveled to the t and z cities. The d_{tz} matrix is the distance matrix that shows the cost between t and z cities to visit. With the restrictions given in Eqs. (2) and (3), the traveling salesman person is only allowed to visit each city once. The restrictions given in Eq. (4) prevents the formation of sub-tours, and thus preventing the traveling salesman to visit a city in a second time. In Eq. (5), a binary constraint condition is given for cities to visit.

3.2. Basic Jaya algorithm

Jaya is a population based algorithm proposed by [48], and it is a simple and new optimization algorithm in order to solve constrained and unconstrained optimization problems. Jaya is easily adaptable because of not containing any algorithm specific parameters, and has an effective exploration in consequence of using best and worst individuals in solution update rule [27,62].

Jaya position update equation is given in Eq. (6).

$$X'_{k,j} = X_{k,j} + r_1 * \left(Best_j - \left|X_{k,j}\right|\right) - r_2 * \left(Worst_j - \left|X_{k,j}\right|\right) \quad (6)$$

Where N indicates the population size and k is an integer value in range of [1, N], X_k shows kth current individual and X'_k shows the new position of kth individual in the population, D indicates dimensionality of the problem and j is an integer value in range of [1, D], Best individual indicates the position with the best fitness value in the population, Worst individual indicates the position with the worst fitness value in the population, r_1 and r_2 are random continuous values in range of [0, 1]. When a candidate solution position is getting updated current, Best and Worst individuals are used. Hence, candidate solution tries to get closer to the Best position and avoid from the Worst position. This update process is the peculiar feature of Jaya. The flowchart of Jaya algorithm is given in Fig. 1.

3.3. A novel discrete Jaya algorithm (DJAYA)

Basic Jaya algorithm proposed for continuous optimization problems. Therefore, in order to solving discrete optimization problems, it needs some proper modification. Proposed discrete Jaya algorithm (DJAYA) uses current, best and worst individuals for position update rule like basic Jaya algorithm. Kiran [73] used search tendency parameter in basic tree-seed algorithm (TSA) to improve exploration capability of TSA method. In order to construct a balance between exploration and exploitation on the solution space of DJAYA two search tendency (ST) parameters called $ST1$ and $ST2$ are incorporated to Jaya. According to $ST1$ and $ST2$ parameter, a parent (X_k, best or worst individual) will be selected to produce a candidate solution. This selection is controlled by $ST1$ and $ST2$ in range of [0, 1]. A parameter analysis of $ST1$ and $ST2$ is being performed in Section 4 for determine the values of ST1 and ST2.

Discrete tree seed algorithm [47], state transition algorithm and discrete state transition algorithm were used swap, shift and symmetry transformation operators for create candidate solutions. we inspired from these algorithms and then integrated transformation operators into proposed algorithm. On the other hand, unlike these algorithms a roulette wheel selection [34] is enclosed into the proposed algorithm. The reason of the using roulette wheel selection is to give a chance both weak solutions and strong solutions to be chosen. Because sometimes weak solutions can lead us to reach better solutions and avoid from local optimum. In proposed algorithm swap, shift and symmetry transformation operators start with an equal selection probability and in later iterations the selection probability of transformation operator's changes according to their success.

The initial population is generated with a random permutation, except the first individual and the first individual is produced by nearest neighbor tour like DTSA [47]. Eight different combination of transformation operators; swap, shift, symmetry, *swap+shift* with equal selection, *swap+symmetry* with equal selection, *shift+symmetry* with equal selection, Combined 1 (swap, shift and symmetry with equal selection) and Combined 2 (swap, shift and symmetry according to roulette wheel selection), respectively are performed in update mechanism and a parameter analysis of transformation operators is being performed in Section 4 to make a decision for position update rule. After the termination condition is satisfied, 2-opt [49] local search method implemented on the best individual to enhanced the solution quality. Transformation operators used for producing candidate solution (X'_k) are described as follow:

Swap transformation: Two different integer numbers are randomly generated from 1 to problem dimensionality (D). D refers to number of the cities for TSP. After two different positions are determined these two position updated by swap operator. The illustration of swap transformation operator which carried out on an example with 7 cities is given in Figs. 2 and 2th and 5th dimension of problem is being swapped.

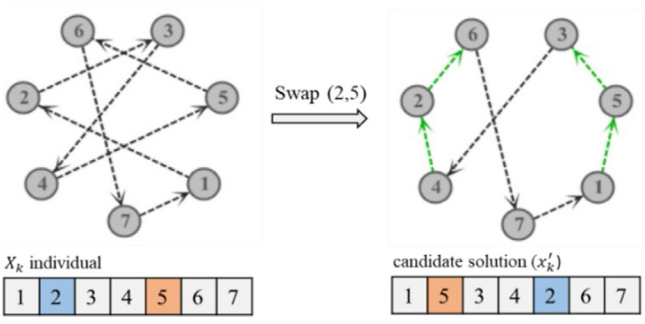


Fig. 2. The model of swap transformation for TSP.

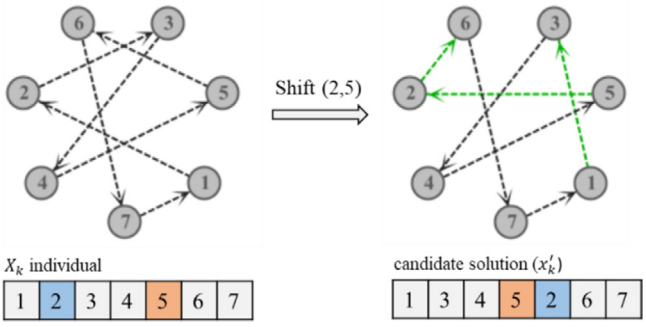


Fig. 3. The model of shift transformation for TSP.

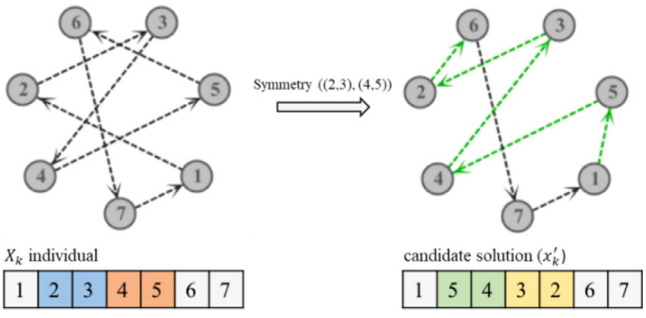


Fig. 4. The model of symmetry transformation for TSP.

Shift transformation: Two different integer numbers x and y are randomly generated from 1 to D. Then, xth position is stored and other positions are shifted to the left in the range of $[x, y]$. After all, the xth position is assigned to the position of y [47]. The illustration of shift transformation operator which carried out on an example with 7 cities is given in Fig. 3.

Symmetry transformation: Two sequences with consecutive positions to be symmetrized are both created randomly. Each component of sequences is inversed in their sequence and then the sequences positions swapped [47]. The symmetry transformation operator performs big changes on the current position (X_k). The illustration of symmetry transformation operator which carried out on an example with 7 cities is given in Fig. 4.

After these explanations the flowchart of discrete Jaya algorithm is given in Fig. 5.

