Punjab University
Journal of Mathematics (ISSN 1016-2526)
Vol. 51(5)(2019) pp. 61-77

A Simulated Study of Genetic Algorithm with a New Crossover Operator using Traveling Salesman Problem

Abid Hussain
Department of Statistics,
Quaid-i-Azam University, Islamabad, Pakistan.
Email: abid0100@gmail.com

Yousaf Shad Muhammad
Department of Statistics,
Quaid-i-Azam University, Islamabad, Pakistan.
Email: yousuf@qau.edu.pk

Muhammad Nauman Sajid
Department of Software Engineering,
Foundation University, Islamabad, Pakistan.
Email: mn.nauman@gmail.com

Received: 05 July, 2018 / Accepted: 18 October, 2018 / Published online: 05 March, 2019

Abstract. This work shows improvement with a modified form of the existing partially-mapped crossover operator for the traveling salesman problem. This novel crossover approach has been presented to get solutions by order of a list, and "permutation crossover"operators, while preserving the legality of offspring. Results are compared with many existing schemes for permutation representation, like partially-mapped, order, and cycle crossovers, etc. Our modified form of partially-mapped crossover operator searches the existing bits outside the crossover sites, whereas the existing partially-mapped crossover searches within the crossover sites. This approach is easy to understand as well as to apply on benchmark problems. Comparison of the proposed operator with traditional ones for several benchmarks TSPLIB instances widely shows its advantages at the same accuracy level. Also, it requires less time for tuning of genetic parameters and provides much narrower confidence intervals on the results, compared to other operators.

AMS (MOS) Subject Classification Codes: 65C60; 90C05; 97K80
Key Words: NP-hard, Traveling salesman problems, Genetic algorithms, Path-representation, Crossover operators.

1. INTRODUCTION

The traveling salesman problem (TSP) is one of the most famous benchmark, significant, and historic hard combinatorial optimization problems. The main objective of TSP is to find the shortest Hamiltonian tour in a complete graph with 'n' nodes. It was documented by Euler in 1759 (his interest was how to get rid of the knight's tour problem [19]). It is a fundamental problem in the fields of computer science, engineering, operations research, discrete mathematics and graph theory. It has a formal mathematical definition as a direct or undirected graph and the family of all Hamiltonian tours (cycles) [21]. In this problem, a salesman visits all cities (nodes) exactly once (the constraint), then returns to the starting point to complete a tour. On the basis of distance evaluation plan, we divide the TSP into two different groups, called symmetric and asymmetric. The TSP is symmetric if $c_{ij} = c_{ji}, \forall\, i, j$, where i and j represent the row and column of a distance (or cost) matrix respectively, otherwise asymmetric i.e. $c_{ij} \neq c_{ji}$. The given 'n' cities, a distance matrix $C = [c_{ij}]_{n\times n}$ is searched for a permutation $\lambda : \{0, ..., n-1\} \longrightarrow \{0, ..., n-1\}$, where c_{ij} is the distance from city i to city j, which minimizes the traveled distance, $f(\lambda, C)$.

$$f(\lambda, C) = \sum_{i=0}^{n-1} d(c_{\lambda(i)}, c_{\lambda(i+1)}) + d(c_{\lambda(n)}, c_{\lambda(1)}) \tag{1. 1}$$

where $\lambda(i)$ represents the location of city i in each tour, $d(c_i, c_j)$ is the distance between city i to city j and (x_i, x_j) is a specified position of each city in a tour in the plane, and the Euclidean distances of the distance matrix C between the city i and j is expressed as:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1. 2}$$

TSP is easy to understand but very difficult to solve. For example, for 'n' cities, there are $n!$ possible ways to find the tour for asymmetric and $\frac{n!}{2}$ for symmetric TSP. If we have only 10 cities then 362,880 and 181,440 ways for asymmetric and symmetric TSP, respectively. This is the reason to say TSP is a non-deterministic polynomial (NP-hard) problem [3]. These type of problems cannot be solved using traditional optimization approaches like derivative-based methods. To achieve the optimal solution within reasonable time, heuristic approaches are efficient at handling the NP-hard problems [20]. TSP has many applications such as variety of routing and scheduling problems, computer wiring, movement of people, X-ray crystallography [6] and automatic drilling of printed circuit boards and threading of scan cells in testable Very-Large-Scale-Integrated (VLSI) circuits [31].

TSP is very carefully studied problem and received considerable attention over the last five decades. A lot of algorithms have been presented to solve this problem by researchers. These algorithms are generally divided in two classes; exact and approximate algorithms. The branch-and bound (BB) [12] and cutting planes (CP) [23] are the two examples of exact algorithms which are excessively time consuming especially in large scale problems. Approximate algorithms are further classified into heuristic and meta-heuristic algorithms. There are three classes of heuristic algorithm; tour construction, tour improvement and composite methods. To add an unvisited city to the solution at each step and try to shorten the initial solution are tour construction and tour improvement methods, respectively. Finally, the composite method is the combination of these two algorithms.

During last three decades with the appearance of meta-heuristic algorithms, a new era of study about optimization problems and their applications has been started. These search algorithms have also been applied on TSP; 2-opt [20], particle swarm optimization (PSO) [8, 34–36], simulated annealing (SA) [33], ant colony optimization (ACO) [11, 22], neural network (NN) [5], tabu search (TS) [16], and genetic algorithms (GAs) [1,2,4,8,18,24,25, 29].

The rest of this article is presented as follows: in Section 2 we present the overview of GAs and their related work in TSP. Crossover operators for TSP are presented in Section 3, proposed crossover operator for path representation discussion in Section 4. Performance evaluation of the proposed operator given in Section 5 and conclusions in Section 6.

