Paper 161-2012

The Traveling Salesman Problem: Optimizing Delivery Routes Using Genetic Algorithms

Sabah Sadiq, Chicago, IL, USA

ABSTRACT

The purpose of this paper is to discuss the methodology of optimizing delivery route scheduling using genetic algorithms to solve the Multiple Traveling Salesman Problem (*m*TSP). The traveling salesman problem (TSP) is a combinatorial optimization problem where a salesman must find the shortest route to *n* cities and return to a home base. While the TSP restricts itself to one salesman, the *m*TSP generalizes the problem to account for multiple salesmen.

With robust clustering and genetic algorithm procedures, SAS ® provides an ideal environment to solve this type of problem. In order to utilize SAS to optimize the delivery routes, clustering will be used, via PROC FASTCLUS, to transform the *m*TSP problem into a set of traditional TSP problems. These single TSP problems will then be solved using genetic algorithms, via PROC GA, and visualized using PROC SGPLOT. Finally, the genetic algorithm solutions will be compared to the greedy algorithm solution in terms of optimal tour distances and processing speed.

INTRODUCTION

A company must optimize their daily delivery route scheduling from a central warehouse. One hundred packages must be delivered and five trucks are available. The purpose is to assign each truck a set of packages and find the route that each truck should take to minimize distance. Additional costs such as traveling time and gas are implied in the distance traveled. Table 1 lists the addresses for the first five packages that must be delivered.

Customer ID	Address	Latitude	Longitude
1	100 Renaissance Center, Detroit, MI	42.329921	-83.040613
2	22930 Woodward Avenue, Ferndale, MI	42.461841	-83.13517
3	42855 Woodward Avenue, Bloomfield Township, MI	42.602327	-83.263204
4	105 East 2nd Street, Rochester, MI	42.678235	-83.133347
5	1816 Maplelawn Dr., Troy, MI	42.551198	-83.175077

Table 1: Delivery Addresses Sample Data

SAS provides the perfect platform to solve this problem using traditional TSP techniques such as genetic algorithms and the greedy algorithm. The data provided in this section was read into a SAS dataset that was used to cluster the packages together, solve the clusters using genetic algorithms, graph the solution, and compare the genetic algorithm solution to the greedy algorithm solution.

THEORY

THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem (TSP) is categorized as being a combinatorial optimization problem. Combinatorial optimization problems are problems that attempt to find an optimal object from a finite set of objects. In this case, the goal is to find the optimal tour (path to visit cities) given all possible tours. In this sense, an optimal tour is one in which distance is minimized. Computationally, the TSP is difficult to solve for a large number of cities as the solving time increases exponentially for every increase in *n*. Thus, approximation and heuristic algorithms are often used to solve these problems.

The delivery route scheduling problem is one representation of the TSP, where a city is a delivery address (i.e. package) and a salesman is a truck. Additional TSP applications include planning, logistics, microchip manufacturing, etc. In these examples, a city can represent customers, soldering points, DNA fragments, etc. Similarly, distance represents traveling time or cost.

THE MULTIPLE TRAVELING SALESMAN PROBLEM

The multiple traveling salesmen problem expands the traditional TSP to allow for multiple salesmen. Thus, each city must be visited exactly once by *any* salesman. The key characteristics of the *m*TSP determine the depot (single depot, multiple depots) and destination (fixed destination, non-fixed destination). In our case, every salesman departs from a single warehouse or depot. Additionally, every salesman must return to the starting city (i.e. GIS coordinate) and thus has a fixed destination. To date, no efficient algorithm exists for the solution of a large-scale *m*TSP. Traditionally, cities are clustered together and assigned to different salesman, thus converting the *m*TSP problem into multiple TSP problems.

GENETIC ALGORITHMS

One method used to solve this type of problem is genetic algorithms. Genetic algorithms are search heuristics that attempt to find an optimal solution based on a process of natural selection and evolution. The process involves generating random solutions and applying genetic operators in hopes that the solution will evolve to find the optimal solution. This process can be characterized by four stages:

