Research Journal of Mathematical and Statistical Sciences _____ISSN 2320–6047
Vol. **3(2),** 1-3, February **(2015)** *Res. J. Mathematical and Statistical Sci.*

A Representation with Novel Crossover Technique of the Genetic Algorithm for the Travelling Salesmen Problem

J. Heymendran[1], U. Priyatharsan[2] and P. Hemija Sarawana[2]
[1]University of Colombo School of Computing, Colombo, SRILANKA
[2]Department of Physical Science, Vavuniya Campus of the University of Jaffna, SRILANKA

Available online at: www.isca.in, www.isca.me
Received 8th December 2014, revised 18th January 2015, accepted 9th February 2015

Abstract

Genetic Algorithm (GA) is a search technique that mimics the process of natural selection. It is routinely used to generate useful solutions to optimization problems. In a GA, we simulate the survival of the fittest among individuals over consecutive generation for solving problems. Choosing a representation in the design of the GA is the major problem. In this research, Travelling Salesmen Problem (TSP) is solved by a new representation of GA. An encoding mechanism is developed and selection, crossover and mutation operators are defined. We have presented a GA variant for solving the TSP that uses the novel cross over method. In the crossover that uses one of the parent's position of the gene structure to mate with other parent's chromosomes. Our GA representation is tested with 17, 26 and 42 cities and found that our algorithm performs accordingly and generates expected near optimal results within acceptable levels. Using Brute force search for 17 cities problem, TSP would have needed checking 2.092279 x 10^{13} possibilities, But GA within 1000 iterations gives the optimal solution. The actual answer for 17 cities TSP is 2085. We also ran GA for 26 cities and 42 cities TSP's with average results of 1034 (actual 937) and 944 (actual 699) respectively for 5 runs. Population size, Number of generations are increased to 500, 2000 and to 8 respectively, we got average of 994 and 837 for 26 cities and 42 cities TSP's respectively. So we could see that tuning the GA parameters optimizes the result.

Keywords: Genetic algorithm, Travelling Salesmen problem, optimization.

Introduction

In this paper we introduce Genetic Algorithms (GA)[1], its concepts and techniques used in modeling optimization problems. Also we will apply GA techniques to the Travelling Salesman Problem (TSP)[2], discuss the methods used and finally we explain the results obtained.

We define TSP as given a collection of cities and the cost of travel between each pair of them, the problem, is to find the cheapest way of visiting all of the cities and returning to your starting point. TSP is more complicated that it might appear. By using a Greedy approach; starting from city and keep going to the city nearest to it, you cannot get optimal solution. One way to find the optimal solution is use Brute force search[3-5]. There are other techniques which use heuristics to search, but here we limit our discussion to GA and Brute force search techniques.

In Brute force search we search every possible solution possible to arrive at the optimal solution. Assuming a 1GHz processor can process up to 1 x 10^9 computations per second; using Brute force to a 17 city TSP you need to run the algorithm for 5.8 hours, for 26 city TSP and 42 city TSP, need to run for 491857243.9 and 1.06 x 10^{33} years respectively. In real world applications, we won't have resources and time check all those possible solutions. We need another way, and GA is one of techniques that can be used to get at least to a near optimal solution without searching the entire solution space.

GA is a technique inspired by biological processes such as selection, crossover and mutation[6,7]. GA follows the natural evolution in search of optimal solution of a problem. Here individuals or solutions are competing to survive[8]. Only the fittest individuals survive and reproduce (selection) others die off. The individuals are encoded into genes and chromosomes[9]. The selected parent's genetic material are mixed (crossover) during reproduction. A gene can also randomly change, this is called mutation[10]. We used these concepts in formulating our algorithm.

Methodology

Algorithm: Obtain the initial *population* and the maximum number of generations *max crossover gene.* Set *Generatation Counter* as 1.
Use *kill percent*parameter to kill the worse individuals from *population.*
Generate the solutions for the next iterations: i. Keep (100-*kill_percent*) * *population* of the solutions with best fitness as surviving individuals. ii. Generate solutions via *cross over.* iii. Select (100-*kill_percent*) * *population* solution from the previous population randomly and mutate them

Update *Generatation_Counter = Generatation_Counter + 1.* If *Generatation_Counter<= max_crossover_gene.,* go to step 1. Otherwise stop.

Figure 1 shows 2 sample routes and its decimal representation (the number list) which is the chromosome of each route. Each chromosome includes a list of cities starting with gene 0 the origin city and cities visited thereafter. A chromosome represents a single solution to the TSP. The GA algorithm starts with choosing the best individuals with the select operator.

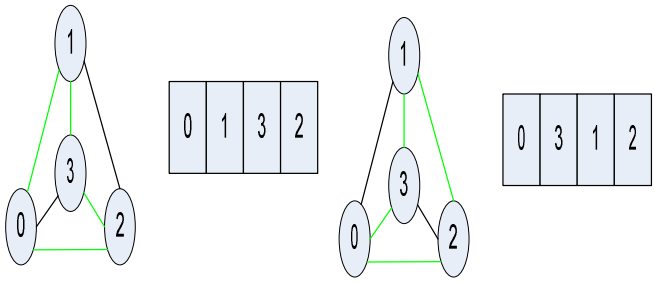

Figure-1
Paths from origin city 0 in green, and its chromosomes

The select operator uses *kill percent*parameter to kill the worse individuals from population. From the surviving individuals, parents are selected randomly to breed. The breeding process involves applying crossover operator. The crossover is done in a novel way that uses one of the parent's gene structure (position) information to affect the gene crossover in one of the parent's chromosomes. First, the parent "A", which will give the structure information, is selected randomly. Then we select the genes that will crossover, for this we randomly select a gene from the gene pool, gene pool being the cities in the search space. The number of genes selected will be an even random number between *min crossover genes* and *max crossover genes* parameter. In the next step the list of positions of the selected genes in Parent "A" is collected. For each position value in the list; in Parent "B" (partner to Parent "A"), the gene in the same position will swap its gene with, the gene in the next position value.

