A Comparative Study of Eight Crossover Operators for the Maximum Scatter Travelling Salesman Problem

Zakir Hussain Ahmed
Department of Mathematics and Statistics, College of Science
Al Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh, Kingdom of Saudi Arabia

***Abstract*—The maximum scatter traveling salesman problem (MSTSP), a variation of the famous travelling salesman problem (TSP), is considered here for our study. The aim of problem is to maximize the minimum edge in a salesman's tour that visits each city exactly once in a network. It is proved be NP-hard problem and considered to be very difficult problem. To solve this kind of problems efficiently, one must use heuristic/metaheuristic algorithms, and genetic algorithm (GA) is one of them. Out of three operators in GAs, crossover is the most important operator. So, we consider eight crossover operators in GAs for solving the MSTSP. These operators have originally been designed for the TSP which can also be applied on the MSTSP after some modifications. The crossover operators are first illustrated manually through an example and then executed on some well-known TSPLIB instances of different types and sizes. The obtained comparative study clearly demonstrates the usefulness of the sequential constructive crossover operator for the MSTSP. Finally, a relative ranking of the crossover operators is reported.**

Keywords*—Traveling salesman problem; maximum scatter; genetic algorithms; crossover operators; sequential constructive crossover

I. INTRODUCTION

The travelling salesman problem is a famous problem (TSP) that aims to find shortest tour of a salesman who starts his journey from depot node and visit all remaining n nodes (cities) such that each node is to be visited only once and then returns to the depot. It is a NP- Hard problem [1] that is very easy to define but difficult to solve. Several researches have been done to deal with the problem and consequently numerous good algorithms have been reported in the literature. However, few circumstances require different restrictions on the acceptability of a tour as solution. One such restriction is to maximize the minimum cost edge in a tour of the salesman, which is named as maximum scatter TSP (MSTSP). In MSTSP, given a weighted graph, the aim is to find a Hamiltonian circuit so that the minimum cost edge is maximized. That is, the aim is to make each point away from (scattered) its previous and next points in the circuit. It is also called the max-min 1-neighbour TSP. In the max-min m-neighbor TSP, the aim is to maximize the minimum cost between any city and all its m-neighbours in the Hamiltonian circuit. These problems are close to the bottleneck TSP (BTSP) [2].

The MSTSP, defined first in [3], has application in operations involving heating workpiece, where it is equally important to keep each point away from its immediate ancestor and successor along with its m-neighbors for allowing cooling period in each operation. It has application in some other manufacturing processes that attach metal sheets together. After required alignment, the topmost sheet has some pre-specified points where riveting operations are applied to attach the sheets together. To avoid nonuniform deformation of the sheets, it is required to arrange the riveting process such that the distance between any rivet and its next rivet is very large; that means, the riveting operations must be scattered. It has application in some kind of medical imaging also. During imaging physical functions by Dynamic Spatial Reconstructor, radiation sources are positioned on the upper half of a circular ring and sensors are positioned directly opposite on the lower. The 'firing sequence' decides the sequence of radiation sources along with their associated sensors, generally periodically. The sensors gather energy intensity which goes through the patient positioned in the middle of the ring. It is required that if the i^{th} source is activated, then its neighbour sources (for example, $(i-1)^{th}$, $(i+1)^{th}$, $(i+2)^{th}$, etc.) must not be activated, and hence some amount of scattering occurs [1]. The problem can be applied to a case where someone is falsely accused of a crime and given is death penalty. Now, he tries to escape from the police by visiting different safe places across his country to avoid the capture. Throughout his journey, he looks for a tour such that the smallest distance between consecutive places is very big [4].

The MSTSP can be formally defined as follows: Let a network with a set of n nodes, considering node 1 as depot node and a travel cost (time or distance, etc.) matrix $C=[c_{ij}]$ of order n connected with ordered pair (i, j) of nodes is given. Let $(1=\alpha_0, \alpha_1, \alpha_2, ...,\alpha_{n-1}, \alpha_n=1) \equiv \{1\rightarrow\alpha_1\rightarrow\alpha_2\rightarrow.... \rightarrow\alpha_{n-1}\rightarrow1\}$ be a tour. The tour cost is defined as $\min\{c_{\alpha_i,\alpha_{i+1}} : i = 0, 1, 2, \dots, n-1\}$. The aim is to find a tour that has maximum tour cost. The problem can be transformed to a BTSP by the transformation $d_{ij} = L\text{-}c_{ij}$ where $D = [d_{ij}]_{nxn}$ is equivalent BTSP's cost (or distance) matrix and L is very large number [5].

Since the problem is NP-hard, obtaining optimal solution using exact method is very hard, if not possible. The moderate sized TSP instances have been effectively solved by using

operations research methods, like branch-and-bound [6], lexisearch [7], branch-and-cut [8] and local search [9]. As the problem size increases, obtaining exact solution is very hard. For solving large sized instances, one must go for heuristic algorithms, which, of course, don't promise to obtain optimal solution of a problem instance; however, they give near exact solution very quickly. Hence heuristic algorithms are used to solve some difficult problems. The most current algorithms that are used to solve various difficult optimization problems are termed as metaheuristics. There are metaheuristic algorithms based on simulated annealing [10], tabu search [11], insertion heuristic [12], ant colony algorithm [13], genetic algorithms [14], variable neighbourhood method [15], etc. However, genetic algorithms (GAs) are extensively applied methods amongst modern metaheuristics, and hence, we are applying GAs to solve the MSTSP.

Genetic Algorithms (GAs) first developed by John Holland in 1975, based on imitating the Darwinian survival-of-the-fittest theory among different species created by arbitrary changes in the chromosomes' structure in the natural biology [14]. They are powerful and robust metaheuristic algorithms for solving large-sized problem instances. They have been fruitfully applied to numerous combinatorial optimization problems to find their solutions. Each feasible solution of a problem may be assumed as a chromosome whose fitness is measured by its objective function value [16].

In general, simple GAs begin using randomly created a set of chromosomes called initial population, also termed as pool of genes, and then apply, mainly three, genetic operators to produce new, and possibly, better populations in subsequent generations. The first operator is selection which probabilistically copies and discards some of the chromosomes of the present generation to the next generation. Crossover is the second operator that selects randomly a pairs of chromosomes and mates to produce new chromosomes. The third operator is mutation, which randomly alters some position values (genes) of a chromosome. Crossover is very powerful operator in the GA search. Mutation diverges the GA search space. Generally, probability of applying mutation operator is fixed very low comparative to probability of crossover operator [14].

The crossover operators which have been developed for the usual TSP are also applied on the variant TSP after some modification. Since the MSTSP is a variant TSP, we consider eight crossover operators in simple GAs for solving the MSTSP. The crossover operators are first illustrated manually through an example and then executed on some well-known TSPLIB instances of different types and sizes. The obtained comparative study clearly demonstrates the usefulness of the sequential constructive crossover operator [16] for the MSTSP. Finally, a relative ranking of the crossover operators is reported.

This paper is organized as follows: A survey of the literature for the MSTSP is reported in Section II. Section III develops simple genetic algorithms using eight crossover operators for the problem, whereas, Section IV reports computational experiments for eight crossover operators. Finally, Section V presents conclusion and future works.

II. Related Work

There are few literatures about MSTSP, and the relevant papers are as follows. Arkin et al. [1] developed the first method for solving the problem. The problem was shown be NP-hard and unless P = NP, any no constant-factor approximation method can be designed. They developed factor-2 (which is best factor) approximation method with the triangle inequality for the max-min 1-neighbor TSP, for the cycle and path versions. Further, the method expanded to obtain a factor-2 approximation solution for the max-min 2-neighbor TSP, for cycle as well as some cases of path version. They also developed methods for the max-min 2-neighbor TSP with the triangle inequality, for both the path and cycle versions. The methods also expanded to obtain an approximation solution for path version of the max-min m-neighbor TSP.