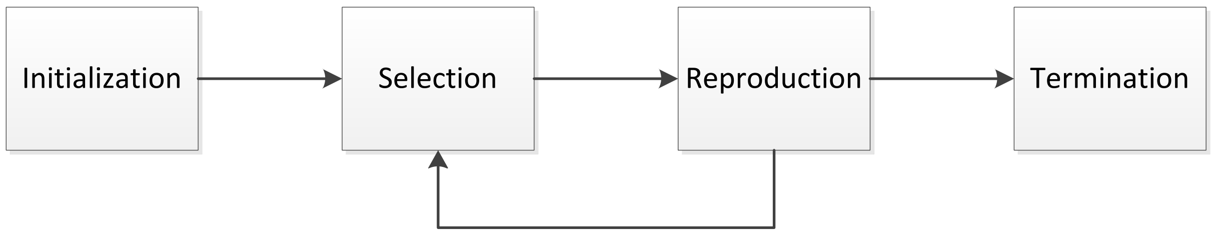

Figure 1: Four Stages of Genetic Algorithms

During initialization, a random set of solutions is generated to form an initial population. These random solutions are generated from the entire range of possible solutions (the search space). Once an initial population has been formed, the solutions are evaluated based on the fitness function (equivalent to an objective function). The best solutions are then selected to breed a new population. During the reproduction phase, genetic operators are applied to the solutions in order to breed new characteristics. The most commonly used operators are mutation and crossover:

- Crossover operations involve combining two solutions (parents) to produce more solutions (children). Genetic material from the previous generation is given to the subsequent generation maintaining fitness strength.
- Mutation operations involve randomly changing some characteristics of a solution. This introduces a certain amount of randomness into the search and prevents premature convergence to a sub-optimal solution.

Once a new population has been bred, it is again evaluated for fit and the best solutions are selected to continue. This process repeats until a terminating condition has been reached. In this case, the terminating condition is a set number of iterations. Once terminated, the best solution in the current population is selected as being the optimal solution.

MODELING APPROACH

In order to solve the delivery route scheduling assignment, a three step approach was used. The SAS code used to generate this solution can be found in the references section.

STEP 1: CLUSTER PACKAGES TOGETHER

The k-means algorithm was used to group similar GIS coordinates together using PROC FASTCLUS. PROC FASTCLUS is a SAS procedure that performs a disjoint cluster analysis using Euclidean distances. This method involves assigning each observation to a cluster through calculating cluster centroid distances. K-means (i.e. PROC FASTCLUS) is a more efficient clustering algorithm for larger datasets as compared to the lengthier hierarchical clustering (i.e. PROC CLUSTER). When using PROC FASTCLUS, the following considerations should be made:

1) The variables that are being clustered may need to be standardized if they have different scales. This is done to ensure that the scale of the variable does not unduly influence the Euclidean distance calculations.
2) PROC FASTCLUS can be extremely sensitive to outliers. In fact, FASTCLUS can be a good outlier detection method as these observations appear as single stand-alone clusters in the FASTCLUS output.
3) With smaller datasets (<100), PROC FASTCLUS is sensitive to the order of observations.

In order to implement this solution using SAS, the latitude and longitude variables were fed into the clustering algorithm. As both of these variables have the same scale, no standardization was needed. Additionally, the constraint that only five trucks were available to deliver the packages was enforced using the maxclusters= option.

```
/*Cluster Packages*/
Proc fastclus data=cords out=clust maxclusters=5;
      var latitude longitude;
run;
```

Essentially, clustering the packages together breaks the large multiple traveling salesman problem into smaller single traveling salesman clusters. These TSP clusters can then be solved using traditional traveling salesman techniques. Table 2 lists the clusters that were developed using FASTCLUS and the number of packages assigned to each truck.

Truck	# Packages
1	50
2	1
3	16
4	31
5	2

Table 2: Cluster Assignment

Table 2 indicates that there are two clusters which have a lower number of assignments (Trucks 2 and 5). In order to examine these clusters, the cluster assignments were graphed according to latitude and longitude. Figure 2 displays the visual representation of these clusters.

