Improved Genetic Algorithm for the Traveling Salesman Problem Using Neighbor-Based Constructive Crossover

Yuki Takahashi[1], Alessia Romano[2], Ahmed El-Mansouri[3], Maria Fernandez[4], Prof. Hideo Nakamura[5]

[1],[2],[3],[4]Department of Computer Science, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto, Japan

[5]Professor and Head, Department of Artificial Intelligence, University of Tokyo

ABSTRACT

In this paper, a new crossover operator named Neighbor-based Constructive Crossover (NCX) is evolved for a genetic algorithm that generates high quality solutions to the Traveling Salesman Problem (TSP). The proposed crossover operator uses the better edges present in parents' structure by comparing the neighboring nodes of a node in order to generate off-springs. The efficacy of the proposed crossover operator, NCX is set against two other crossover operators, single point crossover (SPCX) [19] and sequential constructive crossover (SCX) [1] for several standard TSPLIB instances [2]. Empirical results and observations illustrate that the new crossover operator is better than the SPCX and SCX in terms of quality of solutions.

KEYWORDS: Traveling Salesman Problem, NP-complete, Genetic Algorithm, Sequential Constructive Crossover, Neighbor-based Constructive Crossover.

I. INTRODUCTION

The Traveling Salesman Problem (TSP) is an antique problem in Computer Science and Operations Research. It can be stated as:
A network with 'n' cities (or nodes) with 'node 1' as 'source' and a travel expense (or distance, or travel time etc.,) matrix C= [c_{ij}] of order n associated with ordered node pairs (i, j) is given. Thus, the problem is to find a least cost Hamiltonian cycle.

On the basis of the structure of the cost (or expense) matrix, the TSPs are classified into two groups – symmetric and asymmetric. The TSP is symmetric if $c_{ij} = c_{ji}$, ∀ i, j and asymmetric otherwise. For an n-city asymmetric TSP, there are (n-1)! possible solutions, one or more of which gives the minimum cost. For an n-city symmetric TSP, there are $\frac{(n-1)!}{2}$ possible solutions along with their reverse cyclic permutations having the same total cost. In either case the total number of solutions becomes extremely humongous for even moderately high value of n, thereby, making the exhaustive search impracticable.

TSP remains an active research discipline and has captivated the attention of researchers because it is a proven NP-Complete problem [3]. Also, a large number of real-world problems can be modeled by TSP. Some of them are:- Drilling of printed circuit boards, VLSI circuits [4], Overhauling gas turbine engines [17], X-ray crystallography [5], Computer wiring [17], Vehicle routing [17], Mask plotting in PCB production [17], Warehouse automation system [17].
The methods that provide the exact optimal solution to the problem are called exact methods. An implicit way (i.e. a brute force approach) of solving the TSP exactly is simply to list all the feasible solutions, evaluate their objective function values and pick out the best. Nevertheless, it is evident that this "exhaustive search" is grossly inefficient and infeasible because of boundless number of possible solutions to the TSP even for problem of an average size. All practical applications require solving larger problems, hence emphasis has shifted from the aim of finding exactly optimal solutions for TSP to the aim of getting, heuristically, 'better solutions' in a reasonable time and 'establishing the degree of goodness'. Several intelligent algorithms are available to solve the TSP, some of them are:- artificial neural network [20], genetic algorithms [21], simulated annealing algorithm [22], ant colony optimization algorithm[23], particle swarm optimization [24], consultant-guided search algorithm [25] and many more. Nevertheless, Genetic algorithm (GA) is one of the best heuristic search algorithms that have been used widely to solve the TSP instances.

The new crossover operator, Neighbor-based Constructive Crossover (NCX) discussed in the paper tends to provide a better quality of solutions in solving the TSP, which is manifested by low excess percentages observed for various standard TSPLIB instances.

The organization of paper is as follows: Section 2 develops a background study about genetic algorithm. Section 3 provides the specific details regarding some related works. Section 4 explains the proposed crossover operator (NCX). Section 5 describes computational experiments and results for three crossover operators. Section 6 presents comments and concluding remarks, which is then followed by the acknowledgement section and reference section.

II. GENETIC ALGORITHMS- BACKGROUND STUDY

In computer science and operations research, genetic algorithms are a metaheuristic which are based substantially on the notion of survival of the fittest among the species produced by transmutations in chromosome gene-structure and are inspired by the practice of natural selection which is an inherent division of evolutionary biology [6]. To solve any problem using GA, a string should be able to constitute a solution and an objective (or fitness) function measuring the goodness of a solution must be defined.
Genetic algorithms are often used to produce high-quality solutions to various search and optimization problems using bio-inspired operators such as reproduction (or selection), crossover and mutation.

