2009 International Conference of Soft Computing and Pattern Recognition

A New Approach to Solve Traveling Salesman Problem Using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator

Gohar Vahdati[1], Mehdi Yaghoubi[4]
Computer Department
Islamic Azad University
Mashhad Branch, Iran
[1]Elham62.vahdati@gmail.com
[4]yaghobi@mshdiau.ac.ir

Mahdieh Poostchi[2]
Computer Department
Iran University of Science and Technology (IUST)
Tehran, Iran
[2]ma_poostchi@comp.iust.ac.ir

M. B. Naghibi S.[3]
Electrical Department
Ferdowsi University of Mashhad
Mashhad, Iran
[3]mb_naghibi@um.ac.ir

***Abstract*— This paper proposes a new solution for Traveling Salesman Problem (TSP), using genetic algorithm. A heuristic crossover and mutation operation have been proposed to prevent premature convergence. Presented operations try not only to solve this challenge by means of a heuristic function but also considerably accelerate the speed of convergence by reducing excessively the number of generations. By considering TSP's evaluation function, as a traveled route among all *n* cities, the probability of crossover and mutation have been adaptively and nonlinearly tuned. Experimental results demonstrate that proposed algorithm due to the heuristic performance is not easily getting stuck in local optima and has a reasonable convergent speed to reach the global optimal solution. Besides, implementation of the algorithm does not have any complexities.**

Keywords- Genetic Algorithm; Fitness Function; Heuristic Crossover; Mutation; Traveling Salesman Problem

I. INTRODUCTION

Traveling Salesman Problem (TSP) is one of the most significant optimization problems. TSP, as a general NP-complete problem can be developed to be an admissible solution for any other problems that belongs to NP-complete class. Several heuristic algorithms, such as genetic algorithm (GA) [1-6], Tabu search [7-9], neural network [10-12] and ant colony [13-14], have been suggested to solve this problem. Among all of these soft computing techniques, genetic algorithms, due to a good performance in finding a solution near the optimal one and requiring small computational time, have been much more mentioned. A GA population has a group of individuals that each of them presents a solution. Individuals can obtain higher fitness value through the selection strategy, crossover and mutation operations, and seeking for the best solution. One of the common challenges of a standard genetic algorithm is to prevent premature convergence. This emerges due to a sudden and fast reduction of search space. To avoid this problem, to generate and to reinforce optimal chromosomes, heuristic crossover and mutation operations have been suggested. Probability of these operations have been tuned adaptively and nonlinearly to avoid premature and slow convergence simultaneously, as well as low stability. Proposed method will not easily get stuck in local optima and has a reasonable convergent speed to reach global optimal solutions. Standard problems Kora100, ST70, Eil76 and Eil51 have been used to prove the performance of suggested algorithm.

Next, in section II, TSP will be reviewed briefly. Section III, includes basic idea and describes the proposed algorithm process. Finally, section IV, contains the experimental results of the algorithm and its comparison to the other mentioned methods.

II. TRAVELING SALESMAN PROBLEM

Traveling Salesman Problem (TSP) is one of the most significant optimization problems. If $\{1,2,\ldots,n\}$ be the labels of n cities, the traveling salesman must explore a path to visit each city just once and already costs the minimum total distance. So, the objective is to find the shortest path among n cities. The cost function for this problem is as follows:

$$Cost = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1} \tag{1}$$

where $C_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, is the cost of traveling from city i to city j.

III. BASIC IDEA AND THE ALGORITHM PROCESS

A. Proposed GA for Solving TSP

1) Chromosome Design

First, GA requires a chromosome representation for genetic information. Classic types of GA utilize binary string to represent chromosomes, whereas it is not applicable for problems, such as TSP. The reason is that there is no direct and effective solution for mapping possible solutions to binary strings. Here chromosomes represent the cities' permutations that salesman must pass through. Consequently, the integers will represent genes, and their sequence is the proposed order of the cities where the salesman must visit first. Figure 1 shows a sample chromosome for a problem with 8 cities:

1	**5**	**8**	**2**	**3**	**4**	**7**

Figure 1. Chromosome representation

978-0-7695-3879-2/09 $26.00 © 2009 IEEE
DOI 10.1109/SoCPaR.2009.33

B. Proposed Heuristic crossover operation

So far, so many common crossover operators, such as PMX, CX and OX, have been introduced. But none of them consider the relation between edges in TSP. So they may not accelerate the speed of the algorithm. Here, a heuristic crossover operator, which can increase the algorithm speed, has been proposed. Although it must be mentioned that due to the goal function of TSP which is finding the shortest total distance in one closed cycle, parent chromosomes will be represented in a closed cycle, too. For example, if we consider the information of two parents A and B, as illustrated in figure 2, their correct representation will be as what is shown in figure 3 and 4.