2. GENETIC ALGORITHMS

Genetic algorithms (GAs) are derivative free stochastic approaches which are based on biological evolutionary processes proposed by John Holland [17]. The random population of individuals with different encodes such as binary, real and permutation is created first. In nature, the most suitable individuals are likely to survive and mate. GAs iteratively generate new chromosomes with the help of crossover and mutation operators. The process is repeated until or unless the required criteria such as convergence or fixed number of iterations is met. The objective is the solution with high astounding fitness values which are remarkable in the search process towards the optimal solution. The most attractive features of GAs are that they have the ability to explore the search space with the help of entire population of chromosomes [27]. A lot of work and applications have been highlighted about GAs in [15]. The GAs have the following basic steps:

(1) initial population of chromosomes is generated,
(2) evaluate the fitness of each individual,
(3) apply selection operator for individuals,
(4) apply crossover and mutation operators on these selected individuals,
(5) evaluate the reproduced individuals.

The sketch of a typical genetic algorithm is depicted in Figure 1.

2.1. Genetic Algorithms for TSP.

In literature, there are binary, path, adjacency, real and matrix representations of the chromosomes to solve the TSP using the GAs. A detailed study of these approaches is presented by Larranaga et al. [19]. Other than path, all techniques have complex nature to complete a legal tour for next generation. A path representation is desired because it is the most natural and elaborated way to represent a tour. For example, a nine-city tour $7 \longrightarrow 2 \longrightarrow 6 \longrightarrow 1 \longrightarrow 4 \longrightarrow 8 \longrightarrow 3 \longrightarrow 9 \longrightarrow 5$ can be represented simply as (7 2 6 1 4 8 3 9 5). The selection criteria, crossover and mutation are three major operators which GAs have been implemented in the following way:

Selection Operator: Parents are selected for mating process through selection operator. Several different approaches have been developed in literature, such as roulette wheel, ranking and tournament etc. In this study, we used roulette wheel selection (RWS) operator. This operator gives individuals a probability P_i of being selected (2.3) that is directly proportionate to their fitness.

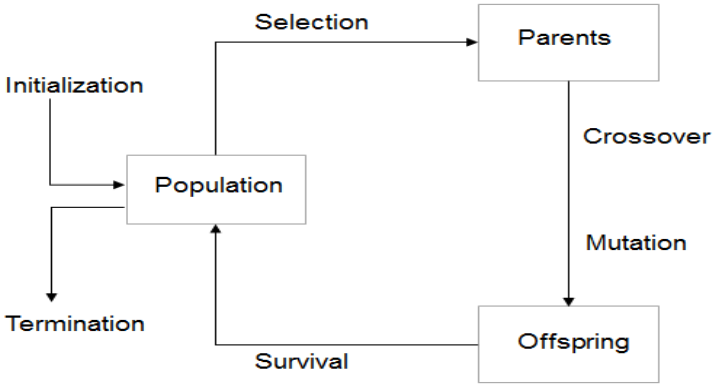

FIGURE 1. Layout of a Typical Genetic Algorithm

$$P_i = \frac{1}{N-1} \times (1 - \frac{f_i}{\sum_{j=1}^{N} f_j}) \tag{2. 3}$$

Where f_i is the value of fitness function for the individual i. Thus, individuals who have lower values of fitness function may have a high chance of being selected among the individuals to cross reproduce. Figure 2 shows that how can we select individuals for mating process.

Crossover Operator: Crossover method creates offspring with the help of those parents which are selected through selection operator. It combines the features of parents to form offspring, that may have new sequences compared to those of their parents and plays a vital

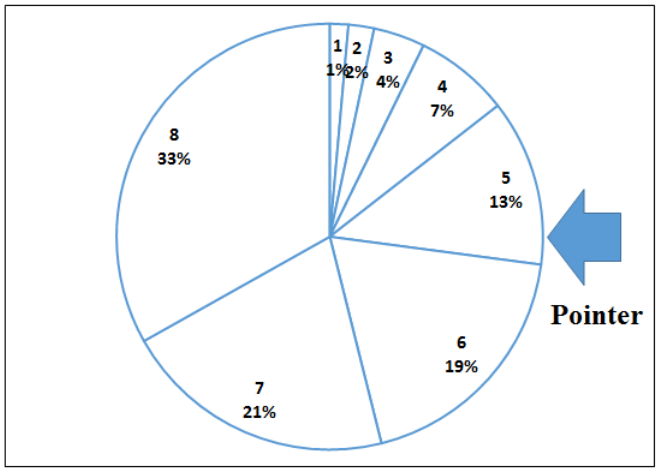

FIGURE 2. Roulette wheel selection (RWS) operator

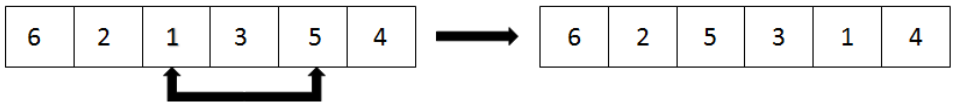

FIGURE 3. Exchange mutation (EM) operator

role in the GAs. A lot of crossover operators have been introduced in literature and all have their own significant importance (Section 3 for details). In this study, we proposes a new crossover operator which is explained in Section 4.

Mutation Operator: A mutation operator is used to enhance the diversity and provide a chance to escape from local optima. In literature, several mutation operators have been proposed such as exchange, insertion, inversion, swap and heuristic mutation etc. This study adopts the well-known exchange mutation (EM) operator, which interchange the bits at two randomly selected loci. In the EM, the two bits selected randomly along the string are exchanged as shown in Figure 3.