Chiang [17] developed approximation methods for the max-min 2-neighbor TSP that follows the triangle inequality. He developed approximation methods for the path and cycle versions by improving methods in [1]. As mentioned, both algorithms are much simpler. John [4] also studied many works of MSTSP and its relevant models. Kabadi and Punnen [18] obtained an approximation method for the MSTSP that satisfies the triangle inequality, which is claimed to be the best bound for the case. Hoffmann et al. [19] extended the algorithm in [1] that produces optimal solutions for the nodes on a line to a regular mxn-grid. As reported, in some particular cases, the algorithm takes linear computational time to find an optimal tour.

The MSTSP is close to the BTSP, where the aim is to minimize the maximum cost edge in a Hamiltonian circuit [20]. Exact algorithms based on lexisearch approach have been developed ([21], [22]). Also, hybrid algorithms have been proposed for solving the problem ([23], [24]). Another closely related problem of the MSTSP is the maximum TSP (MaxTSP), in which the aim is to maximize total length of a tour in a Hamiltonian circuit [25]. A hybrid GA is proposed for solving the problem [26].

Dong et al. [27] proposed the multi-salesmen version of the MSTSP, multiple MSTSP (MMSTSP). They developed three improved GAs using greedy initialization, hill-climbing and simulated annealing algorithms to improve GAs for solving the MMSTSP. As claimed the improved algorithms are efficient algorithms and can reveal several characteristics in finding the solution of the problem.

A multi-start iterated local search approach is proposed in [28] for the MSTSP. Two local search algorithms based on insertion and modified 2-opt moves have been developed as part of our approach. To investigate the effectiveness of the method, it is tested on the TSPLIB instances, and found very good results.

III. Simple Genetic Algorithms for the MSTSP

Beginning with an initial population, a simple GA recurrently applies three genetic operators, selection, crossover and mutation, until the stopping criterion is satisfied. Though GA is among the best metaheuristic algorithms, but its performance verily depends on initial chromosome population,

three operators and some parameters [14] that are discussed in this section.

A. Chromosome Representation and Initial Population

There are numerous ways to represent solutions as chromosomes for the TSP and its variants. Path representation is considered for the MSTSP that lists labels of nodes so that no any node is repeated in a chromosome. Suppose, {1, 2, 3, 4, 5, 6, 7, 8} represents the node labels in an 8-node instance, then the tour {1→7→2→3→8 → 4→6→ 5 →1} can be denoted by (1, 7, 2, 3, 8, 4, 6, 5). The objective function is defined as the sum of the costs of edges in the tour. Since the problem is a maximization problem, fitness and objective functions are same. Usually a simple GA begins with a pool of chromosomes called initial population. Here randomly created initial population is considered.

B. Selection Operator

In selection process, strings/chromosomes are replicated to the mating pool of next generation based on probabilities associated with their fitness function values. By transferring a higher portion of fitter chromosomes to the next generation, selection imitates the Darwinian survival-of-the-fittest in natural biology. Here, no any new chromosome is formed. Generally, the proportionate selection is applied in which any chromosome is chosen based on a probability that is calculated as proportional to its fitness function value. For example, roulette wheel selection, tournament selection, stochastic remainder, etc. are some of them. We consider stochastic remainder selection method [29] for our GAs.

C. Crossover Operators

Crossover operators selects two parent chromosomes and a point throughout the length of the chromosomes and exchanges their information after the crossover point. It performs a very significant role in GAs. Several good crossover methods are suggested for the TSP in the literature which are supposed to be good for the MSTSP. For example, partially mapped crossover [30], ordered crossover [31], alternating edges crossover [32], cycle crossover [33], edge recombination crossover [34], generalized N crossover [35], greedy crossover [32], sequential constructive crossover [16] are some of them. We are going to investigate these eight crossover methods.

1) Partially mapped crossover operator. The partially mapped crossover (PMX) uses two crossover points and produces two offspring chromosomes [30]. It defines exchange mappings in the segment between the crossover points. It is the first crossover operator designed for the TSP in GAs. We illustrate the PMX through the 8-node example instance along with its cost matrix given in Table I and the parent chromosome pair P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7) with costs 3 and 1 respectively. We start journey (computation) from the first gene (headquarters), node 1.

Let the arbitrarily assumed cut points are after 3rd and 6th genes that are marked with "|", as follows:

P_1: (1, 5, 4 | 7, 8, 2 | 3, 6) and

P_2: (1, 8, 3 | 4, 5, 6 | 2, 7)

TABLE I. THE COST MATRIX

Node	1	2	3	4	5	6	7	8
1	0	10	15	95	66	55	29	2
2	61	0	55	22	50	72	1	58
3	45	50	0	69	7	89	22	78
4	91	67	75	0	35	27	34	89
5	60	36	90	31	0	50	61	77
6	3	82	20	70	39	0	77	28
7	16	57	26	86	53	19	0	69
8	13	14	54	8	84	37	87	0

The mapping segments are between these cut points. So, the exchange mappings are 7↔4, 8↔5 and 2↔6. These mapping segments are copied to the offspring chromosomes as follows:

O_1: (1, *, * | 7, 8, 2 | *, *),

O_2: (1, *, * | 4, 5, 6 | *, *)

We now add some more genes from the alternative parent chromosomes that do not form invalid chromosome as follows:

O_1: (1, *, 3 | 7, 8, 2 | *, *),

O_2: (1, *, * | 4, 5, 6 | 3, *)

The node 8 should be in the place of first * in O_1 which comes from P_2, but, since it is available in O_1, so after checking the mapping 8↔ 5, node 5 is placed there. The second * in O_1 should be 2 which comes from P_2, but, since it is available in O_1, so after checking the mapping 2↔6, node 6 is place there. Finally, 4 is added at third *. So, the first complete offspring becomes.

O_1: (1, 5, 3 | 7, 8, 2 | 6, 4) with cost 14.

Similarly, one can create the second complete offspring as:

O_2: (1, 8, 7 | 4, 5, 6 | 3, 2) with cost 2.

2) Ordered crossover operator. To create offspring chromosomes, the ordered crossover (OX) selects a subsegment of a route from one parent chromosome and then preserves the relative order of genes from the other one [31]. We choose the same parent chromosomes and cut points marked with "|" as:

P_1: (1, 5, 4 | 7, 8, 2 | 3, 6) and

P_2: (1, 8, 3 | 4, 5, 6 | 2, 7)

We always fix first gene as 'node 1'. At first, the offspring are created by simply copying the segments between these cuts into the offspring as:

O_1: (1, *, * | 7, 8, 2 | *, *),

O_2: (1, *, * | 4, 5, 6 | *, *)

Now, starting from 2nd cut of one parent chromosome, the genes (un-available) from the other chromosome are copied in the same sequence. The order of genes in P_2 from the 2nd cut is

{2 →7→8→3→4→5 →6}. After ignoring the already available genes 7, 8 and 2 in O_1, the order becomes {3→4→5 →6}, which is added in O_1 starting from the 2nd cut point:

O_1: (1, 5, 6 | 7, 8, 2 | 3, 4) with cost 14.

Similarly, second offspring is created as:

O_2: (1, 8, 2 | 4, 5, 6 | 3, 7) with cost 2.

