Sigma Journal of Engineering and Natural Sciences, Vol. 42, No. 6, pp. XX-XX, December, 2024
Yildiz Technical University Press, Istanbul, Turkey

Modified genetic algorithm with novel crossover and mutation operator for travelling salesman problem

M.K. SHARMA[1], Sadhna CHAUDHARY[1], Laxmi RATHOUR[2,*], Vishnu Narayan MISHRA[3*]

[1]*Department of Mathematics, Chaudhary Charan Singh University, Meerut, 250004, India*
[2]*Department of Mathematics, National Institute of Technology,Chaltlang, Aizawl 796 012, Mizoram, India*
[3*]*Department of Mathematics, Faculty of Science, Indira Gandhi National Tribal University, Lalpur, Amarkantak,Anuppur, Madhya Pradesh 484 887,India*

*Corresponding author

ABSTRACT

In this study, Genetic Algorithm (GA), a sort of randomized direct, iterative search methodology built around natural selection, is employ in computers to discover approximations of solutions to optimisation and search issues. GA employs operators including selection, crossover, and mutation to tackle. In case of NP-hard issues, particularly for travelling salesman problem (TSP), the GAs is beneficial. To reduce the overall distance, we propose a novel crossover operator with its python code for the TSP. Along with the Python pseudo coding, we additionally introduced a mutation operator to enhance the consummation of GA in determining the shortest distance in the TSP. To emphasize the proposed crossover and mutation operator, we also illustrate different steps using examples. We integrated path representation with our developed crossover and mutation operator as it is apparent method to represent a tour.

Keywords: *Crossover Operator; Genetic Algorithm; Muation Operator; Python Coding; Travelling Salesman Problem*

INTRODUCTION

Almost The basic concept of genetic algorithms (GA) is a search-based optimisation approach and is introduced by Holland [1]. The 'survival of the fittest' premise is the foundation for GA, which are metaheuristics relies on natural selection and Genetics principles. GA are often utilised to produce high quality and superior solutions for search and optimisation challenges. In other words, GA tend to find and offer near-optimal solutions to scenarios that could require a lot of time. GA are frequently employed in numerous disciplines, comprises of soft computing, machine learning, and operations research. It can optimise for continuous or discrete variables without needing to know the derivatives. Additionally, it works with numerically generated data, experimental data, or analytical data and delivers a set of optimal variables rather than simply one solution. GA processes sustain a population of individuals and are iterative in character. In essence, GAs can be described as composed of two primary phases: the first is "Selection" for producing the next generation, and the second is "Manipulation," which manipulates the selected individuals to produce the next generation using various techniques, including crossover and mutation. Each iteration in GA is referred to as "a generation," and a population of new candidate solutions is created utilising various biologically inspired operators including mutation, crossover, and selection. In GA, each individual is represented by a string known as chromosome and may also be regarded as a problem-solving strategy. These strings comprise characters known

This paper was recommended for publication in revised form by Regional Editor Ahmet Selim Dalkilic
[1] Department of Mathematics, Chaudhary Charan Singh University, Meerut, 250004, India
[2] Department of Mathematics, National Institute of Technology,Chaltlang, Aizawl 796 012, Mizoram, India
[3] Department of Mathematics, Faculty of Science, Indira Gandhi National Tribal University, Lalpur, Amarkantak,Anuppur, Madhya Pradesh 484 887,India*
** E-mail address:* vnm@igntu.ac.in ; vishnunarayanmishra@gmail.com
Orcid id: https://orcid.org/0000-0003-3071-5931 M. K. Sharma, 0000-0003-1255-0395 Sadhna Chaudhary, 0000-0002-2659-7568 Laxmi Rathour, 0000-0002-2159-7710 Vishnu Narayan Mishra
Manuscript Received 11 July 2023, Revised 19 September 2023, Accepted 11 October 2023

as genes, which contain certain values known as alleles. GAs is appropriate candidate to tackle the constrained, unconstrained and combinatorial problem.

Using genetic operations like selection, fitness, crossover, and mutation processes, GA seeks for the optimal results.

- **Fitness:** The fitness value quantifies the similarity between two individuals and is a favourable utility metric that is determined for every individual in the population.
- **Selection:** Each member of the population receives several copies, which are used up in the mating pool to create an entirely novel population. Therefore, the likelihood that an individual will produce additional copies in the mating pool increases as fitness value increases**.**
- **Crossover:** Recombination of individuals generates new individuals known as offspring or children. One-point and two-point crossover are popular recombination strategies.
- **Mutation:** Maintaining diversity in the population can be done through mutation. Each individual is mutated with a minute or extremely low chance, such as less than 1.0.