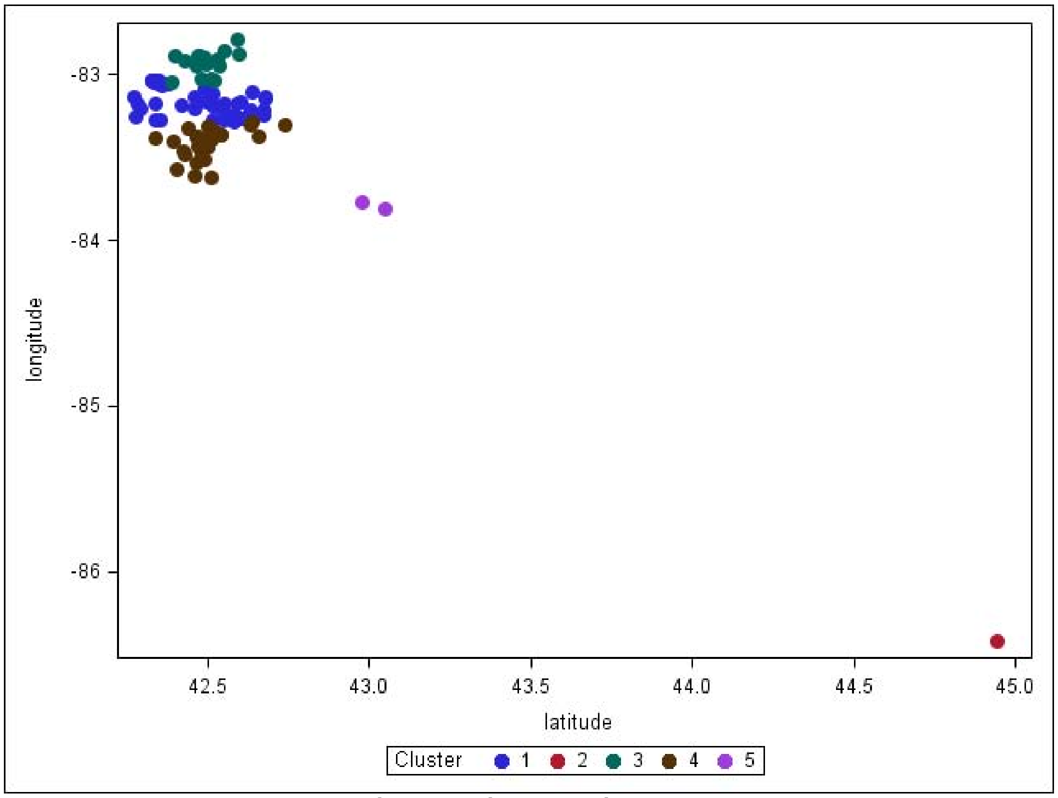

Figure 2: Cluster Assignment

As seen in Figure 2, Clusters 2 and 5 were assigned to their own clusters due to the large distance between cluster centroids. After verifying that the clusters were logical in terms of their package assignments, the five TSP problems were solved using genetic algorithms.

STEP 2: SOLVE THE TSP CLUSTERS USING GENETIC ALGORITHMS

The TSP clusters were solved using genetic algorithms, via PROC GA. PROC GA is a SAS procedure used to implement genetic algorithm searches. This procedure combines the efficiency of SAS programming techniques and genetic algorithm function calls. The data is initialized using programming statements similar to a SAS DATA step, allowing the user to leverage efficient SAS programming statements to manipulate the data and ready it for analysis.

Genetic algorithm function calls are then used to utilize the genetic algorithm solver to arrive at a solution.

In the PROC GA call, the final solution (i.e. optimal tour) was outputted to a SAS dataset using the lastgen= option. This dataset was used to create a graphical representation of the optimal tour after solving the TSP problem.

```
/*Solve Truck 1*/
Proc ga data1 = truck1packages lastgen=truck1solution seed = 48304;
```

In addition to outputting the final solution, the following actions were performed to solve the TSP problems:

1) The fitness function was initialized using SAS DATA step programming
2) Function calls were used to access the genetic algorithm solver

The first concern when modeling a TSP using genetic algorithms is specifying a fitness function (i.e. objective function). Since the goal is to minimize the distance traveled, the objective can be modeled as the sum of the Euclidean distances between the city locations in a tour. In order to model this using SAS, a distance matrix of the Euclidean distances must be created so that the fitness can be evaluated for all possible solutions using array indices. The following code displays this process for the first TSP cluster (Truck 1).

```
/*Initialize Fitness Function*/
call SetEncoding('S51');
array distances[51,51] / nosym;
do i = 1 to 51;
      do j = 1 to i;
            distances[i,j] = sqrt((latitude[i] - latitude[j])**2 +
            (longitude[i]- longitude[j])**2);
            distances[j,i] = distances[i,j];
      end;
end;
```

The SetEncoding call is used to represent the solution type and size. Possible solution types include Real Numbers (R), Integers (I), Booleans (B), and Sequence (S). For Truck 1, the TSP solution consists of a sequence of 51 cities (the 50 cities clustered together using PROC FASTCLUS and the home city). Thus, the solution is encoded as 'S51'. After specifying the solution type, the distance array is initialized using SAS DATA step programming. Additionally, the NOSYM operator is used to ensure that the array does not link to other variables for efficiency purposes.

