Engineering Science and Technology, an International Journal 23 (2020) 879–890

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

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

Full Length Article

A discrete tree-seed algorithm for solving symmetric traveling salesman problem

Ahmet Cevahir Cinar [a,*], Sedat Korkmaz [b], Mustafa Servet Kiran [b]

[a] Turkish State Meteorological Service, Konya Division, 42090 Konya, Turkey
[b] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Konya Technical University, 42075 Konya, Turkey

ARTICLE INFO

Article history:
Received 22 June 2019
Revised 23 August 2019
Accepted 13 November 2019
Available online 22 November 2019

Keywords:
Tree seed algorithm
Transformation operator
Traveling salesman problem
Discrete optimization
Metaheuristic

ABSTRACT

Tree-Seed algorithm (TSA) is a recently developed nature inspired population-based iterative search algorithm. TSA is proposed for solving continuous optimization problems by inspiring the relations between trees and their seeds. The constrained and binary versions of TSA are present in the literature but there is no discrete version of TSA which decision variables represented as integer values. In the present work, the basic TSA is redesigned by integrating the swap, shift, and symmetry transformation operators in order to solve the permutation-coded optimization problems and it is called as DTSA. In the basic TSA, the solution update rules can be used for the decision variables whose are defined in continuous solution space, this rules are replaced with the transformation operators in the proposed DTSA. In order to investigate the performance of DTSA, well-known symmetric traveling salesman problems are considered in the experiments. The obtained results are compared with well-known metaheuristic algorithms and their variants, such as Ant Colony Optimization (ACO), Genetic Algorithm (GA), Simulated Annealing (SA), State Transition Algorithm (STA), Artificial Bee Colony (ABC), Black Hole (BH), and Particle Swarm Optimization (PSO). Experimental results show that DTSA is another qualified and competitive solver on discrete optimization.

1. Introduction

Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem. Although the mathematical model of the algorithm and understanding are too easy, effectively solving this problem is really difficult because TSP is in category of NP-Hard problems [1] so an algorithm that solves this problem at the polynomial time has not yet been proposed. The main problem is finding a Hamiltonian path with minimum cost on a weighted graph. TSP is an important benchmark problem because it models very significant problems such as scheduling, routing, threading of scan cells in a testable Very-Large-Scale-Integrated (VLSI), computer wiring, automatic drilling of printed circuit boards and circuits, movement of people, X-ray crystallography [2]. For solving TSP, some exact and heuristic methods have been proposed. Some exact methods are the branch and bound [3], branch and cut [4], branch and price [5], cutting planes [6], and Lagrangian dual [7]. Heuristic methods are ABC [8–12], PSO [13–18], SA [16], ACO [11,15,16,18–22], neural network [23], tabu search [24], artificial immune systems [25], cuckoo search [26,27], BH [28], STA [29,30], fruit fly [31], imperialist competitive algorithm (ICA)[32], physarum-energy optimization (PEO) [33], and GA [16,18,34–50]. Generally, two main groups of approaches are outstanding for solving permutation coded optimization problems. These approaches are based on path construction and path improvement [11].

Path construction methods usually use problem information, therefore they spend more time to create new solutions for problems. Path improvement methods create new solutions via crossover, mutation and swap etc. Usually, path improvement methods do not use problem information and they consume less time to create candidate solution but they need long iteration time to obtain a quality solution. Our approach only uses information in the population during the candidate solutions are created but in the initialization of the algorithm, a tour that is created by using the nearest nodes are assigned to a tree in order to improve the convergence characteristics of the proposed DTSA. For the rest of trees, random permutation solutions are generated and assigned. After initialization, these solutions are improved by swap, shift and symmetry transformation operators [29,30]. After the termination criterion is satisfied, obtained solution is improved with 2-opt algorithm. 2-opt algorithm [51] is a local search technique which is used for enhancing the solution quality.

* Corresponding author.
E-mail address: ahmetcevahircinar@gmail.com (A.C. Cinar).
Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2019.11.005
2215-0986/

The rest of the paper is arranged as follows. The literature review is given in Section 1.1, Main contribution and motivation of study are given in Section 1.2. In Section 2, the basic TSA is given, and DTSA is introduced with two subsections which are Section 3.1 (transformation operators) and Section 3.2 (proposed method). TSP is mentioned in Section 4. Section 5 is reserved for experimental results and discussions. Finally, our work is concluded in Section 6.

1.1. Literature review

In this section, we briefly talk about some well-known evolutionary computation and swarm intelligence methods for solving TSPs. The methods are given chronologically in this section.

Goldberg and Lingle [52] proposed the partially-mapped crossover (PMX) operator for solving TSP. Grefenstette et al. [38] discussed ordinal and adjacency representation in genetic algorithm for TSP. Heuristic Crossover (HX) is proposed in this work. HX solved synthetically created problems which have 50, 100, and, 200 cities, and obtained results showed that HX is not a good solver for TSP. But this work pointed some important future directions for solving TSPs. Fogel [53] discussed the performance of evolutionary algorithm (EA) on TSP. This approach compared with PMX integrated GA. Reported experiments show that EA outperforms GA in terms of solution quality. In another study, Braun (Braun, 1990) used GA for solving large TSPs. In this work, GA initialized with the nearest neighbor heuristic and it finalized with 2-opt and or-opt heuristics. Braun [36] proposed Insular GA which is a parallel approach using order crossover (OX) for creating offspring. Experimental investigations showed that this approach useful for solving TSPs. Ulder et al. [47] analyzed 2-opt and Lin-Kernighan local search methods on Genetic Local Search GA and Multi-start Local Search GA. Obtained results are compared with 2-opt SA and 2-opt Threshold Accepting. Genetic Local Search with Lin-Kernighan neighborhoods outperformed other approaches. Starkweather et al. [46] discussed the differences between blind TSP and classic TSP. In this work, improved edge recombination crossover (ERX), order based crossover (OBX), position-based crossover (PBX), PMX, and cycle crossover (CX) are compared. Enhanced ERX method has also been developed for preserving the important edges in parent individuals. OLIVER30 TSP is solved with these six methods and the results are compared. Also, a warehouse/shipping scheduling problem is solved with these methods. The performance of the algorithm for solving OLIVER30 TSP is acceptable, on the contrary is not for solving scheduling problem. This situation proves that the success of the operators is problem-dependent.

Potvin [34] prepared a survey about genetic algorithms for solving TSP. In this work, the author discussed crossover and mutation techniques for TSP and touched on parallel implementations. PMX, CX, modified crossover, OX, OBX, PBX, ERX, alternate edges crossover, HX are discussed as crossover techniques. Swap, local hill-climbing, and scramble techniques are mentioned as mutation approaches. Larranaga et al. [41] discussed various representations such as binary, path, adjacency, ordinal and matrix. In this work, 48 combinations of 8 crossover operators (alternating position crossover (APX), CX, ERX, OX, OBX, PMX, PBX, and voting recombination crossover (VRX)) and 6 mutation operators (exchange mutation (EM), inversion mutation (IVM), displacement mutation (DM), insertion mutation (ISM), simple inversion (SIM), scramble mutation (SM)) in GA are examined on three TSPs. There are three important results are obtained in this study. First, ERX outperformed all other crossover operators. Second, there is no significant difference between mutation operators. Third, the best representation type for TSP is path representation.

Bryant and Benjamin [37] investigated the effects of crossover and mutation methods in GA on TSPs. The results show that matrix representation and HX gave good results in terms of solution quality. Üçoluk [35] proposed a new permutation crossover genetic operator which based on PMX technique. The proposed method produced worse results than PMX on BAYS29, BERLIN52 and EIL101 problems. Moreover, both PMX and the proposed method did not achieve optimum results. Wang et al. [13] used PSO algorithm which equipped with swap operator and swap sequence mutation methods for BURMA14 problem. Tsai et al. [50] researched the performance of genetic operators on TSPs and proposed an approach which maintains the population diversity. DPSO algorithm is presented by Shi et al. [17] for solving symmetric and generalized TSPs. 5 symmetric and 19 generalized TSPs are solved by DPSO. Results show that DPSO is an alternative solver for these types of problems. In DPSO, candidate solutions are created by the slide and reverse operators. Also, delete-crossover process (similar to 2-opt) is integrated this algorithm for improving the solution quality. Ahmed [42] proposed sequential constructive crossover (SCX) technique for TSPs. SCX compared with ERX and generalized N-point crossover (GNX) on symmetric and asymmetric TSPLIB instances. Experimental results show that SCX is better than ERX and GNX.