Genetic Encoding
The process of Genetic Encoding is important for generating feasible chromosomes. In this process, the solution of a TSP is often represented as chromosome length (i.e. the number of nodes in the problem). There are mainly two representation methods for representing tour of the TSP – adjacency representation and path representation. In this research, the path representation for a tour is considered, which simply lists the label of nodes. For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5 node instance, then a tour {1→ 5→2→ 4→ 3 →1} may be represented as (1, 5, 2, 4, 3).

Fitness Function
The GAs are used for both minimization and maximization problems. Since, TSP is a minimization problem; a fitness function, f(x) is considered which calculates total distance travelled as cost of the tour represented by a chromosome and the tour with minimum cost is chosen.

Selection Operation using Elitism
In the process of selection (alias, reproduction) operation using Elitism, the best fit chromosomes are carried forward to the next generation. Due to such assignment of the highly fit chromosomes to the succeeding generation, elitism imitates the Darwinian concept of survival-of-the-fittest in the natural world.

Crossover Operator
In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Crossover is a process of taking more than one parent solution and producing a child solution from them [18].

Mutation Operator
The primary purpose of Mutation operation is to prevent the algorithm from getting caught in local minima. The mutation operator performs modification of information in the chromosome by selecting an index randomly and altering it. As it is well known that, after successive generations the less fit members are discarded and some aspect of genetic material might get lost permanently. Therefore, mutation maintains the mating pool diversity by preventing the complete loss of important features. For this investigation, the reciprocal exchange mutation that selects two nodes randomly and swaps them, is considered.

Genetic Algorithm Controlling Parameters
The parameters that govern the whole GA search process are referred as GA controlling parameters. Some of them are:
(a) Population size: - It controls how many chromosomes and thereafter, how much genetic material is available for use during the search process. The search has no chance to adequately cover the space if the genetic material is too little. Nonetheless, if it is too much, a lot of time is squandered in evaluating chromosomes. Henceforth, the population size value must be chosen aptly.
(b) Reproduction probability: - It defines the probability of the population generated by the reproduction operation.
(c) Crossover probability: - It defines the probability of the population generated by the crossover operation taking place between any two chromosomes.
(d) Mutation probability: - It defines the probability of population generated by mutation operation.
(e) Termination criteria: - It defines when to terminate or stop the genetic search process.

Structure of a Genetic Algorithm

A Genetic Algorithm can be recapitulated as follows:

```
GA( ) {
        Initialize population in random fashion;
        Evaluate the population by calculating the fitness of the individuals in the population;
        Set Generation = 0;
        Loop until the termination condition is not satisfied{
                Generation = Generation + 1;
                Select good chromosomes (having high fitness value i.e. using elitism) by reproduction
                procedure with probability of reproduction (Pr);
                Perform crossover operation with probability of crossover (Pc);
                Perform mutation with probability of mutation (Pm);
                Evaluate the population;
        }
}
```

III. LITERATURE REVIEW

In GA, the most important role is played by the crossover operator, and therefore, many crossover operators have been proposed for disentangling the TSP. An operator named PMX (partially mapped crossover), defined by Goldberg and Lingle [7] uses two crossover points. The section between these points defines an interchange mapping. The PMX operator was the first attempt to apply GAs to the TSP, in which near-optimal solutions to a well-known 33-node problem was found. The OX (ordered crossover) operator developed by Davis [8] builds offspring by choosing a subsequence of a tour from one parent and preserving the relative order of nodes from the other parent. Another crossover operator, named CX (cycle crossover) operator was proposed by Oliver et al. [9], where offspring are built in such a way that each node (and its position) comes from one of the parents. Whitley et al. [10] proposed edge recombination crossover (ERX) operator that uses an 'edge map' to construct an offspring that inherits as much information as possible from the parent structures. This edge map stores all the connections from the two parents that lead into and out of a node. A crossover operator based on the conventional N-point crossover operator, named as generalized N-point crossover (GNX), was proposed by Radcliffe and Surry [11]. Poona and Carter [12] developed a tie break crossover (TBX), which was then modified by Choi et al. [13] by combining PMX and TBX operators. Moon et al. [14] proposed a new crossover operator named Moon Crossover (MX), which mimics the changes of the moon such as waxing moon → half-moon → gibbous → full moon. As per what is reported, the performance of MX operator and OX operator is nearly same, but OX never reached an optimal solution for all trials. The Sequential Constructive Crossover Operator (SCX) developed by Zakir H. Ahmed [1] sequentially selects the legitimate nodes and generates the offspring.

We here consider the algorithms of two crossover operators - single point crossover operator (SPCX) [19] and Sequential Constructive Crossover (SCX) [1] for producing the offspring chromosome and comparing their merits and demerits with our proposed approach.

Single Point Crossover (SPCX)

The single point crossover (SPCX) [19] operator constructs an offspring by selecting a crossover site (an index) in parent chromosomes and copying the nodes before the crossover site of first parent chromosome into a new chromosome and then copying the nodes of other parent chromosome such that already visited node do not appear in the new chromosome.

