Genetic Algorithms for the Multiple Travelling Salesman Problem

Maha Ata Al-Furhud[1], Zakir Hussain Ahmed[2]
Department of Computer Science, College of Computer and Information Sciences, Jouf University, Al-Jouf, KSA[1]
Department of Mathematics and Statistics, College of Science[2]
Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, KSA[2]
Department of Computer Science, College of Computer and Information Sciences[1, 2]
Al Imam Mohammad Ibn Saud Islamic University, Riyadh, KSA[1, 2]

***Abstract*—We consider the multiple travelling salesman Problem (MTSP) that is one of the generalization of the travelling salesman problem (TSP). For solving this problem genetic algorithms (GAs) based on numerous crossover operators have been described in the literature. Choosing effective crossover operator can give effective GA. Generally, the crossover operators that are developed for the TSP are applied to the MTSP. We propose to develop simple and effective GAs using sequential constructive crossover (SCX), adaptive SCX, greedy SCX, reverse greedy SCX and comprehensive SCX for solving the MTSP. The effectiveness of the crossover operators is demonstrated by comparing among them and with another crossover operator on some instances from TSPLIB of various sizes with different number of salesmen. The experimental study shows the promising results by the crossover operators, especially CSCX, for the MTSP.**

Keywords—Multiple travelling salesman problem; NP-hard; genetic algorithm; sequential constructive crossover; adaptive; greedy; comprehensive

I. Introduction

The travelling salesman problem (TSP) is one well-known multidisciplinary problem in operations research and computer science, which aims to find a least length (cost) Hamiltonian cycle (circuit) in a network of cities. The problem can be defined as: Given a set of cities (nodes) and the distances among them. Starting from and finishing at single depot city, a salesman should visit all remaining cities exactly once such that the total travelling distance (cost) by the salesman is minimized. The TSP has been extensively studied by several researchers, and hence, several useful approaches have been suggested to solve it. However, certain problems require additional salesman, and thus, the multiple TSP (MTSP) is defined to generalize the usual TSP. In MTSP, all salesmen begin from and finish their journey at a single depot city. Each city, except the depot city, should be visited by only one salesman such that the total travelling distance (cost) by all salesmen is minimized [1].

The MTSP can be formally defined as: Let there are m salesmen placed at single depot in a n-city network, d_{ij}, (i, j=1, 2, ..., n) be the distance (cost) between the cities i and j, and 'city 1' be the 'depot' with the remaining cities, 2, 3, …, n be the intermediate cities. Each of the salesmen is to start from the depot and after touring his set of cities should return to the depot. The tours should have no common cities (except the depot). The purpose is to obtain the optimum tour plan, i.e., the order of cities for each salesman, so that the total distance(cost) of the tour is minimum. Clearly, if m = 1, the problem becomes usual TSP.

The distance matrix may represent cost, time, etc. Depending on the nature of the distance matrix, the TSPs are divided into two types - asymmetric and symmetric. If $d_{ij} = d_{ji}$, $\forall$i, j, then it is symmetric; otherwise, asymmetric. For n-city usual asymmetric TSP, there are (n-1)! possible number of routes. So, for 5-city problem instance, there are 24 probable routes, and there are possibly 120 routes for 6-city problem. However, for 10-city problem, there are 362,880 possible routes, which is huge. Thus, the computational work is directly proportional to the problem size. It is very hard to solve large sized instances, if not impossible. In addition, the MTSP needs first to determine the cities allocated to each salesman, then to order the optimal sequence of cities in each salesman's tour, so, it is more complicated than TSP. Since, the TSP is NP-hard, hence, MTSP is also NP-hard [2].

The MTSP is the most challenging optimization problem in operations research and computer science paving the ways to various scheduling and routing problems. The MTSP seems to be more appropriate than the TSP for practical applications and can be used to simulate many real-life applications. The problem can be applied on job scheduling where multiple parallel production lines are present [3]. Also, the vehicle routing problem can be modelled as the MTSP. The MTSP can be applied to another kind of TSP variant where a salesman visits n cities over a period spanning m weeks but returns to the home city during weekends [4]. The school bus scheduling problem is an application of the MTSP that obtains a bus loading pattern so that the total number of ways is minimized, the total distance travelled by all buses is kept at minimum, no bus is overloaded and the time required to traverse any route does not surpass a maximum allowed policy [5]. Crew scheduling is another application of the MTSP as reported in [6], where investigated the problem to schedule multiple photographers' groups to many schools. The applications also include print press scheduling [4], interview scheduling [7], mission planning [8], and the design of global navigation satellite surveying system networks [9].

The MTSP may be extended to many variations [1]. Number of depots may be single or multiple. Similarly, paths (tours) may be closed or open. A closed path begins and ends at a single depot, whereas an open path does not require returning to the original depot. This paper considers the MTSP that allows all salesmen to start from same depot and end their tours at the original depot.

The MTSP is very difficult, and no any polynomial-time algorithm is available for the problem. So, finding its optimal solution is very tough, if not impossible. Hence, researchers are looking for finding better heuristic solutions within an acceptable computational time, rather, finding accurate optimal solutions to the MTSP as well as other difficult optimization problems. Therefore, one must go for heuristic methods for solving the MTSP. Artificial neural network (ANN) [10], simulated annealing (SA) [11], genetic algorithm (GA) [12], particle swarm optimization (PSO) [13], ant colony optimization (ACO) [14], etc. are a few such approaches.

