



A New Solution for Travelling Salesman Problem By Genetic Algorithm

Shirin Hatami*

Young research club, Islamic Azad University,
Babol-branch, Babol, Iran
Shirin_hatami@yahoo.com

Morteza Babazadeh

Faculty member of Islamic Azad University,
Babol-branch, Babol, Iran
Morteza_babazade@yahoo.com

Abstract: In this paper we have proposed a new solution for TSP. in the beginning of the algorithm we produce pool of solutions randomly and then we try to obtain better solutions by mutation, cross over and selection best chromosomes. We have designed a special kind of cross over and mutation operator in this algorithm. Each chromosome has a fitness that represents length of a tour and an importance number that represents number of links in proposed tour of chromosome.

Keywords: TSP; Genetic Algorithm; NP-complete

I. INTRODUCTION

There are many algorithms for solving TSP problem. Some of these algorithms can find the optimal solution of this problem. For example dynamic programming[1]. The main weakness of this algorithm is the execution time of them. They have a complexity greater than polynomial functions that called NP-Complete algorithms. Some other algorithms try to solve this problem in an acceptable time. For example greedy method[2]. But these solutions cannot guarantee to obtain an optimum tour for salesman. In this paper we have used genetic algorithm to solve this problem. So first we have to determine inputs of algorithm. We have a matrix that represents a map and the roads between them. each row and each column equivalent a city. m_{ij} in matrix M represents distance of a direct road between cities i and j . If there is no any road between cities i and j then m_{ij} will be ∞ . Fig 1 shows a map and its equivalent matrix. In this paper we have supposed that all roads are flat and there is now any difference between roads.

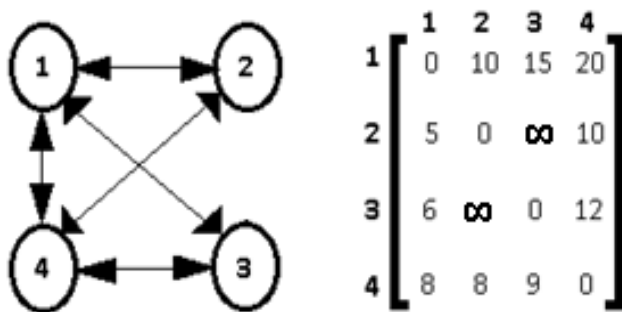


Figure 1. samples

II. CHROMOSOME

In proposed chromosome we have n genes where n is number of cities. Gene number one contains the first city in tour. The second gene contains next city in tour and so on. Finally the last gene represents the city that we have to go to the first city from it. For example for the graph in Fig 1 we have a chromosome by 4 genes. (1,3,4,2), (2,3,4,1), etc are sample configuration for this chromosome. Although some of this configuration are not a valid solution for our problem.

Each chromosome has a fitness that represents length of tour in this chromosome. Another field for measuring chromosome efficacy is importance number. Importance number is number of valid trips between cities that proposed by a separate chromosome. For example importance number of (2, 3, 4, 1) is 3 because there are direct road between cities 3 and 4, cities 4 and 1, cities 1 and 2. But there is no any direct road between cities 2 and 3. Obviously maximum value of importance number will be n where n is number of cities. For many chromosomes the value of fitness is ∞ , because it is possible that there was no any direct road between proposed trips by chromosome. By importance number we can judging between chromosomes that have ∞ value in their fitness to create better generation. Now we have to introduce proposed mutation and cross over operators. Fitness and importance number of each chromosome, evaluate by algorithm in Fig 2

```
function caculate(chromosome Ch)
{
    fitness = 0;
    number = 0;
    for(i = 1 to n)
    {
        k = ch.genes[i];
        if (i == n)
            l = 1;
        else
            l = ch.genes[i + 1];
        fitness = fitness + mkl;
        if (mkl !=  $\infty$ )
            number ++;
    }
    return fitness and number;
}
```

Fig2. Algorithm of calculating fitness & importance number

III. CROSS OVER OPERATOR

In each chromosome, the value of genes has to different and n cities had to be distribution in n genes. In fact we have permutation for cities in genes. So we have a cyclic cross over operator that after cross overing we will obtain chromosomes with considering this limitation.

Suppose that we two chromosome like A= (1,3,4,2,5,6) and B=(4,5,6,1,3,2) and we want to execute cross over operation on them to create child C. In the beginning we select the first gene of A that equals 1 and put this gene in child chromosome as first gene($c = (1, *, *, *, *, *)$). Whereas the first gene in B is equal 4, we have to put city 4 from A to C($c = (1, *, 4, *, *, *)$). Now 4 from A placed on third gene of child and in third gene B contains 6 so $c = (1, *, 4, *, *, 6)$. then 6 placed on child and then $c = (1, *, 4, 2, *, 6)$. Now 2 placed on forth gene in B contains 1, that number 1 previously copied to children. in this moment cycle is finished. We have to select the reminder genes from B. finally the child is equal C= (1, 4, 5, 2, 3, 6). As you see we have succeed to build a new children from two parents by using of a cyclic cross over.