Sequential Constructive Crossover (SCX)

The sequential constructive crossover (SCX) [1] operator constructs an offspring using better edges on the basis of their values present in the parents' structure. Furthermore, it also uses the better edges, which are present neither in the parents' structure. SCX sequentially searches both of the parent chromosomes and considers the first legitimate node (i.e. unvisited node) which appeared after the previous visited node and in case, if no legitimate node is found in either of the parent chromosomes, it sequentially searches for the legitimate node (s) and then compares their associated cost to decide the next node of the child chromosome.

Juxtaposition: SPCX vs SCX

The SPCX crossover operator is very fast in terms of convergence time, however, proves to be bad in terms of quality of solution (i.e. minimum total cost or distance). On the other hand, SCX appears to be comparatively slower in terms of convergence time, nevertheless, provides better quality of solutions. However, still by looking at the quality of solutions (as mentioned in the tables-Table 2 and Table 3), it can be understood that the quality

of solution for various TSPLIB instances is not good enough and there is a huge scope of improvement, which, thereby, arises the need of a new crossover operator.

IV. PROPOSED CROSSOVER OPERATOR

The search of the solution space is accomplished by generating novel chromosomes from antiquated ones. In that the most vital search process is crossover. The Neighbor-based Constructive Crossover (NCX) operator constructs off-springs using better edges on the basis of their values present in the parents' structure. Unlike SCX, NCX uses both of the neighbors of a node. Nonetheless, like SCX, it also uses better edges, which are not present in either of the parents' structure. Additionally, it sometimes introduces novel, but good, edges to the offspring, which are not even present in the present population. Hence, the chances of producing a better offspring are more than SPCX and SCX.

The algorithm for the NCX is as follows:

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

Step 2: - Search both of the parent chromosomes and consider their neighboring 'legitimate nodes' (the nodes that are not yet visited) of 'node p' in each parent. If no 'legitimate node' after 'node p' is present in any of the parent, search sequentially the nodes {2, 3, …, n} and consider the first 'legitimate' node, and go to Step 3.

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

Step 4: Compare the costs of adding all nodes after 'node p' and then select the node with the minimum cost 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.

Let us illustrate the NCX through the example given as cost matrix in Table 1 [1]. Let a pair of selected chromosomes be P1: (1, 5, 7, 3, 6, 4, 2) and P2: (1, 6, 2, 4, 3, 5, 7) with values 312 and 331 respectively.

Table 1. The cost matrix [17]

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

Select 'node 1' as the 1st node. The 'legitimate' nodes after 'node 1' in P1 and P2 are 'node 5', 'node 2' and 'node 6', 'node 7' respectively with c_{15}=35, c_{12}=75, c_{16}=63 and c_{17}=8. Since c_{17} has lowest value, 'node 7' is accepted. Now, the PCC (partially constructed chromosome) becomes (1, 7). The 'legitimate' nodes after 'node 7' in both P1 and P2 are 'node 3', 'node 5' and c_{73}=43, c_{75}=52, so now 'node 3' is accepted, and the PCC becomes (1, 7, 3). The 'legitimate' nodes after 'node 3' in P1 and P2 are 'node 6' and 'node 4', 'node 5' and c_{34}=16, c_{35}=28, c_{36}=35, so now 'node 4' is accepted and the PCC becomes (1, 7, 3, 4). Now, the 'legitimate' node(s) after 'node 4' in P1 are 'node 2' and 'node 6' with c_{42}=45, c_{46}=53 and 'node 2' in P2, so 'node 2' is accepted and the PCC becomes (1, 7, 3, 4, 2). The 'legitimate' node after 'node 2' in P1 is none and P2 is 'node 6' and c_{26}=29, so 'node 6' is accepted and the PCC becomes (1, 7, 3, 4, 2, 6). The 'legitimate' node after 'node 6' in P1 and P2 is none, so now 'legitimate' node is searched sequentially. Since, only one node 'node 5' is left, it is accepted and added to the solution. Thus the complete offspring chromosome will be (1, 7, 3, 4, 2, 6, 5) with value 248 (including the cost c_{51}) which is less than value of both the parent chromosomes (312 & 331) and 266 (1, 5, 7, 2, 4, 3, 6), the result produced by SCX

and 304 (1, 5, 7, 3, 6, 2, 4), the result produced by SPCX [19]. The crossover is shown in Figure 1. The parents are showing as (a) and (b), while (c) is a possible offspring.