3) Alternating edges crossover operator. The alternating edges crossover (AEX) operator considers a chromosome as a cycle of arcs [32] that creates only one offspring by choosing alternative arcs from the parents. In case of invalid offspring, random arc is chosen to create valid offspring. We choose the same example chromosomes P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7).

At the beginning the arc (1, 5) is chosen from P_1 and the arc (5, 6) from P_2 are added to the offspring. Next, the arc (6, 1) is chosen P_1, as 6 is the last node, but this arc creates a cycle. So, an arc leaving node 6 to an unvisited node is chosen randomly. Suppose the arc (6, 2) is chosen. Next, the arc (2, 7) from P_2, (7, 8) from P_1 and then (8, 3) from P_2 are added to the current offspring. Finally, the following offspring may be created:

O: (1, 5, 6, 2, 7, 8, 3, 4) with cost 1.

All arcs present in the offspring (O) are inherited from either of the parents.

4) Cycle crossover operator. The cycle crossover (CX) creates offspring in which every node and its corresponding location are originated from either of the parent chromosomes [33]. We choose the same example chromosomes P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7).

The first gene is 1 and for the 2nd position, we choose randomly either 5 or 8. Suppose we choose node 5, then the offspring chromosome becomes:

O_1: (1, 5, *, *, *, *, *, *)

All genes in the offspring is chosen from either of the parents in the same location, so the next gene to should be 8, which is located in P_2 just below the present node 5. In P_1, this node 8 is located at 5th position; so, the offspring chromosome becomes:

O_1: (1, 5, *, *, 8, *, *, *)

Since, next node to be selected is 5 that is already available in O_1; thus, a cycle is completed and so, the remaining blank locations will be filled up by the genes of those locations that are present in P_2. This way the offspring is built as follows:

O_1: (1, 5, 3, 4, 8, 6, 2, 7) with cost 1.

Similarly, the 2nd offspring is created:

O_2: (1, 8, 4, 7, 5, 2, 3, 6) with cost 2.

5) Edge recombination crossover operator. The edge recombination crossover (ERX) is proposed in [34]. Most operators consider the position and the order of the node. This operator considers the links between these nodes. To apply this operator, we first construct the edge list of the parents P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7).

Table II shows the edge list of all the nodes for the given example. Since the 1st gene of the offspring is 1, the nodes 6, 5, 7 and 8 are the candidates for the next gene. The nodes 6, 7 and 8 have three edges: initially four node minus the present node 1. Similarly, the node 5 has two edges. Among them, node 5 has minimum edges, thus it is chosen, and the offspring becomes (1, 5).

Node 5 has edges to nodes 4 and 6, so node 4 is chosen randomly as both have equal two edges, and the offspring becomes (1, 5, 4).

Node 4 has edges to nodes 7 and 3. Nodes 7 and 3 have 2 and 3 edges. So, node 7 is chosen next and the offspring becomes (1, 5, 4, 7).

Node 7 has edges to nodes 8 and 2. Nodes 8 and 2 have 2 and 3 edges. So, node 8 is chosen next and the offspring becomes (1, 5, 4, 7, 8).

Node 8 has edges to nodes 2 and 3. Both nodes have 2 edges. So, node 2 is chosen randomly and the offspring becomes (1, 5, 4, 7, 8, 2).

Node 2 has edges to nodes 3 and 6. Both nodes have 1 edge. So, node 3 is chosen randomly and the offspring becomes (1, 5, 4, 7, 8, 2, 3). This way the final offspring is created as: (1, 5, 4, 7, 8, 2, 3, 6) with cost 3. Here all edges are chosen from either of the parents.

6) Generalized n-point crossover operator. Radcliffe and Surry [35] developed generalized N crossover (GNX). Suppose N=2, and P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7). Now, if crossover points are 4 and 6, then the bold face nodes would usually be selected by G2X. Suppose the segments are tested in the order (3, 2, 1). Then the 3rd segment of random parent, suppose of P_2, will be added to give the proto child (*, *, *, *, *, *, 2, 7). Next, the nodes in 2nd segment from P_1 will be tested in random order. The node 8 is accepted to give the proto child (*, *, *, *, 8, *, 2, 7). Then nodes in 1st segment from P_2 is tested, and nodes 1, 3 and 4 are accepted to give the final proto child after the 1st phase: (1, *, 3, 4, 8, *, 2, 7).

Now, the untested segments of both parents are the tested in arbitrary order. Only the 2nd segment for P_2 is relevant here and node 6 is accepted. So, the proto child after the 2nd phase is (1, *, 3, 4, 8, 6, 2, 7).

Since this offspring is yet incomplete, we fill it up randomly. So, the final offspring may be (1, 5, 3, 4, 8, 6, 2, 7) with cost 1. Only four edges are chosen from either of the parents.

TABLE II. THE EDGE LIST OF THE NODES FOR THE PARENTS P_1 AND P_2

Node	Edge list	Node	Edge list
1	6, 5, 7, 8	**5**	1, 4, 6
2	8, 3, 6, 7	**6**	3, 1, 5, 2
3	2, 6, 8, 4	**7**	4, 8, 2, 1

4	5, 7, 3	**8**	7, 2, 1, 3

7) Greedy crossover operator. The greedy crossover (GX) selects the first node randomly [32]. Since the MSTSP is a maximization problem, hence some steps of the GX must be modified. So, our modified GX for the problem is as follows. In each step, total four neighbor nodes of the present node are considered from the parents, and the (unvisited) node having the largest cost is selected, because it is best at present. If either this best node or all neighbour nodes are available in the offspring, then any other unvisited node is chosen randomly. GX produces one offspring only from the parents. We consider the same chromosomes P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7).

We have the initial offspring (1). The nodes 5 and 8 are neighbour nodes of node 1 with their costs 66 and 2 respectively. Having higher cos, the node 5 is better, so, it is added to the offspring: (1, 5).

The nodes 4, 1, 6 and 4 are neighbour nodes of node 5 with their costs 31, 60, 50 and 31 respectively. Though the node 1 is the best, since it is available in the offspring, node 2 is chosen randomly and added to the offspring: (1, 5, 2).

The nodes 3, 8, 7 and 6 are neighbour nodes of node 2 with their costs 55, 58, 1 and 72 respectively. Node 6 is added to the offspring, as it the best node: (1, 5, 2, 6).

The nodes 3, 2 and 5 are neighbour nodes of node 6 with their costs 20, 82 and 39 respectively. Though node 2 is the best, since it is available in the offspring, node 3 is chosen randomly and added to the offspring: (1, 5, 2, 6, 3). Finally, the complete offspring may be: (1, 5, 2, 6, 3, 4, 7, 8) with cost 13.

8) Sequential constructive crossover operator. The sequential constructive crossover (SCX) operator creates only one offspring by using better arcs available in the parents' structure ([16], [36]). Additionally, sometimes it uses better arcs those are not available in either of the parents' structure. It sequentially searches both parent chromosomes and selects first legitimate (unvisited) node that appears after the present node. If no any legitimate node is available in either of the parents, it sequentially searches from the beginning of the chromosome and then compares their associated cost to decide the next node of the offspring chromosome. This operator is found to be very effective for the TSP and some other problems ([37]-[40]). The SCX is slightly modified for the MSTSP as below:

Step 1: Start from 'node 1' (i.e., current node p =1).

Step 2: Sequentially search both parent chromosomes and consider the first 'legitimate node' (the node that is not yet visited) appeared after 'node p' in each parent. If no 'legitimate node' after 'node p' is present in any of the parents, search sequentially from the starting of the parent and consider the first 'legitimate node', and go to Step 3.