In the recent years, several GAs have been developed successfully for various difficult optimization problems, for example the quadratic assignment problem [15], the minimum spanning tree problem [16], and the TSP [17]. GAs first developed by John Holland in 1970s that are based on survival-of-the-fittest theory among different species created by arbitrary variations in the chromosomes' structure in the biology. The GA is very successful because it is simple, flexible and easy to encode. A GA always begins with an initial chromosome population that goes through mainly three basic operations, namely selection, crossover and mutation, in successive generations to produce better populations. In selection method, chromosomes are probabilistically copied to the next (iteration) generation. Crossover selects randomly two parent chromosomes and mates them to form new offspring chromosome(s). Mutation occasionally alters value (gene) at a chromosome position. The crossover along with selection is the most influential process in genetic search. Mutation diverges the search space and defends genetic material losses that may resulted from selection and crossover operators. Hence, probability of implementing mutation operator is fixed very low, while probability of implementing crossover is fixed very high [18]. Out of three genetic operators, crossover is the most vital operator, and hence, several crossovers have been used in GAs for the MTSP which are proposed for the TSP. Still, most crossover operators do not lead good GA. Selecting good crossover can lead to a successful GA. An experimental study reported on six crossover operators in [19] showed that sequential constructive crossover (SCX) is the best operator. Recently, several modified versions of SCX, namely adaptive SCX (ASCX) [17], greedy SCX (GSCX) [20], reverse greedy SCX (RGSCX) [21] and comprehensive SCX (CSCX) [21], and were suggested for the TSP which showed very good results for the TSP.

In this study, we first reduce the MTSP to the TSP by introducing some artificial depots and then develop different simple GAs using five crossover operators - SCX, ASCX, GSCX, RGSCX and CSCX for the MTSP. These crossover operators are first applied manually on a pair of parents to create offspring(s). The effectiveness of the crossover operators is demonstrated by comparing among them and with two-part chromosome crossover (TCX) [12], on some instances from TSPLIB of various sizes with different number of salesmen. The comparative study shows the effectiveness of the crossover operators, especially CSCX, for the MTSP.

This paper is prearranged as follows. Section II reviews the related research. Simple genetic algorithms for the MTSP are described in Section III. The comparative study is described in Section IV. Finally, conclusions and future investigations are reported in Section V.

II. Literature Review

The MTSP is one of the most tough NP-hard problems. The MTSP is not well-studied like the usual TSP. For a detailed survey of MTSP and its variations, its practical applications, solution approaches proposed so far, the reader is referred to [1]. As this study proposes genetic algorithms (GAs) for solving the problem, we give literature survey on GAs, especially crossover operators used in different GAs for the MTSP.

For solving the MTSP, the first GAs were proposed in [6]. For solving the MTSP that models hot rolling scheduling in Shanghai Baoshan Iron and Steel Complex, GAs are proposed in [22]. First, the problem is modeled as an MTSP, which is then converted into usual TSP and finally, applied a modified GA for finding solution of the problem.

In [23], grouping GAs have been proposed for the MTSP that obtain better solutions. Additionally, another objective function that minimizes the maximum (cost) distance travelled by any single salesman is considered.

In addition, several crossover operators that were developed for the TSP have been modified and applied to the MTSP. The ordered crossover (OX) [24], cycle crossover (CX) [25], partially mapped crossover (PMX) [26], edge recombination crossover (ERX) [27], alternating edges crossover (AEX) [27] and sequential constructive crossover (SCX) [28] etc. are the most widely used crossover operators for the TSP. However, these crossover operators cannot be applied directly in GAs with the two-part chromosome representation.

In [12], the two-part chromosome crossover (TCX) that minimizes the size of search space of the MTSP is developed to find solution for the problem. A comparative study has been reported against three distinct crossover methods, namely OX+A, PMX+A and CX+A, for bi-objective MTSP that considers total distance and longest tour as objective functions to be minimized. As reported, TCX finds better solutions than other three crossover operators.

In [29], a modified two-part chromosome crossover is proposed for the problem. As reported, the algorithm assigns various number of cities for different salesman and obtained good solutions.

In [4], a combined crossover operator, OX + A (OX combined with an asexual crossover [30]) was proposed for the two-part chromosome method. The OX and the asexual crossover were employed for the 1st and 2nd parts respectively of the chromosome. That is, each part of the chromosome is considered and processed separately using two different crossover methods. The theoretical properties of the method is investigated as well as computational efficiency of the method is reported. As reported, the newer technique minimizes the search space and obtains better solutions.

In [19], one chromosome representation is used and applied six crossover operators, OX, PMX, CX, ERX, AEX and SCX for solving the MTSP. As reported, the SCX is the best operator.

III. Simple Genetic Algorithms for the MTSP

A. Reduced Distance Matrix

The multiple-TSP can be transformed to the single-TSP by assuming single salesman. Similarly, the problem with n cities and m salesmen can be transformed into the usual TSP with n+m-1 cities by adding m-1 artificial depots (namely, n+1, …, n+m-1) [31]. Further, the VRP can be transformed to the MTSP by deleting capacity constraints [32]. However, we transform the MTSP to the TSP by adding m-1 artificial depots. An example of a solution of the MTSP with n=8, m=2 is shown in Fig. 1(a), whereas its transformation to the TSP is depicted in Fig. 1(b).

Further, the original distance matrix and the reduced distance matrix with one artificial depot ‘city 9’, for the same problem are shown in Tables I and II, respectively.