The procedures below can be used to define an uncomplicated genetic algorithm and the flow chart for illustrating various steps is shown in Figure 1. The 2-diemnsional array encompassing population size and chromosome size defined the population. Here, population initialization can be done by utilizing two methods namely random and heuristic initialization.The fitness function ought to be quick enough to calculate. It must quantify the degree to which a given solution is fit or the degree to which fit people can be created from the provided solution.

When a GA run ends, a lot depends on the termination condition of the GA. In general, we want a termination condition that, at the end of the run, puts the outcome very near to the bestone. The following are the termination criteria's for the GA-

- when X iterations have passed with no population improvement.
- when the number of generations is fixed.
- when the value of the objective function reaches a specific, predetermined value.

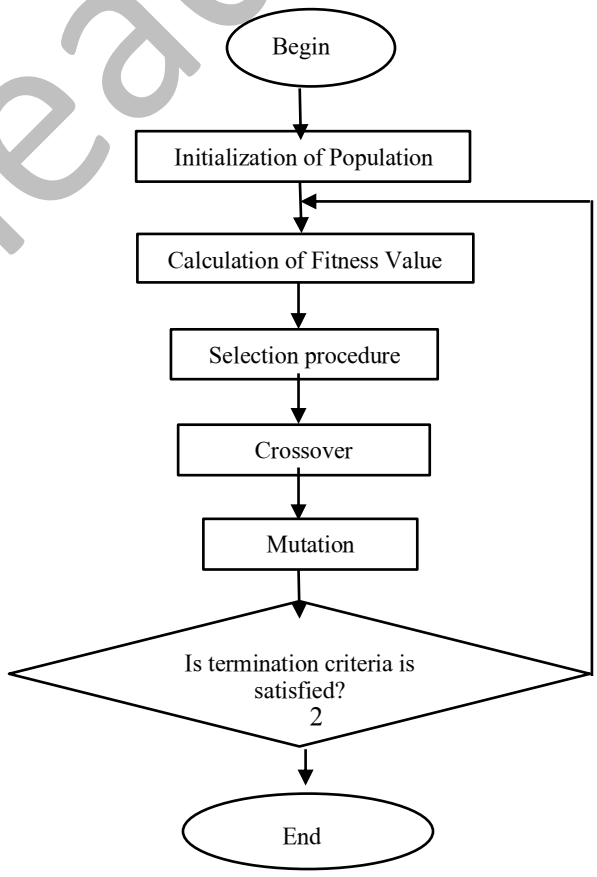

Figure 1. Flow chart of GA

I. Using n chromosomes, create a starting generation. Here the population is initialized.
II. Assess each chromosome's fitness.
III. Proportionally pick $n/2$ parents of the present population.
IV. Use the crossover operator to produce children by picking two parents at random.
V. Employ mutation to vary findings a little bit.
VI. Until all parents have been chosen and mated, repeat steps 4 and 5.
VII. An entirely new population of chromosomes will replace the old one.
VIII. Determine each chromosome's level of fitness in novel population.
IX. Stop when the number of generations reaches a predetermined maximum; otherwise, proceed to Step Ⅲ.

We have numerous representations in literature employing the GAs. Path, binary, adjacency,ordinal and matrixrepresentation are some significant representations and the summary of these representation with novel crossover operator is given in Table 1.

Table 1: list of different representations

Representation	**Crossover Operator**	**Author**
Binary	Classical, repair	Lidd (1991) [2]
Path	Partially-mapped	Goldberg and Lingle, (1985) [3]
	Order	Davis (1985) [4]
	Sorted match	Brady (1985) [5]
	Heuristic	Grefenstette (1987b) [6]
	Maximal preservative	Muhlenbein et al. (1988) [7]
	Voting recombination	Muhlenbein et al. (1989) [8]
	Order based	Syswerda (1991) [9]
	Heuristic	Grefenstette (1987) [6]
	Order based	Syswerda (1991) [10]
	Position based	Syswerda (1991) [10]
	Alternating-positions	Larranaga et al. (1996a) [11]
Adjacency	Alternative edge	Grefenstette et al. (1985) [12]
	Heuristic 1	Grefenstette et al. (1985) [12]
	Subtour chunks	Grefenstette et al. (1985) [12]
Ordinal	Classical operator	Grefenstette et al. (1985) [12]

Arqub et al. [13] employ the continuous GA to solve singular two-point boundary value problems. Arqub and Hammour ([14] discussed a method for employing continuous GA for solving systems of second-order boundary value problems. In order to validate this method, a few test problems were created and solved. Hammour et al. [15] presented a GA approach for the modelling of dynamical systems. For numerically approximating the solutions of Troesch's and Bratu's problems, Hammour et al. [16] introduce continuous GA. Recently, to approximate a class of Lebesgue integrable functions, Raiz et al. [17] introduced a novel sequence of linear positive operators. Schurer Beta bivariate operators were initially developed by Mishra et al. [18] in terms of generalisation exponential functions and their approximation characteristics.