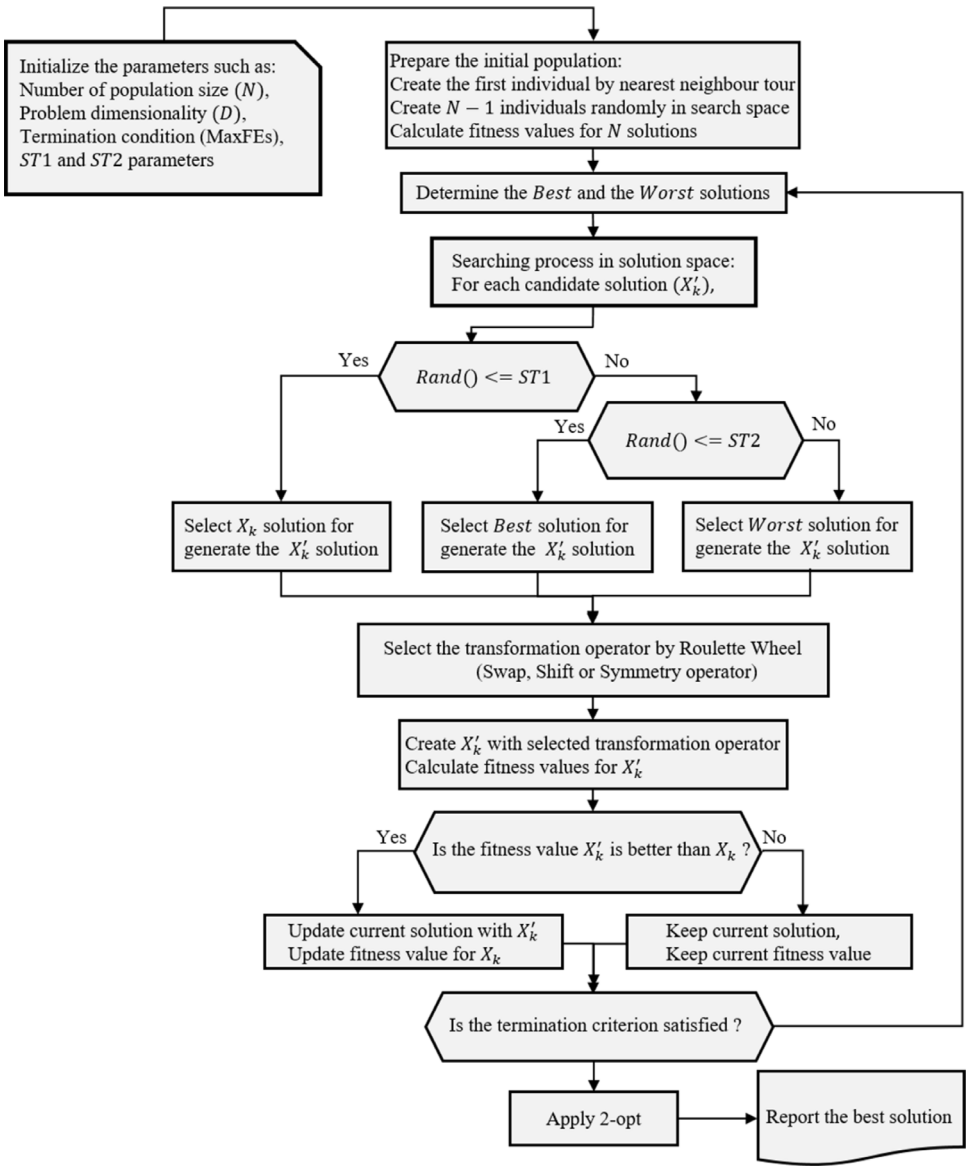


Fig. 5. The flowchart of discrete Jaya algorithm (DJAYA).

4. Experimental results

The proposed discrete Jaya algorithm is performed on some benchmark instances of symmetric TSP taken from TSPLIB library [50]. The optimum route of a lot of TSPs in TSPLIB are known. To indicate potency of proposed algorithm, DJAYA tested on low, middle, and large dimensional 14 different symmetric TSPs. Test problems used in the experiments are given in Table 1. These instances labeled with the number of cities; for example, *Berlin52* TSP means that a problem with 52 cities.

Euclidean distances of these TSP instances are used for experiments and comparisons. All the methods in the comparison are run 20 times and experimental results are presented as mean, standard deviation (Std. Dev.) and relative error (RE). RE value is mean the difference between the optimal cost and the mean cost of 20 runs found by the DJAYA which is calculated by Eq. (7).

$$RE = \frac{f(mean) - f(opt)}{f(opt)} \times 100 \tag{7}$$

where $f(opt)$ indicates to the optimum route cost and $f(mean)$ indicates to the average cost of 20 runs solutions found by the proposed algorithm. All experiments are executed on a Windows 10 Pro OS laptop using Intel(R) Core(TM) i7-6700HQ 2.60 GHz CPU, 24 GB of RAM and the codes were implemented in MATLAB. A parameter analysis of DJAYA is tested on TSP225 benchmark

Table 1
Test problems used in the experiments.

Problem	Dimension size	Tour length
OLIVER30	30	423.74
EIL51	51	428.87
BERLIN52	52	7542 (7544.37)
ST70	70	677.11
EIL76	76	545.38
PR76	76	108159.44
KROA100	100	21282 (21285.44)
KROB100	100	22141
KROC100	100	20749
KROD100	100	21294
KROE100	100	22068
EIL101	101	642.31
CH150	150	6532.10
TSP225	225	3859

Table 2
A performance analysis of transformation operators on TSP225 problem.

Method	DJAYA Mean	DJAYA+2-opt Mean	Std. Dev.	RE (%)
Swap	4416.20	4253.34	29.24	10.22
Shift	4313.97	4127.88	36.87	6.97
Symmetry	4165.15	4048.98	38.58	4.92
Swap+shift	4287.21	4119.37	33.52	6.75
Swap+symmetry	4065.91	4045.20	33.44	4.83
Sift+symmetry	4024.10	4002.56	34.42	3.72
Combined 1	4026.68	4014.73	28.31	4.04
Combined 2	**4009.96**	**3995.88**	42.91	**3.55**

instance, because of the TSP225 instance is a problem within 225 cities, this instance is selected for parameter analysis. The parameter analysis is given in Section 4.1.

4.1. Parameter analysis of DJAYA on TSP225

The control parameters of the DJAYA has been analyzed, before comparing DJAYA with the other algorithms. For this analysis, the number of population size (N) is set from 10 to 100, eight different combination of transformation operator's; swap, shift, symmetry, *swap+shift*, *swap+symmetry*, *shift+symmetry*, Combined 1 and Combined 2 are performed in update mechanism, $ST1$ and $ST2$ are set from 0.1 to 0.9. The termination condition is maximum number of function evaluations (MaxFEs) and it is set to 8×10^5 for parameter analysis. The first analysis is performed on transformation operators. Table 2 is demonstrated a performance analysis of transformation operators on TSP225 for DJAYA and DJAYA with 2-opt local search and the convergence graph of relative error for 8 different cases is given in Fig. 6.

Table 2 shows the performance of eight different transformation operators on TSP225. When Table 2 examined, it can be seen that without 2-opt local search the performance of swap operator is the worst one. And also it is understood from Fig. 6, the convergence of swap operators to mean results is quite slowly. In addition, when shift, symmetry and *swap+shift* operators mean results are considered without the 2-opt, the performances of these operators are not good enough like swap operator. The convergences of these operators to optimum result are slowly and it is seen from Fig. 6, after an iteration the best results found by the shift, symmetry and *swap+shift* operators are remained stationary. However, when Table 2 and Fig. 6 are examined, *swap+symmetry*, *shift+symmetry*, Combined 1 and Combined 2 operators performance are better than swap, shift, symmetry and *swap+shift* operators. And, it is realized that convergence of Combined 2 operator which uses roulette wheel selection to optimum result is the best among of eight transformation operators. In light of this information, Combined 2 operators is selected for the other parameter analysis and experiments.