In general, GAs are very successful heuristic methods in obtaining solutions for the usual TSP and its different variations. GAs do not assure that the obtained solutions are optimal, but they usually obtain better and near optimal solutions very quickly.

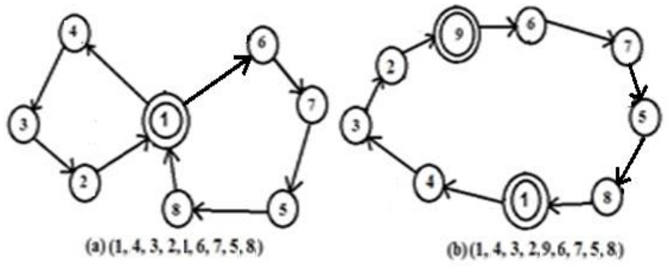
(a) (1, 4, 3, 2, 1, 6, 7, 5, 8)
(b) (1, 4, 3, 2, 9, 6, 7, 5, 8)

Fig. 1. Example of a Solution of the MTSP and its Transformation to the TSP with Artificial city 9.

TABLE I. The Distance Matrix

City	1	2	3	4	5	6	7	8
1	999	75	99	9	35	63	8	11
2	51	999	86	46	88	29	20	15
3	100	5	999	16	28	35	28	2
4	20	45	11	999	59	53	49	8
5	86	63	33	65	999	76	72	5
6	36	53	89	31	21	999	52	6
7	58	31	43	67	52	60	999	9
8	15	95	66	14	54	8	87	999

TABLE II. The Reduced Distance Matrix

City	1	2	3	4	5	6	7	8	9
1	999	75	99	9	35	63	8	11	999
2	51	999	86	46	88	29	20	15	51
3	100	5	999	16	28	35	28	2	100
4	20	45	11	999	59	53	49	8	20
5	86	63	33	65	999	76	72	5	86
6	36	53	89	31	21	999	52	6	36
7	58	31	43	67	52	60	999	9	58
8	15	95	66	14	54	8	87	999	15
9	999	75	99	9	35	63	8	11	999

B. Chromosome Representation

To use GA for any optimization problem, one should find a method to represent solutions by legal chromosomes so that crossover produces legal chromosomes. There are three representation methods used for chromosome for the MTSP. They are one chromosome method [22], two chromosomes method [33], and two-part chromosome method [34-35]. Brown et al. [36] proposed another chromosome representation method which is inspired from the chromosome representation described in [34]. This representation consists of two sections - main section and group section. The main section of the chromosome consists of n real-valued genes and group section consist of m integer-valued genes. The integer part of the real-valued gene i in the main section indicates the salesman assigned to city i, whereas fractional part determines the order in which city i is visited. Group section simply shows various groups exist in the solution in the way they appear in the main section. This is a deviation from the representation described in [34], where groups can appear in any order in the group section. Singh and Baghel [23] proposed another chromosome representation method which represents chromosome as a set of m tours, i.e., there is no sequence among the tours. As reported, GA using this representation method is found to be superior.

We are going to choose one-chromosome with artificial depots. An example of our chromosome for *n = 8* with *m = 2* is shown in Fig. 2.

1	4	3	2	9	6	7	5	8

Fig. 2. An Example of our One Chromosome for 8 Cities with 2 Salesmen.

In this chromosome, (1, 4, 3, 2, 9, 6, 7, 5, 8), there are (8+2-1=) 9 genes including artificial depot city 9. This chromosome represents the tour $\{1\rightarrow4\rightarrow3\rightarrow 2\rightarrow9\rightarrow6\rightarrow 7\rightarrow5\rightarrow8\rightarrow1\}$. That means, the 1st salesman visited cities 1, 4, 3 and 2 sequentially, and the 2nd salesman visited cities 6, 7, 5, and 8 sequentially. The objective function is defined as total distances travelled by the salesmen, which is (9+11+5+51+63+52+52+5+15=) 263 for this tour.

A set of random of chromosomes is generated initially (initial population) to start the genetic search process, which are then evaluated and gone through a selection procedure for creating mating pool. In GAs, the crossover operator executes very important role. Generally, the crossover operators developed for the usual TSP, are considered for its variations also. Numerous crossover operators are developed for the usual TSP and among them SCX, adaptive SCX, greedy SCX, reverse greedy SCX and comprehensive SCX are considered and discussed here.

C. Sequential Constructive Crossover Operator

In [28], the sequential constructive crossover (SCX) operator is developed for the TSP, and then is updated in [37]. It creates only one offspring at a time that uses better edges available in the parents. Moreover, it uses some better edges which are not available in either of the parents. The main characteristic of SCX is to sequentially examine the parents to consider first legitimate (unvisited) cities seen after the current city. If no legitimate city is seen in a parent, it examines from the beginning of (wrapping around) the parent. It then compares their distances from the present city to choose the next city of the child. It is effectively applied to various combinatorial optimization problems ([15], [38]-[41]). We apply this crossover operator for the MTSP. It is seen that the SCX may lead to infeasible chromosomes (tours). So, we applied swap algorithm on the infeasible chromosomes to make them feasible. The swap algorithm randomly selects two cities and exchanges them. We now illustrate the SCX using following parent chromosomes with 9-city and 2-salesman, P_1: (1, 5, 7, 2, 9, 4, 3, 6, 8) and P_2: (1, 4, 6, 3, 7, 9, 2, 5, 8) with total travel distances 265 and 420 respectively using the reduced distance matrix given in Table II. This SCX operator results single offspring from two parents.

