Implementation of Generative Crossover Operator in Genetic Algorithm to Solve Traveling Salesman Problem

Devasenathipathi N. Mudaliar and Nilesh K. Modi

Abstract The research work aims to solve symmetric traveling salesman problem more efficiently. In this research paper, a different crossover operator is proposed, which produces 18 valid offsprings from two parents. The performance of proposed crossover operator is compared with three other existing crossover operators by maintaining the selection technique, mutation technique, and fitness function identical. This crossover operator is tested with data from TSP dataset. The intercity distance table of cities in which distance is measured with L1 norm formed the input to the coded C program that implemented the proposed crossover operator. The same dataset was used to compare the performance of this crossover operator with other three crossover operators. The comparative study indicates that proposed crossover operator performs well compared to other crossover operators in solving traveling salesman problem.

Keywords Symmetric traveling salesman problem · Multiple offspring producing crossover operator · Performance of crossover operator · Intercity distance table · Fitness function

1 Introduction

Traveling salesman problem is an NP hard, combinatorial optimization problem, where a salesman must visit all the cities in his territory exactly once by covering least total distance. The distance between any two given cities (to and fro) is same,

D.N. Mudaliar (✉)
MCA Department, SVIT, Vasad, India
e-mail: devas_mca@yahoo.co.uk

D.N. Mudaliar
R & D Centre, Bharathiar University, Coimbatore, India

N.K. Modi
MCA Department, SVICS, Kadi, India
e-mail: drnileshmodi@yahoo.com

and so the traveling direction is not a hindrance. There exist many problems such as student group formation problem, genome sequencing, and vehicle routing that have traveling salesman problem structure [1–3]. Efficiently solving traveling salesman problem would solve many other problems related to it as no polynomial time algorithm can be formulated for this. Considering a bruteforce approach to solve this problem is infeasible.

Exact algorithms, tour construction, and tour improvement are some of the approaches to solve traveling salesman problem. Linear programming and branch and bound form the types of exact algorithms, while insertion heuristics, closest neighbor heuristics, and greedy heuristics are types of tour construction approach. Finally, tour improvement approach consists of genetic algorithms, ant colonization algorithms, tabu search, etc. However, the above-mentioned approaches and their types have their own set of opportunities and challenges.
Many researchers have applied genetic algorithm to solve the traveling salesman problem or problems having traveling salesman problem structure. Three operators, viz. selection, crossover, and mutation, are used in solving a problem by genetic algorithm. As the first step, some defined number of random feasible solutions (called population) is generated by the genetic algorithm. The selection operator in the genetic algorithm then selects the most fittable solutions (of some defined proportion from the randomly generated population using some fitness function) through various selection techniques such as tournament selection and roulette wheel selection. The filtered (or selected) solutions are now paired to produce new breed of chromosomes. A pair of parent solutions cross over to produce another set of offspring solutions. This process called crossover has various techniques to achieve the task. In case of traveling salesman problem, famous crossover techniques called partially mapped crossover, order crossover, cycle crossover, etc., exist. To bring variation in the offspring population (so that the solutions do not get trapped in the local minima), mutation operator is executed, which randomly changes a gene or two of offspring solutions. The above process from selection to mutation is repeated till a definite number of times or for the time indicative improvement occurs in the population set [4].

In this research work, the authors have tried to present a different crossover operator in genetic algorithm that is able to solve traveling salesman problem. The significance of this crossover operator is that it takes two valid parent solutions and produces 18 valid offspring solutions. Even though published research work exists for crossover work in genetic algorithm that produces more than two offspring solutions, they did not focus to solve the traveling salesman problem. In addition to this, it has to be brought to notice that traditional crossover operators cannot be applied to solve traveling salesman problem as they may end up with invalid solutions. The next section of the paper represents the work done by different researchers in connection with solving traveling salesman problem or its variants through different approaches. The third section details on the actual implementation of the research work. The fourth section elucidates on the results obtained and comparison of the results. The last section concludes the research paper with achievable future directions.

2 Proposed Methodology

Most of the two-point crossover operator in genetic algorithm takes in two parent chromosomes as input and produces two valid offspring chromosomes as output. Additionally, the two offspring chromosomes contain the features of both the parents. The authors propose a two-point crossover approach that produces 18 valid offspring chromosomes and all the offspring chromosomes contain the features of both the parents.