The Travelling Salesman Problem (TSP) is one of the most well-known combinatorial issues in optimising. The TSP is one of the most recent optimisation problems to have undergone extensive deliberation. It was initially

formulated as an optimisation problem in 1930. In TSP, the objective is to determine probable tour such that a travelling salesman visits each city exactly once and back to the initial city in order to minimize the total cost devoted or total distance covered.Since there are n! various approaches to locate the tour for n cities, specifically for 11 cities, there are 39916 800 possible route to optimize the total cost. So, the complexity of finding the best route increases as the number of cities increases. Thus, TSP is a candidate of NP (Non- Polynomial) hard combinatorial optimization problems.

Table 2: Summary of approaches for TSP

Exact Approach	Heuristic Approach
Branch and bound	Genetic Algorithm
Cutting Planes	Neural Network
Branch and Cut	Simulated Annealing
Others	Tabu Search
	Particle Swarm Optimization
	Ant Colony Optimization

In existing literature, exact and metaheuristic algorithms are two approaches to tackle TSP as shown in table 1. In case of exact algorithms, following are major exact algorithms in literature introduced to encounter with TSP; Dantzig et al. [19] introduced a methodology to solve the large-scale TSP. Later, Petberg [20] proposed a branch and cut method to get the optimal solution of symmetric TSP. A cutting plane approach is proposed by Fleischmann [21] in order to tackle the TSP in case of a road network. Thereafter, Petberg and Homg [22] proposed branch and cut method to optimize symmetric TSP with 532 cities. On the other hand, various heuristic algorithms are also introduced in order to tackle the TSP. Initially, Brady [5] proposed a GA approach deal with TSP. Bhide et al. [23] proposed a Boolean approach with the help of neural network to deal with TSP. Dorigo and Gambardella [24] proposed an approach with the help of ant colony system to tackle the TSP. Knox [25] proposed the tabu search approach to solve the symmetric TSP. Later, Chiang and Russell [26] proposed simulated annealing algorithms to cope with vehicle routing problem. Thereafter, Focacci et al. [27] developed a hybrid exact method for TSP. A local search strategy was presented by Ibarki et al. [28] for addressing and arranging issues with extensive time window limitations. Larranaga et al. [11] reviewed the various methodologies used to resolve TSP by utilizing GA. Also, presented different crossover and mutation operators which are proposed to tackle the TSP with the GA. Thereafter, An amalgam GA was suggested by Nguyen et al. [29] to discover the TSP solution. Ghadle and Muley [30] proposed a modified version of GA encoded by using MATLAB to tackle the TSP. Kumar and Gupta [31] proposed a methodology to solve the TSP with fuzzy L-R parameters. Majumdar and Bhunia [32] modelled an asymmetric TSP in a way that the distance between each pair of cities travelled is denoted as an interval value instead of a precise value. Thereafter, Changdar et al. [33] modelled a multi-objective TSP with triangular fuzzy parameters and, proposed an effectual GA to tackle this modelled TSP. Maity et al. [34] proposed a modified GA to cope with constrained solid TSP in different settings including fuzzy and crisp.

We suggest a novel crossover operator for the TSP along with its Python source code. We entailed a mutation operator in addition to the Python pseudo coding to improve the effectiveness of GA in calculating the shortest distance in the TSP. We additionally employ examples to demonstrate the proposed crossover and mutation operator at various stages. We combined our newly designed crossover and mutation operator with path representation because this is an obvious way to express a tour.

This article is organized as follows; Section one is completely devoted to the basics of GA and the literature review of GA and TSP. The mathematical formulation of the TSP is described in section two. Several types of representation involved in GA are described in section three. Major crossover operators for path representation are presented in

section four. Novel crossover operator is illustrated in section five. On the other hand, the novel mutation operator is given in section six. Finally, the section seven is devoted to the conclusion of the article.

MATHEMATICAL FORMULATION OF TSP

One way to represent the TSP is as an integer linear programming. The Dantzig-Fulkerson-Johnson (DFJ) formulation and the Miller-Tucker-Zemlin (MTZ) formulation are well-known formulations of TSP available in the literature (Dantzig [35] & Velednitsky [36]). In some circumstances, the MTZ formulation is still beneficial, although the DFJ formulation is stronger.