IV. MUTATION

For mutation operator we select one gene from chromosome and then exchange it to another number. For example suppose that in (1, 2, 3, 4, 5, and 6) after mutation number 3 convert to 5. Accordingly we have to change number 5 to 3 in chromosome. Finally we (1, 2, 5, 4, 3, 6) as a new tour.

V. SELECTION

As we explained before we have a fitness and a importance number in each chromosome. All chromosomes that have a fitness less than ∞ will be select for next generation, because these are complete tours for salesman. Then will sort the reminder chromosomes on importance number and select best of them. Finally 5% of reminder chromosomes will be select randomly.

VI. EVALUATION ALGORITHM MODEL

For evaluation proposed algorithm first we implement proposed algorithm, classical algorithm based on branch and bound and a classic intelligent method. Input graph has created randomly. We can just determine number of cities. Map created by set randomly the proximity matrix row to row.

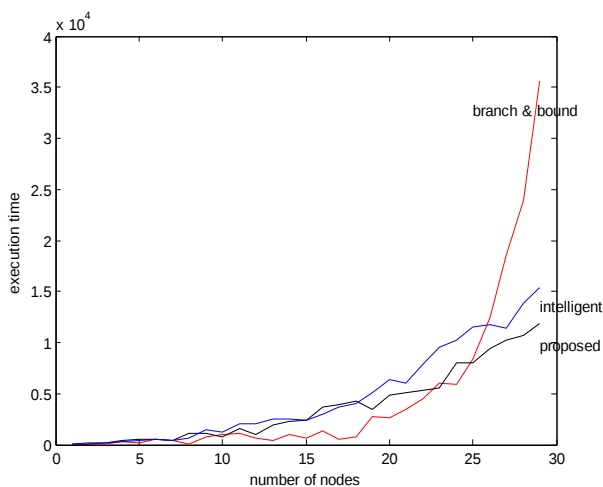


Figure 3. efficiency of algorithm

execution time of classical intelligent, branch & bound and the proposed method has shown in Fig 3. As you see for about 25 city, branch and bound is better. For this beyond the proposed algorithm and the classical intelligent method are better.

For determine difference between proposed method and classical intelligent method we have solved several random problems by both algorithms and the solutions shown in Fig 4. In all executions basic population is 1000, probability of mutation is 0.1 and number of generation is 20.

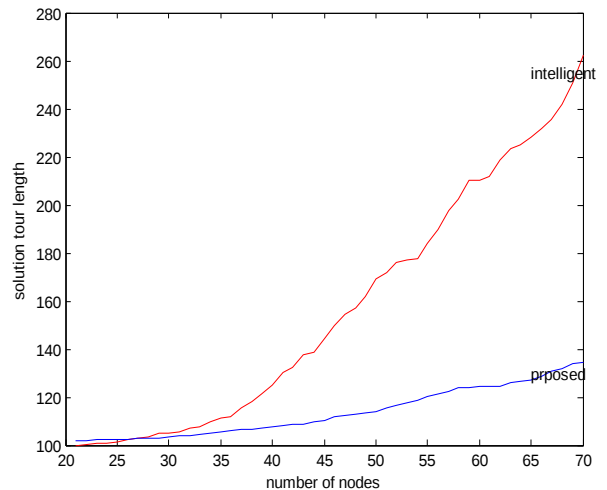


Figure4. compares algorithms

VII. CONCLUSION

Soft computing is the best method for solving Tsp because all the existence classic method is NP-complete. By these algorithms we can find an optimum solution in an acceptable time for travelling salesman problem.

VIII. REFERENCES

- [1] Richard Newpolitan; kewmars naeemipoor. Algorithm design
- [2] Ellis Horowitz, Sartaj Sahni; Sanguthevar Raja. Sem, Computer Algorithm
- [3] Milena Karova; Vassil Smarkov; Stoyan Penev, Genetic operators crossover and mutation in solving the TSP problem, International Conference on Computer Systems and Technologies - CompSysTech' 2005, pages IIIA.20-1 to IIIA.20-6
- [4] Chiung Moon ; Jongsoo Kim ; Gyunghyun Choi ; Yoonho Seo, An efficient genetic algorithm for the traveling salesman problem with precedence constraints, European Journal of Operational Research 140 (2002) 606–617
- [5] J. Renaud; F.F. Bector; J. Ouenniche, A heuristic for the pickup and delivery traveling salesman problem, Computers and Operations Research 27 (2000) 905–916.