2.1 Example of M-Crossover Operator

We try to solve the Traveling Salesman Problem with 9 cities using two-point crossover operator.
First cut point—after third gene (even though a different value less than second cut point can be set).

Second cut point—after sixth gene

Parent 1—1 2 3 4 5 6 7 8 9
Parent 2—9 1 2 8 7 4 5 6 3

Using the given cut points, the parent chromosomes can be cut into three parts, viz.

Parent 1—[1 2 3] [4 5 6] [7 8 9]
Parent 2—[9 1 2] [8 7 4] [5 6 3]

2.2 Creating the First Offspring Chromosome

Inserting the first part of Parent 2 before the first part of Parent 1, we get

[9 1 2] [1 2 3] [4 5 6] [7 8 9]

Striking the matching chromosomes of Parent 2 part from Parent 1 parts, we get

[9 1 2] [~~1~~ ~~2~~ 3] [1 2 3] [4 5 6] [7 8 ~~9~~]

Deleting the striked genes and grouping the rest of the genes according to the cut points, we get

[9 1 2] [3 4 5] [6 7 8]

By simply removing the partition, we get the first valid offspring—9 1 2 3 4 5 6 7 8.

2.3 Creating the Second Offspring Chromosome

Inserting the first part of Parent 2 after the first part of Parent 1 and before the second part of Parent 1, we get

[1 2 3] [9 1 2] [4 5 6] [7 8 9]

Striking the matching chromosomes of Parent 2 part from Parent 1 parts, we get

[~~1~~ ~~2~~ 3] [9 1 2] [4 5 6] [7 8 ~~9~~]

Deleting the striked genes and grouping the rest of the genes according to the cut points, we get

[3 9 1] [2 4 5] [6 7 8]

By simply removing the partition, we get the second valid offspring—3 9 1 2 4 5 6 7 8.

2.4 Creating Other Offspring Chromosomes

In the above manner, we create the remaining 7 more chromosomes with the help of our proposed crossover. The change that is to be followed in getting the remaining chromosomes is identifying which part of Parent 2 chromosome is to be put before which part of Parent 1 chromosome. This is illustrated with the help of Fig. 1.

Figure 1 represents that the first part of Parent 2 be inserted before the first part of Parent 1 and we get the first chromosomes after following the above steps. Following the above steps, the remaining 7 offspring chromosomes are as follows: 345 691 278, 874 123 569, 123 874 569, 123 568 749, 563 124 789, 125 634 789, and 124 563 789.

To obtain the next set of 9 offspring chromosomes, just interchange the contents of Parent 1 and Parent 2 and perform the above-mentioned steps. The following offspring chromosomes will be obtained by performing this step:

123 987 456, 912 387 456, 987 412 356, 456 912 873, 912 456 873, 912 874 563, 789 124 536, 127 894 563, and 124 789 563.

Once all 18 valid offspring chromosomes are obtained, two best chromosomes with respect to fitness value are selected and sent for further stage and the remaining chromosomes are simply ignored.

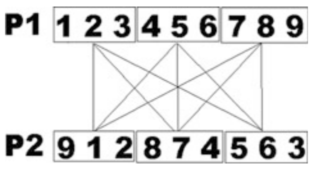

Fig. 1 Part of Parent 2 (P2) to be kept before part of Parent 1 (P1)

3 Actual Experiment

The proposed crossover operator was tested with test data from TSPLIB dataset. Four different C programs were developed to implement and compare the proposed crossover operator in genetic algorithm. All contents of the C programs were same except the crossover operator implementation part. We coded the following functions in C language to accomplish the above tasks.

1. Random initialization of population
2. Selection of chromosomes
3. Cloning of chromosomes
4. Crossover of chromosomes
5. Mutation of chromosomes.

The following steps describe C program coded to implement the experiment.