PARENT_A	7	1	2	8	6	3	5	4
PARENT_B	3	1	5	6	2	7	4	8

Figure 2. Father chromosomes representations

The generation process of child chromosomes will be as follows:

1) First, one city will be selected randomly as a start point (This city is called c).
2) Four pointers will be put at the position *c* of two parents (one pair for each chromosome, A and B, where one rotates clockwise and the other counterclockwise). Figure 3 and 4 show a sample.
3) City c, as the first gene will be inserted into the child chromosome A, and then the both pointers, each in its own direction, will go forward from c to the next city. Then, the value of position c in both parent chromosomes A and B will be replaced by zero.
4) Now to determine the kth-gene of child chromosome, a heuristic function which is defined as the inverse distance between the cities, will be used to calculate the evaluation value of each pointer. Evaluation value μ_{ij} is as follows: $$\mu_{ij} = \frac{1}{dist(c_i, c_j)} \quad (2)$$ where c_i is the city that sits in k-1th gene of child chromosome and c_j is the city that the pointer points to.
5) Only the pointer that gets higher evaluation value will go to one city forward in its direction. If both pointers have the same evaluation value, one will be chosen randomly and goes forward to the next city. City that the pointer comes from, will be inserted into the child chromosome A, and then it will be replaced by zero in both parent chromosomes A and B.

Step 4 to 5 will be repeated until all genes in parent chromosomes A and B become zero. Each visited city will be replaced by zero; therefore, there is no chance for a city to be selected twice and the positions with the zero value will be just bypassed.

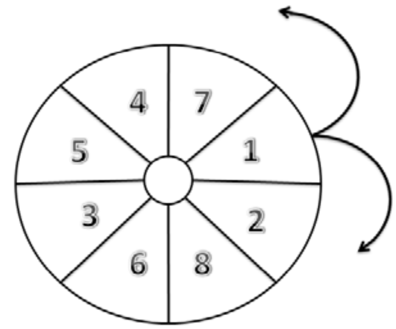

Figure 3. First father chromosomes A

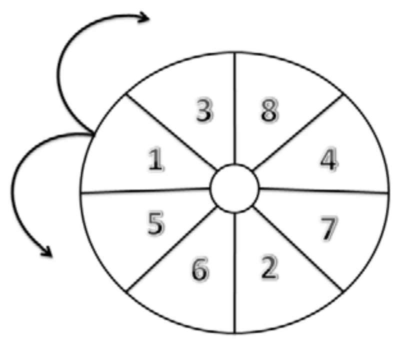

Figure 4. Second father chromosomes B

To generate the child chromosome B, the same process will be done by choosing once more a random start point. To explicit the method, a problem with 8 cities is considered. Distance matrix is included in table I.

TABLE I. DISTANCE MATRIX BETWEEN CITIES

City	1	2	3	4	5	6	7	8
1	0	14	68	75	25	46	55	80
2	25	0	45	87	37	70	95	20
3	71	45	0	50	65	82	63	74
4	75	65	47	0	27	43	57	68
5	35	24	65	27	0	56	44	71
6	23	70	82	48	56	0	25	90
7	17	95	63	57	44	25	0	34
8	54	20	74	68	71	90	71	0

When we start the generation process of child_A, where city 1 is selected as city c, child chromosome A will be generated by proposed method as is illustrated in figure 5:

CHILD_A	3	5	4	6	7	8	2	1

Figure 5. child chromosome representation

C. The Proposed Mutation Operation

There are various common mutation operators as inverse mutation operators. But these operators cannot accelerate the convergent speed of the algorithm. The proposed heuristic mutation operator will consider the relation between cities in TSP, accelerates its convergence and act as follow:

1) First, it will choose a city randomly and called it c.
2) Second, the closest city to c will be found.
3) Third, the order of cities between these two cities containing c and its closest city, in clockwise direction, will be inversed.

