Intelligent Route Construction Algorithm for Solving Traveling Salesman Problem

Md. Azizur Rahman[1†], Ariful Islam[2††], and Lasker Ershad Ali[3†††]

Mathematics Discipline, Science, Engineering and Technology School, Khulna University, Khulna-9208, Bangladesh

Abstract
The traveling salesman problem (TSP) is one of the well-known and extensively studied NPC problems in combinatorial optimization. To solve it effectively and efficiently, various optimization algorithms have been developed by scientists and researchers. However, most optimization algorithms are designed based on the concept of improving route in the iterative improvement process so that the optimal solution can be finally found. In contrast, there have been relatively few algorithms to find the optimal solution using route construction mechanism. In this paper, we propose a route construction optimization algorithm to solve the symmetric TSP with the help of ratio value. The proposed algorithm starts with a set of sub-routes consisting of three cities, and then each good sub-route is enhanced step by step on both ends until feasible routes are formed. Before each subsequent expansion, a ratio value is adopted such that the good routes are retained. The experiments are conducted on a collection of benchmark symmetric TSP datasets to evaluate the algorithm. The experimental results demonstrate that the proposed algorithm produces the best-known optimal results in some cases, and performs better than some other route construction optimization algorithms in many symmetric TSP datasets.

Key words:
Combinatorial Optimization, Traveling Salesman Problem, Ratio Value, Constructive Algorithm.

1. Introduction

Combinatorial optimization problems are broadly studied in the fields of theoretical computer science, artificial intelligence, operations research, and discrete mathematics. The traveling salesman problem (TSP) is one of the most used problems in combinatorial optimization, specifically in transportation and distribution logistics. It is a problem of finding the shortest route among a set of cities that visits each city exactly once and finally returns to the starting city. The TSP is easy to understand but is inherently intractable. Indeed, it has been proven to be an NPC problem which means that no polynomial time algorithm exists to effectively solve it [1]. The TSP provides an ideal platform for the study of combinatorial optimization problems. It's not like that anyone would want to plan to visit some cities or places, rather many real life optimization problems can be formulated as TSP problem. Examples of such real world applications are the movement of people, drilling of printed circuit boards, vehicle routing and scheduling, postal delivery, computer wiring, control of robots, garbage collection, construction of smart cities, and machine scheduling, to quote a few. Thus, the study of solving TSP has not only academic interest, but also has important theoretical, engineering, and practical significance and, consequently, it has been an important research topic of active research.

In the graph theory, the TSP problem can be expressed by a weighted complete graph: $G=(V,E,W)$, where $V=\{v_1,v_2,\cdots,v_n\}$ denotes the vertices, which represents the nodes of the graph, $|V|=n$ is the total number of vertex, $E=\{e_{ij}\}(1\le i\ne j\le n)$ is the set of edges which represents the interconnections between the vertices or nodes, and $W:E\to\mathbb{R}^+$ is the weight function which is associated with each edge of the graph. The TSP problem may be symmetric or asymmetric. For symmetric TSP, the weight of each edge of the graph G is equal in both directions, i.e., $W(e_{ij})=W(e_{ji});\forall e_{ij}\in E$. On the other hand, in case of asymmetric TSP, there exists at least one edge $e_{ij}\in E$ in G for which $W(e_{ij})\ne W(e_{ji})$. In this paper, we concentrate on the solution of symmetric TSP problem. The objective of the TSP problem is to construct a feasible route with n distinct cities such that the total travel cost (distance) $f(Z)$ of the route is minimized. Let $d_{ij}=d(v_i,v_j)$ be the weight (distance) of the edge e_{ij} between the node v_i and v_j .Then, the mathematical model of TSP problem can be formulated as below [2]:

$$\left.\begin{aligned}&\text{Determine } x_{ij},\forall i,j\in V \,\&\, i\ne j \text{ to}\\&\text{Minimize } f(Z)=\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ij}d_{ij}\\&\text{subject to}\sum_{i=1}^{n}x_{ij}=1,\forall\, j\in V\\&\qquad\sum_{j=1}^{n}x_{ij}=1,\forall i\in V\end{aligned}\right\}\quad(1)$$

Manuscript received April 5, 2021
Manuscript revised April 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.4.5

where $x_{ij}=1$ if the salesman travel from node v_i to node v_j, otherwise $x_{ij}=0$. Let $(x_1, x_2, \cdots, x_n, x_1)$, $\forall$ distinct $x_i \in V, \forall i \in V$ be a feasible route and $f(Z)$ be its route cost (distance). Then, the optimization formulation Eq.(1). of TSP problem is reduced to the following formulation [3]:

$$\left.\begin{array}{l}\text{Generate a feasible route } (x_1, x_2, \cdots, x_n, x_1) \\ \text{to minimize } f(Z)=\sum_{i=1}^{n-1} d_{x_i x_{i+1}} + d_{x_n} d_{x_1}\end{array}\right\} \quad (2)$$