Let n be the number of cities, c_{ij} be the cost (distance) form ith to jth city, and u_i be the dummy variable. x_{ij} is a binary variable and defined as:

$$x_{ij} = \begin{cases} 1 & the\ path\ goes\ from\ city\ i\ to\ city\ j \\ 0 & otherwise \end{cases}$$

The MTZ formulation of TSP is as follows:

$$min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij}\, x_{ij}:$$

$$x_{ij} \in \{0,1\} i, j = 1,2 \dots, n;$$
$$u_i \in \boldsymbol{Z} i, j = 1,2 \dots, n;$$

$$\sum_{j \neq i, i=1}^{n} x_{ij} = 1 j = 1,2 \dots, n;$$
$$\sum_{j \neq i, j=1}^{n} x_{ij} = 1 i = 1,2 \dots, n;$$
$$u_i - u_j + (n-1) x_{ij} \leq n - 22 \leq i \neq j \leq n;$$
$$2 \leq u_i \leq n2 \leq i \leq n;$$

The DFJ formulation of TSP is as follows:

$$min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij}\, x_{ij}:$$

$$\sum_{j \neq i, i=1}^{n} x_{ij} = 1 j = 1,2 \dots, n;$$
$$\sum_{j \neq i, j=1}^{n} x_{ij} = 1 i = 1,2 \dots, n;$$
$$\sum_{i \in Q}^{n} \sum_{j \neq i, j \in Q}^{n} x_{ij} \leq |Q| - 1 \forall Q \subsetneq \{1, \dots, N\} |Q| \geq 2$$

Here, the last constraint, known as a subtour elimination constraint, assures that no appropriate subset Q can form a sub-tour, resulting in a single tour as the solution and not a union of smaller tours.

DIFFERENT TYPES OF REPRESENTATION

There have been a wide range of representations of a chromosome to solve TSP problem by employing GA. Binary, path, adjacency, ordinal and matrix representation are major representation forms available in literature.

Binary Representation

Each city in the n-cities TSP is represented in binary as a string of [log2n] bits, and an individual is represented as a string of n[log2n] bits.

Example: In case of a 6-cities TSP, each city assigned by 3-bit string. The tour 2-1-3-6-5-4 depicted as by using table 3.

(001 000 010 011 100 101)

Table 3. An illustration of a visit of six cities in binary

i	City i	i	City i
1	000	4	101
2	001	5	100
3	010	6	011

Path Representation

The elementary representation of a tour in TSP can be done in a more appropriate way by using path representation. For a tour of n cities, if city i is the jth element of the list, city i is the jth city to be visited.

Example: If n =8. Then the tour is 4-2-3-1-7-5-8-6 is represented as a string (4 2 3 1 7 5 8 6).

Ordinal Representation

In ordinal representation, ith member in set is a integer between 1 to $n - i + 1$ and an ordered set consisting of various destinations act as guide also exists.

Example: Let $n = 8\ and$ O= (1 2 3 4 5 6 7 8) be a reference list, then the tour 1-5-3-2-8-4-7-6 is represented by

$$T = (1\ 4\ 1\ 2\ 1\ 4\ 1\ 2\ 1)$$

Matrix Representation

In matrix representation, member ith row and jth column is 1 iff city i is visited before the city j.

Example: The matrix representation of tour 2-3-1-4 is

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

CROSSOVER OPERATORS FOR TSP IN PATH REPRESENTATOION

Partially Mapped (PMX), Cycle (CX), Cycle (CX2) are predominantly used crossover operators of path representation in the current literature.

Partially Mapped (PMX)

Partially Mapped Crossover operator (PMX) was introduced by **Goldberg and Lingle [37]** for path representation of chromosomes in path representation. In PMX, after choosing two random cut locations on the parents to produce

offspring, one parent's string is mapped onto the other parent's string. Thereafter, a remaining bit are filled with the help of mapping with the constraint that no bit is repeated in the offspring.