D. Nonlinear and Adaptive Tuning of Crossover and Mutation Probability

In this paper, probabilities of crossover and mutation operations are tuned adaptively and nonlinearly. Tuning curves of crossover and mutation will change slowly with D_{avg}. Therefore, crossover and mutation probabilities of chromosomes that their evaluation values are close enough to the average evaluation value of population will be increased. Equation 4 and 5 describe consequently the crossover and mutation probabilities.

$$P_c = \begin{cases} p_{c1} - \dfrac{p_{c1} - p_{c2}}{1 + \exp\left(a\left(\dfrac{D_{avg} - D'}{D_{avg} - D_{\min}}\right)\right)} & D' \le D_{avg} \\ p_{c1} & D' > D_{avg} \end{cases} \tag{4}$$

$$P_m = \begin{cases} p_{m1} - \dfrac{p_{m1} - p_{m2}}{1 + \exp\left(a\left(\dfrac{D_{avg} - D}{D_{avg} - D_{\min}}\right)\right)} & D \le D_{avg} \\ p_{m1} & D > D_{avg} \end{cases} \tag{5}$$

Where, D_{min} is the highest evaluation value of the population (the shortest path) and D_{avg} is the average evaluation value (the average distance). D' is the higher evaluation (shorter distance) between two chromosomes in crossover operation. And finally D is the chromosome's evaluation (distance) in mutation operation. When the distances of most chromosomes tend to the average one, probability of crossover and mutation will be increased.

It is also obvious that the rate of probability's ascending in adaptive and nonlinear GA is more than linear adaptive GA and other algorithms [15]. A general pseudo-code of proposed GA is shown in figure 6.

```
begin GA
  create initial population
  while generation_count < k do
     /* k = max.number of generation. */
     begin
        Selection and Elitism
        Heuristic Crossover ← Nonlinear adjusting Crossover Probability
        Heuristic Mutation ← Nonlinear adjusting Mutation Probability
        Increment generation_count
     end
   Output the best individual found
end GA
```

Figure 6. Pseudo code of the proposed GA

IV. EXPERIMENTAL RESULTS

To demonstrate the performance of the proposed algorithm, standard problems as Kora100, Eil51, St70 and Eil76 [16] are chosen. Here, there are some variables which must be first initialized. Table II and III contain parameters value required for the computations. These are essential to calculate P_c and P_m of equation 4 and 5.

TABLE II. PARAMETERS VALUE OF THE PROPOSED METHOD

Population Size	Generation	*Pc1*	*Pc2*	*Pm1*	*Pm2*	*a*
100	100	0.9	0.7	0.1	0.05	40

TABLE III. INITIALIZE PARAMETERS FOR OX_SIM[1], MOC_SIM[1],SWAP_GATSP[5]

Population Size	Crossover probability (*Pc*)	Mutation probability (*Pm*)
100	0.6	0.02

As table II and III include the number of generation and population size has been considered to be 100. Parameters of nonlinear and adaptive probabilities of crossover and mutation as in sequence *Pc*, *Pm* and also factor *a* in equations 4 and 5 for computations of proposed method, table II, and compared methods, table III are presented.

Table IV contains the results of the proposed method that have been compared with the results of IGA[4], MMGA[3], SOM[10], ACO[13] and IWD[17] for some standard problems such as Eil51, St70, Eil76 and Kora100. Contents of table IV illustrate the shortest path length for each method and the special problem. It should also be mentioned that IGA and MMGA have not converged to the global optima. Table V compares the results of proposed algorithm and an algorithm based on ant colony. This table's contents have also shown the shortest path length, and values in parentheses are the required number of generations in order to reach the global optima.

TABLE IV. AVERAGE RESULTS OF SOM[10],IGA[4],MMGA[3] AND THE PROPOSED GA FOR TSP

PROBLEM	OPTIMAL	IGA	MMGA	SOM	PROPOSED ALGORITHM
eil51	426	499	446	432	431
st70	675	-	-	683	685
eil76	538	611	568	556	552
Kora100	21282	24921	22154	-	21353

TABLE V. COMPARISON RESULTS OF PROPOSED GA WITH ACO [13]

PROBLEM	OPTIMAL	ACO[13]	PROPOSED ALGORITHM
eil51	426	427(90)	428(27)
St70	675	676(450)	679(30)

TABLE VI. COMPARISON RESULTS OF PROPOSED GA WITH IWD [17]

problem	optimal	IWD	Proposed Algorithm
eil51	426	471(50)	428(27)
eil76	538	559(300)	548(43)

Table VI also includes the results of proposed algorithm compared with an evolutionary algorithm, IWD. Here its contents are the required shortest path length to reach global optima, and values in parentheses are the number of generations for proposed algorithm, and required iteration number for IWD.