Step 3: Suppose the 'node α' and the 'node β' are found in 1st and 2nd parent respectively, then for selecting the next node go to Step 4.

Step 4: If $c_{p\alpha} > c_{p\beta}$, then select 'node α', otherwise, 'node β' as the next node and concatenate it to the partially constructed offspring chromosome. If the offspring is a complete chromosome, then stop, otherwise, rename the present node as 'node p' and go to Step 2.

We consider the same example P_1: (1, 5, 4, 7, 8, 2, 3, 6) and P_2: (1, 8, 3, 4, 5, 6, 2, 7). Node 1 is the 1st gene. After node 1, nodes 5 in P_1 and 8 in P_2 are legitimate nodes with costs c_{15}=66 and c_{18}=2. Since $c_{15}>c_{18}$, node 5 is accepted and the offspring becomes (1, 5). =

After node 5, nodes 4 in P_1 and 8 in P_2 are legitimate nodes with costs c_{54}=31 and c_{56}=50. Since $c_{56}>c_{54}$, node 6 is accepted and the offspring becomes (1, 5, 6).

After node 6, nodes 4 in P_1 and 2 in P_2 are legitimate nodes with costs c_{64}=70 and c_{62}=82. Since $c_{62}>c_{64}$, node 2 is accepted and the offspring becomes (1, 5, 6, 2).

After node 2, nodes 3 in P_1 and 7 in P_2 are legitimate nodes with costs c_{23}=55 and c_{27}=1. Since $c_{23}>c_{27}$, node 3 is accepted and the offspring becomes (1, 5, 6, 2, 3).

After node 3, there is no legitimate node in P_1 and node 4 is legitimate in P_2. So, for P_1, search continues from its starting and finds same legitimate node 4 with c_{34}=69. So, node 4 is accepted and the offspring becomes (1, 5, 6, 2, 3, 4). Finally, offspring (1, 5, 6, 2, 3, 4, 7, 8) with cost 13 is obtained.

D. Mutation Operator

Mutation operator increases variety in the population by applying random changes in the population. For example, swap mutation, inversion mutation, insertion mutation, adaptive mutation [14], etc. are some of them. We have implemented swap mutation for our simple GAs.

E. Control Parameters

Control parameters rule the genetic process at some extent. They are - population size that decides number of chromosomes available during the process, crossover probability that fixes the probability of performing crossover between parents, mutation probability that fixes the probability of performing gene-wise mutation and stopping criterion that fixes when to stop the genetic process [16]. A simple GA may be summarized as follows:

```
SimpleGA( )
{ Initialize population randomly;
Evaluate the population;
Generation = 0;
While stopping criterion is not satisfied
{ Generation = Generation + 1;
Select better chromosomes by selection operator;
Perform crossover using crossover probability (Pc);
Perform mutation using mutation probability (Pm);
Evaluate the population;
 }
}
```

IV. C Computational Experiences and Discussions

To perform compare study among eight different crossover operators, simple GAs using these crossover operators have been encoded in Visual C++ on a Laptop with i7-1065G7 CPU@1.30 GHz and 8 GB RAM under MS Windows 10, and then run for twenty TSPLIB instances [41]. Out of the twenty, the nine instances ftv33, ftv38, ftv44, ft53, ftv64, ft70, ftv70, kro124p and ftv170 are asymmetric, and the remaining eleven instances dantzig42, eil51, st70, lin105, ch130, kroA150, si175, d198, pr226, a280 and lin318 are symmetric. We run GAs for different setting of parameters, and selected parameters are listed in Table III.

Fig. 1 presents results for ftv170 (by considering only 100 generations) by all GAs. Each curve is for one crossover, and it shows improvement of current solution in the successive generations. The figure shows some variations of SCX and shows that SCX is the best. ERX also has some variations and is place in second position. But GX and AEX have no variations and get trapped in local maximum very quickly and shown to be the worst.

The comparative study among the eight simple GAs are summarized in two tables: Tables IV and VIII. These tables are prepared similarly: each row is for an instance and each column is for one GA using a particular crossover operator. The result is defined best solution cost, average solution cost, standard deviation (S.D.) of solution costs, and average convergence time (in second). The best result for a particular instance among all GAs is marked by bold face.

TABLE III. Parameters for the GAs

Parameters	Values
Population size	50
Crossover probability	100%
Mutation probability	10%
Termination criterion	1,000 generations
No. of runs for each instance	50 times

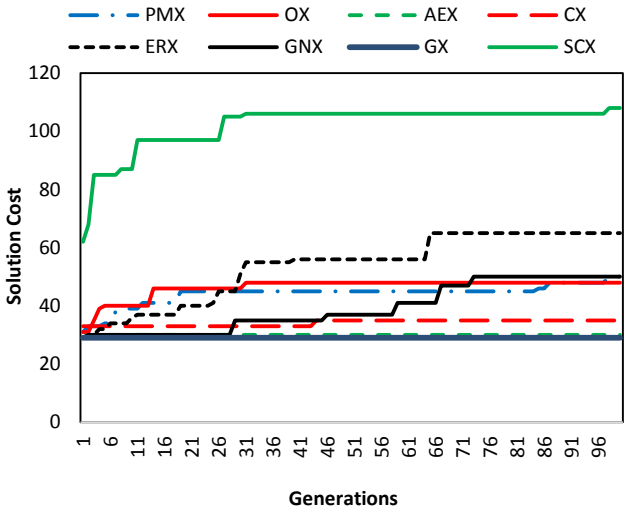

Fig. 1. Result by GAs using different Crossover Operators for ftv170.

From the Table IV, it is seen that the crossovers OX, AEX, CX, ERX and GX could not obtain either best solution or best average cost for any asymmetric instance. The crossover PMX obtains best average costs with lowest S.D. for the instances ftv33, ftv38 and ftv44, whereas SCX obtains best lowest average costs with lowest S.D. for the remaining six instances. So, SCX is shown to be the best. These results are shown in Fig. 2 that also shows the usefulness of crossover SCX. The crossovers ERX and GNX are competing, and GX is the worst.

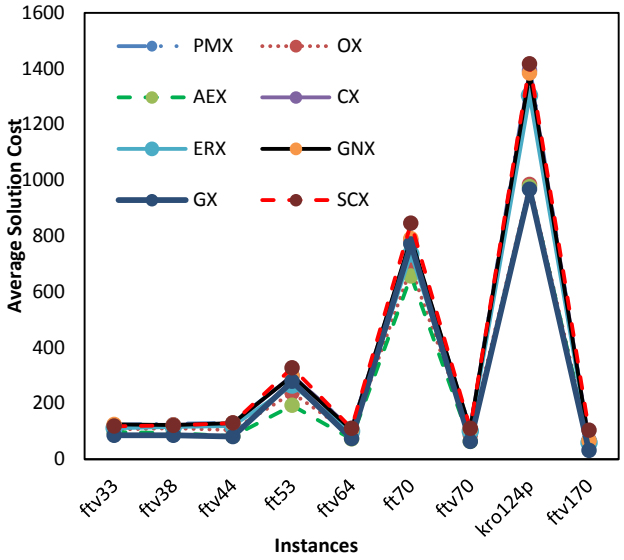

Fig. 2. Average Solution Cost by different GAs for Asymmetric Instances.