Example: Let us consider two parents P_1 and P_2 with two random cut points

$$P_1 = (1\ 2 \mid 3\ 4\ 5\ 6 \mid 7\ 8)$$
$$P_2 = (2\ 7 \mid 5\ 8\ 4\ 1 \mid 6\ 3)$$

Then, the mapping segment between the cut points replicated with each other in order to build offspring, the mapping is 2 ↔ 1,7 ↔ 6,1 ↔ 8.

$$O_1 = (\times\times \mid 1\ 4\ 5\ 8 \mid \times\times)$$
$$O_2 = (\times\times \mid 3\ 4\ 5\ 6 \mid \times\times)$$

Cycle Crossover Operator (CX)

Cycle crossover (CX) operator was first introduced by **Oliver et al. [38]**. Any bit in this operator is obtained via any of the parents to determine its position. Let P_1 and P_2 be two parent string to illustrate the operator

$$P_1 = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$$
$$P_2 = (2\ 4\ 6\ 8\ 7\ 5\ 3\ 1\ )$$

Now, select 1st element of offspring from first element of either of P_1 or P_2. Here from two options (1or 2), we will choose 1. For last element of O_1, we must consider 8, as if 1 is taken then it would not result as a legal tour.

$$O_1 = (1\ \times\times\times\times\times\times\ 8)$$

Similarly, for 2nd and 4th element, we have 2 and 4 respectively.

$$O_1 = (1\ 2\ \times\ 4\times\times\times\ 8)$$

The relative positions of the elements selected up until this point are considered to form a tour. Think about the 3rd element of the O_1. Any of the parents can be the source for this component. Let's say we choose it to come from parent 2. This indicates that the second parent must also be picked for the 5th, 6th, and 7th components of offspring because they make up another cycle. As a result, we discover the following offspring:

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

Cycle Crossover Operator (CX) 2

Hussain et al. [39] proposed cycle crossover operator 2 (CX2) for TSP to optimize distance travelled.

Step1. Let us consider two parents for mating. Choose the 1st bit of the 1st offspring using 2nd parent string.
Step2. The bit selected in Step 1 is present in the 1st parent, followed by the same similar location bit selected in the 2nd parent, which is present in the 1st parent, and the similar location bit selected in the 2nd parent, which is then selected for the first bit of the 2nd offspring.
Step3. The 1st parent will have the chosen bit from Step 3, and the 1st offspring would contain the following bit in the 2nd parent's identical place.
Step 4. Continue Steps 2 and 3 until the 1st bit of the 1st parent does not appear in the 2nd offspring, at which point the procedure may be stopped.
Step 5. If any bits remain, they will be similar in the 1st parent and the 2nd offspring up to this point, and vice versa for both parents.
Example:
Let us consider two parent chromosomes for mating

$$P_1 = (3\ 4\ 8\ 2\ 7\ 1\ 6\ 5\ )$$
$$P_2 = (4\ 2\ 5\ 1\ 6\ 8\ 3\ 7)$$

By employing above steps, the two offspring initiated are as follows:

$$O_1 = (4\ \ 8\ 6\ 2\ 5\ 3\ 1\ 7)$$
$$O_2 = (\ 1\ 7\ 4\ 8\ 6\ 2\ 5\ 3)$$

PROPOSED CROSSOVER OPERATOR

In this section we will propose the Increasing partially mapped crossover operator (IPMX) and the python coding of the proposed operator is given in Figure 2. The proposed operator is defined by the following steps.

Step 1. Select a pair of chromosome parents for mating.
Step 2. Pick out two random cut points on each parent to construct two offspring.
Step 3. First arrange the bits between two random cuts in the increasing order for each parent chromosome
Step 4. Now these portions between cut points are mapped onto other parent strings.
Step 5. Fill the remaining bits from the primary parent such that there is no dispute.
Step 6. To fill the bits with conflict, use the notion of partial mapping.

Example: Let us consider two parent chromosomes for mating with randomly two cut points marked by "|".

$$P_1 = (1\ 2\ |\ 3\ 4\ 5\ 6\ |\ 7\ 8)$$
$$P_2 = (2\ 7\ |\ 5\ 8\ 4\ 1\ |\ 6\ 3)$$

The two children $O_1\ and\ O_2$ are constructed as follows:
Initially arrange all bits of randomly selected segment in the increasing order. Then mapped the resulted strings between cut points onto other parents.

$$O_1 = (\times\times\ |\ 1\ 4\ 5\ 8\ |\ \times\times)$$

$$O_2 = (\times\times\ |\ 3\ 4\ 5\ 6\ |\ \times\times)$$

Now, fill the bits from original parent which does not have any conflict. Here for O_1, 2,7 are filled and for O_2, 2,7 are filled.

$$O_1 = (\times\ 2\ |\ 1\ 4\ 5\ 8\ |\ 7\ \times)$$
$$O_2 = (2\ 7\ |\ 3\ 4\ 5\ 6\ |\ \times\times)$$