As it can be concluded from tables, for aforementioned problems and in comparison with different algorithms, proposed algorithm has converged to optimal solution with extremely smaller number of generations. Table VII, figure 7 and 8 include some other results to prove the preponderance of proposed algorithm as compared to the others. Table VII includes the results of 30 times implementation of SWAP_GATSP, OX_SIM, MOC_SIM and proposed method. Initial population for all of these methods is equal to 100. Contents of this table are as what we have in table V.

V. CONCLUSION

One of the permanent challenges for GAs is how to deal with premature convergence due to a sudden and fast reduction of search space and also getting stuck in local optima. Here a GA based on heuristic crossover and mutation is proposed to solve the traveling salesman problem. Suggested heuristic crossover, actually uses four pointers, one pair for each parent, which will move clockwise and counterclockwise. These pointers will be evaluated by means of a fitness function, equal to the inverse distance between two cities, and try not to get stuck in local optima. This approach improves the convergent speed towards the global optimal solution. The heuristic mutation will prevent premature convergence, too. The role of adaptive and nonlinear probabilities of crossover and mutation is to solve the low stability and also slow convergent. It must be mentioned that the implementation of algorithm has no complexity.

TABLE VII. COMPARISON THE RESULTS OF 30 RUNNING ITERATION OF SWAP_GATSP[5], OX_SIM[1], MOC_SIM[1]

PROBLEM		SWAP_GATSP	OX_SIM	MOC_SIM	PROPOSED ALGORITHM
eil51 n=51 optimal=426	best	439(220)	493(2500)	444(1600)	428(27)
	average	442(700)	540(3000)	453(3000)	431(28)
st70 n=70 optimal=675	best	685(600)	823(4500)	698(4500)	679(30)
	average	701(1000)	920(7500)	748(7500)	685(35)
eil76 n=76 optimal=538	best	548(700)	597(5000)	562(3800)	548(43)
	average	555(1000)	620(7500)	580(7500)	552(43)
kora100 n=100 optimal=21282	best	21397(2000)	21746(10000)	21514(8200)	21285(36)
	average	21740(3000)	22120(12000)	21825(12000)	21353(50)

REFERENCES

[1] S. S. Ray, S. K. Pal, S. Bandyopadhyay, "Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering", Applied Intelligence, vol. 26, no. 3, pp. 183-195, 2007.

[2] S. A. Amraii, M. Ajallooeian, C. Lucas, "A Dynamic Fuzzy-Based Crossover Method for Genetic Algorithms", In Proceedings of the 19th IEEE international Conference on Tools with Artificial intelligence, IEEE Computer Society, Washington, DC, vol. 01, pp.465-471, 2007.

[3] C. F. Tsai, C. W. Tsai, T. Yang, "A Modified Multiple-Searching Method To Genetic Algorithms For Solving Traveling Salesman Problem", 2002 IEEE conference on systems, Man and cybernetics, vol. 3, pp. 6-9, 2002.

[4] L. Jiao, L. Wang, "A Novel Genetic Algorithm Based On Immunity", IEEE Transactions on Systems, Man and Cybernetics, Part A, pp. 552-561, 2002.

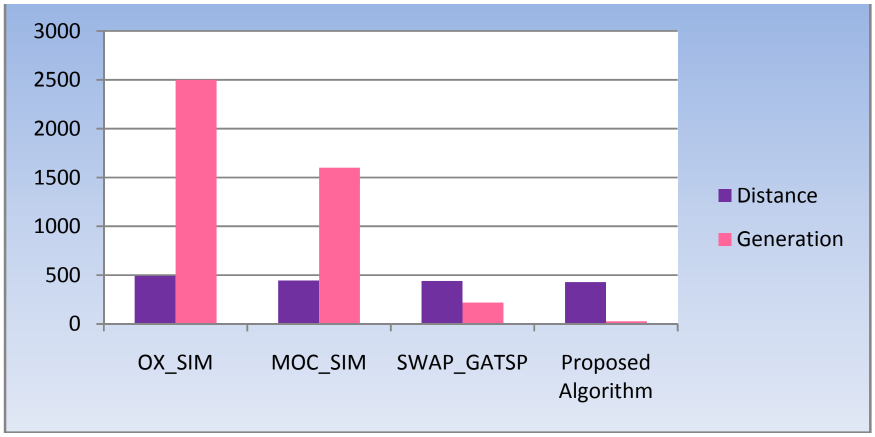

Figure 7. comparison the results of 30 running iteration of OX_SIM[1],MOC_SIM[1],SWAP_GATSP[5] and Proposed GA for standard Eil51 Problem