City 1 is the 1st gene, and after this, 5 and 4 are the legitimate cities in P_1 and P_2 respectively with $d_{15}=35$ and $d_{14}=9$. As $d_{14}<d_{15}$, city 4 is accepted. So, the incomplete chromosome (offspring) becomes (1, 4).

After city 4, 3 and 6 are the legitimate cities in P_1 and P_2 respectively with $d_{43}=11$ and $d_{46}=53$. Since $d_{43}<d_{46}$, we accept city 3. So, the incomplete chromosome becomes (1, 4, 3).

After city 3, 6 and 7 are the legitimate cities in P_1 and P_2 respectively with $d_{36}=35$ and $d_{37}=28$. Since $d_{37}<d_{36}$, we accept city 7. So, the incomplete chromosome becomes (1, 4, 3, 7).

The legitimate cities after city 7 in P_1 & P_2 are 2 & 9 respectively with $d_{72}=31$ and $d_{79}=58$. Since $d_{72}<d_{79}$, we accept city 2. So, the incomplete chromosome becomes (1, 4, 3, 7, 2).

The legitimate cities after city 2 in P_1 & P_2 are 9 & 5 respectively with $d_{29}=51$ and $d_{25}=88$. Since $d_{29}<d_{25}$, we accept city 9. So, the incomplete chromosome becomes (1, 4, 3, 7, 2, 9).

The legitimate cities after city 9 in P_1 & P_2 are 6 & 5, respectively with $d_{96}=63$ and $d_{95}=35$. Since $d_{95}<d_{96}$, we accept city 5. So, the incomplete chromosome becomes (1, 4, 3, 7, 2, 9, 5). By following this way, one can create the complete offspring as: (1, 4, 3, 7, 2, 9, 5, 8, 6) with distance 214, which is better than both parents.

D. Adaptive Sequential Constructive Crossover Operator

In [17], a modified version of the SCX, named adaptive SCX (ASCX), is developed for the TSP which creates only one offspring adaptively, either in forward or backward or mixed direction depending on next city's distance. Eight neighbour (four from each parent) cities of any current city is considered, of which best city in either direction is selected for the offspring. Since number of genes in a chromosome is n, 'city 1' is fixed as 1st and (n+1)th (not shown in the chromosomes) genes. We apply this crossover operator also. For any infeasible chromosomes (tours), swap algorithm is applied to make it feasible. This ASCX is explained using the same example.

For our example, 'city 1' is fixed in 1st and 10th (not shown in the chromosomes) places. In P_1, after 'city 1' (1st gene), legitimate cities in forward and backward (after wrapping around) directions are 5 and 8, respectively; and in P_2 they are 4 and 8, respectively, with their distances 35, 11, 9 and 11 respectively. Among them, the city 4 with distance 9 is the cheapest. From the end, in P_1, before 'city 1' (10th gene), legitimate cities in backward and forward (after wrapping around) directions are 8 and 5, respectively; and in P_2 they are 8 and (after wrapping around) 4, respectively, with their distances 15, 86, 15 and 20, respectively. Among them, the city 8 with distance 15 is the cheapest. Then city 4 is added as the 2nd gene in the offspring, as it is cheaper between the cheapest cities. So, the offspring becomes (1, 4, *, *, *, *, *, *, *).

In P_1, after 'city 4' (2nd gene), legitimate cities in forward and backward directions are 3 and 9, respectively; and in P_2 they are (after wrapping around) 6 and 8, respectively, with their distances 11, 20, 53 and 8, respectively. Among them, the city 8 with distance 8 is the cheapest. From the end, in P_1, before 'city 1' (10th gene), legitimate cities in backward and forward (after wrapping around) directions are 8 and 5 respectively; and in P_2 they are 8 and (after wrapping around) 6 respectively, with their distances 15, 86, 15 and 36 respectively. Among them, the city 8 with distance 15 is the cheapest. Then city 8 is added as the 3rd gene in the offspring, as it is cheaper between the cheapest cities. So, the offspring becomes (1, 4, 8, *, *, *, *, *, *).

In P_1, after 'city 8' (3rd gene), legitimate cities in forward (after wrapping around) and backward directions are 5 and 6 respectively; and in P_2 they are (after wrapping around) 6 and 5, respectively, with their distances 54, 8, 8 and 54, respectively. Among them, the city 6 with distance 8 is the cheapest. From the end, in P_1, before 'city 1' (10th gene), legitimate cities in backward and forward (after wrapping around) directions are 6 and 5 respectively; and in P_2 they are 5 and (after wrapping around) 6, respectively, with their distances 36, 86, 86 and 36 respectively. Among them, the city 6 with distance 36 is the cheapest. Then city 6 is added as the 4th gene in the offspring, as it is cheaper between the cheapest cities. So, the offspring becomes (1, 4, 8, 6, *, *, *, *, *).

