Journal of Engineering and Applied Sciences 12 (Special Issue 2): 6143-6147, 2017
ISSN: 1816-949X

Convergence Analysis of the Ternary Crossover Operator

Apoorva Mishra, Pranav Anand and Anupam Shukla
Soft Computing and Expert System Laboratory, ABV-IIITM, Gwalior, Madhya Pradesh, India

Abstract: Genetic algorithms are inspired by the process of natural evolution and are one of the most widely used optimization algorithms. This study deals with analyzing the convergence behavior of a new variant of the genetic algorithms involving a modified version of the ternary crossover operator. To compare the performance of the proposed variant of genetic algorithms with that of the traditional genetic algorithms both of these are applied to two benchmark datasets of the Travelling Salesman Problem (TSP) taken from TSPLIB. The convergence behavior of both these algorithms is analyzed for 100 iterations with tournament size as 30 and 50. The results indicate that the proposed variant of genetic algorithm involving a modified ternary crossover operator converges to a better solution as compared to that of the traditional genetic algorithms.

Key words: Genetic algorithms, ternary crossover, mutation, convergence, solution, evolution

INTRODUCTION

Evolutionary algorithms are widely used for solving the complex optimization problems (Jong, 2006). The problems which belong to this category are computationally too expensive and have exponential time complexity. For these problems, a method which can give a near-optimal solution is also considered as good. For these types of problem, an algorithm which has randomness in its approach and can consider all the possible combinations is preferable as it considers all the possibilities to achieve the optimal solution in a time-efficient manner. One such type of randomized algorithms is the evolutionary algorithm. Genetic algorithm belongs to the class of evolutionary algorithms (Shukla *et al*., 2010). Genetic algorithms have been applied to solve many problems (Abdelaziz, 2017; Djeffal and Bendib, 2011; Datta *et al*., 2015; Suksonghong *et al*., 2014; Graziela *et al*., 2017).

A genetic algorithm is a meta-heuristic algorithm based on natural selection of the fittest solution. The main use of genetic algorithm is for solving the complex search problems using operators like selection, mutation and crossover. Many variants of these operators have also been proposed (Mishra and Shukla, 2016; Zhu *et al*., 2016; Qiongbing and Lixin, 2016; Banerjee, 2013).

The first step involved in the working of a genetic algorithm is to choose a random population of the individuals and then the possible candidate solutions are evolved using natural process of selection, crossover and mutation to achieve better results.

Selection is a process through which the existing population is allowed to breed to give rise to new generation. Through this process, the solution which is more fit than others in terms of the fitness function is typically more likely to be selected. Different techniques are used for the purpose of selection like round robin, tournament, roulette wheel selection, etc. Each selection scheme has some different algorithms for selecting the individuals but each one of them have the intention to select the more fit individuals present at that stage. After selection, comes the crossover part where parents breed to produce the next generation. This new generation shares quite a few properties with their parents. Some of the major types of crossover are: single-point crossover, multi-point crossover, scattered crossover, heuristic crossover ordered crossover, etc.

Some of the recently proposed crossover operators are: '3P-3C' version of ternary crossover proposed by Mishra and Shukla (2016), adaptive hybrid crossover by Zhu *et al*. (2016), crossover for genetic algorithms having variable length chromosomes by Qiongbing and Lixin (2016) and probabilistically-guided context-sensitive crossover operator by Banerjee (2013). After crossover, the next operation is mutation. It is a process where randomness is introduced in the population by changing the bit representation of the solution; it may or may not provide a better solution. Last step in the process of genetic algorithms is the termination of the process of continuous evolution. Few examples of termination conditions are fixed number of iterations, time limit, fitness function, etc. The working of a traditional genetic algorithm is represented by Fig. 1.

In this study, we propose a modified version of genetic algorithms (by replacing the method of binary crossover with the method of ternary crossover) for solving the Traveling Salesman Problem (TSP). TSP is a NP hard problem as its running time complexity is

Corresponding Author: Apoorva Mishra, Soft Computing and Expert System Laboratory, ABV-IIITM, Gwalior, Madhya Pradesh, India