Due to the importance and applicability, many optimization algorithms have been proposed to deal with TSP problem. However, they do not guarantee for optimality, but provide an approximate solution near to the optimal solution. Indeed, it is not always true that the ultimate goal of a TSP solution is to explore an optimal solution [4, 5]. Rather, a good solution may be required instead of the optimal one in practical application. This practical solution may have some bounds such as a guarantee to be within a certain percentage of the optimal solution, for a certain percentage of the TSP instances [4, 6]. In this context, the heuristic based optimization algorithms are much more suited to handle this complex problem. The simplest and most easily implemented heuristic based optimization algorithms are route construction optimization algorithms. They follow well-defined rules to construct the route gradually [7]. The route is built only once during the running of the algorithm without making changes the part of route that is already built. The solution of these algorithms is usually not good but in some practical situations, they are useful. However, there have been relatively few route construction optimization algorithms in the literature for solving the TSP problem.

Motivated by the insufficiency of route construction algorithm, this paper takes an attempt to solve the symmetric TSP by proposing a route construction optimization algorithm with the help of ratio value. The concept of ratio value was introduced by Rahman and Ma to solve routing problems [8-10]. They adopt a ratio value in the construction process in such a way that an initial ratio value is set up in the first step, and then it is decreased with time in the following steps of the process. As a result, the search engine may not be able to retain enough routes in each step, even retains one single route in some cases. In this paper, we set up a ratio value in the first step, and then the algorithm dynamically adjusts it in the rest of the steps during the optimization process. The remainder of this paper is organized as follows; Section 2 reviews the related literature to solve the TSP problem. Section 3 elaborates the proposed route construction technique. Section 4 presents experimental results and analysis that include the experiment set up, description of tested datasets, description and analysis of the simulated results and comparison of the simulated results with other route construction algorithms. Finally, Section 5 concludes the paper.

2. Related Literature

The route construction algorithms for the TSP generate a feasible route step by step by using various strategies. They start with a sub-route or path and then include the closest city to the sub-route or path at each step until a feasible route is formed. The parts of the route already constructed remain in a certain sense unchanged throughout construction process. The simplest route construction algorithm for TSP is the so-called nearest neighbor (NN) algorithm which attempts to construct the route based on connections to near neighbors [11]. It starts with a randomly chosen city as the starting city of the route and then includes the next city which is nearest to the last city. This process continues until all the cities are included on the route, and finally the last city is joined with the first city to form a feasible route. The performance of this algorithm is very sensitive to the choice of starting city. The way to solve this issue is to repeat the algorithm for each possible city as the starting city. Then the route with the smallest distance from the generated routes is considered as the optimal route. This extension is known as the repetitive nearest neighbor (RNN) algorithm [12].

The main problem with NN algorithm is that several cities are "forgotten" during construction process and have to be inserted with higher cost at the end of the process. As a result, the length of the final feasible route is increased significantly, and hence the effectiveness of the algorithm is decreased. To mitigate this drawback, many researchers have been improved it by applying various strategies. Bentley used a double-sided NN algorithm that permits the path to enhance on both ends [13]. This approach performs NN search on both ends of the route and the path with the smallest length is selected. Burke utilized route concept instead of path during the construction process [14]. Jünger et al. adopted an insertion strategy of forgotten cities to avoid including too many isolated cities at the end of the process [15]. Recently, Klug et al. extended the NN algorithm to k-RNN for solving both symmetric and asymmetric TSPs [16]. In this algorithm, the process starts with all possible sub-routes consisting of k (k=2) cities. After that, each sub-routes repeatedly extend either one side (for 2-RNN) or both sides (for Bi-2-RNN) by using a standard NN algorithm, and finally return the best route found. Their experiments illustrated that overall the 2-RNN performs relatively better than the standard NN, RNN, and Bi-2-RNN.

Another intuitively appealing route construction algorithm is known as the insertion algorithm (IA) [12-13, 15], which begins with sub-routes consisting of a few cities and then extends these sub-routes by inserting the remaining cities. The city is usually inserted into the sub-route at the point in such a way that the net increase in the cost of the route is minimum. This insertion process is continued until all the cities are included in the sub-route. The performance of this algorithm depends on three important factors, such as the choice of starting sub-route, selecting the most useful city from the remaining cities for insertion, and determining the most beneficial part of the sub-route where the selected city needs to be inserted. The initial sub-route typically consists of one to three cities in this approach [15]. There are four variants of this algorithm are available based on the way to choose the next city is to be inserted on the route to extend the current sub-route, which are nearest insertion (NI), farthest insertion (FI), cheapest insertion (CI) and arbitrary insertion (AI) [15]. In general, FI and AI algorithms generate better quality solution than NI algorithm [5].