There are a lot of variety of these operators for TSP in literature. A comprehensive study about GAs approaches are successfully applied to the TSP [13]. A survey of GAs approaches for TSP is presented by Potvin [29]. A new sequential constructive crossover generates with a high quality solution for the TSP by Ahmed [1]. A new genetic algorithm for asymmetric TSP is proposed by Nagata and Soler [25]. Three new variations for order crossover are presented with improvements by Deep and Adane [10]. Piwonska [28] associated a profit based genetic algorithm with TSP and obtaining good results to test on networks of cities of Poland. Best performance is achieved by combining the crossover operators when reported eight different crossover operators on vehicle routing problem by Puljic and Manger [30]. Hussain et al. [18] presented a comparative study of various crossovers with the modified form of cycle crossover operator for TSP.

3. CROSSOVER OPERATORS (PATH-BASED) FOR TSP

Since the TSP in combinatorial with the path representation and the classical crossover operators such as one-point, two-point and uniform crossovers cannot be directly applied to path-based GAs. To address the issue of the legality of an order, various crossover operators for path-based GAs have been proposed. The most popular crossover operators for path-based are:

3.1. **Order Crossover Operator.**

The order crossover (OX) was proposed by Davis [9]. It builds offspring preserving the relative order of bits of one parent by choosing a sub-tour of other parent. Consider, for example two parents tours (with randomly two cut points marked by "|"):

$$P_1 = (9 \quad 4 \quad 5 \,|\, 2 \quad 8 \quad 1 \,|\, 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \,|\, 9 \quad 7 \quad 8 \,|\, 2 \quad 4 \quad 5)$$

The offspring are produced in the following way. First, the bits are copied down between the cuts with similar way into the offspring, which gives:

$$O_1 = (\times \quad \times \quad \times \,|\, 2 \quad 8 \quad 1 \,|\, \times \quad \times \quad \times)$$

and

$$O_2 = (\times \quad \times \quad \times \,|\, 9 \quad 7 \quad 8 \,|\, \times \quad \times \quad \times)$$

After this, starting from the second cut point of one parent, the bits from the other parent are copied in the same order omitting existing bits. As the sequence of the bits in the second parent from the second cut point is:

$$2 - 4 - 5 - 3 - 6 - 1 - 9 - 7 - 8$$

after removal of bits 2, 8 and 1, which already exist in the first offspring, the new sequence is:

$$4 - 5 - 3 - 6 - 9 - 7$$

To complete the first offspring, this new sequence is placed starting from the second cut point:

$$O_1 = (6 \quad 9 \quad 7 \,|\, 2 \quad 8 \quad 1 \,|\, 4 \quad 5 \quad 3)$$

Analogously, we complete second offspring as well:

$$O_2 = (5 \quad 2 \quad 1 \,|\, 9 \quad 7 \quad 8 \,|\, 6 \quad 3 \quad 4)$$

3.2. **Non-Wrapping Order Crossover Operator.** The non-wrapping order crossover operator (NWOX) was proposed by Cicirello [7]. It uses a modified version of the sliding action of the original OX. This scheme is based on the principle of creating and filling holes, while keeping the absolute order of bits of individuals. Consider, for a example of the two parents tours (with randomly two cut points marked by "|"):

$$P_1 = (9 \quad 4 \quad 5 \,|\, 2 \quad 8 \quad 1 \,|\, 6 \quad 7 \quad)$$

and

$$P_2 = (3 \quad 6 \quad 1 \,|\, 9 \quad 7 \quad 8 \,|\, 2 \quad 4 \quad 5)$$

The offspring are produced in the following way. First, all those bits are left as hole which are presenting within the cut-points in other parent, which gives:

$$O_1 = (\times \quad 4 \quad 5 \,|\, 2 \quad \times \quad 1 \,|\, 6 \quad \times \quad 3)$$

and

$$O_2 = (3 \quad 6 \quad \times \,|\, 9 \quad 7 \quad \times \,|\, \times \quad 4 \quad 5)$$

After this, existing bits in cut-points are moved left or right to filled out-sides the unfilled positions as:

$$O_1 = (4 \quad 5 \quad 2 \,|\, \times \quad \times \quad \times \,|\, 1 \quad 6 \quad 3)$$

and

$$O_2 = (3 \quad 6 \quad 9 \,|\, \times \quad \times \quad \times \,|\, 7 \quad 4 \quad 5)$$

Now, the bits are copied down between the cuts from the parents with similar way into the offspring, which gives:

$$O_1 = (4 \quad 5 \quad 2 \,|\, 9 \quad 7 \quad 8 \,|\, 1 \quad 6 \quad 3)$$

and

$$O_2 = (3 \quad 6 \quad 9 \,|\, 2 \quad 8 \quad 1 \,|\, 7 \quad 4 \quad 5)$$

3.3. **Cycle Crossover Operator.** The cycle crossover (CX) operator was first proposed by Oliver et al. [26]. To create offspring using this technique each bit with its location comes from one of the parents. For example, consider the tours of two parents:

$$P_1 = (9 \quad 4 \quad 5 \quad 2 \quad 8 \quad 1 \quad 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \quad 8 \quad 7 \quad 9 \quad 2 \quad 4 \quad 5)$$

Now its up to us that how we choose the first bit for the offspring to be either from the first or from the second parent. In our example, the first bit of the offspring has to be a 9 or a 3. Let us choose it to be 9,

$$O_1 = (9 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

Now offspring is bound to take every bit from one of its parents with the same position and don't have any choice, so the next bit to be considered must be 3, as the bit from the second parent just below the selected bit 9. In first parent this bit is at 9th position, thus

$$O_1 = (9 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad 3)$$

This implies bit 5, which is the bit of second parent just below the selected bit at 3rd position in first parent. Thus

$$O_1 = (9 \quad \times \quad 5 \quad \times \quad \times \quad \times \quad \times \quad \times \quad 3)$$