ABC algorithm is discretized with integrating swap operators by Li et al. in [10]. In this work, discrete ABC (DABC) is compared with PSO in [13] and GA in [36]. The results show that, DABC is better than PSO algorithm on BURMA14 and DANTZIG42 problems, and compete with GA algorithm. Albayrak and Allahverdi [43] proposed a new mutation operator called as Greedy Sub Tour Mutation (GSTM) and compared with EM, DISP, INV, ISM, SIM, SCM, and GSM mutation techniques and DPX, OX, CX, and PMX crossover techniques. Experimental results show that GSTM produced acceptable solutions in a reasonable time when compared the other techniques. BERLIN52, KROA100, PR144, CH150, KROB150, PR152, RAT195, D198, KROA200, TS225, PR226, PR299, LIN318, and PCB442 problems are used as benchmark set. Karaboga and Gorkemli [12] developed the Combinatorial ABC (CABC) algorithm for solving combinatorial optimization problems. At the initialization phase, CABC used the nearest neighbor tour construction heuristic method and GSTM [43] technique is used for producing candidate solutions. KROB150 and KROA200 problems are used for analyze the efficiency of this approach, and good results are obtained. Chen and Chien [16] proposed a new method for solving TSPs, named as genetic simulated annealing ant colony system with particle swarm optimization (GSA-ACS-PSOT). In GSA-ACS-PSOT, initial solutions are created with ACS, after then these solutions are given to genetic simulated annealing techniques. After a certain iteration passes, obtained solutions are improved by particle swarm optimization technique. This approach has been created by hybridizing four different metaheuristic algorithms. The performance of GSA-ACS-PSOT is tested on 25 different TSPs. Chunhua et al. [30] proposed the discrete STA (DSTA). DSTA uses three state transformation techniques as swap, shift and symmetry for creating candidate solutions. In this work, DSTA compared with SA and ACO and the results show that DSTA consumes much less time and has better search ability than SA and ACO. A hybrid approach whose name is GA-PSO-ACO is proposed by Deng et al. [18]. In GA-PSO-ACO, GA and PSO work on exploration phase and ACO works on exploitation phase. Gorkemli and Karaboga [8] modified quick ABC (qABC) for solving TSPs. Onlooker bee phase is different in qABC than the basic ABC. In this work, a new similarity measure is developed and onlooker bees used this measure for new candidate solutions. This approach is named as quick Combinatorial ABC (qCABC). KROB150 and KROA200 problems are solved with qCABC. qCABC is compared with CABC and GSTM and outperforms these methods. Kıran et al. [9] used neighborhood operators for creating candidate solutions in discrete artificial bee colony algorithm. 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) operators are used in this study. Also these operators grouped and used as swapping group (RS, RSS, and RRSS) and insertion group (RI, RIS, and RRIS). OLIVER30, EIL51, BERLIN52, ST70, PR76, KROA100, EIL101, TSP225 and A280 are used as benchmark set for experimental results. Ouaarab et al. [26] proposed discrete cuckoo search (DCS) algorithm. 41 symmetric TSPs are solved by DCS. The obtained results by DCS are compared with some approaches. The results show that DCS is an alternative solver for symmetric TSPs. DCS is equipped with 2-opt and double bridge techniques [54]. 2-opt is used as local search, and double bridge is used as global search method in every iteration although these techniques are very time consuming approaches. TSPs which have 51 cities to 1379 cities are solved with DCS in this work.

Gunduz et al. [11] proposed a hybrid algorithm whose names is ACO-ABC. ACO and ABC are combined for solving TSPs. This hybrid approach has been proposed using the successful aspects of ABC and ACO algorithms. In this work, weaknesses of ABC and ACO, are stated and the hybrid approach is defended. Mahi et al. [15] proposed a hybrid solver which contains PSO, ACO, and 3-opt algorithm. At the starting phase of hybridization, the peculiar parameters of ACO which denoted as α and β are optimized by PSO and ACO solves the TSPs according to these parameters. When termination criterion is satisfied, the best tour which obtained by ACO is given to 3-opt algorithm and this local search method tries to improve the solution. This approach tested on EIL51, BERLIN52, ST70, EIL76, RAT99, KROA100, EIL101, LIN105, CH150 and KROA200 problems. The random key approach is integrated to cuckoo search for solving traveling salesman problem by Ouaarab et al. [27]. Random-key cuckoo search (RKCS) uses random-key encoding scheme for transferring the variables in continuous search space to integer search space. In RKCS, 2-opt local search works until a better result is obtained. The hybrid max–min ant system (HMMA) with a local search algorithm is proposed by Yong [20]. Zhou et al. [29] improved DSTA for solving TSPs. DSTA uses four state transformation techniques as swap, shift, symmetry, and substitute for creating candidate solutions. In this work, KROA100, KROB100, KROC100, KROD100 and KROE100 problems are used for performance testing and obtained results are compared with SA. Gulcu et al. propose a parallel cooperative hybrid algorithm (PACO-3Opt) based on ant colony optimization for solving TSPs [21]. PACO-3Opt uses 3-opt local search algorithm and this is a very time-consuming process. Therefore parallelization is used to accelerate the algorithm. The authors claimed that their approach overcomes the premature convergence of ACO. Hussain et al. [39] proposed a new crossover operator (CX2) which is based CX operator. This operator is compared with PMX, OX and CX operators. Experiments showed that CX2 is better than PMX and OX operators on some TSPs. Jubeir et al. [40] investigated GA's selection method's effect on TSPs. Stochastic Universal Selection (SUS), Rank Selection (RS), Tournament Selection and Roulette Wheel Selection (RWS) are generally used for parental selection in GA. In this work, authors proposed a new enhanced parent selection method for GA. This selection method uses minimum distance knowledge and similar to RS but without sorting. Proposed selection method outperformed SUS on GR24, BRAZIL58, SIL75, and PA561 problems of TSPs. Fruit fly optimization algorithm is modified for solving TSPs in another study [31]. In this work, a real-world problem is solved by the proposed approach. Hatamlou solves TSPs with black hole (BH) algorithm in [28]. Yildirim and Karci [55] redesign artificial atom algorithm for small-scale TSPs. In this work, candidate solutions are created by 2-opt local search method. The performance of this method is examined on WI29, ATT48, EIL51, BERLIN52, ST70, PR76 problems. Chen et al. [32] merged the ICA with a policy-learning function for solving the TSPs. Feng et al. proposed the Physarum-energy optimization (PEO) algorithm for solving TSPs [33].

Generally speaking, metaheuristic algorithms produce candidate solutions in three main stages: Initialization, Crossover, and Mutation. According to the literature review, we collected strategies for producing candidate individuals for permutation-coded path representation. In the initialization phase, the population is created with random permutation or nearest neighbor tour construction heuristic. In the crossover phase, more than one individuals are used for creating new individuals. Some crossover techniques found in literature are PMX [52], OX [56], CX [45], OBX [57], PBX [57], HX [58], sorted match crossover (SMX) [59], ERX [48,49], maximal preservative crossover (MPX) [60], VRX [44], APX [61], and SCX [42]. In the mutation phase, only one individual is used for creating a new individual. Some mutation techniques found in literature are EM [62], DM [63], IVM [64], ISM [53], SIM [38], SM [57], GSTM [43], neighborhood operators (RS, RI, RSS, RIS, RRS, RRSS, RRIS) [9], and transformation operators [29,30]. After termination criterion is met, obtained best result can be improved by local search techniques. The widely used local edge exchange heuristics are 2-opt [51], Or-opt [65], double bridge [54] and Lin and Kernighan neighborhoods [66].