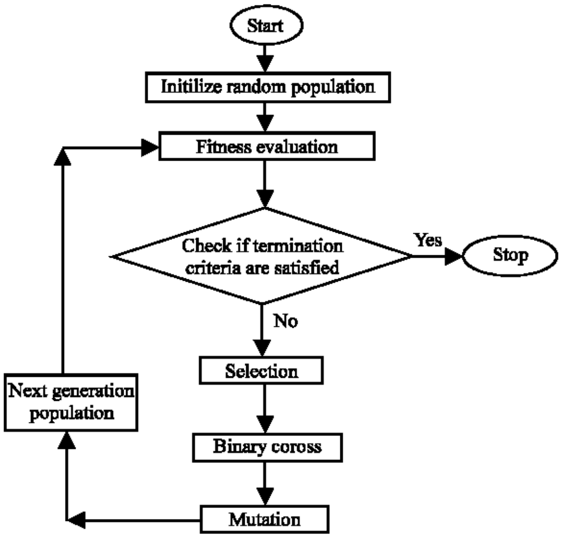

Fig. 1: Working of traditional genetic algorithms

exponential. This problem is a very old problem and many algorithms including genetic algorithms have been used in the past to solve it (Wang *et al*., 2016; Yuan *et al*., 2013; Liu and Zeng, 2009; Nguyen and Yoshihara, 2007). The novel crossover operator introduced in this study involves three parents that participate in the crossover operation to produce one offspring as an outcome of the operation.

MATERIALS AND METHODS

Proposed algorithm:

Ternary crossover operator (3P-1C): "Ternary crossover is an approach where three parents are combined together to produce one offspring".

Let us take an example where there are three parents P1 (111001101), P2 (110111010), P3 (000111011) and one offspring O1 is generated from them. Let the points of crossover be '3' and '6'. The total number of permutations in which the three parents could combine to produce the offspring is 3 = 6. The offspring contains some parts of all the three parents. The formation of offspring for all these six scenarios is represented below to explain the working of the ternary crossover operator. (P1-P2-P3):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	1	1	1	1	1	1	0	1	1

(P1-P3-P2):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	1	1	1	1	1	1	0	1	0

(P2-P1-P3):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	1	1	0	0	0	1	0	1	1

(P2-P3-P1):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	1	1	0	1	1	1	0	1	1

(P3-P1-P2):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	0	0	0	0	0	1	0	1	0

(P3-P2-P1):

		8	7	6	5	4	3	2	1	0
P1	=	1	1	1	0	0	1	1	0	1
P2	=	1	1	0	1	1	1	0	1	0
P3	=	0	0	0	1	1	1	0	1	1
O1	=	0	0	0	1	1	1	1	0	1

In all these scenarios (P1-P2-P3), (P1-P3-P2) and (P3-P2-P1) represents the order in which the parents combine to form the offspring.

Working of the proposed genetic algorithm with ternary crossover: The working of the proposed genetic algorithm with ternary crossover operator is similar to the working of traditional genetic algorithms, except the fact that instead of having a binary crossover in the crossover phase, ternary crossover operator is used. The working of Genetic algorithm with ternary crossover operator is represented by Fig. 2.

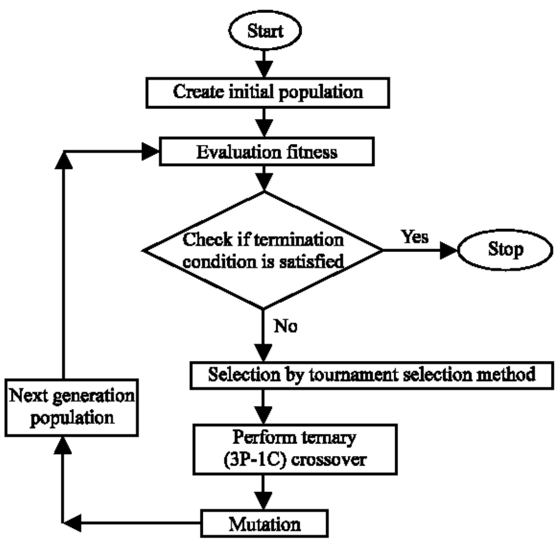

Fig. 2: Working of genetic algorithms with ternary crossover

RESULTS AND DISCUSSION

Description of the dataset and simulation environment: Simulation of the proposed algorithm is done on two data sets "ATT48" and "BERLIN52" which are taken from TSPLIB. The code of the genetic algorithm is written in java programing language on Net beans IDE with the mac-ox operating system with 2.7 GHz Intel Core i7. The first dataset "ATT48" is a set of 48 cities (US state capitals) taken from TSPLIB. The minimal tour has length 10628 and the other one is a set of 52 locations in Berlin taken from TSPLIB. The count of iteration is 100.

Simulation results (comparison of the ternary crossover and traditional binary crossover): In this study, we compare the performance of the proposed new version of the ternary crossover operator (3P-1C) and the traditional binary crossover operator. The results obtained by applying genetic algorithm having ternary crossover and traditional genetic algorithm to the two data sets "ATT48" and "BERLIN52" for two different values of tournament size (30 and 50) are depicted by Fig. 3-6.

