Discrete Teaching–learning-based optimization Algorithm for Traveling Salesman Problems

Lehui Wu, Feng Zou[a], Debao Chen
School of Physics and Electronic Information, HuaiBei Normal University, Huaibei, 235000, China

Abstract. In this paper, a discrete variant of TLBO (DTLBO) is proposed for solving the traveling salesman problem (TSP). In the proposed method, an effective learner representation scheme is redefined based on the characteristics of TSP problem. Moreover, all learners are randomly divided into several sub-swarms with equal amounts of learners so as to increase the diversity of population and reduce the probability of being trapped in local optimum. In each sub-swarm, the new positions of learners in the teaching phase and the learning phase are generated by the crossover operation, the legality detection and mutation operation, and then the offspring learners are determined based on greedy selection. Finally, to verify the performance of the proposed algorithm, benchmark TSP problems are examined and the results indicate that DTLBO is effective compared with other algorithms used for TSP problems.

1 Introduction

The traveling salesman problem (TSP) is the most used and famous optimization problem within the kind of combinatorial optimizations [1]. The main aim of TSP is to obtain the shortest tour that starts from one city and visits each of the other cities once before returning to the starting city. This is one of the most fundamental NP-complete optimization problems [2]. Since the TSP problem belongs to the class of NP complete problems, its solution grows exponentially with the increase in distribution points. Thus, exact algorithms [3-4] are not capable of solving problems with large dimensions. On the other hand, heuristics [5-6] are thought to be more preferable and efficient for a complex TSP problem and have become very popular with researchers.

Although heuristic methods can be used to solve complex TSP problems, they are easy to be trapped into local optimum and cannot obtain the optimum solution. As a result, a new kind of optimization algorithms named meta-heuristics has been proposed in recent years and these relevant meta-heuristics have been proposed to solve complex problems [7-8]. In contrast to heuristic algorithms, meta-heuristic algorithms make use of the randomization of the search space, effectively explore and exploit some promising solution regions. Hence, meta-heuristic algorithms have many advantages of implementation simplicity, adaptability, flexibility and robustness, and the algorithms can obtain a more better or near-optimal solution of higher quality in relatively low time. Here many meta-heuristic algorithms have been developed to solve complex TSP problems [9-10].

Recently, a meta-heuristic algorithm called Teaching–learning-based optimization algorithm (TLBO) has been developed [11]. TLBO and its variants have shown significant performance for combinatorial optimization problems [12, 13]. In this paper, we present a new discrete version of TLBO to solve TSP problems. In the proposed method, the learner representation scheme is redefined based on the characteristics of TSP problem, and the updating rules for learners are also redesigned. Finally, to verify that our proposed DTLBO algorithm is a promising method, 7 different TSP problems are examined. The results indicate that DTLBO is effective compared with other algorithms used for TSP problems.

2 Teaching–learning-based optimization

Teaching–learning-based optimization algorithm (TLBO) is a population-based optimization algorithm inspired from the philosophy of teaching and learning. The learners correspond to the individuals of evolutionary algorithms. The teacher is the learner with the best fitness and she/he can share her/his knowledge to all the learners, thus enhancing the average grade of the class in teaching phase. Each learner also learns knowledge from another randomly selected learner in the class, thus enhancing the average grade of the class in learning phase.

2.1 Learner initialization

In the initial phase, all learners in the class are randomly initialized. The *i*-th learner $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ in the class can be generated as follows.

$$X_i = X_i^{\min} + rand \square (X_i^{\max} - X_i^{\min}) \quad (1)$$

where $X_i^{\min}$ and $X_i^{\max}$ are the lower and upper bounds of the control variables respectively, *rand* is a random vector within the range [0, 1].