To fill the remaining bits, use the notion of partially mapping. The first × in first offspring O_1 is 1 which comes from P_1 but 1 is already present in O_1. So, in P_1, we check the element corresponding to 1 of P_2, here $1 \leftrightarrow 6$, first × is occupies by 6. Again, the second × in the O_1 is 8 but 8 is already present in O_1. So, in P_1, we check the element corresponding to 8 of P_2, here $8 \leftrightarrow 4$, but 4 is present in O_1, again check mapping $4 \leftrightarrow 5$. But 5 is existing in O_1, again check $5 \leftrightarrow 3$, 3 occupies the second × of in P_1.

$$O_1 = (6\ 2\ |\ 1\ 4\ 5\ 8\ |\ 7\ 3)$$
$$O_2 = (2\ 7\ |\ 3\ 4\ 5\ 6\ |\ 1\ 8)$$

Similarly, we complete the second offspring O_1.

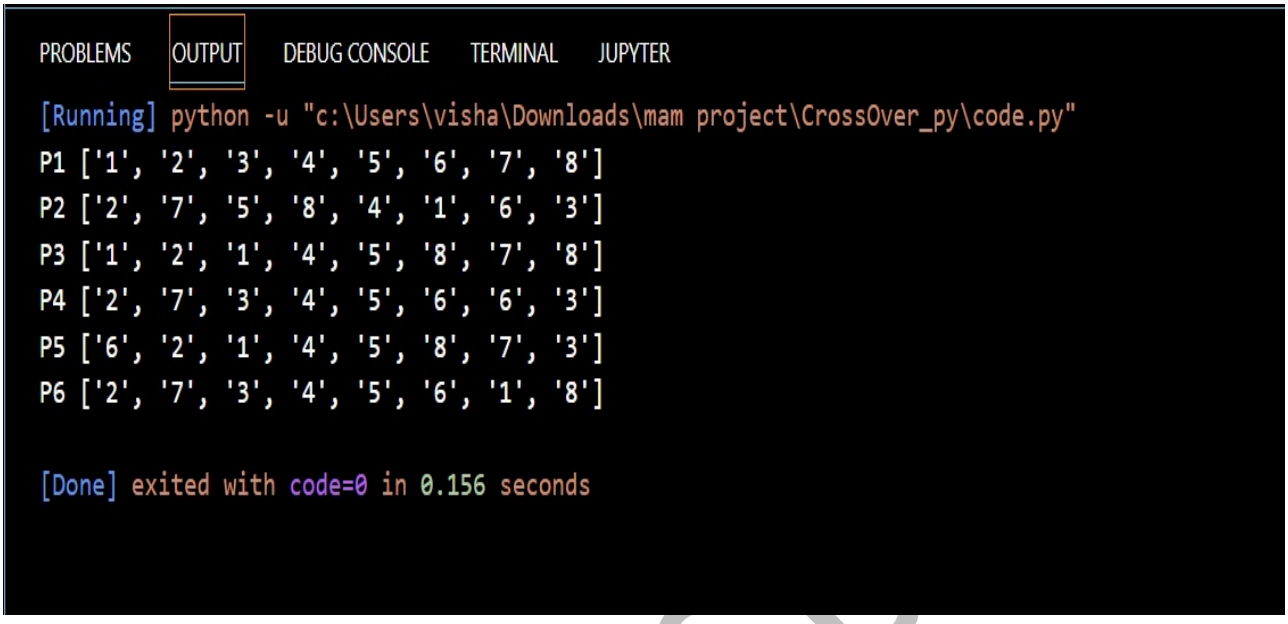

Figure 2. Python coding of proposed crossover operator

PROPOSED MUTATION OPERATOR

GAs employ mutation to help the procedure evade local solutions and give the population newly developed, completely improbable instances. The summary of mutation operators is presented in Table 4. In this section, we propose a novel mutation operator of path representation in GA.

This operator is completely defined by a linear function and is mathematically defined as follows:
Let x_i be the bits in chromosome of n size
$L(x_i) = x_i + 1$ $\forall\, x_i$ where x_i represent bits in chromosome

The python coding of the proposed operator is presented in Figure 3.

Table 4. Summary of Mutation Operator in GA

Mutation Operator	Author
Simple	Holland (1975) [1]
Insertion	Fogel (1988) [40]
Exchange	Banzhaf (1990) [41]
Scramble	Syswerda (1991) [10]
Displacement	Michalewicz (1992) [42]
Inversion	Fogel (1993) [43]

Example: Let us consider $P = (2\ 3\ 5\ 6\ 1\ 4\ 7\ 8)$ be a parent chromosome to emphasize the proposed mutation operator in a better way.