The route results for eight different cases just for one attempt on TSP225 before 2-opt and after 2-opt are given in Figs. 7a–7b. The edges drawn with the red line means that after applied 2-opt these red line areas are being updated. When Fig. 7a and Fig. 7b are analyzed, it is seen that if the combined 2 is used for TSP225, just a few changes made by 2-opt. Because before 2-opt is applied, DJAYA found the route length 3988.43, and after 2-opt 3986.10 is found. The difference between them is just 2.33. But for the other cases more changes made by 2-opt. According to Figs. 7a and 7b, if swap or shift operators used for experiments, the results are very poor. By using swap before 2-opt tour length shortened nearly 200. Therefore, it is seen from Figs. 7a and 7b, 2-opt provides a better route for swap, shift, symmetry, *swap+shift* cases. However, without 2-opt the results of *swap+symmetry*, *shift+symmetry*, combined 1 and combined 2 are better than the other operators.

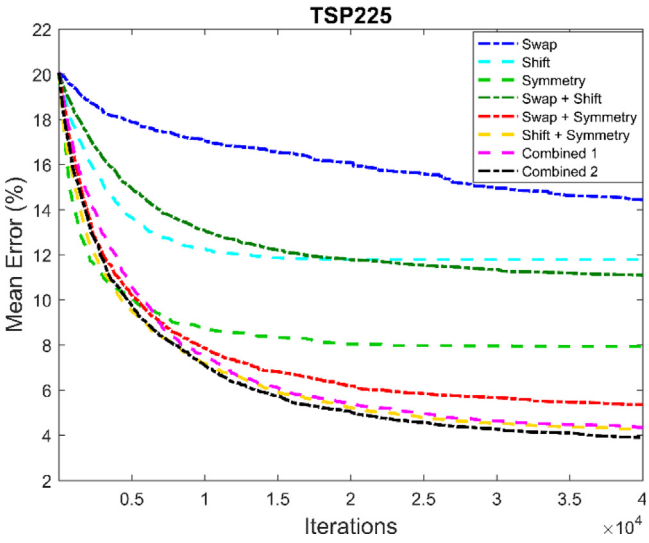


Fig. 6. Convergence graph of mean error for transformation operators.

Table 3
Parameter analysis of number of population size on TSP225.

N	Mean	Std. Dev.	RE (%)
10	4001.25	44.61	3.69
20	**3995.88**	42.91	**3.55**
30	4012.55	42.04	3.98
40	4014.62	42.47	4.03
50	4011.95	27.08	3.96
60	4028.86	38.37	4.40
70	4005.20	34.23	3.79
80	4027.70	36.79	4.37
90	4017.53	40.41	4.11
100	4019.96	30.16	4.17

The second analysis is performed for determine the number of population size (N). For this purpose, N is set from 10 to 100, $ST1 = 0.5$, $ST2 = 0.5$ and combined 2 is used for position update rule. The result for analyzed of N is given Table 3.

According to Table 3, when N is selected 20, the results are slightly better. So for analyses of $ST1$ and $ST2$ parameters process N is chosen 20. When combined 2 is selected for update process, the selection rate of transformation operator used by proposed algorithm according to roulette wheel selection is given in Fig. 8 for each number of population size from 10 to 100.

It is realized from Fig. 8 symmetry operator is selected more than from the other transformation operator according to the selection of roulette wheel. it is understood that when symmetry operator is selected to produce a candidate solution, it is generally produced a better position to previous one, so the selection probability of symmetry operator is higher than swap and shift operators. When N is selected as 20, selection probability of shift and symmetry is near to each other. But, according to the roulette wheel selection swap is not selected to much because of it is not produced a better position to previous one in general.

Another analysis is performed for determine $ST1$ parameter and the results are given in Table 4. For this purpose, $ST1$ is set from 0.1 to 0.9. The other parameters; number of population size N and $ST2$ are chosen 20 and 0.5, respectively. It is seen from Table 4, for $ST1 = 0.5$, the results are better than the other values and it makes a balance between exploration and exploitation.

Fig. 9 demonstrates that When $ST1 = 0.5$ the selection rate of shift and symmetry is almost same and it is understood that shift and symmetry produced good positions for candidate solution, but according to the roulette wheel selection swap is not selected to much for create a new position for candidate solution.

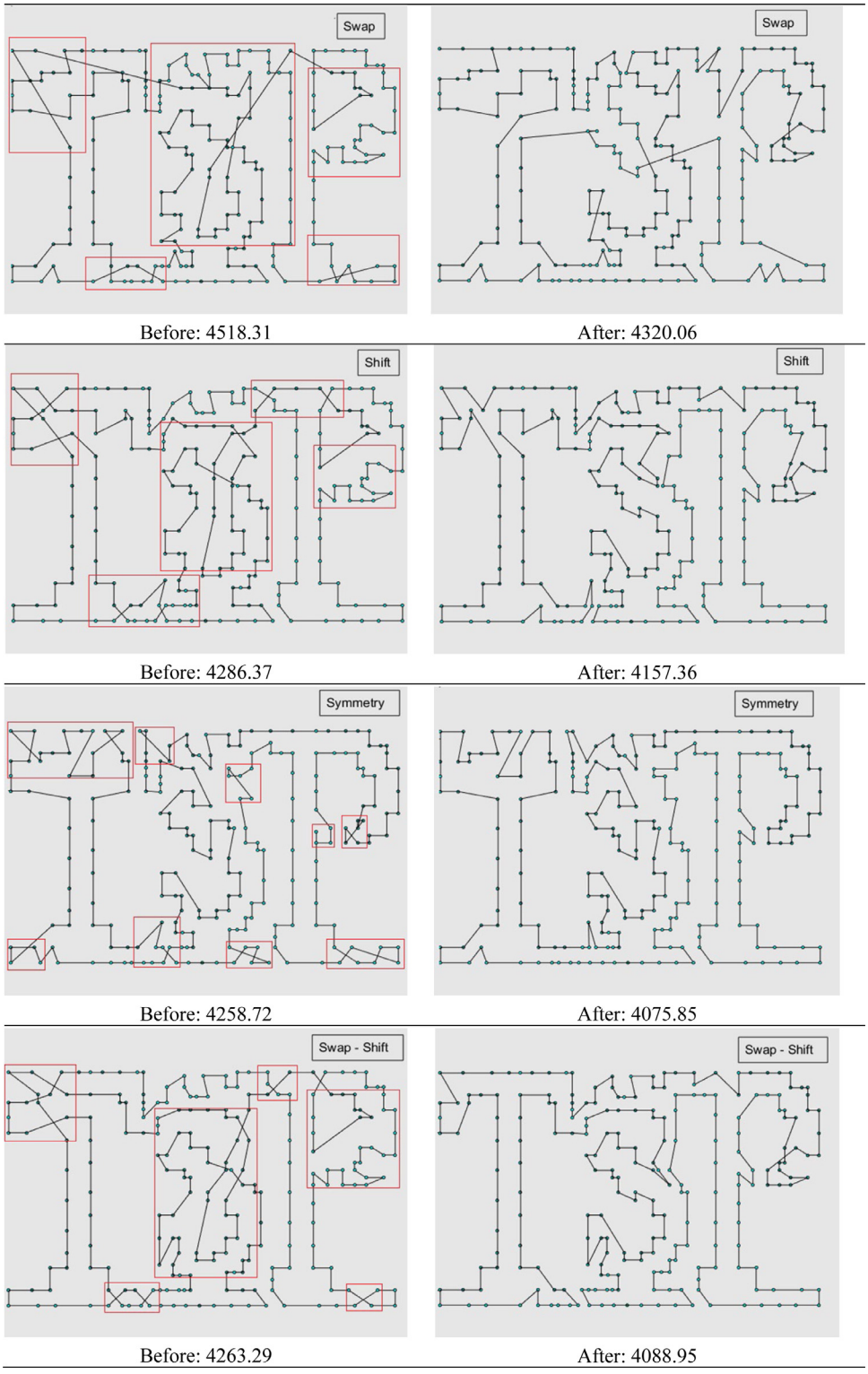


Fig. 7a. The route results of swap, shift, symmetry and *swap+shift* operators on TSP225.