To confirm whether SCX-based GA average is statistically and significantly different from the averages found by other crossover-based GAs, Student's t-test is performed. It is to be mentioned that 50 runs have been performed for each instance. Following t-test formula is used here [42]:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{SD_1^2}{n_1 - 1} + \frac{SD_2^2}{n_2 - 1}}}$$

where,

$\bar{X}_1 - average\ of\ first\ sample,$

$SD_1 - standard\ deviation\ of\ first\ sample,$

$\bar{X}_2 - average\ of\ second\ sample,$

$SD_2 - standard\ deviation\ of\ second\ sample,$

$n_1 - first\ sample\ size,$

$n_2 - second\ sample\ size,$

The values of $\bar{X}_2$ and SD_2 are found by the SCX-based GA, and $\bar{X}_1$ and SD_1 values are found by other GAs. The t-statistic are reported in Table V. The t-values may be positive or negative. Since the problem is maximization problem, the negative value shows that SCX found better solution than the competitive crossover. In the positive case, the competitive crossover found better solution. The confidence interval at the

95% confidence level ($t_{0.05}$ = 1.96) is used. When t-value is greater than 1.96, the difference between them is significant. In this condition, if t-vale is negative then SCX-based GA solution is better, otherwise the competitive crossover-based GA solution is better. If t-value is less than 1.96, then there is no significant difference between the obtained values. The table also shows the information about the crossovers that found significantly better solutions.

On four instances there is no statistically significant difference between SCX and PMX. On three instances SCX is found better than PMX, whereas, PMX is found better than SCX on two instance. There is no significant difference between SCX and CX on two instances. On five instances SCX performed better than CX, whereas, on two instances CX is better than SCX. Next, there is no significant difference between SCX and GNX on three instances. SCX is better than GNX on five instances, whereas, GNX is better than SCX on only one instance. On all nine instances, SCX is found better than OX, AEX and ERX. From this study we can say that SCX is the best for asymmetric instances.

TABLE IV. COMPARATIVE STUDY OF 8 CROSSOVER-BASED GAS FOR ASYMMETRIC TSPLIB INSTANCES

Instance	n	Result	PMX	OX	AEX	CX	ERX	GNX	GX	SCX
ftv33	34	Best Sol	**134**	122	107	133	122	133	99	125
		Avg. Sol	**123.25**	106.25	96.8	121	111.4	124.4	85.5	118.3
		S.D.	6.72	8.1	11.25	6.16	5.57	5.64	11.15	5.04
		Avg. Time	0.02	0.04	0.07	0.06	0.85	0.04	0.08	0.07
ftv38	39	Best Sol	**137**	121	114	135	125	136	103	133
		Avg. Sol	**126.70**	110.70	94.35	123.35	115.35	122.30	85.45	121.10
		S.D.	6.07	5.28	8.79	4.64	7.12	8.49	11.17	5.80
		Avg. Time	0.03	0.05	0.07	0.08	1.14	0.04	0.11	0.11
ftv44	45	Best Sol	140	125	94	142	130	**143**	99	137
		Avg. Sol	**130.6**	104.7	82.45	125.40	118.80	129.00	81.20	129.85
		S.D.	6.56	8.68	8.87	8.00	5.48	7.85	10.07	5.34
		Avg. Time	0.03	0.05	0.08	0.10	1.44	0.05	0.14	0.14
		Best Sol	329	275	223	321	286	327	323	**360**
ft53	53	Avg. Sol	299.35	237.50	192.95	300.30	265.30	297.90	279.00	**327.80**
		S.D.	15.07	12.66	12.51	15.46	11.59	16.16	31.84	10.98
		Avg. Time	0.04	0.05	0.10	0.13	1.99	0.06	0.22	0.22
ftv64	65	Best Sol	**123**	106	85	122	110	118	88	120
		Avg. Sol	110.7	90.25	72.60	109.30	99.70	105.90	73.90	**110.90**
		S.D.	7.88	7.37	5.99	7.31	5.10	7.74	7.44	5.76
		Avg. Time	0.05	0.10	0.15	0.19	2.87	0.06	0.28	0.31
ft70	70	Best Sol	816	707	673	823	768	822	816	**884**
		Avg. Sol	778.95	685.95	656.60	785.3	735.35	791.05	770.85	**845.85**
		S.D.	25.70	7.67	11.01	17.57	24.53	23.2	26.78	17.90
		Avg. Time	0.06	0.07	0.17	0.22	3.45	0.07	0.30	0.32
ftv70	71	Best Sol	121	111	85	118	108	**125**	81	**125**
		Avg. Sol	109.95	87	63.75	106.95	99.95	108.1	63.1	**110.70**
		S.D.	6.57	6.57	8.22	5.42	5.17	5.31	6.39	6.23
		Avg. Time	0.06	0.09	0.15	0.19	3.49	0.07	0.25	0.29
kro124p	100	Best Sol	1562	1083	1097	1486	1498	**1666**	1069	1553
		Avg. Sol	1406.50	984.35	976.30	1392.30	1302.80	1383.70	966.90	**1416.50**
		S.D.	82.81	43.62	50.12	51.6	83.35	95.47	52.42	81.07
		Avg. Time	0.09	0.14	0.29	0.42	6.98	0.11	0.54	0.68
ftv170	171	Best Sol	71	70	78	71	63	73	37	**112**
		Avg. Sol	63.90	62.40	59.95	67.95	59.35	65.60	31.20	**104.15**
		S.D.	2.74	3.43	16.02	1.83	1.80	2.62	2.68	4.09
		Avg. Time	0.16	0.31	0.28	0.66	13.81	0.11	0.06	1.36

TABLE V. THE T-VALUES AGAINST SCX AND THE INFORMATION ABOUT CROSSOVERS THAT FOUND SIGNIFICANTLY BETTER SOLUTIONS

Instance	PMX	OX	AEX	CX	ERX	GNX	GX
ftv33	4.13	-8.84	-12.21	2.37	-6.43	5.65	-18.76
Better	PMX	SCX	SCX	CX	SCX	GNX	SCX
ftv38	4.67	-9.28	-17.78	2.12	-4.38	0.82	-19.83
Better	PMX	SCX	SCX	CX	SCX	---	SCX
ftv44	0.62	-17.27	-32.05	-3.24	-10.11	-0.63	-29.88
Better	---	SCX	SCX	SCX	SCX	---	SCX
ft53	-10.68	-37.72	-56.71	-10.15	-27.40	-10.71	-10.14
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
ftv64	-0.14	-15.45	-32.26	-1.20	-10.19	-3.63	-27.53
Better	---	SCX	SCX	---	SCX	SCX	SCX
ft70	-14.95	-57.48	-63.04	-16.90	-25.47	-13.09	-16.30
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
ftv70	-0.58	-18.32	-31.86	-3.18	-9.29	-2.22	-37.34
Better	---	SCX	SCX	SCX	SCX	SCX	SCX
kro124p	-0.60	-32.86	-32.33	-1.76	-6.85	-1.83	-32.60
Better	---	SCX	SCX	---	SCX	---	SCX
ftv170	-57.23	-54.75	-18.71	-56.55	-70.18	-55.56	-104.43
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX

To rank the other crossover operators, the t-values against PMX is calculated and reported in Table VI. There is no significant difference found between PMX and GNX on five instances. Each of them performed better than the other one on two instances. There is no significant difference found between PMX and CX on five instances. On three instances PMX is found better than CX, whereas, CX is found better than PMX on one instance. It shows that PMX and GNX are sharing 2nd rank. We further carried out an adequate statistical analysis. The results of our hypotheses testing are summarized in Table VII. In the table, each row contains two columns, where the first lists a crossover operator and the second column lists its inferior crossover operators. Each crossover is ranked according to its number of inferior crossover operators. No significant difference is found between AEX and GX, and hence, they share the worst rank.