The next it compelled us to put the 1 at 6th position, as

$$O_1 = (9 \quad \times \quad 5 \quad \times \quad \times \quad 1 \quad \times \quad \times \quad 3)$$

After this, a cycle is completed because the upcoming bit 9 already exists in this offspring, thus filling the remaining blank positions with the bits of those positions which are in second parent.

$$O_1 = (9 \quad 6 \quad 5 \quad 8 \quad 7 \quad 1 \quad 2 \quad 4 \quad 3)$$

Similarly the second offspring is:

$$O_2 = (3 \quad 4 \quad 1 \quad 2 \quad 8 \quad 9 \quad 6 \quad 7 \quad 5)$$

But there is a drawback that some times this technique produces same offspring, for example the following two parents:

$$P_1 = (9 \quad 4 \quad 5 \quad 2 \quad 8 \quad 1 \quad 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \quad 9 \quad 7 \quad 8 \quad 2 \quad 4 \quad 5)$$

After applying CX technique, the resultant offspring are:

$$P_1 = (9 \quad 4 \quad 5 \quad 2 \quad 8 \quad 1 \quad 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \quad 9 \quad 7 \quad 8 \quad 2 \quad 4 \quad 5)$$

Which are the exactly same as their parents.

3.4. **Modified-Cycle Crossover Operator.** The modified-cycle crossover (CX2) operator was proposed by Hussain et al. [18]. Using this technique to create offspring in such a way that first bit of second parent is the the first bit of first offspring and then search that bit in first parent and choose the exact same location bit from second parent and again search it in first parent and again choose exact same location bit from second parent and that bit is the first bit of the second offspring. For example, consider the tours of two parents:

$$P_1 = (9 \quad 4 \quad 5 \quad 2 \quad 8 \quad 1 \quad 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \quad 9 \quad 7 \quad 8 \quad 2 \quad 4 \quad 5)$$

The first bit of second parent is the first bit of first offspring:

$$O_1 = (3 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

The selected bit is 3 and 3 is located at ninth position in first parent and the bit at this position in second parent is 5. For again searching 5 is at third position in first parent and 1 is at same position in second parent, so 8 is selected for second offspring as:

$$O_2 = (1 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

The previous bit was 1 and it is locates at sixth position in first parent and at this position bit is 8 in second parent, so

$$O_1 = (3 \quad 8 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

And for two moves as below 8 is 7 and below 7 is 4, so

$$O_2 = (1 \quad 4 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

Hence similarly;

$$O_1 = (3 \quad 8 \quad 6 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

and

$$O_2 = (1 \quad 4 \quad 9 \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times)$$

Now stop because the bit 9 has comes in second offspring which was in 1st position of first parent. One cycle is over and before starting other cycle, we match first offspring's bits with second parent or vice versa and leave out the existing bits with their position in both parents as:

$$P_1 = (\bullet \quad \bullet \quad 5 \quad 2 \quad 8 \quad \bullet \quad 6 \quad 7 \quad 3)$$

and

$$P_2 = (\bullet \quad \bullet \quad 1 \quad 9 \quad 7 \quad \bullet \quad 2 \quad 4 \quad 5)$$

Now filled positions of parents and '$\times$' positions of offspring are considered 1st, 2nd and 3rd positions etc., so we can completes it as usual:

$$O_1 = (3 \quad 8 \quad 6 \,|\, 1 \quad 4 \quad 9 \,|\, 7 \quad 2 \quad 5)$$

and

$$O_2 = (1 \quad 4 \quad 9 \,|\, 7 \quad 2 \quad 5 \,|\, 6 \quad 3 \quad 8)$$

Hence this scheme is over in three cycles.

3.5. **Partially-Mapped Crossover Operator.** The partially-mapped crossover (PMX) was proposed by Goldberg and Lingle [14]. After choosing two random cut points on parents to builds offspring, the portion between cut points are mapped with each other and the remaining information is exchanged. Consider, for example the two parents tours with randomly one cut point between 3rd and 4th bits and other cut point between 6th and 7th bits(the two cut points marked with "|"):

$$P_1 = (9 \quad 4 \quad 5 \,|\, 2 \quad 8 \quad 1 \,|\, 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \,|\, 9 \quad 7 \quad 8 \,|\, 2 \quad 4 \quad 5)$$

The mapping sections are between the cut points. In this example, the mappings are $2 \longleftrightarrow 9$, $8 \longleftrightarrow 7$ and $1 \longleftrightarrow 8$. Now two mapping sections are copied with each other to make offspring as:

$$O_1 = (\times \quad \times \quad \times \,|\, 9 \quad 7 \quad 8 \,|\, \times \quad \times \quad \times)$$

and

$$O_2 = (\times \quad \times \quad \times \,|\, 2 \quad 8 \quad 1 \,|\, \times \quad \times \quad \times)$$

Then we can fill further bits (from the original parents), for those which have no conflict as:

$$O_1 = (\times \quad 4 \quad 5 \,|\, 9 \quad 7 \quad 8 \,|\, 6 \quad \times \quad 3)$$

and

$$O_2 = (3 \quad 6 \quad \times \,|\, 2 \quad 8 \quad 1 \,|\, \times \quad 4 \quad 5)$$

Hence, the first $\times$ in the first offspring is 9 which comes from first parent but 9 is already in this offspring, so we check mapping $2 \longleftrightarrow 9$, so 2 will occupy first $\times$. Similarly, the second $\times$ in first offspring is 7 which comes from first parent but 7 exists in this offspring, check mapping as well $8 \longleftrightarrow 7$ and see again 8 exists in this offspring, again check mapping $8 \longleftrightarrow 1$ so 1 will occupy second $\times$. Thus the offspring 1 is:

$$O_1 = (2 \quad 4 \quad 5 \,|\, 9 \quad 7 \quad 8 \,|\, 6 \quad 1 \quad 3)$$

and Analogously, we complete second offspring as well:

$$O_2 = (3 \quad 6 \quad 7 \,|\, 2 \quad 8 \quad 1 \,|\, 9 \quad 4 \quad 5)$$

4. Proposed crossover operator

We are proposing a new crossover operator which works similarly as PMX, so we suggest it as PMX2. After choosing two random cut points on parents to builds offspring, the portion between cut points, one parent's string is mapped onto the other parent's string and the remaining information is exchanged. We differentiate PMX2 in the following steps.