In P_1, after 'city 6' (4th gene), legitimate cities in forward (after wrapping around) and backward directions are 5 and 3 respectively; and in P_2 they are 3 and (after wrapping around) 5, respectively, with their distances 21, 89, 89 and 21 respectively. Among them, the city 5 with distance 21 is the cheapest. From the end, in P_1, before 'city 1' (10th gene), legitimate cities in backward and forward (after wrapping around) directions are 3 and 5, respectively; and in P_2 they are 5 and (after wrapping around) 3, respectively, with their distances 100, 86, 86 and 100, respectively. Among them, the city 5 with distance 86 is the cheapest. Then city 5 is added as the 5th gene in the offspring, as it is cheaper between the cheapest cities. So, the offspring becomes (1, 4, 8, 6, *, *, *, *, *). By following this way, one can create the complete offspring as: (1, 4, 8, 6, 5, 3, 2, 9, 7) having distance 201, which is better than both parents.

E. Greedy Sequential Constructive Crossover Operator

In [20], a greedy SCX (GSCX) is proposed for the TSP that modified the SCX as follows. While searching the 'legitimate city' seen after the current city, if 'legitimate city' is not found in a parent, then select the cheapest 'legitimate city' from the set of remaining legitimate cities and add it to the current incomplete offspring chromosome. We apply this crossover operator also. For any infeasible chromosomes (tours), swap algorithm is applied to make it feasible. This GSCX is explained using the same example.

As 'city 1' is the first gene, after this city, the legitimate citys in P_1 is 5 and in P_2 is 4 having d_{15}=35 and d_{14}=9. As $d_{14}<d_{15}$, the city 4 is added as the second gene in the current offspring that leads the incomplete offspring to (1, 4).

City 1 is the 1st gene, and after this, 5 and 4 are the legitimate cities in P_1 and P_2 respectively with d_{15}=35 and d_{14}=9. As $d_{14}<d_{15}$, city 4 is accepted. So, the incomplete chromosome (offspring) becomes (1, 4).

After city 4, 3 and 6 are the legitimate cities in P_1 and P_2 respectively with d_{43}=11 and d_{46}=53. Since $d_{43}<d_{46}$, we accept city 3. So, the incomplete chromosome becomes (1, 4, 3).

After city 3, 6 and 7 are the legitimate cities in P_1 and P_2 respectively with d_{36}=35 and d_{37}=28. Since $d_{37}<d_{36}$, we accept city 7. So, the incomplete chromosome becomes (1, 4, 3, 7).

The legitimate cities after city 7 in P_1 & P_2 are 2 & 9 respectively with d_{72}=31 and d_{79}=58. Since $d_{72}<d_{79}$, we accept city 2. So, the incomplete chromosome becomes (1, 4, 3, 7, 2).

The legitimate cities after city 2 in P_1 & P_2 are 9 & 5 respectively with d_{29}=51 and d_{25}=88. Since $d_{29}<d_{25}$, we accept city 9. So, the incomplete chromosome becomes (1, 4, 3, 7, 2, 9).

The legitimate cities after city 9 in P_1 & P_2 are 6 & 5 respectively with d_{96}=63 and d_{95}=35. Since $d_{95}<d_{96}$, we accept city 5. So, the incomplete chromosome becomes (1, 4, 3, 7, 2, 9, 5). By following this way, one can create the complete offspring as: (1, 4, 3, 7, 2, 9, 5, 8, 6) with distance 214, which is better than both parents.

F. Reverse Greedy Sequential Constructive Crossover Operator

In [21], the GSCX is modified for the TSP by applying it in reverse way and termed as reverse GSCX (RGSCX). It constructs an offspring in reverse way, which is, from the last city (gene) back to the first city (gene). We apply this crossover operator also. For any infeasible chromosomes (tours), swap algorithm is applied to make it feasible. This RGSCX is explained using the same example.

The 'city 1' is the 10th gene. In both P_1 and P_2, 9th genes are city 8, hence, it is added as the 9th gene to the offspring as (8).

Before city 8, in P_1 and P_2, the legitimate cities are 6 and 5 respectively with d_{68}=6 and d_{58}=5. As $d_{58}<d_{68}$, the city 5 is added at the 8th place in the offspring as (5, 8).

Before city 5, legitimate city is not present in P_1. So, the cheapest legitimate city 6, among the remaining cities, is added at the 7th place in the offspring as (6, 5, 8).

Before city 6, in P_1 and P_2, the legitimate cities are 3 and 4 respectively with d_{36}=35 and d_{46}=53. As $d_{36}<d_{46}$, the city 3 is added at the 6th place in the offspring as (3, 6, 5, 8).

Before 'city 3', in both P_1 and P_2, legitimate cities are city 4, hence, it is added at the 5th place in the offspring as (4, 3, 6, 5, 8).

Before city 4, legitimate city is not present in P_2. So, the cheapest legitimate city 9, among the remaining cities, is added at the 4th place in the offspring as (9, 4, 3, 6, 5, 8). By following this way, one can create the complete offspring as: (1, 7, 2, 9, 4, 3, 6, 5, 8) having distance 186, which is better than both parents.

G. Comprehesive Sequential Constructive Crossover Operator

The comprehensive SCX (CSCX) is proposed in [21] that combines GSCX and RGSCX to create two offspring. We apply this crossover operator also. By using same parents' example, the offspring (1, 4, 3, 7, 2, 9, 5, 8, 6) and (: (1, 7, 2, 9, 4, 3, 6, 5, 8) are created, with distances 214 and 186 respectively which are better both parents.

H. Swap Mutation Operator

The mutation operator generally selects a gene (position) randomly in a chromosome and modifies its corresponding allele (city). Since always weaker chromosomes in consecutive generations are rejected in previous operators in GA search, so, some better alleles could be lost permanently. Hence the mutation is used for recovering them. Generally, one can assume that mutation may help the other operators to overcome local optima and obtain better solution. For our simple GAs, the swap mutation operator [18] which randomly chooses two cities, except artificial depots, and swaps them.