Tree-Seed Algorithm (TSA) [67] is proposed by Kiran for solving continuous optimization problems. The parallel version of TSA [68,69] is presented in order to accelerate the algorithm. Kiran [70] investigated the performance of TSA for constrained optimization. In another study [71], search space limitation techniques are analyzed on TSA. Optimal Power Flow Problem in Large-Scale Power Systems is solved by TSA in [72]. TSA based image filter is proposed in [73] by Muneeswaran and Rajasekaran. Kiran [74] modified TSA with the withering process for overcoming stagnation behavior of the algorithm. Parameter identification of equivalent circuit models for Li-ion batteries based on TSA is proposed in [75]. TSA is modified for constrained optimization in [76] and also is modified for solving binary optimization problems in [77–79]. Muneeswaran and Rajasekaran [80] used TSA for performance evaluation of radial basis function neural network (RBFNN). In this work, the values of clustering centers, width and weights of the RBFNN are optimized by TSA. Zheng et al. [81] optimized the parameters of a system related with hydroelectric generating unit by TSA. TSA helped to solve gallbladder shape estimation problem in [82]. Zhou et al. [83] presented a novel variable-length tree-seed algorithm based competitive agglomeration (VTSA-CA) algorithm to determine the optimal number of clusters automatically and improve the clustering performances. Ding et al. [84] solved the structural damage identification problem with clustering based TSA. An improved version of TSA which named as EST-TSA (effective search tendency based to tree seed algorithm) proposed in [85] by Jiang et al.

1.2. Main contribution of the study

The main differences in our study from the studies in the literature and the main contributions to the literature are listed below.

- DTSA is a novel and alternative discrete optimization method for permutation-coded optimization problems.
- Experimental results show that DTSA produces adequate and comparable solutions for symmetric TSPs.
- This study presents the first version of TSA on the permutation-coded discrete form.

In this work, the peculiar parameters of TSA (search tendency, number of seeds, number of trees) have been analyzed for the first time for discrete optimization problems.

2. The basic Tree-Seed algorithm

TSA is proposed by Kiran in 2015 [67] for solving continuous optimization problems. The main idea of the algorithm is the relationship between trees and seeds. In the TSA, trees and seeds represent the possible solutions for the optimization problems. At initialization phase, trees are created randomly in search space. The number of trees is named as "stand size" in TSA. The number of seeds is analyzed in [67] and is suggested as between 10% and 25% of stand size. Seeds are created using Eqs. 1 or (2) for each tree at every iteration.

$$S(k) = T(i) + \alpha(B - T(r)) \quad (1)$$

$$S(k) = T(i) + \alpha(T(i) - T(r)) \quad (2)$$

where, T(i) is ith tree, S(k) is kth seed of T(i), α is a uniformly distributed random number between -1 and 1, B is the best tree obtained so far, T(r) is a random tree which is different from the T (i). Two seed creation equations (Eqs. 1 or 2) are controlled by Search Tendency (ST) parameter has a value between 0 and 1. During the search process, a uniformly distributed random number in range of [0, 1] is generated and compared with ST parameter. If ST smaller than this random number, Eq. (1) is used for seed creation, otherwise Eq. (2) is used for seed creation. Eq.1 provides intensification and Eq. (2) provides diversification in TSA. The flowchart of TSA is given in Fig. 1.

3. The discrete tree-seed algorithm (DTSA)

For solving discrete optimization problems, TSA is modified as follows:

In the initialization phase, trees are created as random permutations and a nearest neighbor tour assigned to one of the trees,

Transformation operators are described in Section 3.1, used for seed creation instead of Eqs. 1 and 2.

After the termination criterion is satisfied, obtained solution is given to 2-opt [51] algorithm in order to improve the solution.

3.1. Transformation operators

In this study, transformation operators such as swap, shift and symmetry [29,30] are used for creating seeds. Any repairing mechanism is not used because the seeds are created in feasible search space. These operators are briefly described below and some examples are given to illustrate the working mechanisms.

Swap transformation: Two different random numbers are created between 1 and N, where N is the total number of cities in

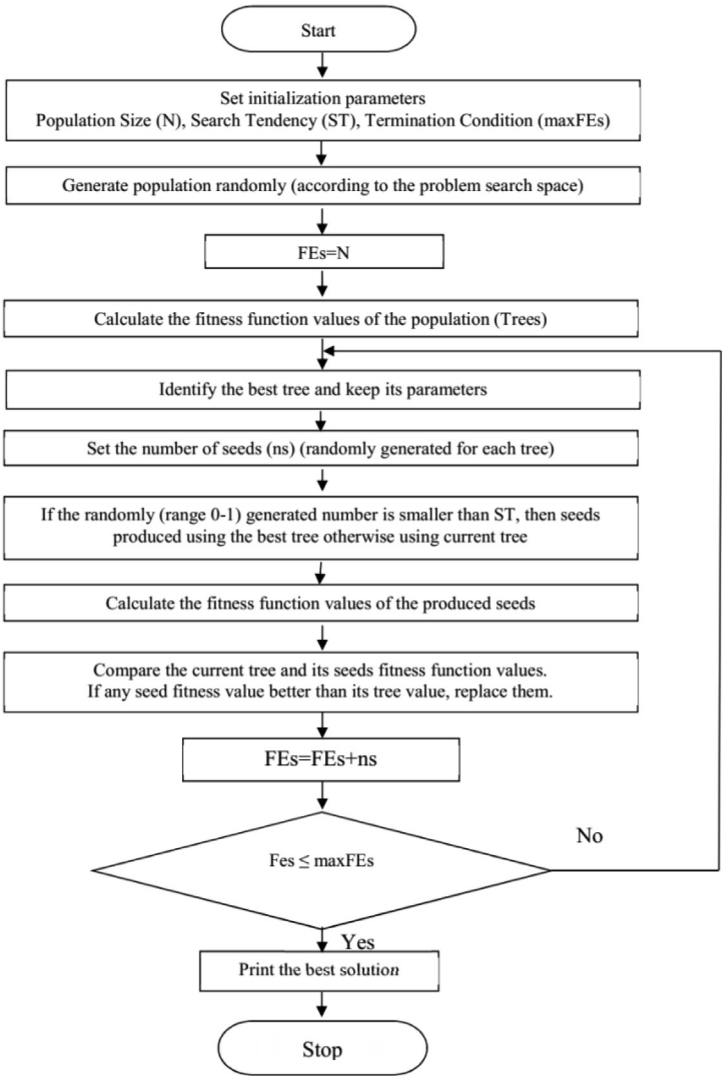

Fig. 1. The flowchart of TSA.

the problem. These two positions in the tree is swapped for creating new seed. The swap transformation makes small changes on the tree. The illustration of swap transformation operation is given in Fig. 2.

Shift transformation: Two different random numbers named as x and y are created between 1 and N, where N is the total number of cities in the problem. Decision variable in the position of x, is memorized and other decision variables are shifted to the left in the range of [x,y]. After then the memorized decision variable is assigned to the position of y. The shift transformation makes medium changes on the tree. The illustration of shift transformation operation is shown in Fig. 3.

Symmetry transformation: Two random positions for blocks which their block size is same random number of elements are determined. Each elements of blocks are inversed in their block and after then determined blocks are swapped. The symmetry transformation makes big changes on the tree. The illustration of symmetry transformation operation is shown in Fig. 4.

3.2. Proposed method

The comprehensive model of the proposed method is shown in Fig. 5. The algorithm initializes with the nearest neighbor tour and N-1 numbers of random permutation tours. Where N is the stand size of the TSA. The TSA algorithm integrated with swap, shift and symmetry transformation operators, starts with iterations with the aim of improving the quality of the initial solutions. After the termination criterion is met, obtained solution is given to 2-opt algorithm. 2-opt algorithm is a local solver and is used only once in the proposed approach. After then the final solution is reported as the best solution of DTSA.

The detailed algorithmic framework (pseudo code of DTSA) is given in Fig. 6.

Tree=	1	2	3	4	5	6
	swap(2,5)					
Seed=	1	5	3	4	2	6

Fig. 2. An illustration of swap transformation operation.

Tree=	1	2	3	4	5	6
	shift(3,5)					
Seed=	1	2	4	5	3	6

Fig. 3. An illustration of shift transformation operation.

Tree=	1	2	3	4	5	6
	symmetry((2,3),(4,5))					
Seed=	1	5	4	3	2	6

Fig. 4. An illustration of symmetry transformation operation.