By applying the Linear function of mutation operator, we get a child chromosome as

$C = (3\ 4\ 6\ 7\ 2\ 5\ 7\ 8\ 1)$

```
a=[2,7,5,8,4,1,6,3]
n=int(input("enter limit: "))

print("Parent: ",a)
for i in range(len(a)):
    if a[i] < n:
        a[i]=a[i]+1
    else:
        a[i]=1
print("Children: ",a)
```

Figure 3. Python coding of proposed mutation operator

CONCLUSION

Utilising the survival of the fittest principle, GA a form of evolution method to deal with optimisation issues. They have been effectively utilized in several types of optimization issues, including the TSP. In TSP, our main aim to locate a tour of every node in a weighted network and minimise the overall weight, we must solve the TSP. We have examined various methods of representations of a tour in TSP which are available in literature that could be employed in genetic algorithms attempting to solve the TSP. We also reviewed partially mapped (PM*X*), cycle (CX) and cycle crossover 2 (CX2) available in literature for path representation as it the any form of tour is originally represented as path. In this article, we proposed modified version of GA by introducing an increasing partially mapped crossover operator (IPMX) and a mutation operator to get optimize solution of a TSP. We also, provide python coding of our novel crossover and mutation operator for the better implementation of modified version of GA. GA and its modified versions are applicable in different types of optimization problems other than TSP such as transportation problem, vehicle routing problem, neural networks, and so on.

CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript.

REFERENCES

[1] Holland J., (1975) Adaption in Natural and Artificial Systems. Ann Arbor: *University of Michigen Press*.

[2] Lidd, M.L., (1991) The travelling Salesman Problem Domain Application of a Fundamentally New Approach to Utilizing Genetic Algorithms. Technical Report, MITRE Corporation.

[3] Goldberg, D. E. & Lingle, Jr., R., (1985) Alleles, Loci and the TSP. In Grefenstette, J. J. (ed.) Proceedings of the First International Conference on Genetic Algorithms and Their Applications,Hillsdale, New Jersey: *Lawrence Erlbaum*,154–159.

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

[5] Brady, R. M., (1985b) Optimization Strategies Gleaned from Biological Evolution. *Nature*,317;804-806.

[6] Grefenstette, J. J., (1987b) Incorporating Problem Specific Knowledge into Genetic Algorithms. In Davis, L. (ed.) Genetic Algorithms and Simulated AnnealingLos Altos, CA: *Morgan Kaufmann*,42-60.

[7] Muhlenbein, H., Gorges-Schleuter, M. & Kramer, O., (1988) Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7,65-85.

[8] Muhlenbein, H., (1988)Parallel Genetic Algorithms, Population Genetics and Combinatorial ¨ Optimization. In Schaffer, J. (ed.) Proceedings on the Third International Conference on Genetic Algorithms, Los Altos, CA: *Morgan Kaufmann Publishers*, 416-421.

[9] Syswerda, G., (199b) Schedule optimization using genetic algorithms. In L. Davis (Ed.), Handbook of genetic algorithms Van Nostrand Reinhold.332-349.

[10] Grefenstette, J. J., (1987b) Incorporating Problem Specific Knowledge into Genetic Algorithms. In Davis, L. (ed.) Genetic Algorithms and Simulated Annealing, Los Altos, CA: Morgan Kaufmann.42-60.

[11] Larranaga, P., Inza, I., Kuijpers, C. M. H., Gra ˜ na, M. & Lozano, J. A., (1996c) Algoritmos Geneticos en el Problema del Viajante de Comercio. ´ Informatica y Automatica (submitted).

[12] Grefenstette, J., Gopal, R., Rosmaita, B. & Van Gucht, D., (1986) Genetic Algorithms for the TSP. In Grefenstette, J. J. (ed.) Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Hillsdale, New Jersey: Lawrence Erlbaum.160-165.

[13] Abu Arqub, O., Abo-Hammour, Z., Momani, S., & Shawagfeh, N., (2012) Solving singular two-point boundary value problems using continuous genetic algorithm. *Abstract and applied analysis* Hindawi.

[14] Arqub, O. A., & Abo-Hammour, Z. (2014). Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. *Inf. Sci.*,*279*,396-415.

[15] Abo-Hammour, Z. E., Alsmadi, O., Momani, S., & Abu Arqub, O., (2013) A genetic algorithm approach for prediction of linear dynamical systems. *Math. Probl. Eng*.

[16] Abo-Hammour, Z., Abu Arqub, O., Momani, S., & Shawagfeh, N., (2014) Optimization solution of Troesch's and Bratu's problems of ordinary type using novel continuous genetic algorithm. *Discrete Dyn Nat Soc*.