I. Structure of our Simple GAs

Our GA is simple that uses traditional genetic procedures and operators without incorporating another heuristic method. Starting with randomly created population, stochastic remainder for selection, only one selected crossover and swap mutation are used in our simple GAs (SGAs) as follows.

```
SGA ()
{ Initialize population of size $P_s$ randomly;
  Evaluate the population;
  Generation = 0;
  While stopping criterion is not fulfilled
  { Generation = Generation + 1;
    Choose fitter chromosomes by stochastic remainder selection;
    Choose a crossover and perform crossover with probability $P_c$;
    Swap randomly chosen genes with probability $P_m$;
    Evaluate the population;
  }
}
```

IV. Computational Experiments

Our proposed SGAs using five crossover operators - SCX, ASCX, GSCX, RGSCX and CSCX, are encoded in Visual C++. In order to demonstrate the efficiency of the algorithm, computational experience is conducted on three benchmark TSPLIB instances [42] with different number of salesmen and run on a Laptop with i3-3217U CPU@1.80 GHz and 4 GB RAM under MS Windows 7. The instances MTSP-51, MTSP-100, and MTSP-150 used in [12] are considered for comparing our five crossovers with state-of-the-art crossover TCX reported in [12]. The success of GAs depends on proper selection of the GA parameters-termination criterion, crossover probability, population size, and mutation probability. But there is no intelligent way to select these parameters. One way to select them is by trial and error method. We run our SGAs for different setting of parameters, and selected parameters are listed in Table III.

We first compare our crossovers in SGAs with TCX [20] for three Euclidean symmetric instances – MTSP-51 with m=3, 5, 10; MTSP-100 with m=3, 5, 10, 20; and MTSP-150 with m=3, 5, 10, 20, 30. There is no restriction on the maximum of cities that a salesman can visit, however, each salesman should visit at least one city. The Table IV reports best solution (BestSol), average solution (AvgSol) and standard deviation (S.D) of the solution. For every instance, the best result over six crossovers is marked by boldface.

TABLE III. Parameter for our Proposed SGAs

Parameters	Values
Population size	50
Crossover probability	100%
Mutation probability	10%
Termination criterion	2000 generations
No. of runs for each instance	30 times

TABLE IV. Results by the Crossover Operators for Symmetric TSPLIB Instances

Instance	**m**	**Results**	**TCX**	**SCX**	**ASCX**	**GSCX**	**RGSCX**	**CSCX**
MTSP-51	**3**	Best Sol	466	456	**454**	460	457	458
		Avg. Sol	510	480	**464**	471	473	470
		S.D.	24	9	5	6	6	4
	5	Best Sol	499	488	486	481	**477**	490
		Avg. Sol	536	496	494	**490**	495	501
		S.D.	26	3	3	4	4	5
	10	Best Sol	602	621	596	**568**	**568**	580
		Avg. Sol	636	631	618	594	**587**	611
		S.D.	17	6	6	12	9	9
MTSP-100	**3**	Best Sol	28943	25107	24443	24875	24063	**22826**
		Avg. Sol	32708	25746	24750	25606	25507	**23731**
		S.D.	2267	477	181	470	512	405
	5	Best Sol	30941	26317	25900	25104	26037	**24196**
		Avg. Sol	34179	27250	26574	26798	27025	**25338**
		S.D.	2006	545	329	619	579	437
	10	Best Sol	32802	31149	30425	29721	**29181**	**29181**
		Avg. Sol	36921	32422	31246	31230	31227	**30256**
		S.D.	1964	565	439	702	874	541
	20	Best Sol	44112	44543	42407	42266	42665	**41465**
		Avg. Sol	46976	45989	44936	44032	44619	**43004**
		S.D.	1773	814	1268	898	1196	730
MTSP-150	**3**	Best Sol	51126	33404	30360	30824	30661	**29390**
		Avg. Sol	55851	34974	31627	32180	32260	**30921**
		S.D.	2588	930	676	531	537	553
	5	Best Sol	51627	34531	31151	32281	32459	**30308**
		Avg. Sol	61596	35844	32660	33403	33397	**31820**
		S.D.	4759	680	607	531	458	592
	10	Best Sol	54473	36514	**35510**	36576	35733	35802
		Avg. Sol	61360	38344	**36933**	37702	37150	37173
		S.D.	3888	643	592	638	835	643
	20	Best Sol	62456	48354	45564	46603	47096	**44697**
		Avg. Sol	69701	49450	47257	48560	49162	**46609**
		S.D.	4340	616	711	883	1064	796
	30	Best Sol	76481	63624	62565	61579	60758	**58757**
		Avg. Sol	84008	65812	65417	63760	64926	**61360**
		S.D.	5285	1389	2335	1198	1888	911