[a] Corresponding author: Feng Zou zfemail@163.com

2.2 Teaching phase

During the teaching phase, the task of learning is to increase the mean marks of the class. In this phase, all learners update their positions according to the difference between the current teacher and the mean position of the class. The updating formula of the *i*-th learner X_i in the class is expressed as follows.

$$newX_i = X_i + r(Teacher - TF \cdot Mean) \quad (2)$$

where, $newX_i$ is the new positions of the *i*-th learner X_i, *Teacher* is the best learner of the class and $Mean = \frac{1}{NP}\sum_{i=1}^{NP} X_i$ is the mean position of the class. $TF = round[1 + rand(0,1)]$ is a teaching factor that decides the value of mean to be changed, the typical value of it is either 1 or 2. *r* is a random number in the range [0, 1].

2.3 Learning phase

Learners also can increase their knowledge by means of group discussions, presentations, formal communications, and so on. In this phase, each learner update their positions learns something new if the other learner has more knowledge than him or her. The updating formula of the *i*-th learner X_i in the class is expressed as follows.

$$newX_i = \begin{cases} X_i + r(X_i - X_k) & if \ \ f(X_i) < f(X_k) \\ X_i + r(X_k - X_i) & otherwise \end{cases} \quad (3)$$

where, $newX_i$ is the new positions of the *i*-th learner X_i, X_k is a randomly selected learner from the class, *r* is a random vector in the range [0, 1], $f(X_i)$ and $f(X_k)$ is the fitness value of the learner X_i and X_k, respectively.

After the new candidate learner has been generated in the teaching and learning phases, the greedy selection is executed to determine the offspring learners. That is, if the value $f(newX_i)$ is better than the value $f(X_i)$, then replace X_i with $newX_i$.

3 The proposed DTLBO

In this section, the discrete variant of TLBO is presented. First, initialize a population with *N* individuals based on a problem-specific strategy. Then, all learners are randomly divided into several sub-swarms with equal amounts of learners. Moreover, the best individual and the mean individual in each sub-swarm are determined, and the crossover operation, the legality detection and mutation operation are performed to generate a candidate learner for each learner in the teaching and learning phases. Finally, the learner is updated by performing a comparison of the current learner and the candidate learner. The above process ends when the termination criterion is met; otherwise, the next iterative process is performed. The detailed description of the DTLBO algorithm is as follows.

3.1 Individual representation and initialization

For a TSP problem, the goal is to obtain a city sequence of with the shortest path. Therefore, it is natural and intuitive to code the solution as a set of real integer values based on the number of cities, whose length is the same as the number of total cities and where each real integer value corresponds to a unique city number. For a TSP problem with 7 cities, the learner can be represented as shown in Figure 1.

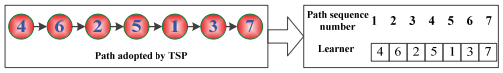


Figure 1. Individual representation of learners

Based on the above individual representation method, the learners of population can be generated by random initialization method. Assume that the population size is *NP*, the city size of the TSP problem is *N*, and then a matrix of $NP \times N$ is generated after initialization. The *i*-th row of the matrix corresponds to the *i*-th learner individual, and each column of the *i*-th row represents a city. The column order represents the order of visiting cities. Concretely, the initialization of the learner can be shown as follows:

$$X_i = randperm(N) \quad (4)$$

where the *randperm*(*N*) representation yields a random permutation of the integers from 1 to *N*.

3.2 The determination of teachers and mean individuals

In the original TLBO algorithm, the teacher tries to disseminate knowledge among learners, which in turn helps learners to achieve good grades and increases the knowledge level of the entire class, thus improving the mean results of the class. The updating of the learner is implemented based on the teacher and the mean individual. However, all learners learn from the random difference between the teacher and the mean individual, and this easily causes learners to become trapped in local optima. To avoid this situation, the multi-swarm mechanism is introduced to maintain the diversity of the population. That is, all learners are randomly divided into several sub-swarms with equal amounts of learners in each iterative process. Suppose that there are *K* (*K*=4 in this paper) learner individuals $X_{i_1}, X_{i_2}, \cdots, X_{i_K}$ in the *i*-th sub-swarm, and $NTeacher_i$ and $NMean_i$ are the best individual and the mean individual respectively in the *i*-th sub-swarm, which can be respectively given as follows:

$$NTeacher_i = Best(X_{i_1}, X_{i_2}, \cdots, X_{i_K}) \quad (5)$$

$$NMean_i = \left(\sum_{k=1}^{K} x_{i_k,1}, \sum_{k=1}^{K} x_{i_k,2}, \cdots, \sum_{k=1}^{K} x_{i_k,D} \right) \quad (6)$$