The last analysis is performed for $ST2$ parameter and the results are given in Table 5. For this purpose, $ST2$ is set from 0.1 to 0.9. The other parameters; number of population size N and $ST1$ are chosen 20 and 0.5, respectively. It is seen from Table 5, for $ST2 = 0.5$, the results are better than the other $ST2$ values. When Fig. 10 is examined, it is seen that when $ST2$ is selected as 0.5, the selection rate of shift and symmetry is near to each other but in general, for position update process swap is not used too much.

4.2. Comparisons with SA, ACO, STA and DTSA

The first comparison is performed with Chunhua et al. [46] and Cinar et al. [47] studies because of they were also used transformation operators for producing candidate solutions.

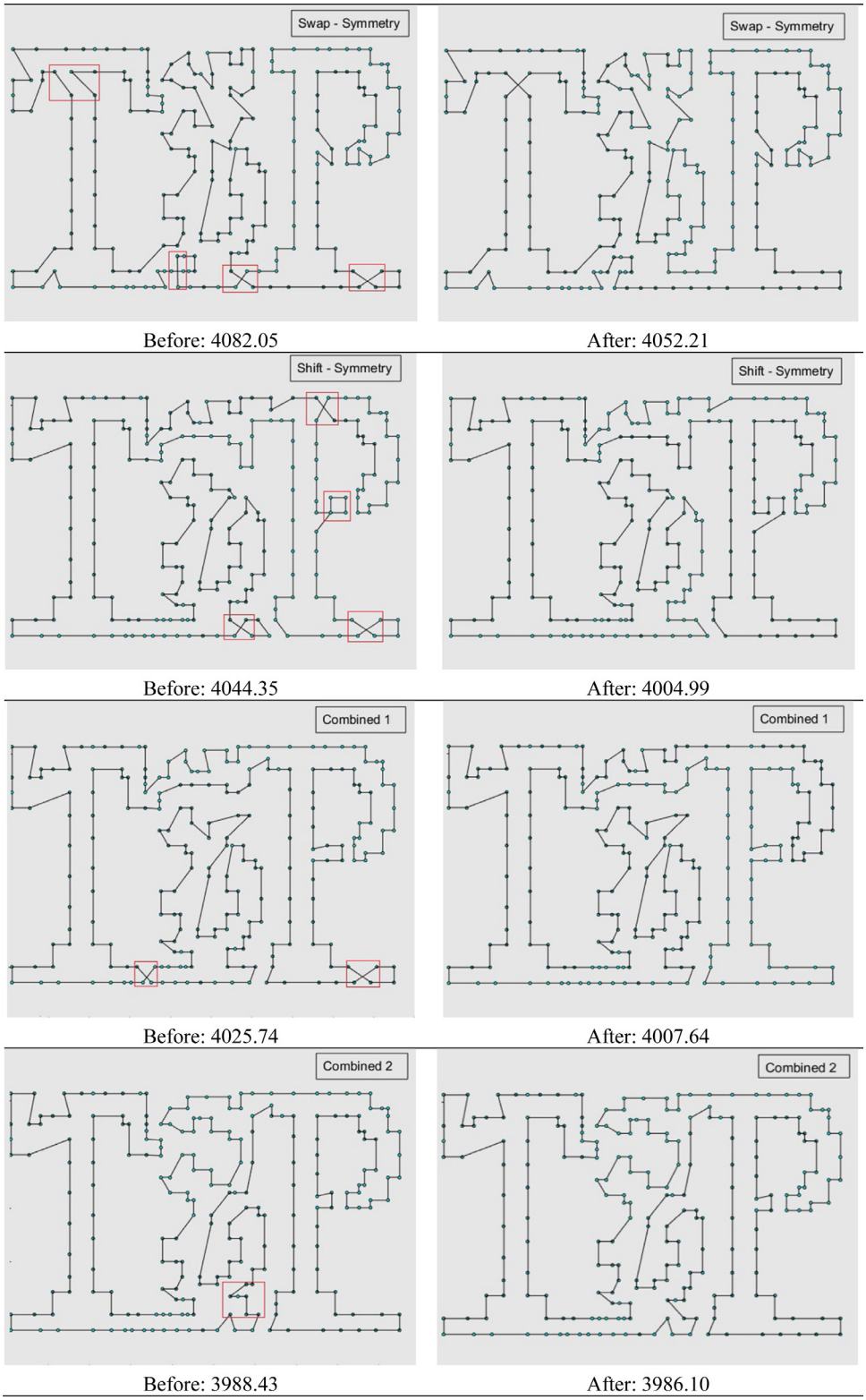


Fig. 7b. The route results of *swap+symmetry, shift+symmetry,* combined 1 and combined 2 on TSP225.

Chunhua et al. [46], proposed STA and compared it with SA and ACO, and BERLIN52, ATT48 and ULYSSES16 problems were used in experiments. Cinar et al. [47] did not make comparison with Chunhua et al. [46] in their study on ATT48 and ULYSSES16 problems because of the faulty coordinate system calculation. So in this study, the proposed algorithm is compared with STA, SA, ACO and DTSA on BERLIN52 problem. Comparison results of STA, SA and ACO are directly taken from [46] and DTSA is directly taken from [47] are compared with DJAYA with the same conditions mentioned before in this study. For a fair comparison MaxFEs is set at 4000 for all methods. The results of DJAYA, ACO, SA, STA and DTSA algorithms for BERLIN52 problem are given in Table 6.

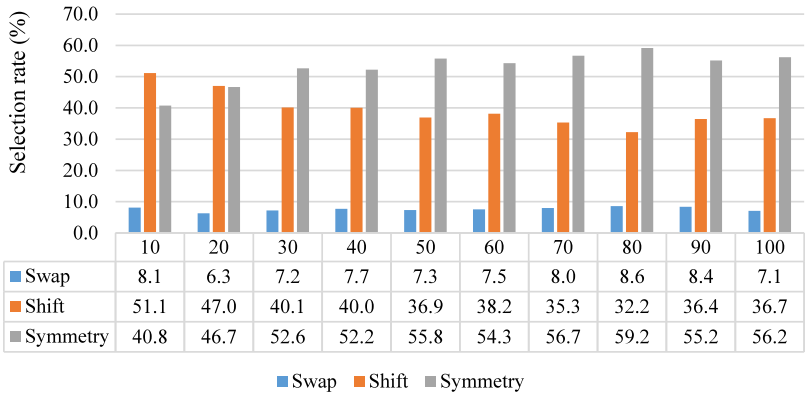


Fig. 8. Selection rate of transformation operators by roulette wheel selection for different N values.

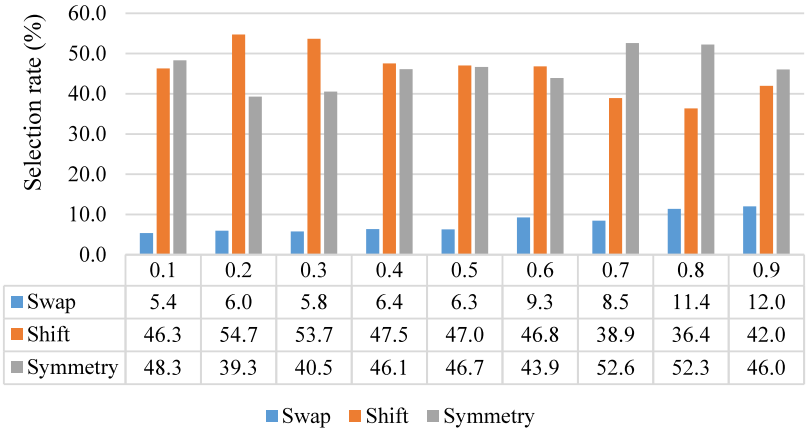


Fig. 9. Selection rate of transformation operators by roulette wheel selection for different $ST1$ values.