Step 1: Choose two parents for mating.
Step 2: Apply two random cut points on these selected parents.
Step 3: Within cut points bits are exchanged into offspring.
Step 4: All those bits of first parent, which are not present in first offspring, placed at same locations.
Step 5: Repeat Step 4 for second parent and offspring.

Step 6: Now for left bits in offspring are mapping within the parents.

Consider, an example of the two parents tours with randomly one cut point between 3rd and 4th bits and other cut point between 6th and 7th bits are (the two cut points marked with "|"):

$$P_1 = (9 \quad 4 \quad 5 \,|\, 2 \quad 8 \quad 1 \,|\, 6 \quad 7 \quad 3)$$

and

$$P_2 = (3 \quad 6 \quad 1 \,|\, 9 \quad 7 \quad 8 \,|\, 2 \quad 4 \quad 5)$$

First two cut point sections are copied with each other to make offspring as:

$$O_1 = (\times \quad \times \quad \times \,|\, 9 \quad 7 \quad 8 \,|\, \times \quad \times \quad \times)$$

and

$$O_2 = (\times \quad \times \quad \times \,|\, 2 \quad 8 \quad 1 \,|\, \times \quad \times \quad \times)$$

Then we can fill further bits (from the original parents), for those which have no conflict as:

$$O_1 = (\times \quad 4 \quad 5 \,|\, 9 \quad 7 \quad 8 \,|\, 6 \quad \times \quad 3)$$

and

$$O_2 = (3 \quad 6 \quad \times \,|\, 2 \quad 8 \quad 1 \,|\, \times \quad 4 \quad 5)$$

Now the mapping of the "$\times$"are outside the cut points from the original tracks as:

$$9 \longleftrightarrow 3 \longleftrightarrow 5 \longleftrightarrow 1 \; and \; 7 \longleftrightarrow 4 \longleftrightarrow 6 \longleftrightarrow 2$$

Thus the offspring 1 is:

$$O_1 = (1 \quad 4 \quad 5 \,|\, 9 \quad 7 \quad 8 \,|\, 6 \quad 2 \quad 3)$$

and Analogously, we complete second offspring as well:

$$O_2 = (3 \quad 6 \quad 9 \,|\, 2 \quad 8 \quad 1 \,|\, 7 \quad 4 \quad 5)$$

In original PMX, the existing bits are mapped within the cut points, but in this novel approach, the existing bits are mapped from outside the cut points. In this way, most of the tracks coming from two approaches are the same. Hence, we suggest it as a modified form of partially-mapped crossover (PMX2) operator.

To apply this crossover operator, we made a MATLAB code for GAs and have given pseudo-code in Algorithm 1.

5. Performance evaluation

This work conducted a series of experiments to analyze the inheritance and evaluate the performance of PMX2. First, the parameter settings for GAs and benchmarks for this study are given in Section 5.1. Second, MATLAB software (version R2017a) was used to compare the simulation study among crossover operators with the help of an average, standard deviation (S.D) and relative error (R.E) and a detailed discussion on results in Section 5.2.

Algorithm 1 The Pseudo-code of PMX2-Operator

```
N ⟵ no. of cities
P1 ⟵ select random parent by selection method
P2 ⟵ select random parent by selection method
cut1 ⟵ select an random cut at same position of both parents
cut2 ⟵ select another random cut at same position of both parents
child1 ⟵ P2(cut1 to cut2)
child2 ⟵ P1(cut1 to cut2)
%For child1
i ⟵ 1
while (i <= N) do
    if(find(child1==p1(i)))
      value ⟵ P1(i)
      while find(P2 == value!=0) do
        location ⟵ find(P2 == value)
        value ⟵ P2(location)
      end while
        child1(i) ⟵ P2(location)
    else
        child1(i) ⟵ P1(i)
    end if
end while
%For child2
i ⟵ 1
while (i <= N) do
    if(find(child2==p2(i)))
      value ⟵ P1(i)
      while (find(P1 == value!=0)) do
        location ⟵ find(P1 == value)
        value ⟵ P1(location)
      end while
        child2(i) ⟵ P1(location)
    else
        child2(i) ⟵ P2(i)
    end if
end while
```

5.1. **Parameter Settings and Test Problems.** The genetic algorithms (GAs) are random search methods and give different results which might be obtained at the end of every run. Therefore, experiments were performed 30 times (30 independent runs) having different random seeds for each instance to achieve an acceptable solution. The selection of parameter values for GA is tedious and reflects the algorithm's performance. Table 1 lists the setting for the GA employed in our simulation study. Two stopping criteria have been

TABLE 1. Parametric configuration for GA

Parameter description	Value
Representation	Permutation
Population size (N)	150
Selection scheme	Roulette-wheel
Crossover probability (P_c)	90%
Mutation method	Exchange
Mutation probability (P_m)	10%
Maximum generation	5000
Number of trails	30
Replacement in GA	Steady-state GA

used; the first is when the maximum number of generations has been reached, and the second is whenever a tour shorter than the current optimum trip has not been found during 300 consecutive generations. All benchmark instances which are used in this study with the known solutions are given in Table 2. These benchmark instances are taken from traveling salesman problem library (TSPLIB) [32].