Different from the original TLBO algorithm, in our approach, $NTeacher_i$ and $NTeacher_i$ are all integer vectors in which each element corresponds to a unique city number. However, there are the same elements in the mean individual according to Eq.(6). For example, the mean individual may be expressed as follows:

$NMean_i$	3	6	6	5	4	3	4

Obviously, there are duplicate elements in the learner $NTeacher_i$, and they are illegal. Hence, the legality detection for the learner individual must be executed.

Generally, the executing process of the legality detection for the individual $NMean_i$ is given as follows:

Step 1: Count the number of times each city appears. For the above individual $NMean_i$, cities marked 3, 4 and 6 appears 2 times, and the city marked 5 appeared once. It can be expressed as a vector A. In the vector A, if a city appeared more than once, the number of times this city appears is written into the last appeared location of this city; otherwise, the other locations are set to 0.

Step 2: Determine the cities that did not appear. For the above individual $NMean_i$, cities marked 1, 2 and 7 have not appeared. It can be expressed as a vector B. In the vector B, the order number of a city that did not appear is written into the associated location of this order number, and the other locations are set to 0.

Step 3: Find out the last location of a city that has appeared more than once. In the vector C, if a city appeared more than once, the order number of this city is written into the last appeared location of this city; otherwise, the other locations are set to 0.

Step 4: Determine the location of cities that did not appear in the individual. For the above individual $NMean_i$, in the vector *C*, the order numbers of cities that did not appear is written into the locations with the values of zeroes, and then the final $NMean_i$ can be obtained.

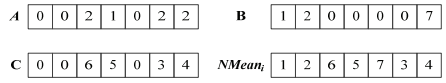


Figure 2. The legality detection for the learner individual

3.3 Teaching phase of DTLBO

In the original TLBO algorithm, the updating of the learner is implemented according to Eq.(2) and it is used to deal with continuous optimization problems. For discrete TSP problems, it needs to redefine the updating strategy of learners.

As a result, assume that there are four learners $X_{i_1}, X_{i_2}, X_{i_3}, X_{i_4}$ in the *i*-th sub-swarm. Their updating strategies can be given as follows:

$$TX_{i_1} = X_{i_1} \otimes Teacher \tag{7}$$

$$TX_{i_2} = X_{i_1} \otimes NTeacher_i \tag{8}$$

$$TX_{i_3} = X_{i_1} \otimes NMean_i \tag{9}$$

$$TX_{i_4} = NTeacher_i \otimes NMean_i \tag{10}$$

where $\otimes$ represents the crossover operation.

Generally, the crossover operation $\otimes$ can be described in detail as follows. Assume that there are two learners *A* and *B* shown as follows:

A = 2 3 |7 5 1| 4 6 B = 1 7 |4 6 5| 2 3

Then, randomly select two gene loci and exchange the corresponding elements between two gene loci in *A* and *B*:

AA = 2 3 |4 6 5| 4 6 BB = 1 7 |7 5 1| 2 3

Due to existing duplicate elements in *AA* and *BB*, they need to be revised by means of the legality detection to obtain *newA* and *newB* as follows:

newA = 2 3 |4 6 5| 7 1 *newB* = 6 4 |7 5 1| 2 3

After the crossover operation has been completed, the legality detection is executed so that the duplicate elements in the learner are revised. And then, the mutation operation is executed to generate the candidate learner shown as follows:

$$newX_i = \Theta TX_i \tag{11}$$

where Θ represents the mutation operation.