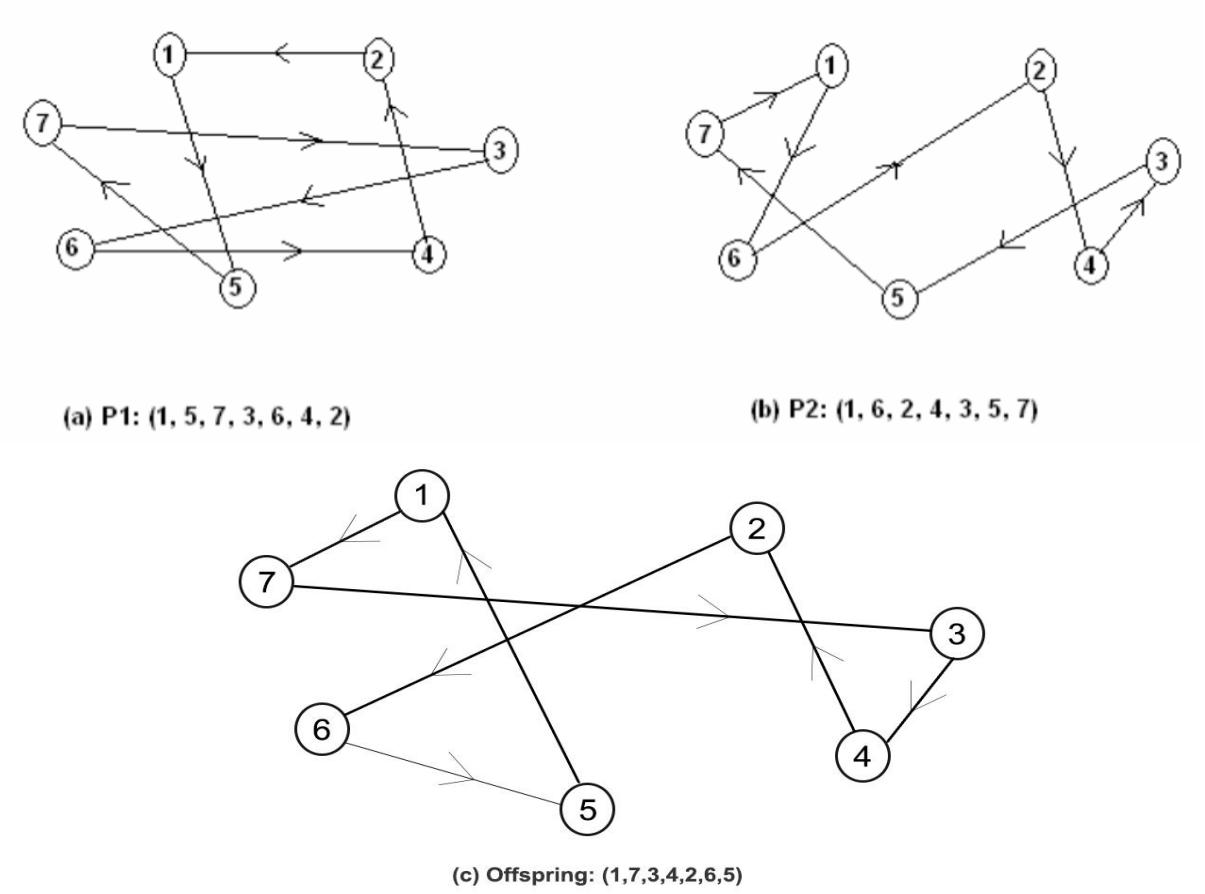

Figure 1: Example of Neighbor-based Constructive Crossover Operator

V. RESULTS AND DISCUSSION

In order to juxtapose the efficiency of the different crossover operators, genetic algorithms using NCX, SCX and SPCX have been coded in JavaScript and executed on an Intel core i5 personal computer with clock-speed 2.5 GHz, 8GB RAM, 4MB L3 cache via the command terminal of Mac OS Sierra for some TSPLIB instances. Initial population is generated randomly.

The following common parameters are selected for the algorithms: (i) population size is 50, (ii) probability of reproduction (i.e. selection using elitism) is 10%, (iii) crossover probability is 80%, (iv) probability of mutation is 10%, (v) maximum of 1,000 generations as the terminating condition. The experiments were performed 10 times for each instance. The solution quality is measured by the percentage of excess above the optimal solution value reported in TSPLIB website, as given by the formula.

$$Excess\ (\%) = \frac{\text{Solution Value} - \text{Optimal Solution Value}}{\text{Optimal Solution Value}} \times 100$$

The excess percentage of best solution values and average solution values over their corresponding optimal solution values of 10 runs and the average time of convergence (in second(s)) of the algorithms is reported in the table 2 and 3.

Table 2 gives the result for fifteen asymmetric TSPLIB instances of size from 17 to 171, whereas, table 3 gives the result for fifteen symmetric TSPLIB instances of size from 17 to 561.