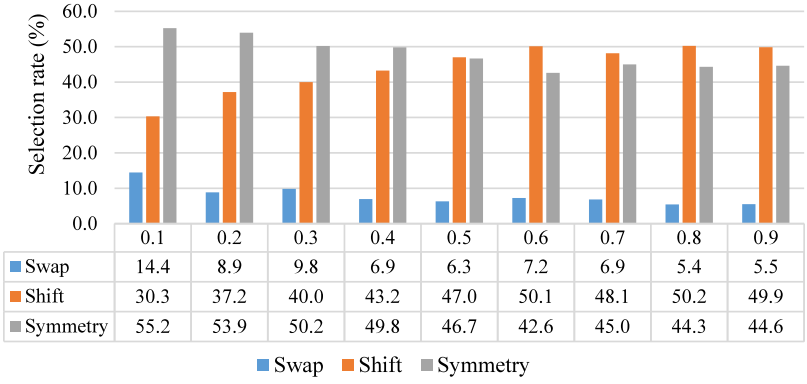


Fig. 10. Selection rate of transformation operators by roulette wheel selection for different $ST2$ values.

Table 4
Parameter analysis of $ST1$ on TSP225.

$ST1$	Mean	Std. Dev.	RE (%)
0.1	4013.95	31.63	4.02
0.2	4005.28	47.28	3.79
0.3	4008.81	33.25	3.88
0.4	4007.12	40.48	3.84
0.5	**3995.88**	42.91	**3.55**
0.6	4012.18	29.87	3.97
0.7	3996.47	38.73	3.56
0.8	4007.05	34.75	3.84
0.9	4014.88	45.75	4.04

Table 5
Parameter analysis of $ST2$ on TSP225.

$ST2$	Mean	Std. Dev.	RE (%)
0.1	4031.82	35.79	4.48
0.2	4011.32	33.97	3.95
0.3	4005.27	31.56	3.79
0.4	4004.29	38.54	3.77
0.5	**3995.88**	42.91	**3.55**
0.6	4012.13	29.31	3.97
0.7	4008.60	39.95	3.88
0.8	4014.67	55.42	4.03
0.9	4003.50	39.65	3.74

Table 6
Comparison of DJAYA with ACO, SA, STA, DTSA.

Problem	Method	Best	Worst	Mean	Std. Dev.
BERLIN52	SA	8186.40	9585.80	8983.80	380.10
	ACO	8240.40	9151.30	8777.60	267.11
	STA	7544.40	8630.50	8247.20	273.45
	DTSA	7542.00	7929.00	7689.17	108.40
	DJAYA	**7542.00**	**7831.00**	**7668.35**	112.50

Table 7
Comparison of DJAYA with SA, DSTA0, DSTAI, DSTAII and DTSA.

Problem	Method	Mean	Std. Dev.	RE (%)	Rank
KROA100	SA	22635.00	778.72	6.36	4
	DSTA0	23213.00	906.11	9.07	6
	DSTAI	22835.00	715.85	7.30	5
	DSTAII	21767.00	221.64	2.28	3
	DTSA	**21506.78**	260.55	1.06	1
	DJAYA	21705.98	290.22	1.99	2
KROB100	SA	23657.00	445.78	6.85	4
	DSTA0	23794.00	517.05	7.47	6
	DSTAI	23734.00	507.38	7.19	5
	DSTAII	**22880.00**	302.14	3.34	1
	DTSA	23139.26	181.74	4.51	3
	DJAYA	22973.73	234.79	3.76	2
KROC100	SA	22223.00	522.20	7.10	5
	DSTA0	22877.00	709.87	10.26	6
	DSTAI	21891.00	536.88	5.50	4
	DSTAII	**21378.00**	246.34	3.03	1
	DTSA	21817.08	217.77	5.15	3
	DJAYA	21702.02	186.32	4.59	2
KROD100	SA	22911.00	483.01	7.59	4
	DSTA0	23043.00	565.80	8.21	6
	DSTAI	22665.00	592.53	6.44	3
	DSTAII	**21991.00**	315.32	3.27	1
	DTSA	22972.26	390.50	7.88	5
	DJAYA	22631.25	487.62	6.28	2
KROE100	SA	23125.00	389.42	4.79	4
	DSTA0	23738.00	450.82	7.57	6
	DSTAI	23371.00	678.69	5.90	5
	DSTAII	22637.00	166.82	2.58	3
	DTSA	**22547.00**	121.96	2.17	1
	DJAYA	22582.47	252.07	2.33	2

When Table 6 is considered, it is understood that the DJAYA algorithm has a better performance than STA, SA, ACO and DTSA on BERLIN52.

4.3. Comparisons with SA, DSTA0, DSTAI, DSTAII and DTSA

The second comparison is performed with Zhou et al. [4] and Cinar et al. [47] studies. Zhou et al. proposed some variants of discrete state transition algorithm such as DSTA0, DSTAI and DSTAII with transformation operators for TSPs and compared with SA. In their experiments, KROA100, KROB100, KROC100, KROD100 and KROE100 problems were tested. In this study, the proposed algorithm is compared with DSTA variants, SA and DTSA for these problems. Comparison results of DSTA0, DSTAI and DSTAII and SA are directly taken from [4] and DTSA is directly taken from [47]. For a fair comparison MaxFEs is set at 9×10^4 for all algorithms. Comparison results of DJAYA with SA, DSTA variants and DTSA are given in Table 7. When Table 7 is examined, DJAYA is obtained efficient results for KROA100 and KROE100 problems although the best solver is DTSA.

The sum ranking of SA, DSTA variants, DTSA and DJAYA are simulated in Fig. 11. It is seen from Fig. 11, DSTAII and DJAYA have obtained better quality and more effective solutions. The total rank value of DSTAII is 9 and the total rank value of DJAYA is 10. However, DSTAII is slightly better and the second best solver is DJAYA.

4.4. Comparisons with ACO, ABC, HA and DTSA

Another comparison is performed with Gündüz et al. [6] and Cinar et al. [47] studies. Gündüz et al. [6] proposed a hierarchic approach (HA) based ACO and ABC for solving TSPs. In their experiments, 10 different benchmark instances such as EIL51, EIL76, EIL101, KROA100, OLIVER30, BERLIN52, ST70, PR76, CH150 and TSP225 were tested. In this comparison, the proposed algorithm is compared with ACO, ABC, HA, and DTSA. Comparison results of ACO, ABC and HA are directly taken from [6] and DTSA is directly taken from [47]. For a fair comparison MaxFEs is set to $D \times 500$ for all algorithms. D indicates the number of cities of the instance and it is mean that MaxFEs is defined depended on the number of cities for each test problem. Comparison results of DJAYA with ACO, ABC, HA and DTSA are given in Table 8. In addition, a comparisons of rank values for these algorithm is given in Fig. 12.

According to Fig. 12, The total rank value of HA and DJAYA are equal to each other and HA and DJAYA are shown effective performances. The total rank value of the DJAYA and HA is 18, and the DJAYA and HA algorithms have shown more effective performances compared to other algorithms used in comparisons. However, if the total number of the problems are taken into account, it is demonstrated that DJAYA is slightly better than HA. According to Fig. 12, the most insufficient algorithm among the algorithms used in comparisons was the ABC algorithm with a total rank value of 50.

As seen from Table 8, DJAYA is found the best routes for 5 problems (number of all problems 10) such as EIL76, EIL101, PR76, TSP225 and CH150 instances. HA is found the best routes for 4 problems and DTSA is found the best route just for one problem. In the light of these results, it is understood that DJAYA has shown a better performance than ACO, ABC and DTSA.