In this paper, the reverse mutation is performed and it is given as follows: randomly select two gene loci in the order of visiting cities, and then reverse the elements between two gene loci to generate the new mutation individual. For example, two gene loci in *newAA* is 3 and 5, the new mutation individual *newAA* can be obtained by the mutation operation:

newA = 2 |3 4 6 5| 7 1 *newAA* = 2 |5 6 4 3| 7 1

3.4 Learning Phase of DTLBO

In the learner phase of the original TLBO algorithm, the learner is updated according to the difference between the current learner X_i and the current random selected learner X_j. In the learning phase of the DTLBO algorithm, the current learner individual X_i is updated as follows:

$$TX_i = X_i \otimes X_j \tag{12}$$

where, $\otimes$ represents the crossover operation, and it is as same as the crossover operation in the teaching phase.

Just like the learning rules in the teaching phase, when the crossover operation is completed, the duplicate elements in the learner are revised, and then the mutation operation is executed to generate the candidate learner shown as follows:

$$newX_i = \Theta TX_i \tag{13}$$

where Θ represents the mutation operation.

Just like the original TLBO algorithm, the greedy selection is executed to determine the offspring learners. That is, after the candidate learner has been generated in the teaching and learning phases, if the value $f(newX_i)$ is better than the value $f(X_i)$, then replace X_i with $newX_i$.

4 DTLBO algorithm for solving TSP

In this section, in order to verify the validity and correctness of the algorithm, we have compared the effectiveness and efficiency of our proposed algorithm on 7 classic benchmark problems for the TSP in comparison with several other meta-heuristic algorithms.

4.1 Parameter setting

The algorithms are tested on several sample instances of TSPLIB [14] with different cities. Every algorithm requires appropriate parameter settings to get proper outcome. For all population-based algorithms, the size of population on the test problems is set to 100, and the maximum number of iteration on the test problems is set to 500 for fair comparison.

4.2 Comparison of simulation result

The simulated experimental results are listed in Table 1. In the table, KOS denotes the known optimal solution as

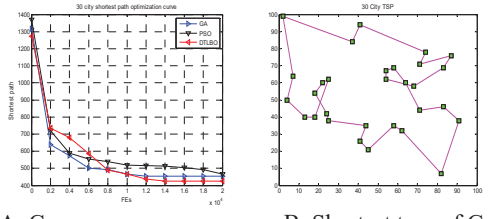
reported in TSPLIB. Best denotes the best solution found by each algorithm and Mean denotes the average value of all solutions by running independently 20 times.

Table 1. Comparison of test results

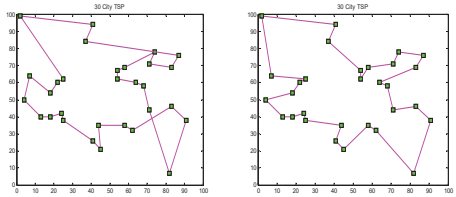
TSP	KOS	ACO[15]	ABC[16]	PSO	GA	DTLBO
Oliver30	420	**423.74**	432	464.71	455.45	**423.74**
Chn31	15377	15602	15942	17097	16646	**15397**
Att48	33522	33536	33541	37882	36458	33524
rand50	5553	—	—	6788	5848	5555
Eil51	426	435	426	563.03	460.48	434.67
rand75	7054	—	—	16544	8506	7170
Eil76	538	555.7	541	1083.91	685.59	565.21

As shown in Table 1, for Oliver30, DTLBO and ACO have the best performance in terms of the average value of 423.74 in all of 20 runs, and the average values obtained by ABC, GA and PSO are 432, 455.45 and 464.71 respectively. For Chn31, DTLBO obtains the average value of 15397, and those of ACO, ABC, PSO and GA are 15602, 15942, 17097 and 16646. It can be clearly seen that DTLBO is closer to the known optimal solution 15397 than the other four algorithms. For Att48, the average value obtained by DTLBO is 33524, while the other four algorithms are worse than the DTLBO. For Eil51 and Eil76, ABC has the best performance in terms of the average value and DTLBO is better than ACO, PSO and GA, While ABC can reach the known optimal solution of 426 for Eil51. For rand50 and rand75, DTLBO performs better than PSO and GA.