5.2. **Simulation Results and Discussion.** To compare the efficiency of various crossover operators (e.g. OX, NWOX, CX, CX2, PMX and the proposed approach PMX2) with the help of MATLAB software, we divided the benchmark instances into two groups as symmetric traveling salesman problems (TSPs) and asymmetric traveling salesman problems (ATSPs). The results of proposed and traditional crossover operators are prepared in the form of tabulated summary with an average and standard deviation (S.D) for TSPs and

TABLE 2. The benchmark problems

Problem name	No. of cities	Optimal tour length
burma14	14	3323
br17	17	39
gr21	21	2707
bayg29	29	1610
ftv33	34	1286
ftv38	39	1530
dantzig42	42	699
p43	43	5743
ft53	53	6905
eil76	76	538
eil101	101	629
ftv170	171	2755
brg180	180	1950
pr226	226	80369
rbg323	323	1326
rbg358	358	1163
rbg443	443	2720
att532	532	27686

ATSPs. Moreover, how much the average solution diverges from the global one, has been indicated by relative error (R.E) and calculated by:

$$Relative\ Error(\%) = \frac{Solution\ Value - Optimal\ Value}{Optimal\ Value} \times 100 \qquad (5.\ 4)$$

The Table 3 shows the effect of six different crossover operators included to compare proposed PMX2 with GAs examined in nine different TSPs solutions with the help of average length, S.D and R.E. The benchmarks in Table 3 are arranged in the ascending order of nodes and quality of the solution is investigated for the purpose of performance evaluation of all six crossover operators. For the benchmarks burma14 and eil76, the OX performs better than all used operators with respect to the average and a low R.E in all 30 runs. For least dispersion, it shared the equal performance with the proposed PMX2 for instance burma14 and a low S.D observed for problem eil76 by PMX operator. The gr21, bayg29, dantzig42 and pr226 benchmarks in Table 3 give a less R.E by the proposed PMX2 with a low mean value. A low S.D computed in all 30 independent runs from the CX2 for gr21, bayg29 and pr226 instances and OX for dantzig42 instance. PMX operator outstanding performs in all three aspect of results measurements for benchmark eil101. The proposed operator PMX2 performs outstanding on the basis of quality of solution, as a whole, for the benchmarks brg180 and att532. The last row of the Table 3 indicates the average performance of average results, S.D and R.E for all nine benchmark instances. This shows the best performance noted by the proposed operator PMX2 which is equal only in one aspect i.e. S.D with OX operator. For visual comparison, the relative errors calculated according to the average solution for all used instances in Table 3 are displayed in Figure 4.

We continue this study for various asymmetric traveling salesman problems (ATSPs) instances, which are reported in Table 4. This elaborates the results according to average, S.D and R.E for nine instances of size 17 to 443. The solution quality of the used operators is insensitive to the number of runs for instance br17. For problem ftv33, the operator CX2 performs better on the average and R.E results but PMX2 performs better in terms of a low S.D. For problem ftv38, the CX operator gives better results on average and R.E, but OX operator put a low S.D. The proposed operator PMX2 gives a better average and least R.E for p43 and ftv170 instances but a low S.D measured by CX and CX2 operators, respectively. For instances ft53 and rbg323, PMX operator performs better on the average and R.E results but CX and CX2 operators are better if we consider S.D. For the last two benchmark instances rbg358 and rbg443, OX operator performs well on the average and R.E, but a small dispersion observed by PMX and CX2 operators respectively. The last row of the Table 4 indicates the average performance on the average, S.D and R.E results for all nine benchmark instances. This shows the best performance posed by the operator PMX but a low average of S.D and R.E results indicated the best performance in all by the proposed operator PMX2. In order to be better understanding, the R.E results were presented with visual graphic in Figure 5 along with displayed results in Table 4.

6. CONCLUSION

This article represents a comprehensive overview of the performance of GAs in NP-hard problems like TSP and keenly observes how GAs create solution without having any prior knowledge about the traveling routes. Unlike other meta-heuristic methods, GA uses natural rules of selection, crossover and mutation to make the computation easier and fast. These aspects make it more valuable, better performing and efficient algorithm over others. The various crossover operators have been introduced for TSP by using GAs. We proposed a new crossover operator for TSP. This proposed operator PMX2 upgrade the path-represented PMX and enhanced the quality of offspring. PMX2 is easy to execute and always generates a valid tour of offspring. Eighteen benchmark instances from the TSPLIB have been used to assess its performance against other operators, for comparison and to investigate how they converge on the basis of average results. We observed their sensitivity to get results in different runs with the help of an absolute measure of S.D. For more close comparison, we measured R.E of the results. All simulation results show that PMX2 is efficient, so proposed operator might be good candidate to get accurate convergent results. Moreover, researchers might be more confident to apply it for comparisons.

DATA AVAILABILITY

The data used to support the findings of this manuscript are taken from the website of TSPLIB (https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/).

ACKNOWLEDGMENTS

The authors are grateful to the editor and referees for their valuable comments and guidance that have made this article worthy to be published.