The Multi-fragment (MF) algorithm is an interesting route construction algorithm that considers the edges as the main parameter to construct the TSP route [7, 11, 13]. It starts with the shortest edge, and then repeatedly added the shortest remaining available edges in the route until a feasible route is formed. This algorithm is also called the greedy algorithm [11]. On the other hand, several route construction algorithms based on the concept of minimum spanning tree (MST) are reported in the literature. The Double-tree and Christofides algorithms are among the two simple MST based route construction algorithms [17-18]. Both algorithms start with an MST and differ only in how a TSP route is constructed from the tree. In the Double-tree algorithm, all the edges of the MST are taken twice and the TSP route is constructed by traversing the Euler cycle and selecting the nodes in the order they are first encountered. In the Christofides algorithm, the minimum cost perfect matching on the odd-degree nodes of MST is obtained first, and then it is added to the MST of the graph. Finally, the TSP route is generated by traversing the Euler cycle and selecting the nodes in the order they are first encountered. However, they are particularly appropriate for the TSP instances satisfying triangular inequality.

3. Proposed Route Construction Framework

The proposed optimization algorithm requires performing several steps to reach an optimal solution of the problem, and it accomplishes two different tasks at each step of the procedure. That is, it first generates possible routes and then keeps the good routes. The whole searching process of the proposed approach can be described as sequence of steps as follows:

Step-1: Let G be a complete graph of n cities TSP problem with vertex list $V=\{v_1, v_2, \cdots, v_n\}$ and edge set $E=\{e_{ij}\}(1\le i\ne j\le n)$. The vertices represent the locations or positions of the cities in a coordinate system and edges' weights (distances) denote travel length, time, cost, etc. In the first step, the search engine generates sub-routes with three cities by considering all possible triplets of the node from graph G. Therefore, the set of possible routes in this step can be expressed as below [8-10]:

$$\{(v_i, v_j, v_k): \forall v_i, v_j, v_k \in V; i\ne j\ne k; i, j, k=1,2,\cdots,n\} \tag{3}$$

After that, the quality or the fitness value of each route is measured based on the Euclidean distance value. Let (x_i, y_i), (x_j, y_j) and (x_k, y_k) be the Cartesian coordinate of the location of the nodes v_i, v_j and v_k, respectively. Then, the fitness value of the route is calculated by using the formula Eq.(4). The quality of the route is inversely proportional to the distance, i.e., the route with a lower fitness value is more fitter and vice versa. Finally, the search engine keeps some good routes based on the fitness value by adopting a ratio value.

$$\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}+\sqrt{(x_j-x_k)^2+(y_j-y_k)^2} \tag{4}$$

Step-2: This step uses good routes obtained from first step to produce sub-routes with five cites. Indeed, each good route of first step is extended on both ends in this step. Thus, the set of possible routes can be constructed as follows [8-10]:

$$\{(v_s, v_i, v_j, v_k, v_t): \forall \text{ good route } (v_i, v_j, v_k); \\ \forall v_s, v_t \in V;\ s\ne t; s, t\ne i, j, k; s, t=1,2,\cdots,n\} \tag{5}$$

Like as Step-1, the fitness value of each route is measured based on the Euclidean distance value, and then the procedure retains some good routes using ratio value following the fitness value. These good routes are used to yield seven cities route in the next step. In this fashion, the procedure is continued until a set of feasible good routes are obtained. Finally, a good feasible route is to find out from there.

4. Performance Evaluation and Comparison

In this section, we conduct a number of experiments to evaluate the effectiveness of the proposed algorithm based on the benchmark symmetric TSP datasets ranging from 14 up to 1032 cities. The numerical figure appears in the dataset name denotes the dimension of the problem, e.g. burma14 is a 14-cities TSP dataset and si1032 is a 1032-cities TSP dataset. The tested datasets and their best-known optimal solutions are taken from the well-known TSPLIB [19-21]. Actually, TSPLIB is a publicly available online TSP library, which was published on the websites of Heidelberg University and the University of Waterloo. It was created and managed by Reinelt [19-20] and Cook [21], respectively. A detailed description of this data library is reported in [22]. The data library also provides the best-known optimal solution for each dataset. All the technical computations in this research are carried out on a 2 core GPU system, while the implementation software is MATLAB R2016b. The ratio value is fitted by trial and error method, and the proper adjustment for this work is presented in Eq.(6).

$$\frac{\alpha}{n+\sqrt{\lambda}} \tag{6}$$

In the above equation, n denotes the dimension of the problem, $\lambda(\lambda = 1, 2, \cdots\cdots)$ indicates the step number, and α is the constant. Through the experiment we found that $\alpha \in \left[\frac{2n+2}{n^3-3n^2+2n}, \frac{1300n+1300}{n^3-3n^2+2n}\right] \subset \square^+$ is the proper adjustment for this research. In addition, if the search engine keeps a single route at a step, the ratio value is increased in such a case that the number of retaining routes does not exceed 50. Performance evaluation and comparison are given in the following two subsections consecutively.