4.5. Comparisons with ACO, PSO, GA, BH and DTSA

The last comparison is made with Hatamlou [45] and Cinar et al. [47] studies. Hatamlou [45] is presented a method based on black hole algorithm (BH) for solving TSP and 9 different benchmark instances such as ULYSSES22, BAYG29, ATT48, GR96, EIL51, EIL76, EIL101, BERLIN52 and ST70 were tested for experiments. Cinar et al. [47] did not make a comparison with Hatamlou [45] in their study on ULYSSES22, BAYG29, ATT48 and GR96 problems because of the faulty coordinate system calculation. Therefore, the proposed algorithm is compared with BH, ACO, PSO, GA and DTSA on EIL51, EIL76, EIL101, BERLIN52 and ST70 instances. Comparison results of BH, ACO, PSO, GA are directly taken from [45] and DTSA is directly taken from [47]. For a fair comparison MaxFEs is set to 2×10^4, the number of population size N is chosen 100 for all algorithms and DJAYA executed 5 independent runs. Comparison results of DJAYA with BH, ACO, PSO, GA and DTSA are given in Table 9.

As seen from Table 9, DJAYA produces more effective and better quality results for all problems used for comparison. Experimental results confirmed that DJAYA is outperformed the ACO, PSO, GA, BH and DTSA algorithms in terms of the solution quality and robustness. A comparisons of rank values for these algorithm is given in Fig. 13. When the total rank values given in Fig. 13 are considered, the total rank value of the DJAYA is 5, and the DJAYA algorithm has shown a more effective performance compared to other algorithms used in comparisons. The second best solver algorithm was the DTSA algorithm and the rank value of the DTSA algorithm is 11. According to Fig. 13, the most insufficient algorithm among the algorithms used in comparisons was the PSO algorithm with a total rank value of 30.

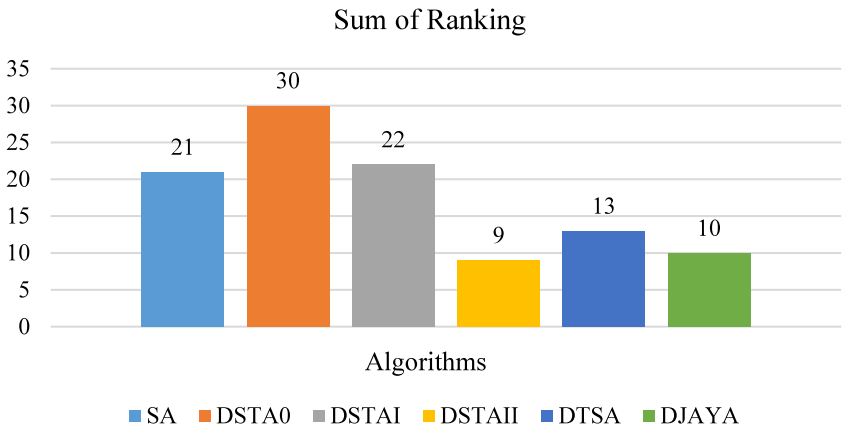


Fig. 11. Comparisons of rank values for SA, DSTA variants, DTSA and DJAYA.

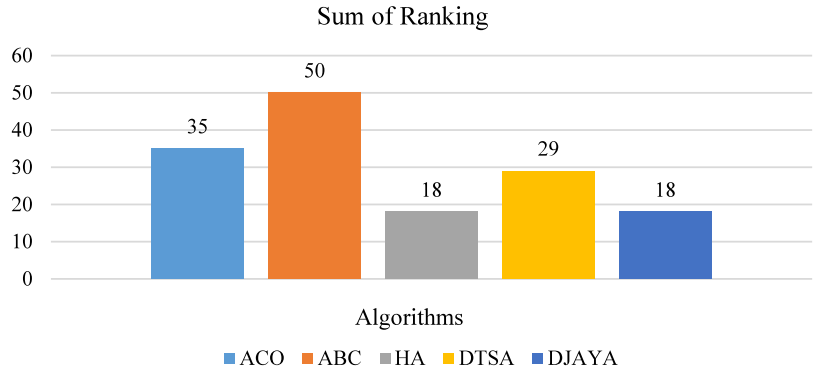


Fig. 12. Comparisons of rank values for ACO, ABC, HA, DTSA and DJAYA.

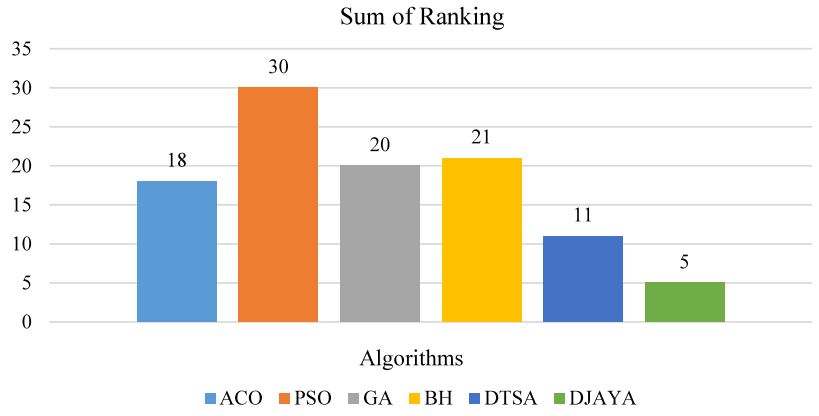


Fig. 13. Comparisons of rank values for ACO, PSO, GA, BH, DTSA and DJAYA.

5. Results and discussion

The DJAYA has been applied to solve 14 symmetric traveling salesman problems and the experimental results are compared with the state-of-art optimization algorithms. The experimental results show that DJAYA show better or similar performance to the compared algorithms. When we analyze the results of DJAYA on the benchmark problems dealt with the study, it can be seen that the DJAYA algorithm shows comparable and competent performance in terms of solution quality and robustness on low, middle, and large dimensional traveling salesman problem benchmark datasets. When we compare DJAYA update mechanism with the other optimization algorithms, there is a considerable distinction among them, and that the DJAYA uses best and worst solutions positions to get a candidate solution. By adding the worst and best solutions information to the update rule in DJAYA and using the transformation operators to generate the new position of candidate solution, the required exploration and perturbation are provided for the proposed algorithm. Also the exploitation capability is extended thanks to using 2-opt local search heuristic. The obtained results on symmetric TSPs demonstrate that the proposed algorithm is an alternative and competitive optimizer for on the permutation-coded discrete optimization problems.

Table 8
Comparison of DJAYA with ACO, ABC, HA and DTSA.