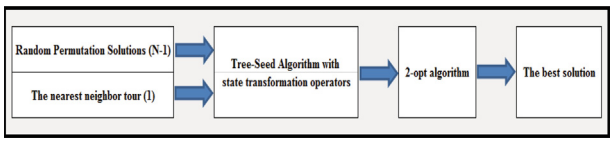

Fig. 5. The comprehensive model of the proposed method.

4. Traveling salesman problem

In TSP, a salesman starts his tour from any city and finished his tour in this city. A tour must include all of the cities which given for sale. The main purpose is to complete this tour on the shortest route. The main difficulty is the numerous number of possible tours: (n-1)!/2 for n cities. In this work, we use Euclidean distance for calculating distance between city i and city j as follows:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + \left((y_i - y_j)^2\right)} \tag{3}$$

where $d_{i,j}$ is the distance between city i and city j, x_i is x cartesian coordinate for city i, x_j is x Cartesian coordinate for city j, y_i is y Cartesian coordinate for city i, y_j is y Cartesian coordinate for city j. For calculating total tour length we use f function as follows:

$$f = d_{n,1} + \sum_{c=1}^{n-1} d_{c,c+1} \tag{4}$$

where n is the total number of cities. If $d_{i,j}$ equal to $d_{j,i,}$ these type problems named as symmetric TSPs. The main objective is finding a Hamiltonian path with minimum cost on a weighted graph.

5. Experimental results and discussions

The problems which are used for experimental results can be found in TSPLIB [86]. Most of the problems in TSPLIB are solved and the optimum values are presented. The numbers in the problem names indicate the city numbers. Used problems and their optimum values are OLIVER30 (423.74), EIL51 (428.87), BERLIN52 (7542/7544.37), ST70 (677.11), PR76 (108159.44), KROA100 (21282/21285.44), KROB100 (22141), KROC100 (20749), KROD100 (21294), KROE100 (22068), EIL101 (642.31), KROB150 (26130), TSP225 (3859), and A280 (2586.77). Comparisons are made with these optimum values. Coordinate types of these problems are Euclidean (EUC). In literature, in some works [28,30] different type problems (for example, geographical-GEO) calculated as EUC and this is causing the fault in the comparisons. Such situations have been identified and specified in the subsections. The relative error (RE) is calculated using Eq. (5) as follows.

$$RE = \frac{Optimum - Result}{Optimum} x100 \tag{5}$$

Here, *Result* is the obtained solution (mean of 30 different runs), *Optimum* is the optimum value of problem. The comparisons were made using *RE* values and the best results were marked in boldface font type to ease the comparison. The experiment set was run 30 times during performance measurements. All experiments are run on a Windows 10 Professional OS laptop using Intel(R) Core(TM) i7-4510U 2.0 GHz CPU, 8 GB of RAM and the codes were implemented in MATLAB. The working conditions of each experimental work are given in the relevant section.

5.1. Experiment 1: Comparisons of transformation operators for TSP

The first experiment is done for determining the best transformation operator for TSP. For this experiment, three transformation

```
Determine the maximum function evaluation number (maxfes)
D is the total number of city in the problem
Determine the number of trees (N)
Determine ST number between 0 and 1
Determine the first tree as nearest neighbor tour
Create all trees (except the first one) with random permutations between 1 and D
Calculate the objective functions of all trees
Set function evaluation number (fes) as N
Determine the best tree via using the objective function values (best)
While fes is smaller than maxfes
  For all trees (Tree(1) to Tree (N) )
    Determine the number of seed (ns) as 6
    Determine a random tree (except the current tree) from the stand (Tree(k))
    Create a random number between 0 and 1 (rand)
    If rand is smaller than ST
      Create a seed with swap operator with using the best tree (s1)
      Create a seed with shift operator with using the best tree (s2)
      Create a seed with symmetry operator with using the best tree (s3)
      Create a seed with swap operator with using the Tree(k) (s4)
      Create a seed with shift operator with using the Tree(k) (s5)
      Create a seed with symmetry operator with using the Tree(k) (s6)
    end if
    If rand is bigger than ST
      Create a seed with swap operator with using the current tree (s1)
      Create a seed with shift operator with using the current tree (s2)
      Create a seed with symmetry operator with using the current tree (s3)
      Create a seed with swap operator with using the Tree(k) (s4)
      Create a seed with shift operator with using the Tree(k) (s5)
      Create a seed with symmetry operator with using the Tree(k) (s6)
    end if
    Calculate the objective function of seeds of current tree and increase the fes value with adding 6
    Determine the best seed from 6 seeds (s1, s2, s3, s4, s5, s6) (bestseed)
    If the bestseed is smaller than current tree, replace the current tree with best seed
  end for
  Determine the best tree via using the objective function values (tempbest)
  If the tempbest is smaller than best, replace the best with the tempbest
end while
Apply the 2-opt algorithm using the best individual.
Report the best as the obtained best solution by DTSA.
```

Fig. 6. The pseudo code of DTSA.

operators which have three different situations analyzed on BERLIN52 TSP. Stand size is 52, the maximum number of evaluation is set at 104000 (D×2000) and results are shown in Table 1. According to the Table 1, the best transformation operator is the symmetry operator because it was produced more diversified solutions.

5.2. Experiment 2: determining the N and ST parameters for DTSA

For determining the optimum parameters for DTSA, the number of trees (N), the number of seeds (NS) and the search tendency (ST) parameters are analyzed. In the base versions of TSA, a random number of seed production mechanisms used in each iteration. But in DTSA we used fixed number of seeds as 6. For the number of trees (N), we proposed the total number of cities of problem. To prove that this suggestion is true, we use N as by increasing 10–300 by 10. Moreover NS, ST and maxfes parameters are used as 6, 0.5 and 800000 respectively. As seen from Table 2, N parameter does not seem to have a significant effect on the results.

For analyzing ST parameter, we use ST as by increasing 0.1 to 0.9 by 0.1. The obtained results are presented in Table 3. As seen from the Table 3, the mean errors are very close together. Therefore, we use ST as 0.5 for giving equal chances to best and random trees. In the light of the obtained information, in subsequent experiments, N, ST and NS parameters will be used as number of cities in the problem, 0.5 and 6 respectively.

5.3. Experiment 3: comparisons with SA, ACO and STA

DTSA is compared with STA firstly because STA used transformation operators for improving solutions. STA compared with SA and ACO in [30]. Comparison results directly taken from [30] and compared with DTSA algorithm under the conditions mentioned before in this work. All algorithm's maximum function evaluation is set at 4000 for a fair comparison as in [30]. The algorithmic parameters of SA are initial temperature and cooling rate and these are set at 5000 and 0.97 respectively. The values of specific parameters of ACO are $\alpha = 1$, $\beta = 5$, $\rho = 0.9$. The number of ants is set at 20 and total maximum of iterations is set at 200 for ACO. For STA, search enforcement is set at 20 and maximum iteration number is set at 200.

According to the Table 4, DTSA has performed well on BERLIN52 instance. In [30] work, results of ATT48 and ULYSSES16 problems are faulty because of coordinate system issue and these results were not included.

5.4. Experiment 4: Comparisons with SA and DSTA variants

In [29] work, KROA100, KROB100, KROC100, KROD100 and KROE100 problems are used for performance testing to DSTA variants and obtained results are compared with SA. DTSA was implemented with the same parameters in [29] for a fair comparison. DTSA results are compared with directly taken results of SA and DSTA variants from [29]. All algorithm's maximum function evaluation is set at 90000. The algorithmic parameters of SA are initial temperature and cooling rate and these are set at 2000 and 0.97 respectively. For DSTA variants, search enforcement is set at 100 and maximum iteration number is set at 900. Comparison of DTSA with SA and DSTA variants are presented in Table 5. When this information is examined, DTSA is the best solver for KROA100 and KROE100 problems, DTSA is the second best solver for other three problems among the other solvers.

Table 1
Transformation operators performance analyze on BERLIN52 TSP.

Method Detail	TSA Mean	TSA + 2Opt Mean	Std.Dev.	RE(%)
swap(current tree)	8133.00	7863.00	0.00	4.08
swap(random tree)	8059.17	7858.00	3.22	4.02
swap(best tree)	8133.00	7863.00	0.00	4.08
shift(current tree)	7816.73	7678.57	83.39	1.78
shift(random tree)	7903.13	7740.23	74.13	2.56
shift(best tree)	7891.37	7758.43	65.71	2.79
symmetry(current tree)	**7683.73**	**7542.00**	**0.00**	**0.00**
symmetry(random tree)	**7697.00**	**7542.00**	**0.00**	**0.00**
symmetry(best tree)	7737.90	7551.03	49.48	0.12

Table 2
N analyses of DTSA on KROB150 TSP.