4.1 Performance Evaluation

Two performance evaluation indicators such as the deviation of the simulated solution from the optimal solution, i.e., Error (measured in percentage) and the required execution time (measured in seconds) are computed to evaluate the performance of the proposed optimization algorithm. The percentage deviation of the simulated solution from optimal (Error (%)) is calculated based on the Eq.(7). To measure the required execution time, the algorithm runs 10 consecutive times independently for each TSP datasets. After that, the average computational time is calculated on the basis of 10 values obtained from these 10 runs.

$$\text{Error}(\%) = \frac{\text{Our Result - Optimum}}{\text{Optimum}} \times 100 \tag{7}$$

The results for the 16 symmetric TSP datasets obtained from our experiments along with the loss of efficiency and required running time are displayed in Table 1. In the table, the first column represents the serial number (S/N) of the datasets, the second column contains the name of each dataset, the third column brings the size (dimension) of each dataset, the fourth column stands for the length of optimal route reported by the data library, the fifth column carries the length of obtained optimal route by the proposed algorithm, the sixth column represents the percentage of excess length by which the obtained length exceeds the optimal route length, and the final column available for the average required computational time (in seconds) of each TSP datasets. The datasets whose best-known optimal solutions are obtained by the proposed approach are highlighted in bold face.

Table 1: Computational results of the proposed algorithm for 16 symmetric TSP datasets taken from TSPLIB.

S/N	Datasets	Scale	Optimum	Our Result	Error (%)	Time(s)
1	burma14	14	3323	**3323**	**0.0000**	2.1148
2	p15	15	291	**291**	**0.0000**	0.0101
3	ulysses16	16	6859	**6859**	**0.0000**	4.1738
4	gr17	17	2085	**2085**	**0.0000**	0.0181
5	gr21	21	2707	**2707**	**0.0000**	0.3412
6	gr24	24	1272	**1272**	**0.0000**	31.278
7	fri26	26	937	**937**	**0.0000**	0.4982
8	bays29	29	2020	2093	3.6139	613.20
9	bayg29	29	1610	1667	3.5404	0.0282
10	hk48	48	11461	**11461**	**0.0000**	206.80
11	att48	48	10628	10671	0.4046	1265.20
12	brazil58	58	25395	25649	1.0002	2.2556
13	si175	175	21407	21913	2.3637	166.04
14	brg180	180	1950	1960	0.5128	347.34
15	si535	535	48450	49918	3.0299	5160.10
16	si1032	1032	92650	95039	2.5785	17620.30
Average Error (SD):					**1.07(1.42)**	

The results of Table 1 indicate that the proposed optimization algorithm produces the closest optimal solutions to the datasets considered for the test, and the errors of the solutions are very small. For some datasets, such as burma14, p15, ulysses16, gr17, gr21, gr24, fri26, and hk48, the error of the solution is 0, which indicates that the proposed algorithm captures exactly the best-known optimal solutions with good robustness in these cases. For the other tested datasets, the loss of efficiency no more than $\approx 3.61\%$ in which the error lies within the interval of $[0.4\%, 1\%]$ in three datasets such as att48, brazil58 and brg180. In addition, the average of error over all of the considered datasets is only 1.07% and the standard deviation (SD) value of the error is 1.42. On the other hand, the proposed algorithm expenses a little amount of run time to solve small size datasets and for others, it takes comparatively more time. Although the

required execution time is comparatively higher in large scale datasets, the quality of the solution is satisfactory and the error lies within the interval of $[0.4\%, 3.61\%]$. In order to facilitate observation, the comparison of the obtained optimal solutions with the best-known optimal solutions is illustrated graphically. This is shown in the form of bar charts in Fig. 1. From the comparison bar chart, one can easily get an idea about the results of the proposed algorithm intuitively. It can be observed from the bar chart that in some cases, especially in last two large datasets, such as si535 and si1032, the computed optimal route length is slightly higher than the best-known optimal route length. However, the algorithm is still capable of producing solution near to the best-known optimal solution for the tested TSP datasets in a reasonable computational time frame.

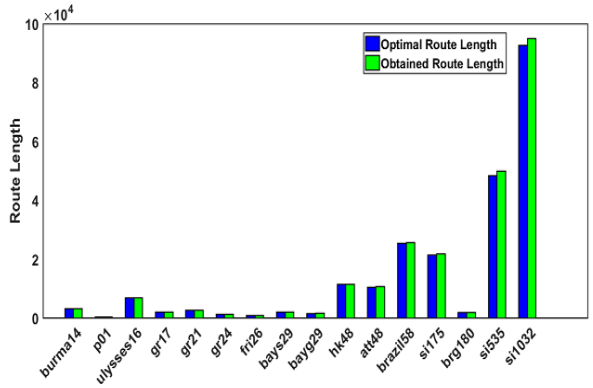

Fig. 1 Comparison of obtained optimal solutions by the proposed optimization algorithm with the best-known optimal solutions.