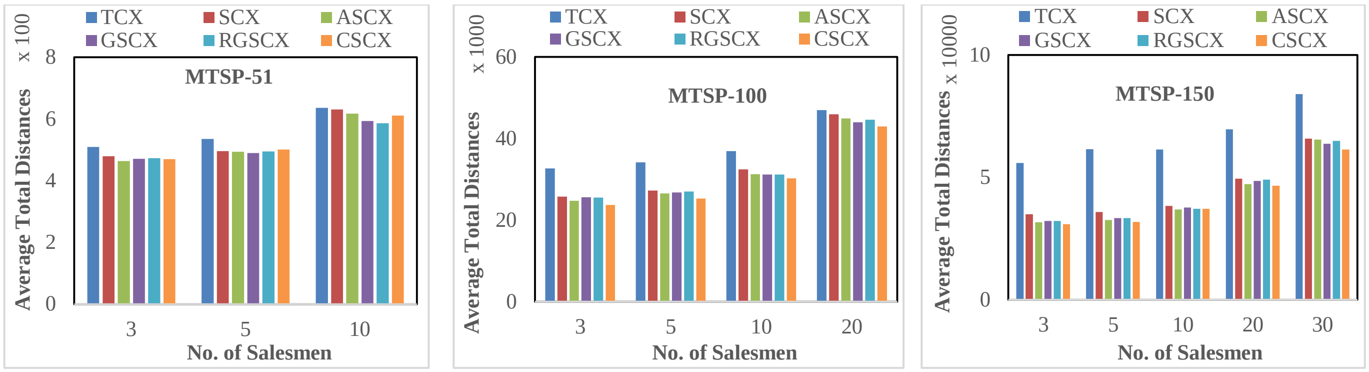

Fig. 3. Average Solution for the MTSP-51, MTSP-100 and MTSP-150 for different Crossover Operators.

The operators TCX and SCX could not find either lowest best or average solution for any instance. The operator GSCX finds lowest best solution for MTSP-51 with m=10 and lowest average solution for MTSP-51 with m=5; ASCX obtains lowest best and average solutions for MTSP-51 with m=3 and MTSP-150 with m=10; RGSCX obtains lowest best solutions for MTSP-51 with m=5,10, and MTSP-100 with m=10, and lowest average solution for MTSP-51 with m=10; and CSCX finds lowest best and average solutions for MTSP-100 with m=3, 5, 10, 20, and MTSP-150 with m=3, 5, 20, 30. From Table IV, it can be seen that CSCX gives more efficient results for most instances, and hence found to be very effective crossover operator. The results are showed in Fig. 3 that further demonstrates the effectiveness of the CSCX. From this study, it is concluded that CSCX is placed in 1st rank, ASCX in 2nd rank and TCX in the worst rank. To confirm these findings, statistical analysis is also carried out and found same results.

V. Conclusion and Future Works

In this study we considered the multiple travelling salesman problem (MTSP) which is a generalization of the travelling salesman problem (TSP). To solve this problem numerous genetic algorithms (GAs) based on numerous crossover operator have been reported in the literature. Choosing effective crossover operator can lead to effective GA. Generally, crossover operators that are proposed for the TSP are used for the MTSP also. We developed five simple GAs using sequential constructive crossover (SCX), adaptive SCX (ASCX), greedy SCX (GSCX), reverse greedy SCX (RGSCX) and comprehensive SCX (CSCX) for solving the MTSP. First, these crossover operators are applied manually on parent chromosomes to create offspring(s) and then encoded in Visual C++. The effectiveness of the crossover operators is demonstrated by comparing among them and with another crossover operator (TCX) on some instances from TSPLIB of various sizes with different number of salesmen. The experimental results show promising results by the crossover operator CSCX for the MTSP.

In this present study, we aimed to develop simple GAs using five crossovers and then compare among them and with a state-of-art crossover operator. We did not aim to develop improved and high quality GA. Though CSCX finds very good solutions, yet it gets trapped in local minimums in the early generations, and for small-sized instances, it does not show promising results. Therefore, effective local search and immigration methods could be combined to hybridize the simple GA for finding better solutions to more instances that is under our study.

Acknowledgment

The authors are very much thankful to the honourable anonymous reviewers for their constructive comments and suggestions which helped the authors to improve this paper. This research was funded by King Abdulaziz City for Science and Technology (KACST), Saudi Arabia, vide grant no. 1-17-00-008-0002. The authors are very much thankful to the KACST for its financial support.

References

[1] T. Bektas, “The multiple traveling salesman problem: An overview of formulations and solution procedures,” Omega, vol. 34, pp. 209–219, 2006.

[2] M.R. Garey, and D.S. Johnson, “A guide to the theory of np-completeness, Computers and intractability,” W. H. Freeman & Co., New York, NY, USA, 1990.

[3] A.E. Carter, and C.T. Ragsdale, “Scheduling pre-printed newspaper advertising inserts using genetic algorithms,” Omega, vol. 30, pp. 415–421, 2002.

[4] A.E. Carter, and C.T. Ragsdale, “A new approach to solving the multiple traveling salesperson problem using genetic algorithms,” European Journal of Operational Research, vol. 175, pp. 245–257, 2006.

[5] R.D. Angel, W.L. Caudle, R. Noonan, and A. Whinston, “Computer assisted school bus scheduling,” Management Science, vol. 18, pp. 279–288, 1972.

[6] T. Zhang, W.A. Gruver, and M.H. Smith, “Team scheduling by genetic search,” In Proceedings of the second international conference on intelligent processing and manufacturing of materials, 1999, vol. 2, pp. 839–844.

[7] K.C. Gilbert, and R.B. Hofstra, “A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem,” Decision Sciences, vol. 23, pp. 250–259, 1992.

[8] B. Brummit, and A. Stentz, “Dynamic mission planning for multiple mobile robots,” In Proceedings of the IEEE international conference on robotics and automation, April 1996.