Graph represented by Fig. 3 is for ATT48 dataset. In this graph, tournament size is taken as 30. Graph represented by Fig. 4 is also for ATT48 dataset. In this graph tournament size is taken as 50.

Here, it can be observed from the graphs represented by Fig. 3 and 4 that the convergence achieved by applying ternary crossover operator is better as compared to the traditional binary crossover operator and the convergence rate for ternary crossover operator with tournament size of 50 is better than the convergence rate for ternary crossover operator with tournament size of 30. So, if the space of selection is increased better results are

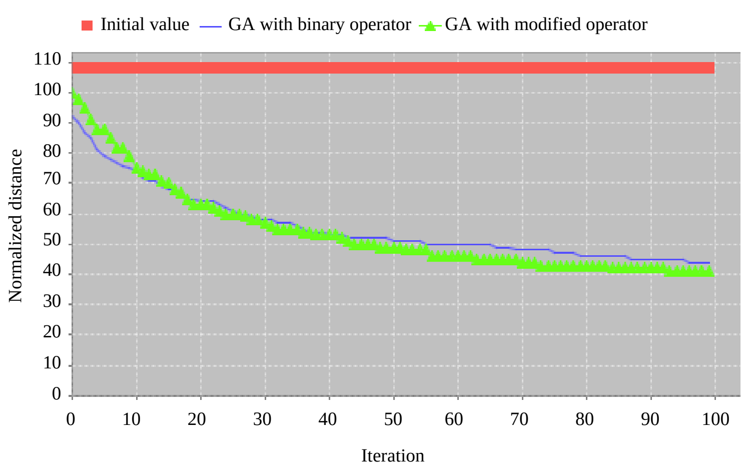

Fig. 3: Comparative analysis of the convergence behaviors of genetic algorithm with ternary crossover and traditional genetic algorithm for ATT48 dataset when tournament size = 30; ATT 48.txt 3-1C 30

Fig. 4: Comparative analysis of the convergence behaviors of genetic algorithm with ternary crossover and traditional genetic algorithm for ATT48 dataset when tournament size = 50; ATT 48.txt 3-1C 50

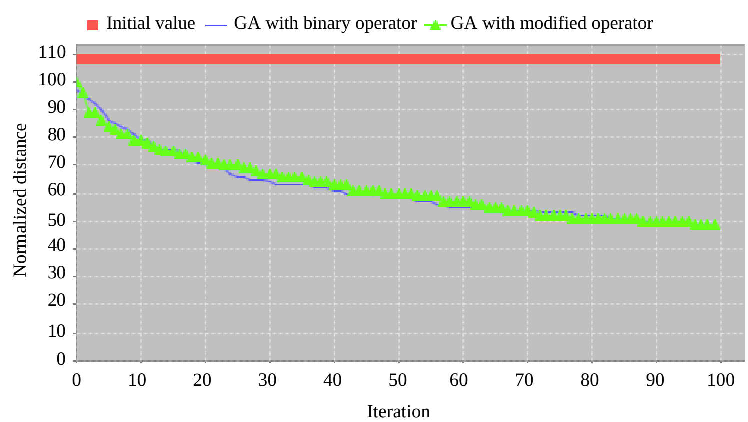

Fig. 5: Comparative analysis of the convergence behaviors of genetic algorithm with ternary crossover and traditional genetic algorithm for BERLIN52 dataset when tournament size = 30; ATT 48.txt 3-1C 30

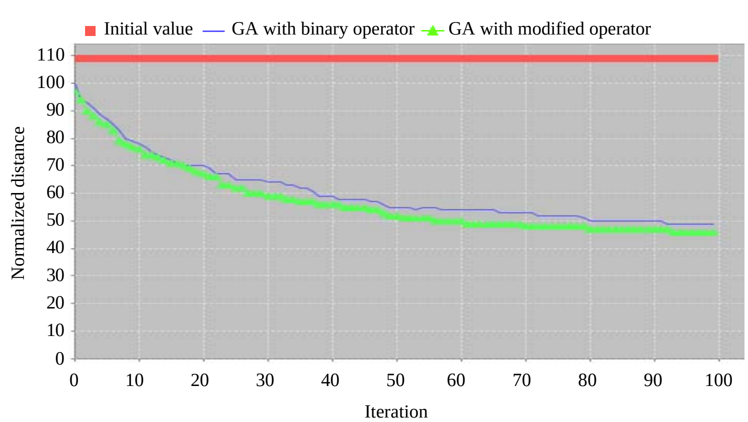

Fig. 6: Comparative analysis of the convergence behaviors of genetic algorithm with ternary crossover and traditional genetic algorithm for BERLIN52 dataset when tournament size = 50; ATT 48.txt 3-1C 50

obtained. The graph represented by Fig. 5 is for BERLIN52 dataset. In this graph, the tournament size is taken as 30.