The quality of solutions of the algorithms is inconsiderate to the number of runs. In the table, the best value, average value and average time is calculated by applying each crossover operator to the same TSPLIB instance. Furthermore, the excess percentage is calculated as per the above formula in order to compare the solution obtained with the optimal solution. Both the tables portray the insights in which the best values and average values

for NCX are better than both SCX and SPCX and the corresponding excess percentages are less. Additionally, SPCX performs worst among the three crossover operators for almost all values; however, SPCX outperforms NCX and SCX in terms of average time or time of convergence (i.e. low time complexity). Though, SCX surpasses NCX in respects of time of convergence, it is noted that by observation, NCX outshines both SCX and SPCX in terms of quality of solutions for all the instances.

Table 2. Summary of the results by the crossover operators for Asymmetric TSPLIB instances

			N C X			**S C X**			**S P C X**		
tsplib file	**n**	**Optimum Value**	**Best Val(Excess%)**	**Avg. Val(Excess%)**	**Avg. Time(in sec)**	**Best Val(Excess%)**	**Avg. Val(Excess%)**	**Avg. Time(in sec)**	**Best Val(Excess%)**	**Avg. Val(Excess%)**	**Avg. Time(in sec)**
br17	17	39	**39(0.00)**	**39(0.00)**	0.4073	**39(0.00)**	39.6(1.53)	0.3707	**39(0.00)**	42.5(8.97)	0.2279
ftv33	34	1286	1350(4.97)	1487.1(15.63)	0.7947	1445(12.36)	1510(17.41)	0.6542	2233(73.63)	2561(99.14)	0.2671
ftv35	36	1473	1659(12.62)	1737.4(17.94)	0.8529	1726(17.17)	1765.2(19.83)	0.7323	2315(57.16)	2823(91.64)	0.2677
ftv38	39	1530	1655(8.16)	1706.9(11.56)	0.9704	1705(11.43)	1732.1(13.20)	0.8567	2552(66.79)	2921.7(90.96)	0.3053
p43	43	5620	5624(0.07)	5632.3(0.21)	1.3895	5636(0.28)	5644.5(0.43)	1.1725	5862(4.30)	7006.4(24.66)	0.3153
ftv44	45	1613	1775(10.04)	1854.9(14.99)	1.1269	1832(13.57)	1911.6(18.51)	0.9898	3288(103.84)	3673.7(127.75)	0.3318
ftv47	48	1776	2022(13.85)	2133.9(20.15)	1.2875	2108(18.69)	2209.3(24.39)	1.0842	3488(96.39)	4057.2(102.44)	0.3383
ry48p	48	14422	15716(8.97)	16054.3(11.31)	1.2457	16223(12.48)	16843.6(16.79)	1.1501	24914(72.74)	29708.3(105.99)	0.3406
ft53	53	6905	8480(22.80)	8708.4(26.11)	1.4578	8612(24.72)	9304.3(34.74)	1.2206	13059(89.12)	15985.5(131.50)	0.376
ftv55	56	1608	1775(10.38)	1844.7(14.72)	1.5312	1802(12.06)	1971.9(22.63)	1.3208	3652(127.11)	4354.9(170.82)	0.3925
ftv64	65	1839	2165(17.72)	2345(27.51)	1.9566	2370(28.87)	2459.4(33.73)	1.6343	4201(128.43)	5305.2(188.48)	0.4661
ft70	70	38673	41342(6.90)	42078.2(8.80)	2.2151	42243(9.23)	43521.9(12.53)	1.8875	53980(39.58)	56904(47.14)	0.5021
ftv70	71	1950	2234(14.56)	2378.5(21.97)	2.17	2452(25.74)	2566.2(31.6)	1.89	4868(149.64)	5754.9(195.12)	0.5101
kro124p	100	26230	43149(64.50)	44161.7(68.36)	3.6342	48074(83.27)	49976.2(90.53)	3.0493	107071(308.20)	119796.8(356.71)	0.7727
ftv170	171	2755	3530(28.13)	3884.1(40.98)	8.3449	4581(66.27)	4811.5(74.64)	6.8825	16530(500)	17372.1(530.56)	1.7855

Table 3. Summary of the results by the crossover operators for Symmetric TSPLIB instances