4.2 Performance Comparison with other Route Construction Algorithms

In this subsection, we examine the performance of the proposed route construction optimization algorithm with other route construction optimization algorithms available in the literature. To do this, seven algorithms are considered in which one algorithm is the recently improved algorithm and the others are conventional algorithms. Actually, the simulated results of the proposed algorithm are compared with the results obtained by the algorithms described in reference articles. The authors of different articles considered different sets of datasets to test their algorithms. For this reason, seven algorithms are divided into three groups, and different set of datasets are used to compare the proposed algorithm with each group. Side by side comparisons for the symmetric TSP datasets are displayed in Table 2 - Table 4. The best results obtained across all algorithms are highlighted in boldface. Furthermore, the comparison between our algorithm and the comparison algorithms are depicted graphically in Fig.2 - Fig.4 for further visualization of the performance of the proposed algorithm. Indeed, the competitive behavior of our algorithm is tested based on the following route construction algorithms:

1. Nearest Neighbour Optimization Algorithm (NN) [23]
2. Extending NN Optimization Algorithm (k-RNN) [16]
3. Multi-fragment Optimization Algorithm (MF) [23]
4. Christofides Optimization Algorithm [23]
5. Farthest Insertion Optimization Algorithm (FI) [7]
6. Boruvka's Optimization Algorithm (Bor) [7]
7. Quick-Boruvka's Optimization Algorithm (Q-Bor) [7]

The comparison of 34 symmetric TSP datasets with the recently improved of NN algorithm named k-RNN are summarized in Table 2. The literature of k-RNN reported three extension of NN algorithm such as 1-RNN, 2-RNN, and Bi-2-RNN. The results of Table 2 show that the proposed algorithm is very competitive with each of three extensions. For the considered 34 datasets, our algorithm yields the best solutions in 22 datasets, as opposed to 1-RNN, 2-RNN and Bi-2-RNN, which give the best solution in 0, 4 and 9 datasets, respectively. Besides this, the average percentage error of our algorithm over all the datasets is 9.38, which is better than the 24.02 of 1-RNN, 11.75 of 2-RNN and 22.25 of Bi-2-RNN, respectively. In addition, our algorithm provides the best-known optimal solutions in some datasets (gr17, gr21, gr24, fri26 and hk48), but none of the comparison algorithms find such type of solution. It is also visible from the table that in large scale datasets such as si535, si1032, and nrw1379, the effectiveness of the proposed algorithm is inferior compared to k-RNN. However, its performance over all the tested datasets is still better than the improvement of NN algorithm. The comparison of the proposed algorithm with k-RNN is also depicted graphically in Fig. 2, which further shows the superior performance of the proposed algorithm. In fact, there is a significant difference between the bar of our algorithm and the compared algorithms.

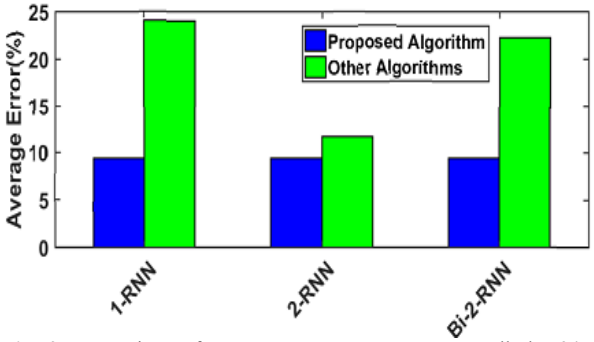

Fig. 2 Comparison of average percentage error over all the 34 symmetric TSP datasets of the proposed algorithm with the recent k-RNN algorithms.

Table 2: Performance comparison of the proposed algorithm with the recently improved of NN algorithm (k-RNN) for 34 symmetric TSP datasets.