1. Initialize a random population of 100 valid chromosomes and perform steps 2–8 for 50 iterations.
2. Obtain the fitness value of each chromosome of population by the fitness function.
3. Select the best 50 % of chromosomes with to respect fitness value.
4. Clone the selected chromosomes by merely creating a copy of those chromosomes and add them to the population.
5. Randomly create pairs of chromosomes and send them for crossover.
6. The crossover results in two new valid offspring chromosomes for every pair of chromosomes sent for crossover.
7. Send 2 % of the entire newly obtained offspring chromosome for mutation (in our case, displacement mutation).
8. Obtain the fitness value of each chromosome and return to step 2.

The developed C programs were tested for test dataset (fri26_d.txt) which provides the intercity distance table for 26 cities. The TSP test data in intercity distance table of the dataset obtained from TSPLIB formed the input to the C programs [5]. The different crossover techniques were proposed: crossover (discussed in methodology section), order crossover, partially mapped crossover, and cycle crossover. The output and results of the experimental work are discussed in the next section.

4 Results and Discussion

As stated in the previous section, we have tried to evaluate the efficiency of our proposed crossover operator by comparing the results of the experiment with the existing crossover operators. Roulette wheel selection technique was used to select the chromosomes. Crossover rate was set to 0.9, and mutation rate was set to 0.02. The initial population was set to 100 valid chromosomes.

Table 1 Comparison of performance of proposed crossover with other crossover operators for 26-city traveling salesman problem

Number of iterations (generations)	Best fitness value obtained by proposed crossover C program	Best fitness value obtained by partially mapped crossover C program	Best fitness value obtained by order crossover C program	Best fitness value obtained by cycle crossover C program
0–10	1,261	1,857	1,724	1,802
11–20	1,144	1,674	1,665	1,721
21–30	1,054	1,606	1,602	1,871
31–40	937	1,616	1,567	1,849
41–50	1,051	1,541	1,549	1,790

Table 1 represents the best fitness value obtained up to a given number of iterations (generations) for 26-city problem (fri26_d.txt). It could be noted that output given by the C program of our proposed crossover approach gives best results quickly. The shortest distance for the 26-city problem (for this test data) obtained till now by researchers is 937, and within 50 iterations (generations), our proposed crossover approach is better close to the optimal solution compared to the other crossover techniques.

The obtained optimal path by m-crossover operator for 26-city problem (FRI26) is as follows: 8-7-5-6-4-2-3-14-15-12-13-11-10-16-9-19-20-18-17-21-26-22-24-23-25-1. The length of this path is 937 km according to the values given in the dataset.

5 Conclusion and Future Work

In this research paper, the authors have tried to propose a new crossover operator of genetic algorithm to solve symmetric traveling salesman problem. Alternately, the performance of the proposed work is compared with other existing crossover operators that aid to solve symmetric traveling salesman problem. C programs were coded to implement and compare the performance of proposed crossover operator and other three crossover operators. A test data from TSPLIB (fri26_d.txt 26-city problem) were considered for the same. The results of the experiment positively proved our proposed crossover approach to solve symmetric traveling salesman problem.

As part of the future work, we plan to implement the proposed crossover operator for more number of cities for symmetric traveling salesman problem. In addition to this, the same crossover approach can be used to solve asymmetric traveling salesman problem as well.

References

1. R. Agarwala, D.L. Applegate, D. Maglott, G.D. Schuler, A.A. Schäffer, A fast and scalable radiation hybrid map construction and integration strategy. Genome Res. **10**, 350–364 (2000)
2. D.N. Mudaliar, N.K. Modi. Evolutionary algorithm approach to pupils' pedantic accomplishment, in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, Advances in intelligent systems and computing, vol. 199 (Springer, Berlin, 2013), pp. 415–423
3. R. Matai, S.P. Singh, M.L. Mittal, in *Traveling salesman problem: an overview of applications, formulations, and solution approaches*, Traveling Salesman Problem, Theory and Applications (InTech, Croatia, 2010), pp. 1–24
4. N. Bansal, A. Blum, S. Chawla, A. Meyerson, Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time-Windows, in *Proceedings of ACM STOC* (2004), pp. 166–174
5. M. Ünal, Ak. Ayça, V. Topuz, H. Erdal, Genetic algorithm optimization of PID controllers using ant colony and genetic algorithm, Studies in computational intelligence. vol. 449 (Springer Berlin, 2013), pp. 19–29