In addition to the fitness function, PROC GA function calls are used to configure the genetic algorithm search. Table 3 lists the PROC GA options used to solve the TSP problems.

Genetic Algorithm Stage	Supporting PROC GA Call	Selected Options
Initialization	Initialize	Initializes a population using the DEFAULT method.
Selection	setSel	The tournament selection method was used to select the fittest solutions.
Reproduction	SetCross SetMut	The ORDER method was used for Cross-over and INVERT method was used for Mutation.
Termination	ContinueFor	Continues solution generation for a set number of iterations. This value was chosen based on trial-and-error.

Table 3: PROC GA Function Call Options

STEP 3: COMPARE GENETIC ALGORITHM SOLUTION TO GREEDY ALGORITHM SOLUTION

After solving and graphing the individual TSP problems using genetic algorithms, the solutions were compared to the greedy algorithm in terms of processing speed and optimal tour distances. The greedy algorithm is a simplistic algorithm used to solve TSP problems that can provide good tour matches in a short amount of time. The greedy algorithm makes "greedy" choices in that it chooses to iteratively visit the closest unvisited city. Essentially, the algorithm sorts the cities based on ascending Euclidean distance and choses to visit the closest unvisited city. This process is repeated until all cities are visited. Although this algorithm produces good tour matches, it does not guarantee that the total distance will be minimized. In fact, for some specific cases the greedy algorithm has been shown to produce the worst possible solution.

RESULTS

After clustering the packages together, PROC GA was used to solve the five TSP problems. Table 4 displays the genetic algorithm results.

Truck	Distance Traveled	First 10 Customer Locations (by PROC GA ID)
1	1.653464087	22 – 31 – 15 – 34 – 40 – 44 – 19 – 20 – 45 – 41
2	7.8641307139	2 – 1
3	1.3607628695	5 – 6 – 7 – 13 – 16 – 17 – 9 – 14 – 1 – 10
4	1.5019987829	14 – 13 – 32 – 25 – 30 – 2 – 15 – 27 – 26 – 17
5	1.4334926297	2 – 1 – 3

Table 4: TSP Solutions Using Genetic Algorithm

The outputted datasets were then used to visualize the solution. Figure 3 displays the optimal path for Truck 1.