Datasets	Scale	Error (%) of k-RNN (2019) [16]			Our Result
		1-RNN	2-RNN	Bi-2-RNN	
gr17	17	4.46	4.46	4.46	**0.0000**
gr21	21	10.93	9.27	10.75	**0.0000**
gr24	24	22.09	10.06	16.04	**0.0000**
fri26	26	2.99	2.35	2.45	**0.0000**
swiss42	42	12.88	11.94	**6.05**	8.0911
dantzig42	42	23.61	18.17	21.32	**9.0129**
gr48	48	15.74	**10.21**	12.86	11.5339
hk48	48	5.90	4.97	4.62	**0.0000**
eil51	51	13.15	10.80	13.38	**7.0094**
berlin52	52	8.47	5.65	11.11	**4.9059**
brazil58	58	7.83	7.16	6.77	**1.0002**
eil76	76	13.01	11.15	**7.06**	11.7955
pr76	76	21.04	19.04	19.70	**15.3302**
kroA100	100	16.05	15.51	15.35	**15.1724**
kroB100	100	16.91	14.06	15.38	**6.4451**
kroC100	100	14.03	13.75	15.52	**13.1062**
kroD100	100	16.71	15.54	**11.40**	13.6448
kroE100	100	12.30	10.77	**9.59**	10.1120
eil101	101	18.60	18.12	17.33	**15.9300**
lin105	105	17.78	12.30	**10.42**	15.7104
gr120	120	21.55	20.07	21.16	**17.1132**
bier127	127	13.25	**8.71**	9.17	11.8682
ch130	130	16.68	12.98	**11.83**	14.5990
kroB150	150	20.98	20.64	14.98	**12.2044**
si175	175	2.77	**2.33**	2.43	2.3637
brg180	180	355.90	3.59	355.90	**0.5128**
d198	198	11.66	10.30	12.50	**10.0507**
kroA200	200	17.62	17.62	20.30	**14.3672**
kroB200	200	20.22	19.86	20.44	**19.6948**
gil262	262	18.71	16.36	16.40	**15.0841**
lin318	318	17.06	17.06	16.58	**16.2543**
si535	535	3.27	3.27	**2.90**	3.0299
si1032	1032	1.55	1.44	**1.17**	2.5785
nrw1379	1379	21.00	19.84	**19.03**	20.3379
Average Error :		**24.02**	**11.75**	**22.25**	**9.38**
(SD):		**(58.97)**	**(5.88)**	**(59.25)**	**(6.49)**

Table 3 and Fig. 3 show the performance comparison of 20 symmetric TSP benchmark datasets with three conventional algorithms namely the nearest neighbour optimization algorithm (NN), Multi-fragment optimization algorithm (MF) and Christofides optimization algorithm. From the comparison, it can be observed that the proposed approach exhibits a competitive behavior with other considered algorithms. It finds the best solution in 9 out of 20 datasets, while NN, MF and Christofides provide the best solution in the 1, 2, and 8 datasets, respectively. Moreover, the average error over all 20 datasets of our algorithm is 11.85, which is better than the 24.88 of NN, 17.38 of MF, and 14.89 of Christofides, respectively. Although Christofides produces better solutions in some specific datasets, still our algorithm performs better over all the considered datasets. From Fig. 3, it can easily be seen that the superior performance of the proposed optimization algorithm.

Table 3: Performance comparison of the proposed algorithm with the Nearest Neighbour (NN), Multi-fragment (Greedy) (MF) and Christofides route construction algorithms for 20 symmetric TSP datasets.

Datasets	Scale	Error (%) [23]			Our Result
		NN	MF	Christofides	
att48	48	20.89	19.80	20.29	**0.4046**
eil51	51	20.57	13.03	17.33	**7.0094**
berlin52	52	19.08	31.98	12.16	**4.9059**
eil76	76	32.34	14.71	17.20	**11.7955**
kroA100	100	26.19	**13.70**	18.53	15.1724
kroB100	100	31.68	16.59	9.19	**6.4451**
kroC100	100	26.88	**12.94**	13.68	13.1062
kroD100	100	26.56	14.82	**13.47**	13.6448
kroE100	100	25.01	12.59	**9.31**	10.1120
lin105	105	41.61	16.64	26.26	**15.7104**
pr107	107	**5.36**	6.60	9.70	13.8704
bier127	127	14.77	14.77	**11.16**	11.8682
ch130	130	23.98	28.40	**13.15**	14.5990
ch150	150	25.53	18.61	**9.11**	10.2298
kroA150	150	26.71	20.24	**16.13**	20.2458
kroB150	150	25.62	20.25	22.54	**12.2044**
d198	198	18.00	22.20	14.57	**10.0507**
kroA200	200	21.90	17.83	**13.43**	14.3672
gil262	262	36.31	13.21	**13.19**	15.0841
lin318	318	28.56	18.75	17.40	**16.2543**
Average Error :		**24.88**	**17.38**	**14.89**	**11.85**
(SD):		**(7.78)**	**(5.69)**	**(4.62)**	**(4.53)**

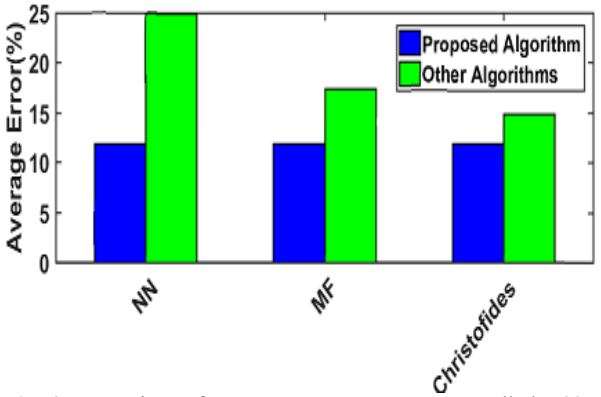

Fig. 3 Comparison of average percentage error over all the 20 symmetric TSP datasets of the proposed algorithm with the Nearest Neighbour (NN), Multi-fragment (Greedy) (MF) and Christofides algorithms over 20 symmetric TSP datasets.