Graph represented by Fig. 6 is also for BERLIN52 dataset. In this graph tournament size is taken as 50. In Fig. 3-6, X-axis represents the number of iterations and the Y-axis represents the normalized distance (distances are very large so they are normalized).

It can be observed from the graphs represented by Fig. 5 and 6 that the convergence achieved by applying ternary operator is better as compared to the traditional binary operator and the convergence rate for ternary crossover operator with tournament size of 50 is better than the convergence rate for ternary crossover operator with tournament size of 30. So, if the space of selection is increased better results are obtained.

From the graphs represented by Fig. 3-6, it is evident that the convergence achieved by applying ternary crossover operator is better as compared to that for the traditional binary operator in all the four scenarios.

CONCLUSION

In this study, a new variant of the genetic algorithms involving modified version of the ternary crossover (involving three parents and one offspring) operator has been proposed. The convergence behavior of this new variant has been analyzed by applying it to two different benchmark data sets of TSP taken from TSPLIB. It is evident that as the number of iterations increases, the proposed variant of the genetic algorithms starts converging towards the near-optimal solution. It can also be concluded from these figures that the proposed variant of the genetic algorithms converges to a comparatively better solution than the traditional genetic algorithms.

REFERENCES

Abdelaziz, M., 2017. Distribution network reconfiguration using a genetic algorithm with varying population size. Electr. Power Syst. Res., 142: 9-11.

Banerjee, A., 2013. A novel probabilistically-guided context-sensitive crossover operator for clustering. Swarm Evol. Comput., 13: 47-62.

Datta, R., S. Pradhan and B. Bhattacharya, 2015. Analysis and design optimization of a robotic gripper using multiobjective genetic algorithm. IEEE. Trans. Syst. Man Cybern. Syst., 46: 1-11.

Djeffal, F. and T. Bendib, 2011. Multi-objective genetic algorithms based approach to optimize the electrical performances of the Gate Stack Double Gate (GSDG) MOSFET. Microelectron. J., 42: 661-666.

Graziela, F., N. Barros, W. Alex, U. Holanda and M. Vinicius *et al*., 2017. Compression of electrical power signals from waveform records using genetic algorithm and artificial neural network. Electr. Power Syst. Res., 142: 207-214.

Jong, K.A.D., 2006. Evolutionary Computation: A Unified Approach. MIT Press, Cambridge, Massachusetts, ISBN:0-262-04194-4, Pages: 110.

Liu, F. and G. Zeng, 2009. Study of genetic algorithm with reinforcement learning to solve the TSP. Expert Syst. Applic., 36: 6995-7001.

Mishra, A. and A. Shukla, 2016. Mathematical analysis of the cumulative effect of novel ternary crossover operator and mutation on probability of survival of a schema. Theor. Comput. Sci., 1: 1-11.

Nguyen, H.D. and I. Yoshihara, 2007. Implementation of an effective hybrid GA for large-scale traveling salesman problems. IEEE. Trans. Syst. Man Cybern. Part B. Cybern., 37: 92-99.

Qiongbing, Z. and D. Lixin, 2016. A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. Expert Syst. Appl., 60: 183-189.

Shukla, A., R. Tiwari and R. Kala, 2010. Towards Hybrid and Adaptive Computing: A Perspective. Springer, Germany.

Suksonghong, K., K. Boonlong and K. Goh, 2014. Electrical power and energy systems multi-objective genetic algorithms for solving portfolio optimization problems in the electricity market. Intl. J. Electr. Power Energy Syst., 58: 150-159.

Wang, J., O.K. Ersoy, M. He and F. Wang, 2016. Multi-offspring genetic algorithm and its application to the traveling salesman problem. Appl. Soft Comput. J., 43: 415-423.

Yuan, S., B. Skinner, S. Huang and D. Liu, 2013. Discrete optimization a new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. Eur. J. Oper. Res., 228: 72-82.

Zhu, Q., Q. Lin, Z. Du, Z. Liang and W. Wang *et al*., 2016. A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm. Inf. Sci., 345: 177-198.