Problem	Method	Mean	Std. Dev.	RE (%)	Rank
OLIVER30	ACO	424.68	1.41	0.22	2
	ABC	462.55	12.47	9.16	5
	HA	**423.74**	0.00	**0.00**	1
	DTSA	428.50	4.21	1.12	4
	DJAYA	426.88	2.74	0.74	3
EIL51	ACO	457.86	4.07	6.76	4
	ABC	590.49	15.79	37.69	5
	HA	443.39	5.25	3.39	2
	DTSA	443.93	4.04	3.51	3
	DJAYA	**440.18**	4.95	**2.64**	1
BERLIN52	ACO	7659.31	38.70	1.52	4
	ABC	10390.26	439.69	37.72	5
	HA	**7544.37**	0.00	**0.00**	1
	DTSA	7545.83	21.00	0.02	2
	DJAYA	7580.30	80.60	0.48	3
ST70	ACO	709.16	8.27	4.73	4
	ABC	1230.49	41.79	81.73	5
	HA	**700.58**	7.51	**3.47**	1
	DTSA	708.65	6.77	4.66	3
	DJAYA	702.30	9.56	3.72	2
EIL76	ACO	561.98	3.50	3.04	2
	ABC	931.44	24.86	70.78	5
	HA	**557.98**	4.10	**2.31**	1
	DTSA	578.58	3.93	6.09	4
	DJAYA	573.17	6.33	5.10	3
PR76	ACO	116321.22	885.79	7.55	4
	ABC	205119.61	7379.16	89.65	5
	HA	115072.29	742.90	6.39	3
	DTSA	114930.03	1545.64	6.26	2
	DJAYA	**113258.29**	1711.93	**4.71**	1
KROA100	ACO	22880.12	235.18	7.49	4
	ABC	53840.03	2198.36	152.94	5
	HA	22435.31	231.34	5.40	3
	DTSA	**21728.40**	358.13	**2.08**	1
	DJAYA	21735.31	331.33	2.13	2
EIL101	ACO	693.42	6.80	7.96	4
	ABC	1315.95	35.28	104.88	5
	HA	683.39	6.56	6.40	2
	DTSA	689.91	4.47	7.41	3
	DJAYA	**677.37**	4.87	**5.46**	1
CH150	ACO	6702.87	20.73	2.61	4
	ABC	21617.48	453.71	230.93	5
	HA	6677.12	19.30	2.22	2
	DTSA	6748.99	32.63	3.32	3
	DJAYA	**6638.63**	52.79	**1.63**	1
TSP225	ACO	4176.08	28.34	8.22	3
	ABC	17955.12	387.35	365.28	5
	HA	4157.85	26.27	7.74	2
	DTSA	4230.45	58.76	9.93	4
	DJAYA	**4095.02**	42.54	**6.12**	1

Table 9
Comparison of DJAYA with ACO, PSO, GA, BH and DTSA.

Problem	Method	Mean	Std. Dev.
EIL51	ACO	461.0175	6.2974
	PSO	574.8022	107.2371
	GA	453.4773	9.4157
	BH	458.9252	38.6365
	DTSA	456.5184	8.9247
	DJAYA	**440.4394**	**3.1055**
EIL76	ACO	594.1442	40.2152
	PSO	975.6397	152.4061
	GA	652.0593	122.0972
	BH	659.1021	152.1754
	DTSA	588.0623	5.7296
	DJAYA	**574.4803**	**5.6710**
EIL101	ACO	763.9207	59.9684
	PSO	1499.9911	319.7468
	GA	838.8307	9.9642
	BH	897.3813	210.1446
	DTSA	689.8384	7.2994
	DJAYA	**686.8843**	**6.0664**
BERLIN52	ACO	8522.9017	1152.2000
	PSO	11089.5286	2067.9323
	GA	9288.4483	1301.2108
	BH	8455.8304	508.9871
	DTSA	7761.6000	62.8594
	DJAYA	**7627.0000**	120.3869
ST70	ACO	757.7540	59.6079
	PSO	1321.8137	269.2793
	GA	1158.8458	52.1734
	BH	797.5745	125.2272
	DTSA	710.4037	2.7956
	DJAYA	**707.2151**	15.3049

6. Conclusion and future works

Optimization problems are gathered under two main categories such as continuous and discrete optimization when considering the type of decision variables. Jaya algorithm is a newly proposed stochastic population-based metaheuristic optimization algorithm to solve constrained and unconstrained continuous optimization problems. There is no permutation-coded discrete version of Jaya in the literature. So in this study, we improved a discrete form of Jaya called as DJAYA for solving TSPs. This study we focus on modification of Jaya algorithm to solve discrete optimization problems using eight transformation operators. Firstly, the effect of the transformation operators to the performance of the discrete version of Jaya algorithm has been analyzed, and then its performance has been compared with state-of-art algorithms on the TSPs. In order to improve the obtained solutions by the proposed algorithm, 2-opt heuristic approach is applied to these solutions. In accordance with the analysis and comparisons, the proposed algorithm produces better or comparable performances on the problem dealt with the study in terms of solution quality and robustness. In near future, we plan to develop new solution update rules for Jaya to solve the discrete optimization problems such as graph coloring and job-shop scheduling.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] K.G. Murty, Optimization Models for Decision Making: Volume, University of Michigan, Ann Arbor, 2003.
[2] A. Ouaarab, B. Ahiod, X.-S. Yang, Discrete cuckoo search algorithm for the travelling salesman problem, Neural Comput. Appl. 24 (2014) 1659–1669.
[3] N. Gould, An introduction to algorithms for continuous optimization, in: Oxford University Computing Laboratory Notes, 2006.
[4] X. Zhou, D.Y. Gao, C. Yang, W. Gui, Discrete state transition algorithm for unconstrained integer optimization problems, Neurocomputing 173 (2016) 864–874.
[5] M. Mahi, Ö.K. Baykan, H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, Appl. Soft Comput. 30 (2015) 484–490.
[6] M. Gündüz, M.S. Kiran, E. Özceylan, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem, Turk. J. Electr. Eng. Comput. Sci. 23 (2015) 103–117.
[7] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, Q. Wang, Particle swarm optimization-based algorithms for TSP and generalized TSP, Inf. Process. Lett. 103 (2007) 169–176.
[8] M.K. Sayadi, A. Hafezalkotob, S.G.J. Naini, Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation, J. Manuf. Syst. 32 (2013) 78–84.
[9] M. Aslan, N.A. Baykan, A performance comparison of graph coloring algorithms, Int. J. Intell. Syst. Appl. Eng. (2016) 1–7.