Performance comparison with Farthest Insertion (FI), Boruvka's (Bor) and Quick-Boruvka's (Q-Bor) optimization algorithms for the 11 symmetric TSP benchmark datasets are displayed in tabular form in Table 4 and graphically in Fig. 4. From Table 4 and Fig. 4, it can be noticed that the proposed algorithm significantly dominates FI and Q-Bor. But its performance almost similar to the Bor over all the considered datasets. In fact, Bor performs well on relatively large datasets where our algorithm finds the best solution in small cases. For the considered 11 datasets, the proposed algorithm captures

the best solution in 5 datasets, where the best performance appears in 1 case by FI and Q-Bor and 4 cases by Bor. Besides this, the overall penalty on the solution of our algorithm is 17.33, which is better than FI's 28.80, Bor's 17.63, and Q-Bor's 22.84. Thus, it can be concluded that the performance of the proposed algorithm is superior than FI and Q-Bor but inferior against Bor in large scale datasets. Thus, it can be said that the performance of the proposed algorithm is better compared to the other considered algorithms in many symmetric TSP datasets. In addition, the average percentage error and the standard deviation value computed by the proposed algorithm over all the datasets in each group are lower than other considered route construction algorithms.

Table 4: Performance comparison of the proposed algorithm with Farthest Insertion (FI), Bor_uvka's (Bor) and Quick-Bor_uvka's (Q-Bor) route construction algorithms for 11 symmetric TSP datasets.

Datasets	Scale	Error (%) [7]			Our Result
		FI	Bor	Q-Bor	
st70	70	27.96	12.15	23.35	**12.1800**
kroA100	100	26.85	19.57	28.47	**15.1724**
pr107	107	**5.18**	7.15	20.46	13.8704
pr136	136	22.18	19.92	21.99	**18.8095**
ch150	150	24.86	22.43	25.87	**10.2298**
kroA200	200	35.48	18.53	20.89	**14.3672**
gil262	262	31.96	15.33	20.15	**15.0841**
rd400	400	34.90	22.48	**18.05**	21.5954
417	417	31.72	**23.85**	37.13	29.8879
rat575	575	36.57	**17.26**	18.22	19.0935
nrw1379	1379	39.09	**15.22**	16.67	20.3379
Average Error :		**28.80**	**17.63**	**22.84**	**17.33**
(SD):		**(9.43)**	**(4.96)**	**(5.87)**	**(5.46)**

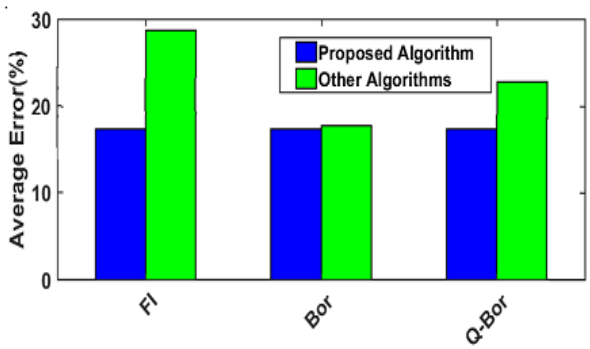

Fig. 4 Comparison of average percentage error over all the 11 symmetric TSP datasets of the proposed algorithm with the Farthest Insertion (FI), Boruvka's (Bor) and Quick- Boruvka's (Q-Bor) algorithms.

5. Conclusion

We have established a route construction optimization algorithm in an intelligent way with the help of ratio value for finding the optimal solutions of the symmetric TSPs. The algorithm starts with the sub-routes and the good routes are enlarged step by step, meanwhile, the worse routes are kicked out from the search environment. Indeed, the algorithm is composed of several steps and in each step, it performs two main tasks. It first generates possible routes in the search environment and then retains the good routes in the environment by using a rigorous ratio value. The proposed optimization algorithm is completely different from other route construction optimization algorithms in the route construction process. The main contribution of this research is to introduce the intuitive assumption that we can construct the best routes step by step by adopting a ratio value in the construction process. It is demonstrated by the experimental results that the proposed algorithm obtains the best-known optimal solutions in some cases and generally shows the competitive behavior with the other route construction optimization algorithms.