			N C X			**S C X**			**S P C X**		
tsplib file	**n**	**Optimum Value**	**Best Val (Excess %)**	**Avg. Val (Excess %)**	**Avg. Time(in sec)**	**Best Val (Excess %)**	**Avg. Val (Excess %)**	**Avg. Time(in sec)**	**Best Val (Excess %)**	**Avg. Val (Excess %)**	**Avg. Time(in sec)**
gr17	17	2085	**2085(0.00)**	2091.9(0.33)	0.3613	**2085(0.00)**	2100.7(0.75)	0.3086	2155(3.35)	2278.6(9.28)	0.199
gr24	24	1272	1328(4.40)	1394.2(9.60)	0.561	1413(11.08)	1444.9(13.59)	0.4913	1600(25.78)	1807.9(42.13)	0.2146
hk48	48	11461	12250(6.88)	12542.6(9.43)	1.3084	12620(10.11)	13425.4(17.13)	1.1398	21666(89.04)	25486.5(122.37)	0.3447
eil51	51	426	460(7.98)	473(11.03)	1.3995	476(11.73)	501.9(17.81)	1.2765	797(87.08)	932.5(118.89)	0.3713
berlin52	52	7542	8009(6.19)	8454.4(12.09)	1.4566	8404(11.42)	9081.4(20.41)	1.2388	12363(63.92)	16305(116.18)	0.3783
eil76	76	538	575(6.87)	586.9(9.08)	2.4262	636(18.21)	681.5(26.67)	1.9737	1340(149.07)	1511.3(180.91)	0.546
pr76	76	108159	123390(14.08)	130222.9(20.39)	2.3859	137127(26.78)	144131.8(33.25)	2.0158	262244(142.46)	339919.9(214.27)	0.5424
kroA100	100	21282	24805(16.55)	25338.3(19.05)	3.717	30514(43.37)	31616.3(48.55)	3.0514	82627(288.24)	98913.8(364.77)	0.7754

kroC100	100	20749	22201(6.99)	23564.7(13.57)	3.6122	29020(39.86)	31448.1(51.56)	3.0306	79006(280.77)	93023.1(348.32)	0.77
eil101	101	629	705(12.08)	735.2(16.88)	3.7871	837(21.30)	852.8(35.58)	3.0854	1860(195.70)	2101.6(234.11)	0.7911
lin105	105	14379	15789(9.80)	16676.5(15.97)	3.9104	19219(33.66)	20779.6(44.51)	3.1575	64052(345.45)	68236.8(374.55)	0.8257
gil262	262	2378	3070(29.10)	3262.7(37.20)	17.298	5160(116.98)	5341.3(124.61)	13.3541	15592(555.67)	17500.7(635.94)	3.4191
a280	280	2579	2907(12.71)	2993.8(16.08)	24.4988	3403(31.95)	3608.3(39.91)	17.9052	20226(684.25)	21346(727.68)	4.0563
lin318	318	42029	52348(24.55)	55273.1(31.51)	25.1861	69377(65.06)	73443.4(74.74)	19.935	351199(735.61)	394607.7(838.89)	5.1213
pa561	561	2763	3480(25.95)	3544.3(28.27)	83.5643	4580(65.76)	4685.8(69.59)	59.0842	24610(790.69)	26212.4(848.69)	14.983

The following figures 2 and 3 depict the graph between the number of generations (x-axis) and tour cost (y-axis) for an asymmetric and symmetric TSPLIB instances respectively. It is clearly observed from the figure 2 that Neighbor-based CX has tour cost (3636) much lower than that of Sequential CX (5603) and basic CX (17569) and very near to the optimum value (2755). Similarly, it can be identified from the figure 3 that the proposed algorithm has tour cost (3191) much lower than that of Sequential CX (5604) and basic CX (18913) and very near to the optimum value (2378).