[17] Raiz, M., Kumar, A., Mishra, V. N., & Rao, N., (2022) Dunkl analogue of Sz $\acute {a} $ sz-Schurer-Beta operators and their approximation behaviour. *Math. Found. Comput.*,*5*(4),315-330.

[18] Mishra, V. N., Raiz, M., & Rao, N. Dunkl analouge of Sz $\acute {a} $ sz Schurer Beta bivariate operators. *Math. Found. Comput.*, 2023;*6*(4), 651-669.

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

[20] Petberg, M. W., & Homg, S., (1980) On the symmetric traveling salesman problems: a computational study. *Math Prog Stud*, *12*; 61-77.

[21] Fleischmann, B., (1985) A cutting plane procedure for the travelling salesman problem on road networks. *Eur. J. Oper. Res.*, *21*(3),307-317.

[22] Padberg, M., & Rinaldi, G. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res.Lett.*,1987;*6*(1); 1-7.

[23] Bhide, S., John, N., & Kabuka, M. R., (1993) A Boolean neural network approach for the traveling salesman problem. *IEEE TransComput.*,*42*(10),1271-1278.

[24] Dorigo, M., & Gambardella, L. M., (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comput*, *1*(1),53-66.

[25] Knox, J. E., (1989) The application of tabu search to the symmetric traveling salesman problem. University of Colorado at Boulder.

[26] Chiang, W. C., & Russell, R. A., (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Ann. Oper. Res.*,*63*,3-27.

[27] Focacci, F., Lodi, A., & Milano, M., (2013) A hybrid exact algorithm for the TSPTW. *INFORMS J Comput*, *14*(4),403-417.

[28] Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., & Yagiura, M., (2007) Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transp.Sci.*, *39*(2);206-232

[29] Nguyen, H. D., Yoshihara, I., Yamamori, K., & Yasunaga, M., (2007) Implementation of an effective hybrid GA for large-scale traveling salesman problems. *IEEE TransSyst, Man, Cybern, Part B (Cybernetics)*, *37*(1), 92-99.

[30] Ghadle, K. P., & Muley, Y. M.,(2015) Travelling salesman problem with MATLAB programming. *International Journal of Advances in Applied Mathematics and Mechanics*, *2*(3),258-266.

[31] Kumar, A., & Gupta, A., (2012) Assignment and travelling salesman problems with coefficients as LR fuzzy parameters. *Int.J. of Appl. Sci.*, *10*(3),155-170.

[32] Majumdar, J., & Bhunia, A. K., (2011) Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times. *J. of Comput. Appl. Math, 235*(9),3063-3078.

[33] Changdar, C., Mahapatra, G. S., & Pal, R. K., (2014) An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. *Swarm Evol. Comput.*,*15*, 27-37.

[34] Maity, S., Roy, A., & Maiti, M., (2012) A modified genetic algorithm for solving uncertain constrained solid travelling salesman problems. *Comput Ind Eng*, *83*;273-296.

[35] Dantzig, G. B., (1963) Linear programming and extensions, princeton, univ. *Press, Princeton, NJ*.

[36] Velednitsky, M., (2018) Short combinatorial proof that the DFJ polytope is contained in the MTZ polytope for the Asymmetric Traveling Salesman Problem. *arXiv preprint arXiv:1805.06997*.

[37] Goldberg, D. E. & Lingle, Jr., R. Alleles, Loci and the TSP. In Grefenstette, J. J. (ed.) (1985) Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Hillsdale, New Jersey: *Lawrence Erlbaum,*154-159.

[38] Oliver, I. M., Smith, D., & Holland, J. R. (2013, August)A study of permutation crossover operators on the traveling salesman problem. *Genetic Algorithms and their Applications*, Psychology Press.224-230.

[39] Hussain, A., Muhammad, Y. S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., & Gani, S., (2017) Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Comput.İntell.Neurosci*.

[40] Fogel, D. B., (1988) An evolutionary approach to the traveling salesman problems. *Biol Cybrn*,60;139-144.

[41] Banzhaf, W. (1990). The 'molecular' traveling salesman. *Biol Cybrn*, 64;7-14.

[42] Michalewicz, Z., (1992) Genetic Algorithms + Data Structures = Evolution Programs. Berlin Heidelberg: *Springer Verlag*.

[43] Fogel, D. B., (1993a) Empirical estimation of the computation required to discovery approximate solutions to the traveling salesman problem using evolutionary programming.Proceedings of 2nd annual conference on evolutionary programming,56-61.