References

[1] Papadimitriou, C. H. (1977). The Euclidean travelling salesman problem is NP-complete. Theoretical computer science, 4(3), 237-244.

[2] Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America, *2*(4), 393-410.

[3] Khanra, A., Maiti, M. K., & Maiti, M. (2015). Profit maximization of TSP through a hybrid algorithm. Computers & Industrial Engineering, 88, 229-236.

[4] Ahrens, B. M. (2013). A tour construction framework for the travelling salesman problem. 1-8. IEEE.

[5] Reinelt, G. (2003). The traveling salesman: computational solutions for TSP applications. Vol. 840. Springer.

[6] Grotschel, M., & Lovász, L. (1995). Combinatorial optimization. Handbook of combinatorics, 2(1541-1597), 168.

[7] El Krari, M., Ahiod, B., & El Benani, B. (2016). An empirical study of the multi-fragment tour construction algorithm for the travelling salesman problem. In. International Conference on Hybrid Intelligent Systems (pp. 278-287). Springer, Cham.

[8] Rahman, M. A., & Ma, J. (2018). Probe machine based consecutive route filtering approach to symmetric travelling salesman problem. In. International Conference on Intelligence Science (pp. 378-387). Springer, Cham.

[9] Rahman, M. A., & Ma, J. (2019). Solving symmetric and asymmetric traveling salesman problems through probe machine with local search. In. International Conference on Intelligent Computing (pp. 1-13). Springer, Cham.

[10] Rahman, M. A., & Ma, J. (2019). Probe Machine Based Optimization Approach for Capacitated Vehicle Routing Problem. In. 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP) (pp. 1-6). IEEE.

[11] Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 1(1), 215-310.

[12] Rosenkrantz, D. J., Stearns, R. E., & Lewis, II, P. M. (1977). An analysis of several heuristics for the traveling salesman problem. SIAM journal on computing, 6(3), 563-581.
[13] Bentley, J. J. (1992). Fast algorithms for geometric traveling salesman problems. ORSA Journal on computing, 4(4), 387-411.
[14] Burke, L. I. (1994). Neural methods for the traveling salesman problem: insights from operations research. Neural Networks, 7(4), 681-690.
[15] Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. Handbooks in operations research and management science, 7, 225-330.
[16] Klug, N., Chauhan, A., Vijayakumar, V., & Ragala, R. (2019). k-RNN: Extending NN-heuristics for the TSP. Mobile Networks and Applications, 24(4), 1210-1213.
[17] Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.
[18] Williamson, D. P., & Shmoys, D. B. (2011). The design of approximation algorithms. Cambridge university press.
[19] http://comopt.i_.uni-heidelberg.de/software/TSPLIB95/
[20] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/
[21] http://www.math.uwaterloo.ca/tsp/data/index.html
[22] Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. ORSA journal on computing, *3*(4), 376-384.
[23] Jazayeri, A., & Sayama, H. (2020). A polynomial-time deterministic approach to the travelling salesperson problem. International Journal of Parallel, Emergent and Distributed Systems, 35(4), 454-460.

Dr. Md. Azizur Rahman received the Bachelor of Science (B.Sc.) degree in Mathematics and Masters of Science (M.Sc.) degree in Applied Mathematics from Khulna University in 2008 and 2011, respectively. He also received the Doctor of Natural Science degree from Peking University in 2020. After working as a Lecturer (from 2015), he has been an Assistant Professor of Mathematics at Khulna University since 2016. His research interest includes Artificial Intelligence, Combinatorial Optimization, Evolutionary Algorithms, Machine Learning, Deep Learning, and Applied Mathematics. He is a member of Bangladesh Mathematical Society (BMS).

Dr. Ariful Islam is currently an Associate Professor at the Mathematics Discipline, Khulna University, Bangladesh. His main area of research involves the study of the behaviour of ultrafine magnetic minerals in microchannels, which is applicable for the mineral processing and medical sciences research. Dr. Islam achieved his PhD in 2021 from the University of Newcastle, Australia. He has been graduated from the Khulna University, Bangladesh in 2005.

Dr. Lasker Ershad Ali received the Bachelor of Science (B.Sc.) degree in Mathematics and Masters of Science (M.Sc.) degree in Applied Mathematics from Khulna University in 2006 and 2008, respectively. He also received the Doctor of Natural Science degree from Peking University in 2018. After working as, a Lecturer (from 2008), an Assistant Professor (from 2010), and an Associate Professor (from 2015), he has been a Professor of Mathematics at Khulna University since 2019. His research interest includes Biometric, Image Processing, Pattern Recognition, Machine Learning as well as Deep Learning, Computer Vision and Applied Mathematics. He is a member of Bangladesh Mathematical Society (BMS).