TABLE 3. Comparison results of crossover operators for TSPs

	Crossover Operator																	
	OX			**NWOX**			**CX**			**CX2**			**PMX**			**PMX2**		
Instance	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E
burma14	**3331**	**8**	**0.24**	3341	10	0.54	3342	12	0.57	3340	10	0.51	3334	11	0.33	3332	**8**	0.27
gr21	2822	72	4.25	2841	91	4.95	2829	49	4.51	2818	**38**	4.10	2854	57	5.43	**2796**	52	**3.29**
bayg29	1648	**12**	2.36	1652	22	2.61	1639	17	1.80	1657	31	2.92	1649	20	2.42	**1637**	18	**1.68**
dantzig42	752	20	7.58	759	37	8.58	761	35	8.87	722	**17**	3.29	722	23	3.29	**718**	22	**2.72**
eil76	**558**	18	**3.72**	597	28	10.97	587	17	9.11	582	21	8.18	562	**15**	4.46	561	21	4.28
eil101	652	17	3.65	661	19	5.09	647	16	2.86	669	22	6.36	**642**	**9**	**2.07**	645	12	2.54
brg180	2058	93	5.54	2109	118	8.15	2087	75	7.03	2071	82	6.21	2048	72	5.03	**2043**	**49**	**4.77**
pr226	82464	243	2.61	82811	394	3.04	82893	423	3.14	82667	**229**	2.86	82690	365	2.89	**82311**	328	**2.42**
att532	29854	517	7.83	31052	814	12.16	30812	671	11.29	29978	528	8.28	29712	469	7.32	**29444**	**492**	**6.35**
Average	13793	**111**	4.20	13980	170	6.23	13955	146	5.46	13837	109	4.75	13801	116	3.69	**13721**	**111**	**3.15**

TABLE 4. Comparison results of crossover operators for ATSPs

	Crossover Operator																	
	OX			NWOX			CX			CX2			PMX			PMX2		
Instance	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E	Average	S.D	R.E
br17	**39**	**0**	**0.00**	40	1	2.56	41	1	5.13	**39**	**0**	**0.00**	**39**	**0**	**0.00**	**39**	**0**	**0.00**
ftv33	1407	93	9.41	1472	118	14.46	1429	82	11.12	**1391**	102	**8.16**	1441	89	12.05	1417	**66**	10.19
ftv38	1687	**68**	10.26	1732	97	13.20	**1618**	84	**5.75**	1664	89	8.76	1678	75	9.67	1669	69	9.08
p43	5862	102	4.31	5919	98	5.32	5847	**67**	4.04	5891	91	4.82	5798	83	3.17	**5761**	72	**2.51**
ft53	7432	162	7.63	8137	197	17.84	7714	**140**	11.72	7704	208	11.57	**7417**	169	**7.41**	7492	202	8.50
ftv170	3138	215	13.90	3216	194	16.73	3076	209	11.65	3119	**169**	13.21	3093	210	12.27	**3057**	182	**10.96**
rbg323	1817	175	37.03	1894	201	42.84	1824	162	37.56	1799	**154**	35.67	**1778**	180	**34.09**	1814	161	36.80
rbg358	**1412**	140	**21.41**	1502	159	29.15	1469	191	26.31	1497	204	28.72	1423	**118**	22.36	1433	192	23.22
rbg443	**3614**	152	**32.87**	3869	214	42.24	3814	197	40.22	3792	**108**	39.41	3652	159	34.26	3641	121	33.86
Average	2934	123	15.20	3087	142	20.48	2981	126	17.06	2988	125	16.70	**2924**	120	15.03	2925	**118**	**15.01**

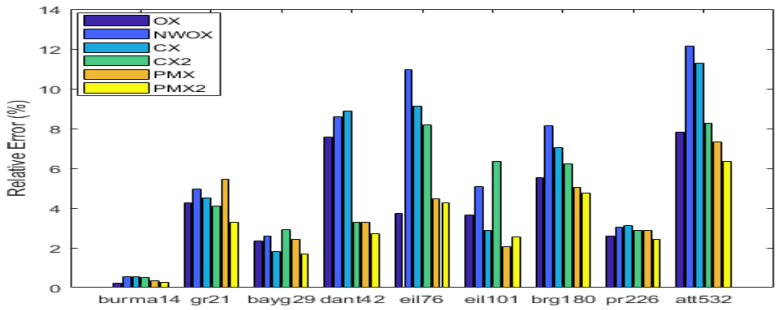

FIGURE 4. The relative error of average length for TSPs

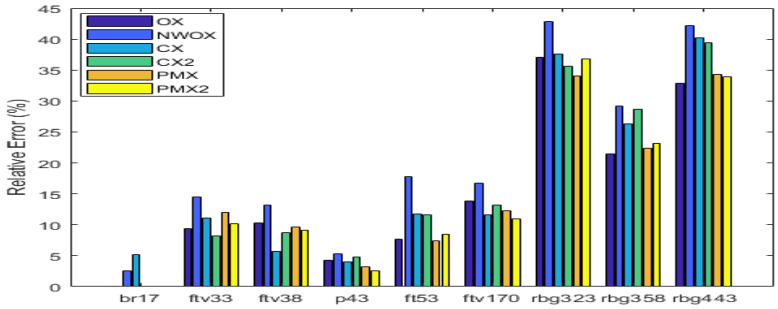

FIGURE 5. The relative error of average length for ATSPs

REFERENCES

[1] Z. H. Ahmed, *Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator*, International Journal of Biometrics & Bioinformatics (IJBB) **3**, No. 6 (2010) 96-105.

[2] B. F. Al-Dulaimi and H. A. Ali, *Enhanced traveling salesman problem solving by genetic algorithm technique*, In: Proceedings of World Academy of Science, Engineering and Technology **38**, (2008) 296-302.

[3] D. Applegate, R. Bixby, V. Chvatal and W. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 1st edition, New Jersey, USA, 2006.

[4] M. Bhattacharyya and A. K. Bandyopadhyay, *Comparative study of some solution methods for traveling salesman problem using genetic algorithms*, CYBERNET SYST **40**, No. 1 (2009) 1-24.

[5] S. Bhide, N. John and M. R. Kabuka, *A Boolean neural network approach for the traveling salesman problem*, IEEE T COMPUT **42**, No. 10 (1993) 1271-1278.

[6] R. G. Bland and D. F. Shallcross, *Large traveling salesmen problems arising from experiments in x-ray crystallography*, OPER RES LETT **8**, (1988) 125-128.