N	Mean	Std.Dev.	Mean Error
10	27011.31	385.22	3.37
20	26981.58	324.98	3.26
30	27005.62	278.70	3.35
40	26975.44	246.02	3.24
50	26879.64	256.89	2.87
60	26835.50	205.93	2.70
70	26776.81	233.30	2.48
80	26966.12	325.13	3.20
90	26841.62	205.05	2.72
100	26801.49	197.21	2.57
110	26825.27	241.02	2.66
120	26714.21	125.24	2.24
130	26725.14	162.88	2.28
140	26760.35	165.55	2.41
150	26763.14	157.57	2.42
160	26817.75	199.53	2.63
170	26853.57	190.64	2.77
180	26804.85	175.04	2.58
190	26833.20	193.97	2.69
200	26884.30	200.10	2.89
210	26845.76	146.88	2.74
220	26880.65	134.48	2.87
230	26881.87	190.82	2.88
240	26913.85	161.44	3.00
250	26846.96	129.83	2.74
260	26971.07	183.25	3.22
270	26928.55	143.85	3.06
280	26974.95	194.46	3.23
290	27027.08	228.64	3.43
300	26986.31	191.51	3.28

Table 3
ST analyses of DTSA on KROB150 TSP.

ST	Mean	Std.Dev.	Mean Error
0.1	26865.36	194.90	2.81
0.2	26854.99	208.38	2.77
0.3	26830.46	161.16	2.68
0.4	26794.43	174.89	2.54
0.5	26792.15	218.49	2.53
0.6	26719.09	134.27	2.25
0.7	26795.28	160.74	2.55
0.8	26760.29	184.34	2.41
0.9	26717.41	151.97	2.25

5.5. Experiment 5: comparisons with ABC variants

In [9] work, ABC integrated with neighborhood operators for solving TSPs. Neighborhood operators are similar to transformation operators. RS, RSS, and RRSS are similar to swap transformation operator, RI and RIS are similar to shift transformation operator. RRS and RRIS operators are similar to symmetry transformation operator. Also, these operators grouped and named as Combined 1 (RS, RSS, and RRSS) and Combined 2 (RI, RIS, and RRIS) in [9]. OLIVER30, EIL51, BERLIN52, ST70, PR76, KROA100, EIL101, TSP225, A280 are used as the benchmark functions for experimental results in [9] work. For a fair comparison, termination criterion (the maximum function evaluation number) is set at ($D \times 100{,}000$). D is the number of cities of the test problem. The algorithmic parameters of ABC and its variants are population size and limit and these are set at D and DxDx100 respectively. The obtained results are shown in Tables 6–14. In these tables, 2-OPT means ABC with 2-opt and 3-OPT means ABC with 3-opt. If we make an overall analysis, DTSA produces more successful results when dimension of the problems increases. For BERLIN52, KROA100, TSP225 and A280 problems, DTSA has produced better results than the other compared algorithms. For the other six problems, DTSA produced competitive solutions.

5.6. Experiment 6: comparisons with ACO, ABC, HA

Hierarchic approach (HA) is an algorithm which consists of both ABC and ACO proposed by Gündüz et al.[11].We compared DTSA with the HA, ACO, ABC. For a fair comparison, termination criterion (the maximum function evaluation number) is set at ($D \times 500$). D is the number of cities of the test problem. The algorithmic parameters of ABC are population size and limit and these are set at D and DxDx500 respectively. The values of specific parameters of ACO are $\alpha = 1$, $\beta = 5$, $\rho = 0.65$. The number of ants is set at D and total maximum of iterations is set at 500 for ACO. HA means 50% ABC and 50% ACO so the parameter settings are same, only the population sizes are half of D. The results of ABC, ACO and HA are directly taken from [11]. The results reported in Table 15.As seen from the Fig. 7, DTSA is a yet another good solver for these ten TSPs. HA is the leader because HA was used problem and population knowledge together but DTSA do not use problem knowledge so DTSA creates competitive solutions but not creates the best solution for all problems. According to the Table 15 and Fig. 7, DTSA is better than ACO, ABC but worse than HA. Using the problem

Table 4
Comparison of DTSA with SA, ACO and STA.

Problem	Performance	SA	ACO	STA	DTSA
BERLIN52	Best	8,186.40	8,240.40	7,544.40	**7,542.00**
	Mean	8,983.80	8,777.60	8,247.20	**7,689.17**
	Worse	9,585.80	9,151.30	8,630.50	**7,929.00**
	Std.Dev.	380.10	267.11	273.45	**108.40**

Table 5
Comparison of DTSA with SA and DSTA variants.

Problem	Optimum	Algorithm	Mean	Std.Dev.	Mean Error
KROA100	21,282	SA	22635.00	778.72	6.36
		DSTA0	23213.00	906.11	9.07
		DSTAI	22835.00	715.85	7.30
		DSTAII	21767.00	221.64	2.28
		DTSA	**21506.78**	**260.55**	**1.06**
KROB100	22,141	SA	23657.00	445.78	6.85
		DSTA0	23794.00	517.05	7.47
		DSTAI	23734.00	507.38	7.19
		DSTAII	**22880.00**	**302.14**	**3.34**
		DTSA	23139.26	181.74	4.51
KROC100	20,749	SA	22223.00	522.20	7.10
		DSTA0	22877.00	709.87	10.26
		DSTAI	21891.00	536.88	5.50
		DSTAII	**21378.00**	**246.34**	**3.03**
		DTSA	21817.08	217.77	5.15
KROD100	21,294	SA	22911.00	483.01	7.59
		DSTA0	23043.00	565.80	8.21
		DSTAI	22665.00	592.53	6.44
		DSTAII	**21991.00**	**315.32**	**3.27**
		DTSA	22972.26	390.50	7.88
KROE100	22,068	SA	23125.00	389.42	4.44
		DSTA0	23738.00	450.82	7.21
		DSTAI	23371.00	678.69	5.56
		DSTAII	22637.00	166.82	2.24
		DTSA	**22547.00**	**121.96**	**1.83**

Table 6
Comparison of DTSA with ABC variants for OLIVER30 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	477.86	477.86	476.37	**426.74**
RSS	**423.74**	**423.74**	**423.74**	426.74
RI	447.36	444.43	444.58	**426.74**
RIS	**423.88**	**423.88**	**423.88**	426.74
RR	425.16	425.16	**424.91**	426.74
RRIS	**423.74**	**423.74**	**423.74**	426.74
RRSS	**423.74**	**423.74**	**423.74**	426.74
Combined 1	**423.74**	**423.74**	**423.74**	426.74
Combined 2	**423.74**	**423.74**	**423.74**	426.74

Table 7
Comparison of DTSA with ABC variants for EIL51 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	506.32	506.32	504.45	**438.25**
RSS	431.11	**431.03**	431.11	438.25
RI	467.62	463.28	465.03	**438.25**
RIS	435.78	**435.77**	435.78	438.25
RR	440.00	440.00	439.92	**438.25**
RRIS	430.41	**430.37**	430.41	438.25
RRSS	**430.25**	**430.25**	**430.25**	438.25
Combined 1	**430.08**	**430.08**	**430.08**	438.25
Combined 2	**430.55**	**430.55**	**430.55**	438.25

knowledge slows down the algorithm, but increase the solution quality. It is a tradeoff between quality and time. Thus, we do not prefer the usage of problem knowledge so DTSA is the second best solver in this experiment.