From the Table VIII, it is seen that the crossovers OX, AEX, CX and GX could not obtain either best solution or best average cost for any asymmetric instance. The crossover PMX and ERX obtain best average costs with lowest S.D. for the instances eil51 and dantzig42 respectively, whereas SCX obtains best lowest average costs with lowest S.D. for the remaining nine instances. So, the crossover SCX is found to be the best.

The results are shown in Fig. 3 that also shows the usefulness of SCX. The crossovers ERX, OX, CX and GNX are competing, and GX is the worst. Based on this study also one can say that SCX is the best and GX is the worst, and others are competing.

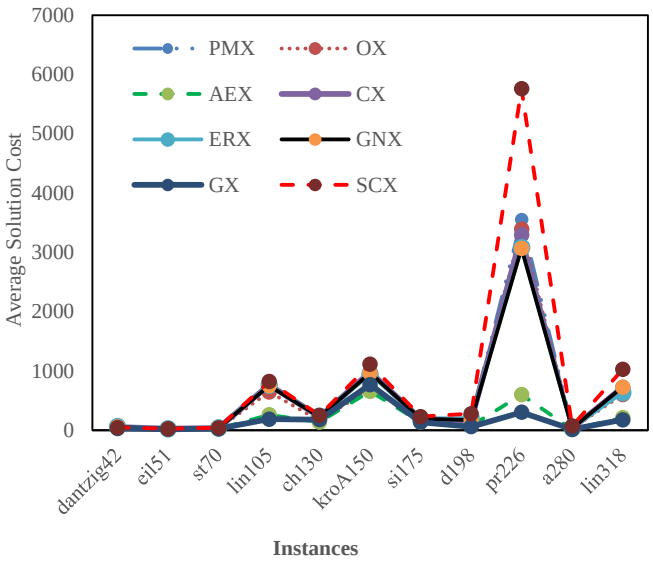

Fig. 3. Average Solution Cost by different GAs for Symmetric Instances.

For these symmetric instances also, to confirm whether SCX-based GA average solution is significantly different from the average solution found by other GAs, Student's t-test is performed, and the calculated t-values are reported in the Table IX.

On two instances there is no statistically significant difference between SCX and ERX. On eight instances SCX is better than ERX, whereas, ERX is better than SCX on one instance only. On one instance, there is no significant difference between SCX and (PMX, OX, CX and GNX). SCX performed better than PMX, CX and GNX on nine instances,

whereas, PMX, CX and GNX are better than SCX on only one instance. Next, on one instance there is no statistically significant difference between SCX and OX. On remaining ten instances SCX is better than OX. From this study we can conclude that SCX is the best. However, to rank the other crossover operators, an adequate statistical analysis is carried out, and the results are summarized in Table X. The crossovers PMX, ERX, GNX, CX, OX, AEX and GX are placed in the 2nd, 3rd, 4th, 5th, 6th, 7th and worst rank, respectively. On both kind of problem instance, SCX is placed the 1st rank, PMX is in the 2nd rank and GX is in the worst rank.

TABLE VI. THE T-VALUES AGAINST PMX AND THE INFORMATION ABOUT CROSSOVERS THAT FOUND SIGNIFICANTLY BETTER SOLUTIONS

Instance	**OX**	**AEX**	**CX**	**ERX**	**GNX**	**GX**
ftv33	-11.31	-14.13	-1.73	-9.50	0.92	-20.30
Better	PMX	PMX	----	PMX	---	PMX
ftv38	-13.92	-21.20	-3.07	-8.49	-2.95	-22.71
Better	PMX	PMX	PMX	PMX	PMX	PMX
ftv44	-16.66	-30.55	-3.52	-9.66	-1.09	-28.77
Better	PMX	PMX	PMX	PMX	---	PMX
ft53	-22.00	-38.03	0.31	-12.54	-0.46	-4.04
Better	PMX	PMX	---	PMX	---	PMX
ftv64	-13.27	-26.94	-0.91	-8.20	-3.04	-23.77
Better	PMX	PMX	---	PMX	PMX	PMX
ft70	-24.27	-30.63	1.43	-8.59	2.45	-1.53
Better	PMX	PMX	---	PMX	**GNX**	---
ftv70	-17.29	-30.73	-2.47	-8.37	-1.53	-35.78
Better	PMX	PMX	PMX	PMX	---	PMX
kro124p	-31.57	-31.11	-1.02	-6.18	-1.26	-31.40
Better	PMX	PMX	---	PMX	---	PMX
ftv170	-2.39	-1.70	8.60	-9.72	3.14	-59.72
Better	PMX	---	**CX**	PMX	**GNX**	PMX

TABLE VII. RESULTS OF STATISTICAL HYPOTHESES TESTING ON ASYMMETRIC INSTANCES

Crossover	**Inferior crossovers**
SCX	PMX, OX, AEX, CX, ERX, GNX, GX
PMX	OX, AEX, CX, ERX, GX
GNX	OX, AEX, CX, ERX, GX
CX	OX, AEX, ERX, GX
ERX	OX, AEX, GX
OX	AEX, GX
AEX	----
GX	----

TABLE VIII. COMPARATIVE STUDY OF 8 CROSSOVER-BASED GAS FOR SYMMETRIC TSPLIB INSTANCES

Instance	n	Results	PMX	OX	AEX	CX	ERX	GNX	GX	SCX
dantzig42	42	Best Sol	65	54	50	62	**69**	62	41	52
		Avg. Sol	57.80	48.75	39.50	53.85	**61.30**	55.50	31.70	49.05
		S.D.	3.78	2.66	4.02	4.29	3.27	4.63	4.28	1.28
		Avg. Time	0.03	0.04	0.07	0.09	1.32	0.04	0.10	0.13
eil51	51	Best Sol	29	25	24	30	29	**31**	21	30
		Avg. Sol	**25.75**	22.10	20.90	25.25	26.00	25.60	18.40	25.60
		S.D.	2.02	1.26	2.12	1.89	1.26	2.40	1.62	1.88
		Avg. Time	0.04	0.05	0.08	0.10	1.47	0.05	0.17	0.13
st70	70	Best Sol	**48**	42	30	44	**48**	45	33	**48**
		Avg. Sol	41.00	32.70	25.55	37.20	41.90	38.55	27.00	**43.15**
		S.D.	3.91	2.81	2.60	2.62	3.69	3.38	2.68	2.65
		Avg. Time	0.06	0.07	0.15	0.23	3.29	0.07	0.29	0.37
lin105	105	Best Sol	906	736	385	850	857	832	282	**914**
		Avg. Sol	768.40	645.80	261.85	765.45	760.95	751.05	189.45	**822.30**
		S.D.	62.20	59.35	66.69	46.63	51.11	45.59	29.66	57.26
		Avg. Time	0.09	0.18	0.10	0.45	8.20	0.10	0.70	0.75
ch130	130	Best Sol	265	173	163	269	253	250	208	**273**
		Avg. Sol	236.00	160.35	140.80	228.60	234.40	227.65	180.35	**251.90**
		S.D.	21.79	7.09	11.45	19.10	13.23	13.20	19.18	12.94
		Avg. Time	0.11	0.20	0.44	0.69	12.10	0.13	1.22	1.19
kroA150	150	Best Sol	1147	791	719	1142	1053	1094	899	**1238**
		Avg. Sol	997.85	706.20	657.65	983.45	944.85	966.10	771.50	**1113.20**
		S.D.	73.05	40.69	39.32	65.35	56.88	65.94	71.82	67.41
		Avg. Time	0.13	0.29	0.51	0.94	16.64	0.15	1.42	1.45
si175	175	Best Sol	211	193	149	211	208	211	149	**248**
		Avg. Sol	196.15	186.60	136.85	189.95	197.10	195.20	136.85	**231.40**
		S.D.	9.93	3.44	5.46	11.74	7.39	10.49	5.46	10.73
		Avg. Time	0.13	0.31	0.14	0.81	15.24	0.13	0.09	1.81
d198	198	Best Sol	265	198	114	218	234	217	92	**309**
		Avg. Sol	191.70	168.15	75.10	165.75	207.70	173.15	62.50	**279.10**
		S.D.	39.29	13.54	16.60	24.58	17.31	28.52	11.77	21.89
		Avg. Time	0.15	0.36	0.06	1.50	23.23	0.16	0.16	1.90
pr226	226	Best Sol	4887	3640	1650	4014	3796	3790	522	**6540**
		Avg. Sol	3555.40	3394.15	600.75	3295.50	3080.35	3069.40	304.30	**5761.80**
		S.D.	595.65	133.00	352.07	389.34	427.27	588.77	83.88	319.06
		Avg. Time	0.18	0.73	0.29	1.89	33.15	0.21	0.25	1.89
a280	280	Best Sol	54	40	23	49	45	45	18	**88**
		Avg. Sol	39.50	34.50	16.50	32.65	35.70	32.85	15.25	**72.90**
		S.D.	6.84	2.01	2.67	7.70	4.06	6.51	1.95	5.44
		Avg. Time	0.20	0.63	0.13	2.35	34.78	0.19	0.09	3.70
lin318	318	Best Sol	860	721	346	860	850	813	202	**1151**
		Avg. Sol	742.15	602.8	213.5	723.6	635.75	726.6	176.3	**1027.6**
		S.D.	75.84	57.18	51.71	81.4	64.36	48.88	12.08	72.64
		Avg. Time	0.26	0.85	0.3	3.59	59.5	0.29	0.3	4.73