We use only one of the parent's genes for crossover but traditional crossover method uses both parents' genes. We could use this method because; by nature of the problem both chromosomes have the same genes and differ only in their positions. Using same chromosome, we avoid chromosomes becoming invalid since chromosome does not lose any of its genes, just rearranges itself and also it's simplifies our implementation. Finally, we apply mutation on population using *mutation percent* parameter. During mutation, we randomly pick two positions in chromosome and swap genes in them. We ran the algorithm for defined number generations with defined population size and plotted each generation's best (minimum distance), average, and worse distances.

Results and Discussion

In this section we discuss the results obtained from running the 17 city problem. In figure 2 early in the generation, the steep drop indicates rapid rate of improvement of the solution because of crossover and mutation operations, but as the solution improves the rate decreases and finally flats out. Given a large number of iterations the best, average and worse should converge along the way.

Table-1
Number times GA executed

No.	Optimum Sol. Found	Change (-2085)
1	2119	34
2	2153	68
3	2090	5
4	2090	5
5	2090	5
6	2088	3
7	2090	5
8	2119	34
9	2120	35
10	2090	5

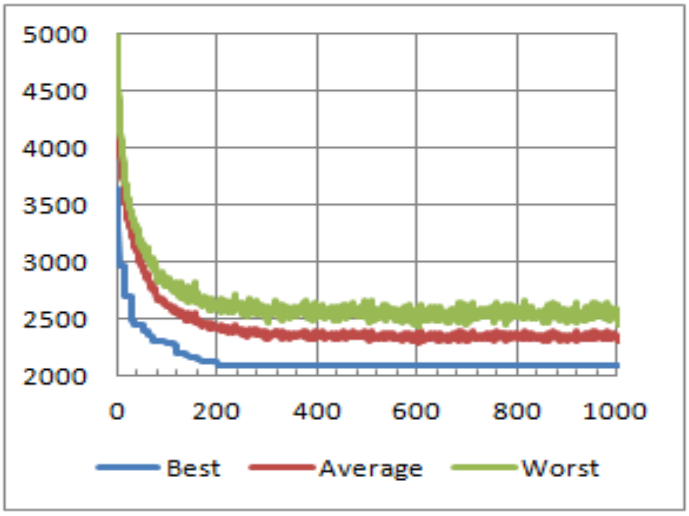

Figure-2
Solution (Distance) vs Generation

Using Brute force search for 17 cities TSP would have needed checking 2.092279 x 10^{13} possibilities, But GA within 1000 iterations gives near the optimal solution. The table 2 presents number of times GA was run and its result. The actual answer for 17 cities TSP is 2085, we can see from table 1 that the optimal solution found each time, and difference from actual result is minimal.

Each GA run produces near optimal solution. In a real world application, to get quick answers at an acceptable level, we can define a threshold that would satisfy the particular domain. We also ran GA for 26 and 42 TSP's with average results of 1034

(actual 937) and 944 (actual 699) respectively for 5 runs. When we increased the population size to 500, number generations to 2000 and *max crossover gene* parameter to 8, we got average of 994 and 837 for 26 cities and 42 cities TSP's respectively. So we could see that tuning the GA parameters optimizes the result.

Conclusion

In this paper we have given a very effective procedure for TSP by using the genetic algorithms. We have presented a GA variant for solving the TSP that uses the novel cross over method. In the crossover that uses one of the parent's gene structure (position) information to affect the gene crossover in one of the parent's chromosomes. The optimal solutions for several TSPs obtained using this technique are generally competitive with the best known solutions. These results suggest that choice of representation plays an important role in the GA's ability to satisfactorily solve the TSP. For the method presented the maximum deviation of the optimal solutions from the best known solutions is less than 2%.

References

1. Michalewicz Z, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin **(1994)**

2. P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, I. Inza and S. Dizdarevic, Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators **(1999)**

3. J. Kirk, Traveling Salesman Problem – Genetic Algorithm, Matlab Central, **(2009)**

4. Mou L., An efficient ant colony system for solving the new generalized traveling salesman problem, CCIS2011 - Proceedings: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, 407 **(2011)**

5. R. Durbin and D. Willshaw, An Analogue Approach to the Traveling Salesman Problem Using an Elastic Net Method, Nature, vol. 326

6. N. Ernest and K. Cohen, Fuzzy clustering based genetic algorithm for the multi-depot polygon visiting Dubins multiple traveling salesman problem, in Proceedings of the 2012 AIAA Infotech@Aerospace, no. AIAA-2012-2562.Garden Grove, CA: **(2012)**

7. N. Ernest and K. Cohen, Self-Crossover Based Genetic Algorithm for Performance Augmentation of the Traveling Salesman Problem, AIAA, Infotech@Aerospace 2011, St. Louis, Missouri **(2011)**

8. Bhattacharya Sourabh, Applications of DSTATCOM MATLAB/Simulation in Power System, Res. J. Recent Sci., 1(ISC-2011), 430-433 **(2012)**

9. PitalúaDíaz N., Lagunas Jiménez R. and González Angelesa, Tuning Fuzzy Control Rules via Genetic Algorithms: An Experimental Evaluation, *Research Journal of Recent Sciences,* **2(10),** 81-87, **(2013)**

10. Esmail Limouzade, Capacitor Replacement in Distribution Networks using Genetic Algorithm, *Research Journal of Recent Sciences,* **2(12),** 54-64, **(2013)**