5.7. Experiment 7: comparisons with ACO, PSO, GA, BH

In 2017, BH is proposed for solving TSPs in [28] work and compared with ACO, PSO, and GA. In [28], ULYSSES22, BAYS29, BAYG29, ATT48, EIL51, BERLIN52, ST70, EIL76, GR96, and EIL101 problems are solved but when we examined the results, a mistake appears in the obtained results. The optimum of ULYSSES22 is 7013 but in this work, Hatamlou found it about 75 because of the false calculation of objective function. Coordinate of the ULYSSES22 problem is GEO type but in this work, it calculated as EUC type as in BAYS29 (2020), BAYG29 (1610), ATT48 (10628), and GR96 (55209) problems. Therefore, only EIL51, BERLIN52, ST70, EIL76, and EIL101 problems used for comparison as seen in

Table 8
Comparison of DTSA with ABC variants for BERLIN52 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	9,177.02	9,177.02	9,117.92	**7,544.37**
RSS	7,562.80	7,562.80	7,562.80	**7,544.37**
RI	8,470.14	8,395.98	8,401.40	**7,544.37**
RIS	7,591.43	7,591.43	7,591.43	**7,544.37**
RR	7,774.77	7,774.77	7,763.02	**7,544.37**
RRIS	**7,544.37**	**7,544.37**	**7,544.37**	**7,544.37**
RRSS	**7,544.37**	**7,544.37**	**7,544.37**	**7,544.37**
Combined 1	7,548.47	7,547.07	7,548.47	**7,544.37**
Combined 2	**7,544.37**	**7,544.37**	**7,544.37**	**7,544.37**

Table 9
Comparison of DTSA with ABC variants for ST70 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	927.69	927.69	922.84	**688.77**
RSS	688.59	**687.15**	688.46	688.77
RI	785.14	773.34	777.03	**688.77**
RIS	690.04	**687.87**	690.03	688.77
RR	695.17	695.17	693.84	**688.77**
RRIS	681.52	**680.85**	681.32	688.77
RRSS	683.18	**682.55**	683.18	688.77
Combined 1	684.17	684.17	**684.05**	688.77
Combined 2	681.60	**680.88**	681.56	688.77

Table 10
Comparison of DTSA with ABC variants for PR76 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	147206.79	147206.79	146809.12	**112747.33**
RSS	110106.91	**109871.01**	110106.91	112747.33
RI	126341.76	124429.93	124738.89	**112747.33**
RIS	110397.12	**110005.95**	110356.60	112747.33
RR	109817.41	109817.41	**109787.82**	112747.33
RRIS	109005.00	**108830.04**	108965.31	112747.33
RRSS	108911.08	**108848.04**	108895.81	112747.33
Combined 1	109164.47	109164.47	**109164.13**	112747.33
Combined 2	108668.29	108642.94	**108624.77**	112747.33

Table 11
Comparison of DTSA with ABC variants for KROA100 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	33761.06	33761.06	33437.72	**21386.20**
RSS	22663.10	22081.42	22560.99	**21386.20**
RI	26204.50	25856.45	25959.69	**21386.20**
RIS	22490.43	21997.55	22331.84	**21386.20**
RR	21845.07	21845.07	21834.45	**21386.20**
RRIS	22080.16	21688.40	22013.77	**21386.20**
RRSS	21825.11	21687.13	21800.82	**21386.20**
Combined 1	21759.41	21747.45	21749.99	**21386.20**
Combined 2	21521.00	21506.96	21493.58	**21386.20**

Table 16. DTSA has executed 5 different runs with 20,000 function evaluations as in related work. The stand size is taken as 100 and ST is taken as 0.5. The values of specific parameters of ACO are α = 1.5, β = 2, ρ = 0.7. The number of ants is set at 100 and total maximum of iterations is set at 200 for ACO. For PSO algorithm social and cognitive constants c1 and c2 are set at 2. The inertia weight is taken as 0.9, the maximum of velocity is taken as 100 and dimension of space as taken as 10. In GA, crossover percentage,

Table 12
Comparison of DTSA with ABC variants for EIL101 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	837.37	837.37	829.74	**670.51**
RSS	683.64	675.86	681.99	**670.51**
RI	730.02	724.38	726.53	**670.51**
RIS	681.49	674.82	679.90	**670.51**
RR	676.34	676.34	674.95	**670.51**
RRIS	670.82	**664.86**	669.15	670.51
RRSS	668.05	**665.35**	666.96	670.51
Combined 1	667.80	667.58	**667.27**	670.51
Combined 2	661.25	660.96	**660.92**	670.51

Table 13
Comparison of DTSA with ABC variants for TSP225 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	7264.12	7219.30	7142.35	**3974.49**
RSS	5578.45	4866.89	5069.09	**3974.49**
RI	5030.64	4916.51	4952.39	**3974.49**
RIS	5273.92	4720.48	4903.77	**3974.49**
RR	4183.45	4170.36	4177.70	**3974.49**
RRIS	5242.72	4693.40	4875.97	**3974.49**
RRSS	5165.84	4666.68	4835.13	**3974.49**
Combined 1	4741.70	4624.32	4672.41	**3974.49**
Combined 2	4439.00	4337.75	4404.18	**3974.49**

Table 14
Comparison of DTSA with ABC variants for A280 TSP.

	Average			
	ABC	2-OPT	3-OPT	DTSA
RS	5759.04	5642.34	5611.08	**2686.87**
RSS	4416.36	3685.15	3849.63	**2686.87**
RI	3674.23	3570.72	3574.11	**2686.87**
RIS	4136.35	3538.74	3714.04	**2686.87**
RR	2912.31	2878.30	2894.80	**2686.87**
RRIS	4082.35	3452.81	3642.13	**2686.87**
RRSS	4156.83	3570.00	3707.45	**2686.87**
Combined 1	3584.49	3407.18	3497.76	**2686.87**
Combined 2	3322.47	3205.48	3267.81	**2686.87**

mutation percentage and mutation rate are set at 0.8, 0.3, and 0.02 respectively. The population size is set at 100 and total maximum of iterations is set at 200 for BH. The comparisons show that, DTSA had the best performance in 4 of 5 problems.

6. Conclusions

Optimization problems are divided into two main groups according to the decision variable types. These are continuous and discrete optimization. TSA is a newly proposed metaheuristic optimization algorithm for solving continuous optimization problems. In the literature, there is no discrete version of TSA, so in this study, we developed a discrete version of TSA and named as DTSA. This discretization process is made with transformation operators. Swap, shift and symmetry transformation operators are used for creating new solutions. Additionally, for improving the solution, the 2-opt local search algorithm is used. For analyzing the success of this approach, we used symmetric traveling salesman problems. These problems names are ULYSSESS16, OLIVER30, EIL51, BERLIN52, ST70, PR76, KROA100, KROB100, KROC100, KROD100, KROE100, EIL101, KROB150, TSP225, and A280. The results of our algorithm are compared with ACO, SA, BH, GA, STA, ABC, PSO and

Table 15
Comparison of DTSA with ACO, ABC, HA.

Problem	Method	Mean	Std.Dev.	RE	maxfes	Rank
OLIVER30	ACO	424.68	1.41	0.22	15,000	3
	ABC	462.55	12.47	9.16	15,000	4
	HA	**423.74**	0.00	**0.00**	15,000	1
	DTSA	428.50	4.21	1.12	15,000	2
EIL51	ACO	457.86	4.07	6.76	25,500	3
	ABC	590.49	15.79	37.69	25,500	4
	HA	**443.39**	5.25	**3.39**	25,500	1
	DTSA	443.93	4.04	3.51	25,500	2
BERLIN52	ACO	7659.31	38.70	1.52	26,000	3
	ABC	10390.26	439.69	37.72	26,000	4
	HA	**7544.37**	0.00	**0.00**	26,000	1
	DTSA	7545.83	21.00	0.02	26,000	2
ST70	ACO	709.16	8.27	4.73	35,000	3
	ABC	1230.49	41.79	81.73	35,000	4
	HA	**700.58**	7.51	**3.47**	35,000	1
	DTSA	708.65	6.77	4.66	35,000	2
EIL76	ACO	561.98	3.50	3.04	38,000	2
	ABC	931.44	24.86	70.78	38,000	4
	HA	**557.98**	4.10	**2.31**	38,000	1
	DTSA	578.58	3.93	6.09	38,000	3
PR76	ACO	116321.22	885.79	7.55	38,000	3
	ABC	205119.61	7379.16	89.65	38,000	4
	HA	115072.29	742.90	6.39	38,000	2
	DTSA	**114930.03**	1545.64	**6.26**	38,000	1
KROA100	ACO	22880.12	235.18	7.49	50,000	3
	ABC	53840.03	2198.36	152.94	50,000	4
	HA	22435.31	231.34	5.40	50,000	2
	DTSA	**21728.40**	358.13	**2.08**	50,000	1
EIL101	ACO	693.42	6.80	7.96	50,500	3
	ABC	1315.95	35.28	104.88	50,500	4
	HA	**683.39**	6.56	**6.40**	50,500	1
	DTSA	689.91	4.47	7.41	50,500	2
CH150	ACO	6702.87	20.73	2.61	75,000	2
	ABC	21617.48	453.71	230.93	75,000	4
	HA	**6677.12**	19.30	**2.22**	75,000	1
	DTSA	6748.99	32.63	3.32	75,000	3
TSP225	ACO	4176.08	28.34	8.22	112,500	2
	ABC	17955.12	387.35	365.28	112,500	4
	HA	**4157.85**	26.27	**7.74**	112,500	1
	DTSA	4230.45	58.76	9.63	112,500	3