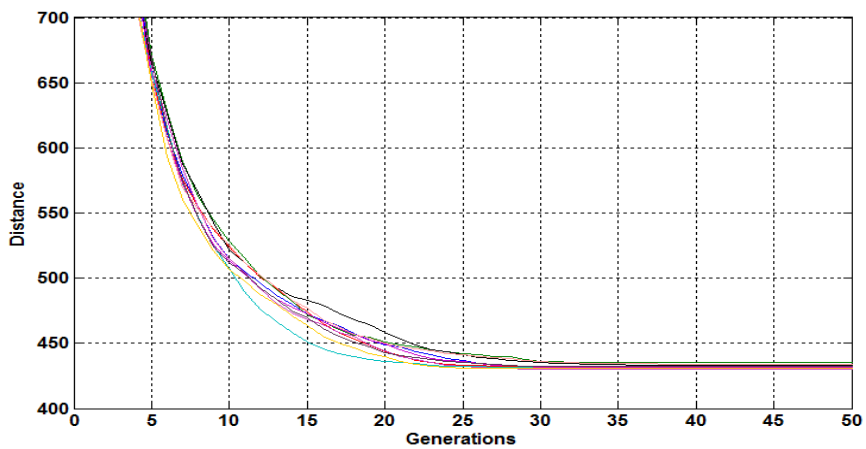

Figure 8. convergent curves of running proposed GA 10 times for Eil51

[5] S. S. Ray, S. Bandyopadhyay, S. K. Pal, "New Operators of Genetic Algorithms for Traveling Salesman Problem", Cambridge, UK, ICPR-042, vol. 2, pp.497–500, 2004.

[6] Z. Tao, "TSP Problem Solution Based on Improved Genetic Algorithm", In Proceedings of the 2008 Fourth international Conference on Natural Computation, ICNC. IEEE Computer Society, Washington, DC, vol. 01, pp. 686-690, 2008.

[7] C. N. Fiechter, "A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems", Discrete Appl. Math. 51, vol. 03, pp.243–267, 1994.

[8] M. Dam, M. Zachariasen, "Tabu search on the geometric traveling salesman problem", Meta-heuristics: Theory and Applications, Kluwer Academic Publishers, Boston, pp. 571-587, 1996.

[9] N. Yang, P. Li, B. Mei, "An Angle-Based Crossover Tabu Search for the Traveling Salesman Problem", Third International Conference on Natural Computation (ICNC 2007), vol. 4, pp. 512-516, 2007.

[10] Y. Bai, W. Zhang, Z.Jin, "An new self-organizing maps strategy for solving the traveling salesman problem", Chaos, Solitons & Fractals, vol. 28, pp. 1082–1089, 2006.

[11] H. Wu, Y. Yang, "Application of continuous hopfield network to solve the TSP", 8th International Conference on Control, Automation, Robotics and Vision Kunming, China, vol. 03, pp. 2258-2263, 2004.

[12] T. Nakaguchi, K. Jin'no, M. Tanaka, "Hysteresis Neural Networks for Solving Traveling Salesperson Problems", 2000 IEEE International Symposium on Circuits and Systems, Switzerland , vol. 03, pp. 153-156, 2000.

[13] Q. Zhu , S.Chen, "A New Ant Evolution Algorithm to Resolve TSP Problem", IEEE Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp. 62-66, 2007.

[14] Z. Cai, "Multi-direction Searching Ant Colony Optimization for Traveling Salesman Problems", In Proceedings of the 2008 international Conference on Computational intelligence and Security, CIS. IEEE Computer Society, Washington, DC, Vol. 02, pp.220-223, 2008.

[15] Y. Xing , Z.Chen, J. Sun, L. Hu, "An Improved Adaptive Genetic Algorithm for Job-Shop Scheduling Problem", In Proceedings of the Third international Conference on Natural Computation (ICNC), IEEE Computer Society, Washington, DC, vol. 04, pp. 287-291, 2007.

[16] TSPLIB,http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/.

[17] H. Shah_Hosseini, " Problem Solving by Intelligent Water Drops", Evolutionary Computation, CEC 2007, Singapore, pp. 3226-3231, 2007.