TABLE IX. THE T-VALUES AGAINST SCX AND THE INFORMATION ABOUT CROSSOVERS THAT FOUND SIGNIFICANTLY BETTER SOLUTIONS

Instance	**PMX**	**OX**	**AEX**	**CX**	**ERX**	**GNX**	**GX**
dantzig42	15.35	-0.71	-15.85	7.51	24.42	9.40	-27.19
Better	**PMX**	---	SCX	**CX**	**ERX**	**GNX**	SCX
eil51	0.38	-10.83	-11.61	-0.92	1.24	0.00	-20.31
Better	---	SCX	SCX	---	---	---	SCX
st70	-3.19	-18.94	-33.19	-11.18	-1.93	-7.50	-30.00
Better	SCX	SCX	SCX	SCX	---	SCX	SCX
lin105	-4.46	-14.98	-44.63	-5.39	-5.60	-6.81	-68.70
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
ch130	-4.39	-43.43	-45.01	-7.07	-6.62	-9.18	-21.65
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
kroA150	-8.12	-36.18	-40.86	-9.67	-13.36	-10.92	-24.28
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
si175	-16.88	-27.83	-54.97	-18.24	-18.43	-16.89	-54.97
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
d198	-13.60	-30.17	-51.98	-24.11	-17.91	-20.63	-61.01
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
pr226	-22.86	-47.95	-76.04	-34.30	-35.20	-28.14	-115.80
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
a280	-26.75	-46.35	-65.15	-29.88	-38.36	-33.05	-69.83
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX
lin318	-19.03	-32.17	-63.91	-19.51	-28.26	-24.06	-80.92
Better	SCX	SCX	SCX	SCX	SCX	SCX	SCX

TABLE X. RESULTS OF STATISTICAL HYPOTHESES TESTING ON SYMMETRIC INSTANCES

Crossover	**Inferior crossovers**
SCX	PMX, OX, AEX, CX, ERX, GNX, GX
PMX	OX, AEX, CX, ERX, GNX, GX
ERX	OX, AEX, CX, GNX, GX
GNX	OX, AEX, CX, GX
CX	OX, AEX, GX
OX	AEX, GX
AEX	GX

V. CONCLUSION AND FUTURE WORKS

Numerous crossover operators have been proposed for the TSP using GAs which can also be used for its variations. In this paper, eight simple GAs using eight different crossover operators, namely PMX, OX, AEX, CX, ERX, GNX, GX and SCX, have been developed for solving the MSTSP. We first applied these operators in manual experiment on two parent chromosomes to produce an offspring, for each crossover operator. We then run the algorithms run on TSPLIB instances of different types and sizes. We set highest crossover probability to show exact nature of crossover operators. We carried out comparative study of the GAs on nine asymmetric and eleven symmetric TSPLIB instances. In terms of solution quality, our comparative study showed that crossover operator SCX is the best, PMX is the second-best and GX is the worst. Our observation is confirmed using Student's t-test at 95% confidence level. Thus, SCX may be good crossover operator to obtain more accurate results, researchers may apply it for other related combinatorial optimization problems. However, it is seen that PMX is better than SCX for small-sized instances.

In this study, our aim was to compare the solution quality found using different crossover operators, neither to improve the solution quality nor to develop the most competitive algorithm for the MSTSP. So, neither any local search technique is used to improve the solution quality nor parallel version of algorithms is developed to find exact solution. Therefore, we have developed simple and pure GAs. Thus, modified SCX operators ([43]-[45]) can be used instead of SCX and then good local search and immigration procedures [46] can be incorporated to hybridize the algorithm to solve the instances more accurately, which is under our investigation.

ACKNOWLEDGMENT

The author is very much thankful to the honourable anonymous reviewers for their constructive comments and suggestions which helped the author to improve this paper. This research was supported by Deanery of Academic Research, Al-Imam Mohammad Ibn Saud Islamic University, Saudi Arabia vide Grant No. 18-11-09-010. The author is also thankful to the Deanery for its financial support.

REFERENCES

[1] E.M. Arkin, Y.-J. Chiang, J.S.B. Mitchell, S.S. Skiena, and T.-C. Yang, "On the maximum scatter traveling salesperson problem," SIAM Journal of Computing, vol. 29, pp. 515–544, 1999.

[2] Z.H. Ahmed, "A hybrid genetic algorithm for the bottleneck traveling salesman problem," ACM Transactions on Embedded Computing Systems, vol. 12, Art. No. 9, 2013.

[3] F. Scholz, "Coordination hole tolerance stacking," Technical Report BCSTECH-93-048, Boeing Computer Services, November 1993.

[4] L.R. John, "The bottleneck traveling salesman problem and some variants," Master of Science of Simon Fraser University, Canada, 2010.

[5] J. LaRusic and A.P. Punnen, "The asymmetric bottleneck traveling salesman problem: Algorithms, complexity and empirical analysis," Computers & Operations Research, vol. 43, pp. 20–35, 2014.

[6] J.D.C. Little, K.G. Murthy, D.W. Sweeny, and C. Kare, "An algorithm for the travelling salesman problem," Operations Research, vol. 11, pp. 972-989, 1963.

[7] S.N.N. Pandit, "The Loading Problem," Operations Research, vol. 11, pp. 639-646, 1962.

[8] D. Applegate, R.E. Bixby, V. Chv´atal and W. Cook, "On the solution of traveling salesman problems," Documenta Mathematica, Extra Vol. ICM III, pp. 645-656, 1998.

[9] D.S Johnson and L.A. McGeoch, "The traveling salesman problem: a case study," in E. Aarts, J.K. Lenstra, eds. Local Search in Combinatorial Optimization. Wiley, Chichester, UK. Pp. 215-310, 1997.

[10] J.W. Ohlmann and B.W. Thomas, "A compressed-annealing heuristic for the traveling salesman problem with time windows," INFORMS Journal of Computing, vol. 19, no. 1, pp. 80–90, 2007.