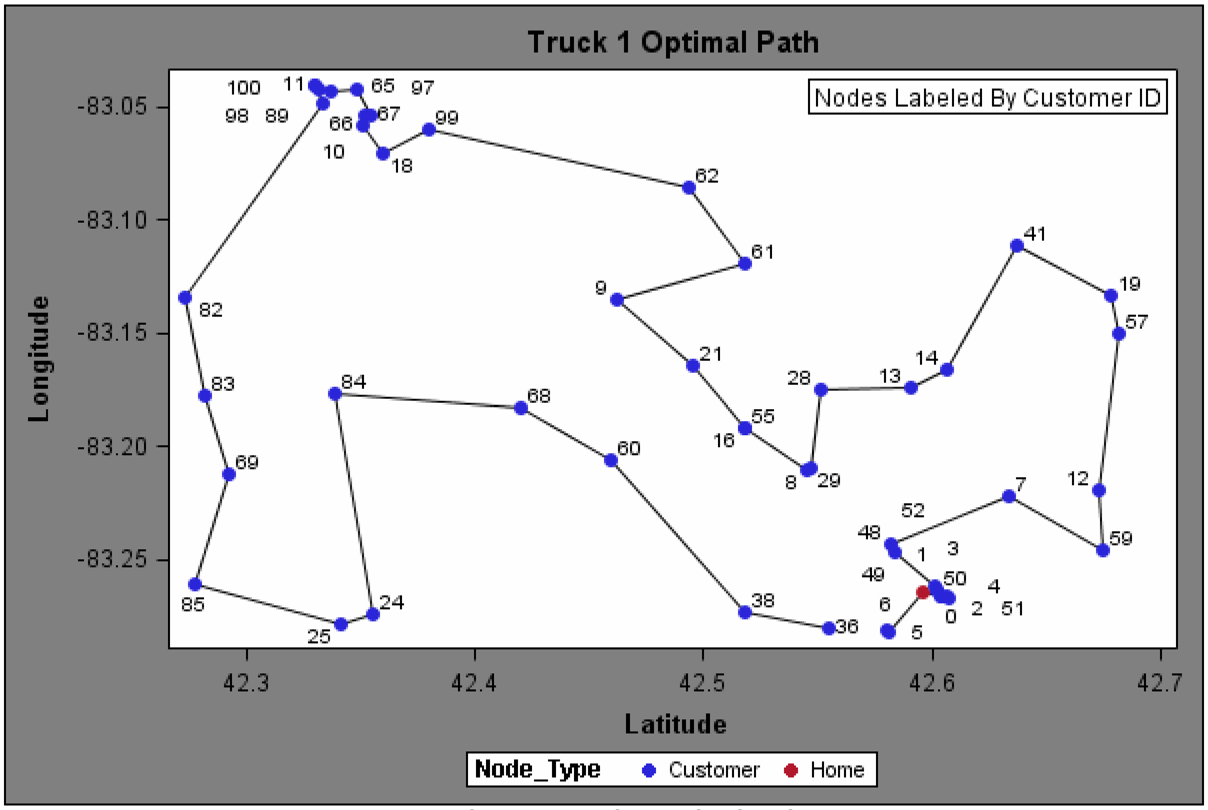

Figure 3: Truck 1 Optimal Path

Figure 3 shows that the optimal path generated by the genetic algorithm seems to be logical. Meaning, no intersecting paths were generated from this solution. After verifying that the solution follows a reasonable path, the solutions were compared to the greedy algorithm to gauge processing speed. Table 4 displays the tour distances for both methods and the increase in processing time incurred through using the genetic algorithm.

Truck	Genetic Algorithm Solution (Euclidean Distance)	Greedy Algorithm Solution (Euclidean Distance)	Genetic Algorithm Processing Time Increase (Seconds)
Truck 1	1.653464087	1.9434231	4.42 s
Truck 2	7.8641307139	7.8641307139	-1.218 s
Truck 3	1.3607628695	1.573486	3.313 s
Truck 4	1.5019987829	1.770256	0.952 s
Truck 5	1.4334926297	1.355027	5.246 s

Table 4: TSP Optimal Path Heuristic Comparison

In comparing the two methods, the genetic algorithm often provided an optimized solution to the traveling salesman problem with shorter tour distances. However, processing time also increased with an average increase of 2.54 seconds. Although, the processing time may change depending on the complexity and size of the problem, the results seen in Table 4 display that genetic algorithms provide a competitive alternative to greedy algorithms in solving the traveling salesman problem. When using genetic algorithms to solve this type of problem, the following considerations should be made:

1) The selected genetic algorithm options (initial size of the population, rate of mutation and crossover, selection type, and termination criterion) may affect the ability to converge to an optimal solution. These values should be selected with care using a trial-and-error approach to ensure that the genetic algorithm does not converge to a sub-optimal solution.
2) Genetic algorithms are not guaranteed to find the global optimum. Various factors including the selected options and deceptive individual strength can cause premature convergence. If a certain individual emerges early in the search as being a strong competitor, it may bias the search to converge on a local optimum that represents the competitor rather than a global optimum.
3) Processing time may increase depending on the size and complexity of the problem.

CONCLUSION

The multiple traveling salesman problem is applicable to a variety of industries and functions. However, exact solutions become infeasible for problems with more than twenty cities. Using genetic algorithms, SAS users can reach valid solutions in a reasonable amount of time. PROC FASTCLUS and PROC GA provide an efficient means to implement this type of solution. Through the use of efficient SAS DATA step programming and function calls, the genetic algorithm was proven to be a competitive alternative to the greedy algorithm. Additionally, the optimal solution can be outputted to a SAS dataset and PROC SGPLOT can be used to visualize the optimal path in order to confirm that the solution is both logical and feasible.

REFERENCES

Sadiq, S (2012). Traveling Salesman. Retrieved February 20, 2012, from https://sites.google.com/site/travelingsalesmansas/

Solomon, Marius M (03/01/1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". *Operations research (0030-364X)*, 35 (2), p. 254.

Paschos, Vangelis Th (01/31/2003). "Approximation algorithms for the traveling salesman problem".*Mathematical methods of operations research (Heidelberg, Germany) (1432-2994)*, 56 (3), p. 387

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Sabah Sadiq
Address: 111 S. Wacker Dr.
City,StateZIP: Chicago, IL 60601
Work Phone: 847-504-9038
Fax: 866-770-0956
E-mail: ssadiq07@gmail.com
Web:http://www.linkedin.com/pub/sabah-sadiq/8/11/31b