[10] A. Hertz, D. de Werra, Using tabu search techniques for graph coloring, Computing 39 (1987) 345–351.
[11] Z.W. Geem, Harmony search in water pump switching problem, in: International Conference on Natural Computation, Springer, 2005, pp. 751–760.
[12] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (1998) 753–782.
[13] J.K. Lenstra, A.R. Kan, Some simple applications of the travelling salesman problem, J. Oper. Res. Soc. 26 (1975) 717–733.
[14] C. Ravikumar, Parallel techniques for solving large scale travelling salesperson problems, Microprocess. Microsyst. 16 (1992) 149–158.
[15] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, Springer-Verlag, 1994.
[16] W.-h. Zhong, J. Zhang, W.-n. Chen, A novel discrete particle swarm optimization to solve traveling salesman problem, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 3283–3287.
[17] Ö. Ergun, J.B. Orlin, A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem, Discrete Optim. 3 (2006) 78–85.
[18] A. Chentsov, L. Korotayeva, The dynamic programming method in the generalized traveling salesman problem, Math. Comput. Modelling 25 (1997) 93–105.
[19] R. Radharamanan, L. Choi, A branch and bound algorithm for the travelling salesman and the transportation routing problems, Comput. Ind. Eng. 11 (1986) 236–240.
[20] M. Padberg, G. Rinaldi, Optimization of a 532-city symmetric traveling salesman problem by branch and cut, Oper. Res. Lett. 6 (1987) 1–7.
[21] H. Hernández-Pérez, J.-J. Salazar-González, A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, Discrete Appl. Math. 145 (2004) 126–139.
[22] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W. Savelsbergh, P.H. Vance, Branch-and-price: Column generation for solving huge integer programs, Oper. Res. 46 (1998) 316–329.
[23] B. Fleischmann, A cutting plane procedure for the travelling salesman problem on road networks, European J. Oper. Res. 21 (1985) 307–317.
[24] G. Laporte, Y. Nobert, A cutting planes algorithm for the m-salesmen problem, J. Oper. Res. Soc. 31 (1980) 1017–1023.
[25] G. Laporte, The traveling salesman problem: An overview of exact and approximate algorithms, European J. Oper. Res. 59 (1992) 231–247.
[26] S. Jaballah, K. Rouis, F.B. Abdallah, J.B.H. Tahar, An improved shuffled frog leaping algorithm with a fast search strategy for optimization problems, in: 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP, IEEE, 2014, pp. 23–27.
[27] M. Aslan, M. Gunduz, M.S. Kiran, JayaX: Jaya algorithm with xor operator for binary optimization, Appl. Soft Comput. 82 (2019) 105576.
[28] J. Knox, Tabu search performance on the symmetric traveling salesman problem, Comput. Oper. Res. 21 (1994) 867–876.
[29] S.-M. Chen, C.-Y. Chien, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, Expert Syst. Appl. 38 (2011) 14439–14450.
[30] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, Appl. Soft Comput. 11 (2011) 3680–3689.
[31] J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, Genetic algorithms for the traveling salesman problem, in: Proceedings of the First International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, 1985, pp. 160–168.
[32] H. Braun, On solving travelling salesman problems by genetic algorithms, in: International Conference on Parallel Problem Solving from Nature, Springer, 1990, pp. 129–133.
[33] N.L. Ulder, E.H. Aarts, H.-J. Bandelt, P.J. Van Laarhoven, E. Pesch, Genetic local search algorithms for the traveling salesman problem, in: International Conference on Parallel Problem Solving from Nature, Springer, 1990, pp. 109–116.
[34] J.-Y. Potvin, Genetic algorithms for the traveling salesman problem, Ann. Oper. Res. 63 (1996) 337–370.
[35] K. Bryant, Genetic algorithms and the travelling salesman problem, 2000.
[36] G. Üçoluk, Genetic algorithm solution of the TSP avoiding special crossover and mutation, Intell. Autom. Soft Comput. 8 (2002) 265–272.
[37] Z.H. Ahmed, Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, Int. J. Biom. Bioinform. 3 (2010) 96.
[38] A. Hussain, Y.S. Muhammad, M. Nauman Sajid, I. Hussain, A. Mohamd Shoukry, S. Gani, Genetic algorithm for traveling salesman problem with modified cycle crossover operator, Comput. Intell. Neurosci. 2017 (2017).
[39] Ş. Gülcü, M. Mahi, Ö.K. Baykan, H. Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem, Soft Comput. 22 (2018) 1669–1685.
[40] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1997) 53–66.
[41] M.S. Kıran, H. Işcan, M. Gündüz, The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem, Neural Comput. Appl. 23 (2013) 9–21.
[42] L. Li, Y. Cheng, L. Tan, B. Niu, A discrete artificial bee colony algorithm for TSP problem, in: International Conference on Intelligent Computing, Springer, 2011, pp. 566–573.
[43] D. Karaboga, B. Gorkemli, A combinatorial artificial bee colony algorithm for traveling salesman problem, in: 2011 International Symposium on Innovations in Intelligent Systems and Applications, IEEE, 2011, pp. 50–53.
[44] K.-P. Wang, L. Huang, C.-G. Zhou, W. Pang, Particle swarm optimization for traveling salesman problem, in: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, IEEE cat. no. 03ex693, IEEE, 2003, pp. 1583–1585.
[45] A. Hatamlou, Solving travelling salesman problem using black hole algorithm, Soft Comput. 22 (2018) 8167–8175.
[46] Y. Chunhua, T. Xiaolin, Z. Xiaojun, G. Weihua, State transition algorithm for traveling salesman problem, in: Proceedings of the 31st Chinese Control Conference, IEEE, 2012, pp. 2481–2485.
[47] A.C. Cinar, S. Korkmaz, M.S. Kiran, A discrete tree-seed algorithm for solving symmetric traveling salesman problem, Eng. Sci. Technol. 4 (2020) 879–890.
[48] R. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, Int. J. Ind. Eng. Comput. 7 (2016) 19–34.
[49] G.A. Croes, A method for solving traveling-salesman problems, Oper. Res. 6 (1958) 791–812.
[50] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA J. Comput. 3 (1991) 376–384.
[51] S. Irnich, B. Funke, T. Grünert, Sequential search and its application to vehicle-routing problems, Comput. Oper. Res. 33 (2006) 2405–2429.
[52] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, 1988.
[53] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (1973) 498–516.
[54] W. Pang, K.-p. Wang, C.-g. Zhou, L.-j. Dong, Fuzzy discrete particle swarm optimization for solving traveling salesman problem, in: The Fourth International Conference on Computer and Information Technology, 2004, CIT'04, IEEE, 2004, pp. 796–800.
[55] S. Lin, Computer solutions of the traveling salesman problem, Bell Syst. Tech. J. 44 (1965) 2245–2269.
[56] B. Görkemli, D. Karaboga, Quick combinatorial artificial bee colony-qCABC-optimization algorithm for TSP, 2013.
[57] B. Emine, E. Ülker, Dıscrete socıal spıder algorıthm for the traveling salesman problem, Artif. Intell. Rev. (2020) 1–23.
[58] M. Akhand, S.I. Ayon, S. Shahriyar, N. Siddique, H. Adeli, Discrete spider monkey optimization for travelling salesman problem, Appl. Soft Comput. 86 (2020) 105887.
[59] Y. Huang, X.-N. Shen, X. You, A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem, Appl. Soft Comput. 107085.
[60] R.V. Rao, Jaya: An Advanced Optimization Algorithm and its Engineering Applications, Springer International Publishing, Switzerland, 2019.
[61] R.V. Rao, D.P. Rai, J. Balic, Surface grinding process optimization using Jaya algorithm, in: Computational Intelligence in Data Mining—Volume 2, Springer, 2016, pp. 487–495.
[62] T. Prakash, V. Singh, S. Singh, S. Mohanty, Binary Jaya algorithm based optimal placement of phasor measurement units for power system observability, 2017.
[63] R.V. Rao, G. Waghmare, A new optimization algorithm for solving complex constrained design optimization problems, Eng. Optim. 49 (2017) 60–83.
[64] R. Rao, K. More, Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm, Energy Convers. Manage. 140 (2017) 24–35.
[65] L. Wang, C. Huang, A novel elite opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models, Optik 155 (2018) 351–356.
[66] R.V. Rao, D.P. Rai, J. Balic, A multi-objective algorithm for optimization of modern machining processes, Eng. Appl. Artif. Intell. 61 (2017) 103–125.
[67] L. Wang, Z. Zhang, C. Huang, K.L. Tsui, A GPU-accelerated parallel Jaya algorithm for efficiently estimating Li-ion battery model parameters, Appl. Soft Comput. 65 (2018) 12–20.
[68] K.C. More, R.V. Rao, Design optimization of plate-fin heat exchanger by using modified Jaya algorithm, in: Advanced Engineering Optimization Through Intelligent Techniques, Springer, 2020, pp. 165–172.
[69] T. Dede, M. Grzywiński, R.V. Rao, Jaya: A new meta-heuristic algorithm for the optimization of braced dome structures, in: Advanced Engineering Optimization Through Intelligent Techniques, Springer, 2020, pp. 13–20.

[70] M. Kumawat, N. Gupta, N. Jain, R. Bansal, Jaya algorithm based optimal allocation of distributed energy resources, in: Intelligent Computing Techniques for Smart Energy Systems, Springer, 2020, pp. 805–814.
[71] A.C. Cinar, Adaptation and analysis of tree-seed algorithm for solving constrained and discrete optimization problems, in: Computer Engineering, Konya Technical University Institute of Graduate Studies, 2020.
[72] G. Pataki, Teaching integer programming formulations using the traveling salesman problem, SIAM Rev. 45 (2003) 116–123.
[73] M.S. Kiran, TSA: Tree-seed algorithm for continuous optimization, Expert Syst. Appl. 42 (2015) 6686–6698.