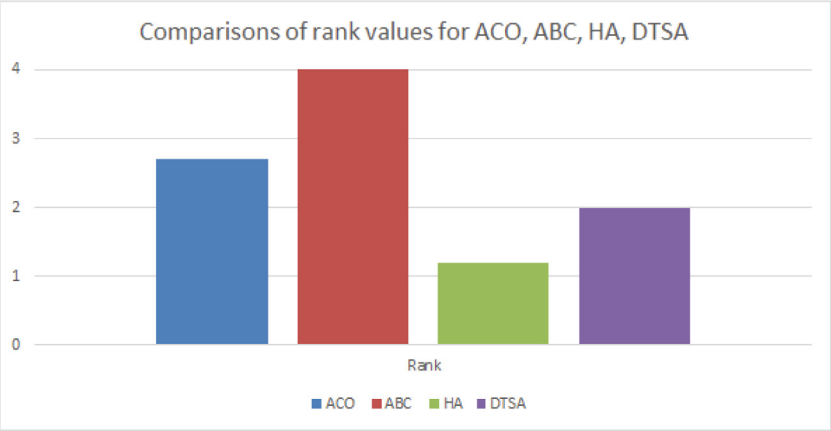

Fig. 7. Comparisons of mean rank values for ACO, ABC, HA and DTSA.

some of their variants. Experimental results confirmed that our approach is another qualified and competitor solver for symmetric traveling salesman problems.

In the future, this approach may be used for solving asymmetric and generalized traveling salesman problems. Also, discrete real-world problems may be solved with this approach. In the near

Table 16
Comparison of DTSA with ACO, PSO, GA, BH.

	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
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**
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**
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**
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**

future, we are thinking about to apply this approach to the job shop scheduling problems.

Acknowledgments

The authors wish to thank Scientific Research Projects Coordinatorship at Selcuk University and Scientific Research Projects Coordinatorship at Konya Technical University for their institutional supports.

References

[1] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM (JACM) 45 (5) (1998) 753–782.
[2] C. Ravikumar, Parallel techniques for solving large scale travelling salesperson problems, Microprocess. Microsyst. 16 (3) (1992) 149–158.
[3] E.L. Lawler, D.E. Wood, Branch-and-bound methods: a survey, Oper. Res. 14 (4) (1966) 699–719.
[4] M. Padberg, G. Rinaldi, Optimization of a 532-city symmetric traveling salesman problem by branch and cut, Operat. Res. Lett. 6 (1) (1987) 1–7.
[5] C. Barnhart et al., Branch-and-price: column generation for solving huge integer programs, Oper. Res. 46 (3) (1998) 316–329.
[6] G. Laporte, Y. Nobert, A cutting planes algorithm for the m-salesmen problem, J. Operat. Res. Soc. (1980) 1017–1023.
[7] G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms, Eur. J. Oper. Res. 59 (2) (1992) 231–247.
[8] Görkemli, B., Karaboga, D. 2013. Quick Combinatorial Artificial Bee Colony-qCABC-Optimization Algorithm for TSP. in The Second International Symposium on Computing in Informatics and Mathematics (ISCIM 2013). Tiran, Albania.
[9] M.S. Kıran, H. İşcan, M. Gündüz, The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem, Neural Comput. Appl. 23 (1) (2013) 9–21.
[10] Li, L. et al. 2011. A discrete artificial bee colony algorithm for TSP problem. In: International Conference on Intelligent Computing. Springer.
[11] M. Gündüz, M.S. Kiran, E. Özceylan, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. Turkish, J. Elect. Eng. Comput. Sci. 23 (1) (2015) 103–117.
[12] Karaboga, D., Gorkemli. B. 2011. A combinatorial artificial bee colony algorithm for traveling salesman problem. In: Innovations in intelligent systems and applications (inista), international symposium on. 2011. IEEE.
[13] Wang, K.-P., et al. 2003. Particle swarm optimization for traveling salesman problem. In: Machine Learning and Cybernetics. International Conference on. 2003. IEEE.
[14] Pang, W., et al. 2004. Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In: Computer and Information Technology. CIT'04. The Fourth International Conference on 2004. IEEE.
[15] 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.
[16] 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 (12) (2011) 14439–14450.
[17] X.H. Shi et al., Particle swarm optimization-based algorithms for TSP and generalized TSP, Informat. Process. Lett. 103 (5) (2007) 169–176.
[18] W. Deng et al., A novel two-stage hybrid swarm intelligence optimization algorithm and application, Soft. Comput. 16 (10) (2012) 1707–1722.
[19] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[20] W. Yong, Hybrid Max-Min ant system with four vertices and three lines inequality for traveling salesman problem, Soft. Comput. 19 (3) (2015) 585–596.
[21] Ş. Gülcü et al., A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem, Soft. Comput. (2016) 1–17.
[22] H. Eldem, E. Ülker, The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere, Eng. Sci. Technol. Int. J. 20 (4) (2017) 1242–1248.
[23] J.-Y. Potvin, State-of-the-art survey—the traveling salesman problem: a neural network perspective, ORSA J. Comput. 5 (4) (1993) 328–348.
[24] J. Knox, Tabu search performance on the symmetric traveling salesman problem, Comput. Oper. Res. 21 (8) (1994) 867–876.
[25] A. Baykasoğlu et al., YAPAY BAĞIŞIKLIK SİSTEMİNİN ÇOKLU ETMEN BENZETİM ORTAMINDA REALİZE EDİLMESİ VE GEZGİN SATICI PROBLEMİNE UYGULANMASI, Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi 27 (4) (2013).
[26] A. Ouaarab, B. Ahiod, X.-S. Yang, Discrete cuckoo search algorithm for the travelling salesman problem, Neu. Comput. Appl. 24 (7–8) (2014) 1659–1669.
[27] A. Ouaarab, B. Ahiod, X.-S. Yang, Random-key cuckoo search for the travelling salesman problem, Soft. Comput. 19 (4) (2015) 1099–1106.
[28] A. Hatamlou, Solving travelling salesman problem using black hole algorithm, Soft. Comput. (2017) 1–9.
[29] X. Zhou et al., Discrete state transition algorithm for unconstrained integer optimization problems, Neurocomputing 173 (2016) 864–874.
[30] Chunhua, Y., et al. State transition algorithm for traveling salesman problem. in Control Conference (CCC), 2012 31st Chinese. 2012. IEEE.
[31] H. Iscan, M. Gunduz, An application of fruit fly optimization algorithm for traveling salesman problem, Procedia Comput. Sci. 111 (Supplement C) (2017) 58–63.
[32] M.-H. Chen, S.-H. Chen, P.-C. Chang, Imperial competitive algorithm with policy learning for the traveling salesman problem, Soft. Comput. 21 (7) (2017) 1863–1875.
[33] X. Feng et al., Physarum-energy optimization algorithm, Soft. Comput. (2017) 1–18.
[34] J.-Y. Potvin, Genetic algorithms for the traveling salesman problem, Ann. Oper. Res. 63 (3) (1996) 337–370.
[35] G. Üçoluk, Genetic algorithm solution of the TSP avoiding special crossover and mutation, Intellig. Automat. Soft Comput. 8 (3) (2002) 265–272.
[36] H. Braun, On solving travelling salesman problems by genetic algorithms. in International Conference on Parallel Problem Solving from Nature, Springer, 1990.
[37] K. Bryant, A. Benjamin, Genetic algorithms and the traveling salesman problem, Department of Mathematics, Harvey Mudd College, 2000, pp. 10–12.
[38] J. Grefenstette et al., Genetic algorithms for the traveling salesman problem. in Proceedings of the first International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, New Jersey, 1985, pp. 160–168.
[39] A. Hussain et al., Genetic algorithm for traveling salesman problem with modified cycle crossover operator, Computat. Intellig. Neurosci. 2017 (2017).
[40] Jubeir, M.B., Almazrooie, M., Abdullah. R. 2017. Enhanced selection method for genetic algorithm to solve traveling salesman problem. In: Proceedings of the 6th International Conference of Computing & Informatics. Kuala Lumpur
[41] P. Larranaga et al., Genetic algorithms for the travelling salesman problem: a review of representations and operators, Artif. Intell. Rev. 13 (2) (1999) 129–170.
[42] Z.H. Ahmed, Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, Int. J. Biomet. Bioinformat. (IJBB) 3 (6) (2010) 96.
[43] M. Albayrak, N. Allahverdi, Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms, Expert Syst. Appl. 38 (3) (2011) 1313–1320.
[44] H. Mühlenbein, Parallel genetic algorithms, population genetics and combinatorial optimization. In: Workshop on Parallel Processing: Logic, Organization, and Technology, Springer, 1989.
[45] I. Oliver, D. Smith, J.R. Holland, Study of permutation crossover operators on the traveling salesman problem, Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA, L. Erlhaum Associates, Hillsdale, NJ, 1987.
[46] Starkweather, T., et al. A Comparison of Genetic Sequencing Operators. In: ICGA. 1991.
[47] N.L. Ulder et al., Genetic local search algorithms for the traveling salesman problem. in International Conference on Parallel Problem Solving from Nature, Springer, 1990.