To demonstrate the effectiveness of the new DTLBO algorithm, the convergence curves and route tracing obtained by PSO, GA and DTLBO are given in Figure 3. However, we are only able to show convergence graphs for the proposed algorithm DTLBO, SPO and GA because the data of ACO and ABC were taken from literature and this information was unavailable. It can be concluded from Table 1 and Figure 3 that the DTLBO has a good performance for the most of all TSP problem instances in this paper.



A. Convergence curves B. Shortest tour of GA



C. Shortest tour of PSO D. Shortest tour of DTLBO

Figure 3. Oliver 30 city TSP optimization curve and path

5 Conclusion

In this paper, a new discrete teaching–learning-based optimization algorithm (DTLBO) is proposed for solving the traveling salesman problem. The proposed approach used multi-swarms mechanism to maintain the diversity of the population and reduce the probability of being trapped in local optimum and introduced the new crossover operation, legality detection and mutation operation into the teaching phase and the learning phase. The numerical simulation tests were carried out on 7 different TSP problems, and the results show that the DTLBO algorithm is a promising method for TSP problems. However, it can also be seen from the results that, the performance of DTLBO is also decreased with the increasing of city size. Hence, future work will concentrate on how to improve DTLBO's performance for TSP problems with large scale cities.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (Grant Nos.61572224, 41475017 and 11504121) and the National Science Fund for Distinguished Young Scholars (Grants No.61425009). This work is also partially supported by Anhui Provincial Natural Science Foundation (Grant No.1708085MF140), the Major Project of Natural Science Research in Anhui Province (Grant No.KJ2015ZD36) and the Natural Science Foundation in colleges and universities of Anhui Province (Grant No.KJ2016A639).

Reference

1. A. Philip, A. A. Taofiki, O. Kehinde. *International Journal of Advanced Computer Science & Applications*, 2(1): 26-29(2011).
2. M. Mavrovouniotis, F. M. Muller, S. Yang. *IEEE Transactions on Cybernetics*, 47(7):1743-1756(2016).
3. V. Mak, N. Boland. *Discrete Applied Mathematics*, 155(16):2093-2110(2007).
4. J. F. Cordeau, M. Dell'Amico, M. Iori. *Computers & Operations Research*, 37(5):970-980(2010).
5. W. Malika. *Operations Research Letters*, 35(6):747-753(2007).
6. A. I. Vakhutinsky, B. L. Golden. *IEEE World Congress on Computational Intelligence*. 7, 4535-4540(1994).
7. D. Simon. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6): 702-713(2008).
8. J. Knox. *Computers & Operations Research*, 21(8): 867-876(1994).
9. B. Akay, D. Karaboga. *Information Sciences*, 192(1): 120-142(2012).
10. A. Ouaarab, B. Ahiod, X. S. Yang. *Neural Computing & Applications*, 24(7-8):1659-1669(2014).
11. R. V. Rao, V. J. Savsani, D. P. Vakharia. *Information Sciences*, 183(1):1-15(2012).
12. A. Baykasoğlu, A. Hamzadayi, S. Y. Köse. *Information Sciences*, 2014, 276(C):204-218(2014).
13. X. Ji, H. Ye, J. Zhou, et al. *Applied Soft Computing*, 57, 504-516(2017).

14. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
15. Liu Yin, Ma Liang. *Application Research of Computers*, 9: 2694-2696(2013).
16. Zhonghua Hu, Min Zhao. *Transactions of Beijing Institute of Technology*, 11:978-982 (2009).