[11] W.B. Carlton and J.W. Barnes, "Solving the travelling salesman problem with time windows using tabu search," IEE Transaction, vol. 28, pp. 617–629, 1996.

[12] M. Gendreau, A. Hertz, G. Laporte and M. Stan, "A generalized insertion heuristic for the traveling salesman problem with time windows," Operations Research, vol. 46, no. 3, pp. 330–335, 1998.

[13] C.-B. Cheng and C.-P. Mao, "A modified ant colony system for solving the travelling salesman problem with time windows," Mathematical Computer Modelling, vol. 46, pp. 1225–1235, 2007.

[14] D.E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," Addison-Wesley, New York, 1989.

[15] R.F. da Silva and S. Urrutia, "A general VNS heuristic for the traveling salesman problem with time windows," Discrete Optimization, vol. 7, no. 4, pp. 203–211, 2010.

[16] Z.H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," International Journal of Biometrics & Bioinformatics, vol. 3, pp. 96-105, 2010.

[17] Yi-J. Chiang, "New approximation results for the maximum scatter TSP," Algorithmica, vol. 41, pp. 309–341, 2005.

[18] S.N. Kabadi and A.P. Punnen, "The bottleneck TSP," In The Traveling Salesman Problem and Its Variations, G. Gutin and A.P. Punnen (eds.), Chapter 15, Kluwer Academic, Dordrecht, 2002.

[19] I. Hoffmann, S. Kurz, and J. Rambau, "The maximum scatter TSP on a regular grid," in Operations Research Proceedings 2015, Springer, 2015, pp. 63–70.

[20] G. Gutin and A.P. Punnen (eds.), "The Traveling Salesman Problem and Its Variations," Kluwer Academic, Dordrecht, 2002.

[21] Z.H. Ahmed, "A lexisearch algorithm for the bottleneck travelling salesman problem," International Journal of Computer Science and Security, vol. 3, no. 5, pp. 569-577, 2010.

[22] Z.H. Ahmed, "A data-guided lexisearch algorithm for the bottleneck travelling salesman problem," International Journal of Operational Research, vol. 12, no. 1, pp. 20-33, 2011.

[23] Z.H. Ahmed, "A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem," International Journal of Computational Intelligence Research, vol. 6, no. 3, pp. 475-484, 2010.

[24] Z.H. Ahmed, "A hybrid genetic algorithm for the bottleneck traveling salesman problem," ACM Transactions on Embedded Computing Systems, vol. 12, Art. No. 9, 2013.

[25] A. Barvinok, S.P. Fekete, D.S. Johnson, A. Tamir, G.J. Woeginger and R. Woodroofe, "The geometric maximum traveling salesman problem," Journal of the ACM, vol. 50, no. 5, pp. 641–664, 2003.

[26] Z.H. Ahmed, "An experimental study of a hybrid genetic algorithm for the maximum travelling salesman problem," Mathematical Sciences, vol. 7, pp. 1-7, 2013.

[27] W. Dong, X. Dong and Y. Wang, "The improved genetic algorithms for multiple maximum scatter traveling salesperson problems," In J. Li et al. (Eds.): CWSN 2017, CCIS 812, pp. 155–164, 2018.

[28] P. Venkatesh, A. Singh and R. Mallipeddi, "A multi-start iterated local search algorithm for the maximum scatter traveling salesman problem," in 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 2019, pp. 1390-1397.

[29] K. Deb, "Optimization for engineering design: algorithms and examples," Prentice Hall of India Pvt. Ltd., New Delhi, India, 1995.

[30] D.E. Goldberg, and R. Lingle, "Alleles, loci and the travelling salesman problem," In J.J. Grefenstette (ed.) Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum Associates, Hilladale, NJ, 1985.

[31] L. Davis, "Job-shop scheduling with genetic algorithms," Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 136-140, 1985.

[32] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Gucht, "Genetic algorithms for the traveling salesman problem," In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, (J. J. Grefenstette, Ed.), Lawrence Erlbaum Associates, Mahwah NJ, pp. 160–168, 1985.

[33] I.M. Oliver, D. J. Smith and J.R.C. Holland, "A study of permutation crossover operators on the travelling salesman problem," In J.J. Grefenstette (ed.). Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hilladale, NJ, 1987.

[34] D. Whitley, T. Starkweather and D. Shaner, "The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination," In L. Davis (Ed.) Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, pp. 350-372, 1991.

[35] N.J. Radcliffe and P.D. Surry, "Formae and variance of fitness," In D. Whitley and M. Vose (Eds.) Foundations of Genetic Algorithms 3, Morgan Kaufmann, San Mateo, CA, pp. 51-72, 1995.

[36] Z.H. Ahmed, "Improved genetic algorithms for the traveling salesman problem," International Journal of Process Management and Benchmarking, vol. 4, no. 1, pp. 109-124, 2014.

[37] Z.H. Ahmed, "The ordered clustered travelling salesman problem: A hybrid genetic algorithm," The Scientific World Journal, vol. 2014, Art ID 258207, 13 pages, 2014.

[38] Z.H. Ahmed, "A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem," Journal of Scientific & Industrial Research, vol. 73, pp. 763-766, 2014.

[39] Z.H. Ahmed, "The minimum latency problem: a hybrid genetic algorithm," IJCSNS International Journal of Computer Science and Network Security, vol. 18, no. 11, pp. 153-158, 2018.

[40] Z.H. Ahmed, "Performance analysis of hybrid genetic algorithms for the generalized assignment problem," IJCSNS International Journal of Computer Science and Network Security, vol. 19, no. 9, pp. 216-222, 2019.

[41] G. Reinelt, TSPLIB, http://comopt.ifi.uni-heidelberg.de/ software/ TSPLIB95/

[42] M. Nikolić and D. Teodorović, "Empirical study of the bee colony optimization (BCO) algorithm," Expert Systems with Applications, vol. 40, pp. 4609–4620, 2013.

[43] Z.H. Ahmed, "Solving the traveling salesman problem using greedy sequential constructive crossover in a genetic algorithm," IJCSNS International Journal of Computer Science and Network Security, vol. 20, no. 2, pp. 99-112, 2020.

[44] Z.H. Ahmed, "Adaptive sequential constructive crossover operator in a genetic algorithm for solving the traveling salesman problem," IJACSA International Journal of Advanced Computer Science and Applications, vol. 11, no. 2, pp. 593-605, 2020.

[45] Z.H. Ahmed, "Genetic algorithm with comprehensive sequential constructive crossover for the travelling salesman problem," IJACSA International Journal of Advanced Computer Science and Applications, vol. 11, no. 5, pp. 245-254, 2020.

[46] Z.H. Ahmed, "A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem," Cogent Engineering, vol. 5, Article 1423743, 2018.

AUTHOR'S PROFILE

Zakir Hussain Ahmed is a Full Professor in the Department of Mathematics and Statistics at Al Imam Mohammad Ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia. Till the end of 2019, he was in the Department of Computer Science at the same University. He obtained MSc in Mathematics (Gold Medalist), Diploma in Computer Application, MTech in Information Technology and PhD in Mathematical Sciences (Artificial Intelligence/Combinatorial Optimization) from Tezpur University (Central), Assam, India. Before joining the current position, he served in Tezpur University, Sikkim Manipal Institute of Technology, Asansol Engineering College and Jaypee Institute of Engineering and Technology, India. His research interests include artificial intelligence, combinatorial optimization, digital image processing and pattern recognition. He has several publications in the fields of artificial intelligence, combinatorial optimization and image processing.