[7] V. A. Cicirello, *Non-wrapping order crossover: An order preserving crossover operator that respects absolute position*, Proceedings of the 8th annual conference on Genetic and evolutionary computation, (2006) 1125-1132.

[8] M. unka and M. Y. Ozsaglam, *A comparative study on particle swarm optimization and genetic algorithms for traveling salesman problems*, CYBERNET SYST **40**, No. 6 (2009) 490-507.

[9] L. Davis, *Applying Adaptive Algorithms to Epistatic Domains*, In: Proceedings of the International Joint Conference on Artificial Intelligence (1985) 162-164.

[10] K. Deep and H. M. Adane, *New variations of order crossover for traveling salesman problem*, International Journal of Combinatorial Optimization Problems and Informatics **2**, No. 1 (2011) 2-13.

[11] M. Dorigo and L. M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem*, IEEE T EVOLUT COMPUT **1**, No. 1 (1997) 53-66.

[12] G. Finke, A. Claus and E. Gunn, *A two-commodity network flow approach to the traveling salesman problem*, Congresses Numeration **41**, (1984) 167-178.

[13] M. Gen and R. Cheng, *Genetic algorithms and Engineering design*, John Wiley and Sons, London, UK, 1997.

[14] D. E. Goldberg and R. Lingle, *Alleles, loci, and the traveling salesman problem*, In: Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications, Hillsdale, New Jersey: Lawrence Erlbaum, (1985) 154-159.

[15] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley Publishing Company, 1989.

[16] Y. He, Y. Qiu, G. Liu and K. Lei, *A parallel adaptive tabu search approach for traveling salesman problems*, In: IEEE International Conference on Natural Language Processing and Knowledge Engineering (2005) 796-801.

[17] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, Oxford, UK, 1975.

[18] A. Hussain, Y. S. Muhammad, M. N. Sajid, I. Hussain, M. A. Shoukry and S. Gani, *Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator*, COMPUT INTEL NEUROSC **2017**, (2017) 1-7.

[19] P. Larranaga, C. M. Kuijpers, R. H. Murga, I. Inza and S. Dizdarevic, *Genetic algorithms for the traveling salesman problem: A review of representations and operators*, ARTIF INTELL REV **13**, No. 2 (1999) 129-170.

[20] S. Lin and B. W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem*, Operations research **21**, No. 2 (1973) 498-516.

[21] M. K. Mahmood and S. Ali, *A Novel Labeling Algorithm on Several Classes of Graphs*, Punjab Univ. j. math. **49**, No. 2 (2017) 23-35.

[22] V. Maniezzo and M. Roffilli, *Very strongly constrained problems: an ant colony optimization approach*, CYBERNET SYST **39**, No. 4 (2008) 395-424.

[23] P. Miliotis, *Using cutting planes to solve the symmetric traveling salesman problem*, MATH PROGRAM **15**, No. 1 (1978) 177-188.

[24] C. Moon, J. Kim, G. Choi and Y. Seo, *An efficient genetic algorithm for the traveling salesman problem with precedence constraints*, EUR J OPER RES **140**, No. 3 (2002) 606-617.

[25] Y. Nagata and D. Soler, *A new genetic algorithm for the asymmetric traveling salesman problem*, EXPERT SYST APPL **39**, No. 10 (2012) 8947-8953.

[26] I. M. Oliver, D. Smith and J. R. Holland, *Study of permutation crossover operators on the traveling salesman problem*, In: Grefenstette, J. J. (ed.) Genetic Algorithms and Their Applications, Proceedings of the Second International Conference. Hillsdale, New Jersey: Lawrence Erlbaum, (1987) 224-230.

[27] A. Osyczka, *Evolutionary algorithms for single and multicriteria design optimization*, Heidelberg, Germany: Springer-Verlag, Physica-Verlag, 2001.

[28] A. Piwonska, *Genetic algorithm finds routes in traveling salesman problem with profits*, Zeszyty Naukowe Politechniki Bialostockiej Informatyka, (2010) 51-65.

[29] J. Y. Potvin, *Genetic algorithms for the traveling salesman problem*, ANN OPER RES **63**, No. 3 (1996) 337-370.

[30] K. Puljic and R. Manger, *Comparison of eight evolutionary crossover operators for the vehicle routing problem*, MATH COMMUN **18**, No. 2 (2013) 359-375.

[31] C. P. Ravikumar, *Parallel techniques for solving large scale traveling salesperson problems*, MICROPROCESS MICROSY **16**, (1992) 149-158.

[32] G. Reinelt, *TSPLIB http://www. iwr. uni-heidelberg. de/groups/comopt/software*, TSPLIB95, 1995.

[33] C. H. Song, K. Lee and W. D. Lee, *Extended simulated annealing for augmented TSP and multi-salesmen TSP*, In: Proceeding of International Joint Conference on Neural Networks **3**, (2003) 2340-2343.

[34] Y. Q. Tao, D. W. Cui, X. L. Miao and H. Chen, *An improved swarm intelligence algorithm for solving TSP problem*, Lecture Notes of Artificial Intelligence, (2007) 813-822.

[35] K. P. Wang, L. Huang, C. G. Zhou and W. Pang, *Particle swarm optimization for traveling salesman problem*, In: Proceedings of the Second International Conference on Machine Learning and Cybernetics **3**, (2003) 1583-1585.

[36] W. Pang, K. P. Wang, C. G. Zhou, L. J. Dong, M. Liu, H. Y. Zhang and J. Y. Wang, *Modified particle swarm optimization based on space transformation for solving traveling salesman problem*, In: Proceedings of the Third International Conference on Machine Learning and Cybernetics **4**, (2004) 2342-2346.