[48] Whitely, D., Starkweather, T. Fuquay, D. Scheduling problems and traveling salesman: The genetic edge recombination operator. In: Proc. 3rd Int. Conf. Genetic Algorithms. 1989.
[49] D. Whitley, T. Starkweather, D. Shaner, The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination, Colorado State University, Department of Computer Science, 1991.
[50] H.-K. Tsai et al., Some issues of designing genetic algorithms for traveling salesman problems, Soft. Comput. 8 (10) (2004) 689–697.
[51] G.A. Croes, A method for solving traveling-salesman problems, Oper. Res. 6 (6) (1958) 791–812.
[52] D.E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem. in Proceedings of an international conference on genetic algorithms and their applications, Lawrence Erlbaum, Hillsdale, NJ, 1985.
[53] D.B. Fogel, An evolutionary approach to the traveling salesman problem, Biol. Cybern. 60 (2) (1988) 139–144.
[54] Martin, O., Otto, S.W. Felten, E.W. 1991. Large-step Markov chains for the traveling salesman problem.
[55] A.E. Yildirim, A. Karci, Applications of artificial atom algorithm to small-scale traveling salesman problems, Soft. Comput. (2017) 1–13.
[56] L. Davis, Applying adaptive algorithms to epistatic domains, IJCAI., 1985.
[57] G. Syswerda, Scheduling optimization using genetic algorithms, Handbook Genet. Algorith. (1991) 322–349.
[58] J.J. Grefenstette, Incorporating problem specific knowledge in genetic algorithms, Genet. Algorith. Simulat. Anneal. (1987) 42–60.
[59] R. Brady, Optimization strategies gleaned from biological evolution, Nature 317 (6040) (1985) 804–806.
[60] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, Evolution algorithms in combinatorial optimization, Parallel Comput. 7 (1) (1988) 65–85.
[61] P. Larrañaga et al., Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms, Statist. Comput. 7 (1) (1997) 19–34.
[62] W. Banzhaf, The "molecular" traveling salesman, Biol. Cybern. 64 (1) (1990) 7–14.
[63] Z. Michalewicz, Genetic Algorithms+ Data Structures= Evolution Programs, Springer Science & Business Media, 1992.
[64] D.B. Fogel, Applying evolutionary programming to selected traveling salesman problems, Cybernet. Syst. 24 (1) (1993) 27–36.
[65] Or, I., TRAVELING SALESMAN TYPE COMBINATORIAL PROBLEMS AND THEIR RELATION TO THE LOGISTICS OF REGIONAL BLOOD BANKING. 1976, Northwestern University.
[66] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (2) (1973) 498–516.
[67] M.S. Kiran, TSA: Tree-seed algorithm for continuous optimization, Expert Syst. Appl. 42 (19) (2015) 6686–6698.
[68] Cinar, A.C., Kiran, M.S. A Parallel Version of Tree-Seed Algorithm (TSA) within CUDA Platform. In: Selçuk ISCAS (International Scientific Conference On Applied Sciences) 2016. 2016.
[69] A.C. Cinar, A Cuda-based Parallel Programming Approach to Tree-Seed Algorithm MSc Thesis, Graduate School of Natural Sciences, Selcuk University, Selcuk University, 2016.
[70] M.S. Kıran, An Implementation of Tree-Seed Algorithm (TSA) for Constrained Optimization, in Intelligent and Evolutionary Systems, Springer, 2016, pp. 189–197.
[71] Cinar, A.C., Kiran, M.S. 2017. Boundary conditions in Tree-Seed Algorithm: Analysis of the success of search space limitation techniques in Tree-Seed Algorithm. International Conference on Computer Science and Engineering (UBMK). 2017.
[72] A.A. El-Fergany, H.M. Hasanien, Tree-seed algorithm for solving optimal power flow problem in large-scale power systems incorporating validations and comparisons, Appl. Soft Comput. (2017).
[73] V. Muneeswaran, M.P. Rajasekaran, Beltrami-Regularized Denoising Filter Based on Tree Seed Optimization Algorithm: An Ultrasound Image Application. in International Conference on Information and Communication Technology for Intelligent Systems, Springer, 2017.
[74] M.S. Kiran, Withering process for tree-seed algorithm, Procedia Comput. Sci. 111 (Supplement C) (2017) 46–51.
[75] W.J. Chen, X.J. Tan, M. Cai, Parameter identification of equivalent circuit models for Li-ion batteries based on tree seeds algorithm, IOP Conference Series: Earth and Environ. Sci. 73 (1) (2017). 012024.
[76] A. Babalik, A.C. Cinar, M.S. Kiran, A modification of tree-seed algorithm using Deb's rules for constrained optimization, Appl. Soft Comput. 63 (Supplement C) (2018) 289–305.
[77] A.C. Cinar, M.S. Kiran, Similarity and logic gate-based tree-seed algorithms for binary optimization, Comput. Ind. Eng. 115 (2018) 631–646.
[78] A.C. Cinar, H. Iscan, M.S. Kiran, Tree-seed algorithm for large-scale binary optimization, KnE Social Sci. (2018) 48–64.
[79] M.A. Sahman, A.C. Cinar, Binary tree-seed algorithms with S-shaped and V-shaped transfer functions, Int. J. Intellig. Syst. Appl. Eng. 7 (2) (2019) 111–117.
[80] Muneeswaran, V., Rajasekaran, M.P. 2016. Performance evaluation of radial basis function networks based on tree seed algorithm. In: 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT). IEEE.
[81] Y. Zheng et al., Design of a multi-mode intelligent model predictive control strategy for hydroelectric generating unit, Neurocomputing 207 (2016) 287–299.
[82] V. Muneeswaran, M.P. Rajasekaran, Gallbladder shape estimation using tree-seed optimization tuned radial basis function network for assessment of acute cholecystitis, in Intelligent Engineering Informatics, Springer, 2018, pp. 229–239.
[83] J. Zhou et al., A heuristic TS fuzzy model for the pumped-storage generator-motor using variable-length tree-seed algorithm-based competitive agglomeration, Energies 11 (4) (2018) 944.
[84] Z. Ding et al., Structural damage identification with uncertain modelling error and measurement noise by clustering based tree seeds algorithm, Eng. Struct. 185 (2019) 301–314.
[85] Jiang, J. et al., 2019. EST-TSA: An effective search tendency based to tree seed algorithm. Physica A: Statistical Mechanics and its Applications, pp. 122323.
[86] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384.