Brute force & Genetic Algorithms

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

We first developed encoding mechanism and we had to define selection, crossover and mutation operators. Then we used the general algorithm [1] for GA to generate the solutions. We used novel way model crossover, where we used one of the parent's structure information to affect the crossover in the other parent. We tested our GA with 17, 26 and 42cities and found that our algorithm performs accordingly and generates expected near optimal results within acceptable levels. In the next section describes in detail the methods we used.

2. METHODOLOGY

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.

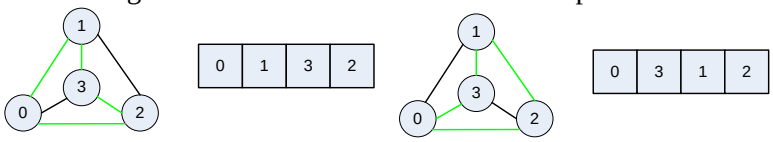

Figure 1: Paths from origin city 0 in green, and its chromosomes.

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 parent's 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.

3. 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.

Using Brute force search for 17 city TSP would have needed checking 2.092279×10^{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 city TSP is 2085, we can see from table 1 that the optimal solution found each time, and difference from actual result is minimal.

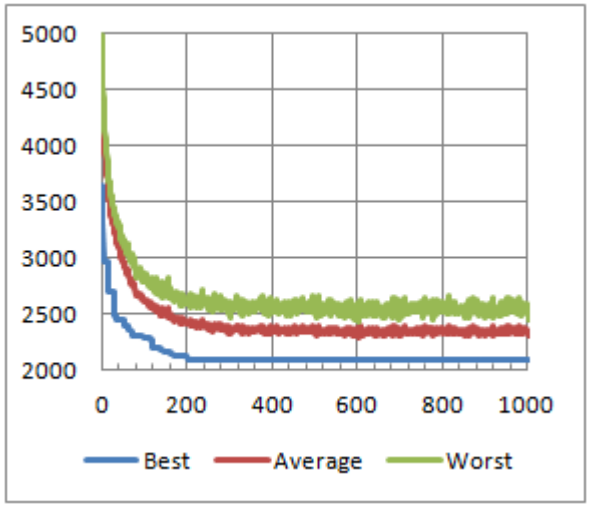

Figure2: Solution(Distance) vs Generation

Table 1: Number times GA executed

No.	Optimum Solution. 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

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 city and 42 city TSP's respectively. So we could see that tuning the GA parameters optimizes the result.

4. CONCLUSIONS

We have seen GA is good candidate for hard problems that require quick answers. In comparison with Brute force search, GA with appropriate parameters was able produce near optimal solutions within relatively small number of iterations. In modeling GA, we found that representation affects uniqueness, optimization and validity of solutions, also simplifies the implementation.

REFERENCES

1. Michalewicz, Z. (1994), Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin
2. Travelling Salesman problem. (n.d.). Retrieved April 3, 2014 from the TSP home page of univ. waterloo: http://www.math.uwaterloo.ca/tsp/problem/index.html