[9] H.A. Saleh, and R. Chelouah, “The design of the global navigation satellite system surveying networks using genetic algorithms,” Engineering Applications of Artificial Intelligence, vol. 17, pp. 111–122, 2004.

[10] E. Wacholder, J. Han, and R.C. Mann, “A neural network algorithm for the multiple traveling salesmen problem,” Biology in Cybernetics, vol. 61, pp. 11–19, 1989.

[11] C. Song, K. Lee, and W.D. Lee, “Extended simulated annealing for augmented TSP and multi-salesmen TSP,” In Proceedings of the

international joint conference on neural networks, 2003, vol. 3, pp. 2340–2343.

[12] S. Yuan, B. Skinner, S. Huang, and D. Liu, "A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms," European Journal of Operational Research, vol. 228, pp. 72–82, 2013.

[13] X.S. Yan, C. Zhang, and W. Luo, "Solve traveling salesman problem using particle swarm optimization algorithm," International Journal of Computer Science, vol. 9(6), pp. 264–271, 2012.

[14] G. Singh, and R. Mehta, "Implementation of travelling salesman problem using ant colony optimization," Journal of Engineering Research and Application, vol. 6(3), pp. 385–389, 2014.

[15] Z.H. Ahmed, "A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem," Journal of Scientific & Industrial Research, vol. 73(12), pp. 763-766, 2014.

[16] K. Singh, and S. Sundar, "A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem," Soft Computing, vol. 24, pp. 2169–2186, 2020.

[17] Z.H. Ahmed, "Adaptive sequential constructive crossover operator in a genetic algorithm for solving the traveling salesman problem," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 11(2), pp. 593-605, 2020.

[18] D.E. Goldberg, "Genetic algorithms in search, optimization and machine learning," Addison-Wesley, Reading, MA, 1989.

[19] M.A. Al-Omeer, and Z.H. Ahmed, "Comparative study of crossover operators for the MTSP," In proceedings of 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-6.

[20] 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(2), pp. 99-112, 2020.

[21] Z.H. Ahmed, "Genetic algorithm with comprehensive sequential constructive crossover for the travelling salesman problem," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 11(5), pp. 245-254, 2020.

[22] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in Shangai Baoshan Iron & Steel Complex," European Journal of Operational Research, vol. 124, pp. 267–282, 2000.

[23] A. Singh, and A.S. Baghel, "A new grouping genetic algorithm approach to the multiple traveling salesperson problem," Soft Computing, vol. 13, pp. 95-101, 2009.

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

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

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

[27] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Gucht, "Genetic algorithms for the traveling salesman problem," In Genetic Algorithms and Their Applications: Proceedings of the 1st International Conference on Genetic Algorithms, J.J. Grefenstette, Ed. Lawrence Erlbaum Associates, Hilladale, NJ, pp. 160–168, 1985.

[28] Z.H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," International Journal of Biometrics & Bioinformatics, vol. 3(6), pp. 96-105, 2010.

[29] R. Kaliaperumal, A. Ramalingam, and J. Sripriya, "A modified two part chromosome crossover for solving MTSP using genetic algorithms." In Proceedings of ICARCSET, New York, 2015, pp. 1–4.

[30] S. Chatterjee, C. Carrera, and L. Lynch, "Genetic algorithms and traveling salesperson problem," European Journal of Operational Research, vol. 93, pp. 490–510, 1996.

[31] J.A. Sveska, and V.E. Huckfeldt, "Computational experience with an m-salesman traveling salesman algorithm," Management Science, vol. 19, pp. 790-799, 1973.

[32] H.C. Lau, T.M. Chan, and W.T. Tsui, "Application of genetic algorithms to solve the multi-depot vehicle routing problem," IEEE Transaction Automatic Science and Engineering, vol. 7, pp. 383–392, 2010.

[33] C. Malmborg, "A genetic algorithm for service level based vehicle scheduling," European Journal of Operational Research, vol. 93(1), pp. 121–134, 1996.

[34] E. Falkenauer, "Genetic algorithms and grouping problems," John Wiley & Sons, New York, 1998.

[35] S. Ross, "Introduction to Probability Models," Macmillian, New York, NY, 1984.

[36] E.C. Brown, C.T. Ragsdale, and A.E. Carter, "A grouping genetic algorithm for the multiple traveling salesperson problem," International Journal of Information Technology, vol. 6, pp. 333–347, 2007.

[37] Z.H. Ahmed, "Improved genetic algorithms for the traveling salesman problem," International Journal of Process Management and Benchmarking, vol. 4(1), pp. 109-124, 2014.

[38] Z.H. Ahmed, "A hybrid genetic algorithm for the bottleneck traveling salesman problem," ACM Transactions on Embedded Computing Systems, vol. 12(1), Article No. 9, 2013.

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

[40] Z.H. Ahmed, "The ordered clustered travelling salesman problem: A hybrid genetic algorithm," The Scientific World Journal, vol. 2014, 2014.

[41] Z.H. Ahmed, "A comparative study of eight crossover operators for the maximum scatter travelling salesman," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 11(6), pp. 317-329, 2020.

[42] Reinelt, G. 1991, TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB9.

AUTHOR PROFILE

Maha Ata Al-Furhud is a Lecturer in the Department of Computer Science at Al-Jouf university. He obtained MSc in Computer Science from Al Imam Mohammad Ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia.

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 at Al Imam University in 2004, 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 these field.