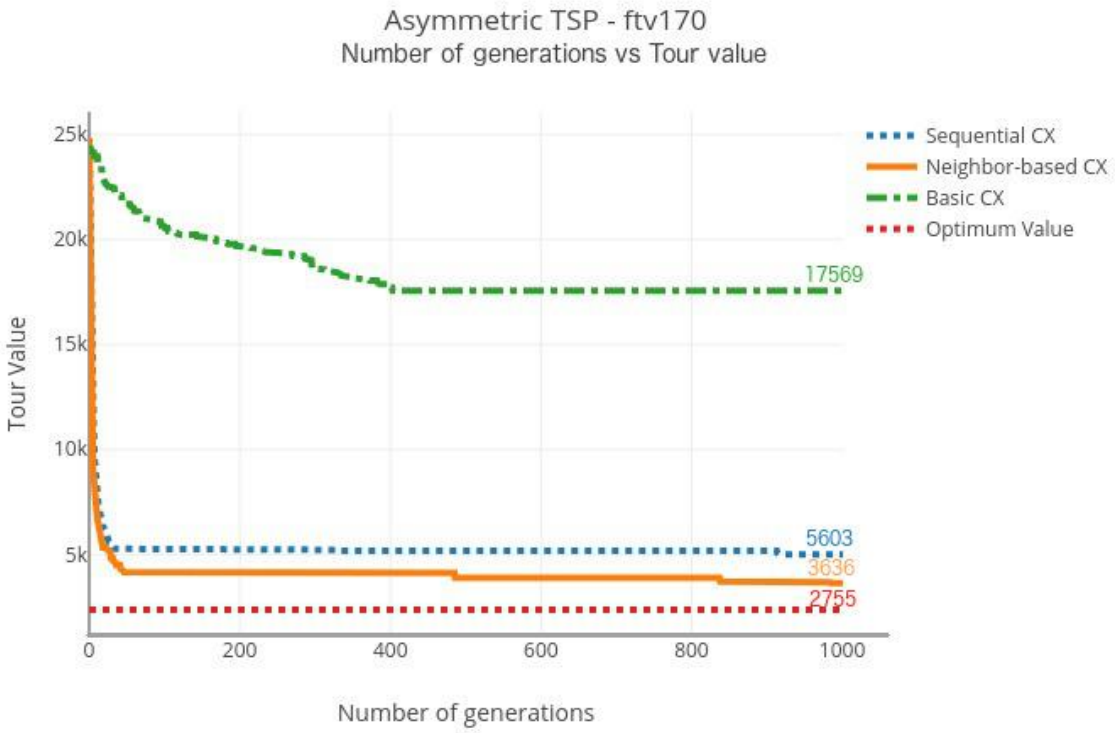

Figure 2. Performance of different crossover operators on Asymmetric TSP instance ftv70(71 nodes) [2]

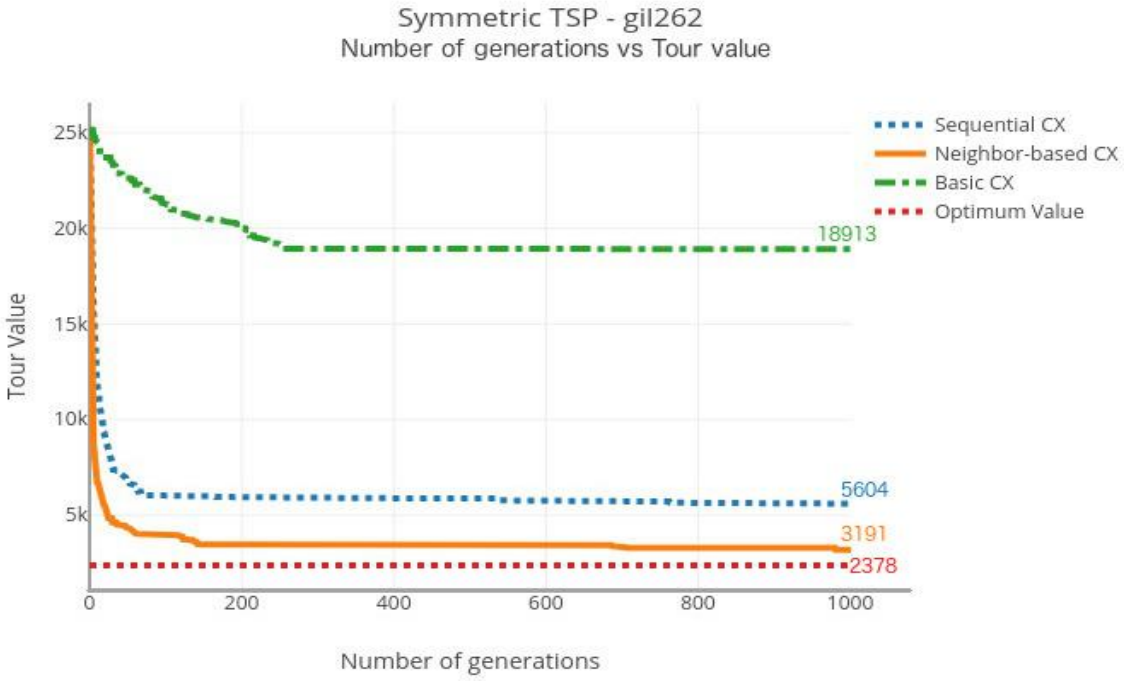

Figure 3. Performance of different crossover operators on Symmetric TSP instance gil262(262 nodes)

VI. CONCLUSION

A new crossover operator named Neighbor-based Constructive Crossover (NCX) for a genetic algorithm for solving the well-known Traveling Salesman Problem (TSP) is proposed. A comparative study among NCX, SCX, and SPCX for some benchmark TSPLIB instances is presented. In terms of quality of solutions, for small, medium and large sized instances, SCX is found to be better than SPCX, which is evident from tables 2 and 3. Also, the experimental results illustrate the proposed crossover operator (NCX) to be better than both SPCX and SCX in terms of quality of solutions which is evident from the graphs shown in Figures 2, 3 and Tables 2, 3. Despite of better quality of solution, it is also observed that the proposed approach may not always converge faster in terms of time as juxtaposed with SCX and SPCX.

The research work only deems the primitive form of SCX and SPCX. The primary focus of this research includes analysis of the quality of solutions by several crossover operators and the work does not aim to improve the quality of solutions by any operator by using any local search technique. The crossover probability is set highest in order to portray the exact working of crossover operators. The mutation operator is applied to prevent the solution from getting stuck in local minima quickly. The population size is not set high and the parallel version of algorithms to obtain the exact solution as was done by Whitley et al. [10] is not considered in this investigation.

In future, some algorithm can be designed to enhance the quality of solutions produced by NCX and overcome the drawbacks of NCX (i.e. by reducing the time of convergence). In order to enhance the quality of solutions by NCX, we also designed and implemented an enhanced version of NCX which we named Window-based Constructive Crossover, in which we considered the neighbors in a particular window size. For example- if window-size is two, then two on either sides of a node in a chromosome i.e. in total four neighbor nodes will be selected from a single parent chromosome. However, after taking a window size of two, the solution quality was good for small-sized TSPLIB instances (having a size less than 25) but the time of convergence increased significantly and the algorithm was found out to be impracticable for even moderate-sized instances.

VII. ACKNOWLEDGEMENTS

This research was supported by Department of Computer Engineering, Shri G.S. Institute of Technology & Science, Indore, India.

VIII. REFERENCES

1. Z.H. Ahmed. “Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator”, Al-Imam Muhammad Ibn Saud Islamic University, Kingdom of Saudi Arabia, IJBB Volume(3) Issue(6) pg. 96-105, 2010
2. TSPLIB instances data source: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html
3. C.H. Papadimitriou and K. Steglitz. “Combinatorial Optimization: Algorithms and Complexity”. Prentice Hall of India Private Limited, India, Courier Corporation 1998.
4. C.P. Ravikumar. "Solving Large-scale Travelling Salesperson Problems on Parallel Machines”. Microprocessors and Microsystems Volume(16) Issue(3), pp. 149-158, 1992.
5. R.G. Bland and D.F. Shallcross. "Large Travelling Salesman Problems arising form Experiments in X-ray Crystallography: A Preliminary Report on Computation". Operations Research Letters, Journal Operations Research letters Volume(8) Issue(3), pp. 125-128, 1989.
6. D.E. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning". AddisonWesley, New York, 1989.
7. 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.
8. L. Davis. “Job-shop Scheduling with Genetic Algorithms”. Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 136-140, 1985.
9. 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.
10. 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.
11. 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.
12. P. Poon and J. Carter. “Genetic algorithm crossover operations for ordering applications”. Computers and Operations Research 22, pp. 135–47, 1995.
13. I. Choi, S. Kim and H. Kim. "A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem". Computers & Operations Research 30, Volume(30) Issue(5) pp. 773 – 786, 2003.
14. C. Moon, J. Kim, G. Choi and Y. Seo. "An efficient genetic algorithm for the traveling salesman problem with precedence constraints". European Journal of Operational Research Volume(140) Issue(3), pp. 606-617, 2002.
15. Z.H. Ahmed. "A sequential Constructive Sampling and Related approaches to Combinatorial Optimization". PhD Thesis, Tezpur University, India, 2000.
16. Z.H. Ahmed and S.N.N. Pandit. “The travelling salesman problem with precedence constraints”. Opsearch Volume(38) Issue(3), pp. 299-318, 2001.
17. Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches- By Rajesh Matai, Surya Singh and Murari Lal Mittal, DOI: 10.5772/12909
18. https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)
19. Study of Various Crossover Operators in Genetic Algorithms, Nitasha Soni, Dr .Tapas Kumar Lingaya’s university, Faridabad et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Volume(5) Issue(6), 2014, 7235-7238.
20. H. Ghaziri and I. H. Osman, “A neural network algorithm for the traveling salesman problem with backhauls.” Computers & Industrial Engineering, Volume(44) Issue(2), pp.267-281, February 2003. http://dx.doi.org/10.106/S0360-8352(02)00179-1
21. Y. H. Liu, “Different initial solution generators in genetic algorithms for solving probabilistic traveling salesman problem,” Applied Mathematics and Computation, Volume(216) Issue(1), pp.125-137, March 2010. http://dx.doi.org/10.106/j.amc.2010.01.021
22. Y. Liu, S. W. Xiong and H. B. Liu, “Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP,” Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp.895-898, Shanghai, June 12-14, 2009.
23. M. Dorigo, M. Birattari and T. Stutzle, “Ant colony optimization IEEE Computational Intelligence Magazine,” Volume(1) Issue(4), pp.28-39, November 2006
24. W. N. Chen, J. Zhang, et al,”A novel set-based particle swarm optimization method for discrete optimization problems,” IEEE T. Evolu. Computation., Volume(14) Issue(2), pp.278-300, April 2010. http://dx.doi.org/10.1109/TEVC.2009.2030331
25. S. Iordache, “Consultant-guided search-a new metaheuristic for combinatorial optimization problems,” Proceedings of the 12th annual conference companion on Genetic and evolutionary computation, pp.225-232, July 7-11, Portland, 2010.