Solving the Vehicle Routing Problem with Multiple Multi-Capacity Vehicles

Michael S. Sanders, Jr.
Computer Systems Lab, 2008-2009
Thomas Jefferson High School for Science and Technology
Alexandria, Virginia

April 1, 2009

Abstract

The Vehicle Routing Problem (VRP) has existed as long as a distributor has needed to deliver items. As such, the VRP has been solved with many different methods, including agent architecture and ant colony modeling. However, these methods have generally been set up for an established organization that has a specific number of vehicles with only a few unique capacities. This project aims to create a program that will solve the VRP, but in a case where all the vehicles could have different capacities. This is the situation faced by some volunteer groups that do not have established vehicle fleets and rely on people volunteering vehicles when something needs to be distributed.

Keywords: Vehicle Routing Problem, heuristics, A* search

1 Introduction

The goal of this project is to create a program to quickly find the most efficient routing of a given number of vehicles with differing capacities to a variety of delivery points with a variety of demands of product. This

is a pure-applied research project. It is being created to assist a volunteer group in their distribution of goods.

The project is created linearly. The first step is the creation of a route finder, which includes the locating and processing of road data. The route finder is built upon through the addition of a heuristic algorithm to decrease the time required to find a route. The second major step is then to create a solution finder. The solution finder will implement another heuristic algorithm to determine a very good route. Neither the route finder or solution finder will be expected to routinely return the best results, given the scope of the problem. It is expected that this deficiency will be compensated for by human intuition, as the drivers of the vehicles can be expected to be familiar with the area road network.

2 Background

A great deal of research has been done into the VRP and its variants, such as the VRP with Time Windows (VRPTW) and Multi-Depot VRP (MDVRP). Projects have looked into using agent architecture and ant colony optimization to solve the problems. These projects have yielded such ideas as Clarke-Wright Algorithm to place delivery points into routes. Mennell, Shmygelska, and Thangiah did a project to evaluate using agents to solve the VRP. The program used agents to represent the vehicles and "auctioneers" that informed the vehicles of the current situation with regards to customers and deliveries. While the resulting program did not find the optimal solutions for any the sample problems, it was extremely adaptable for the MDVRP and VRP with multi-capacity vehicles.

3 Development

The project has two major components: the route finder and the solution finder.

3.1 Route Finder

The route finder began as an extension of an assignment from a previous computer science course. The assignment had dealt with various manners

of searches through a map. The type of search selected for this project was an A* search. This component began as a stand-alone program, and can still function in that manner. The program receeves an input of a starting address and an ending address. It returns a length (in miles) and a list of streets traversed. It does not return good results when the two addresses are very far apart from each other. Given the scope of the problem, however, it returns very acceptable results for the intended purpose.

3.1.1 Road Data

The road data comes from the US Census Bureau. It was produced in 2006 using the Census Bureau TigerLine system. The TigerLine system uses five sets of data to list a road segment's name, geographic coordinates, address ranges, type of road, and alternate road names, among other data. A common TigerLine Identification number (TLID) is used to link entries from each of the data sets together. The following table shows the different pieces of data from each record type for a particular road segment. The particular segment is the section of Braddock Road immediately outside TJHSST.

Record Type (RT) Number	Data
RT1	”11106 76033712 A Braddock Rd A31 6555 6567 6560 656611112231222312 51510590599454394543 45784019124525004521003007 4003 -77166726+38817271 -77168032+38816700”
RT2	”21106 76033712 1 -77167098+38817200 -77167268+38817182 -77167327+38817171 -77167439+38817137 -77167616+38817063 -77167707+38817007 -77167817+38816919 -77167921+38816825+000000000+00000000+000000000+00000000”
RT4	”41106 76033712 1 269 ”
RT5	”5110651059 269 State Route 620 ”
RT6	None

Record Type: 1
Version Number: 1106
TLID: 76033712
Source Code: A
Name: Braddock Rd
Full Name: Braddock Rd
Street Direction: 246.384432363328, Southwest
Length: 0.0843058817053028 miles
CCFC: A31
City: Fairfax County
Start Address, Left: 6555; End Address, Left: 6567
Start Address, Right: 6560; End Address, Left: 6566
Start Impute, Left: 1; End Impute, Left: 1
Start Impute, Right: 1; End Impute, Right: 1
Zip Code, Left: 22312; Zip Code, Right: 22312
Starting Coordinates: +38.817271, -77.166726
Ending Coordinates: +38.816700, -77.168032
Additional Coordinates: [[[”+38.817200”, ”-77.167098”], [”+38.817182”, ”-77.167268”],
[”+38.817171”, ”-77.167327”], [”+38.817137”, ”-77.167439”], [”+38.817063”, ”-77.167616”],
[”+38.817007”, ”-77.167707”], [”+38.816919”, ”-77.167817”], [”+38.816825”, ”-77.167921”]]]
Additional Names: [[””, ”State Route 620”, ””, ””]]
Additional Address: []

3.1.2 A* Search

The route finder implements an A* search for finding a route. After identifying the starting and ending points, the program then begins to iterate through paths. A path is an array containing a list of all the geographic coordinates that that path has passed through. The program then determines what streets intersect at the path's current location. It then assembles new paths by copying the current path and adding on each intersecting street. The program then determines how long each path is (how far the path has traveled from the starting point) and estimates how much farther the path has to go before it reaches the target. Once it has created all the new paths, the program sorts all the paths based on estimated total length. The goal of the A* search is to reduce time used for searching by identifying the paths that are most likely to be the shortest routes.

3.2 Solution Finder

The solution finder is designed to return a better-than-average set of routes that efficiently utilize available vehicles and delivers the product to all delivery points. It has two specific classes, the Route class and the Solution class. A Solution object has an array of Route objects. In terms of genetic algorithms, the Route objects are the "genes" of the Solution object. The solution finder will have a certain number of Solution objects that will be manipulated to find an acceptable result. A Route object has a vehicle assigned and contains an array listing all the deliveries on that route.

3.2.1 Genetic Operators

These are various operators that can be used to perform genetic mutations. Customers can be displaced from routes, deleted from one route and inserted in another, or swapped with other customers. Routes can also be fully or partially inverted.

3.2.2 Initializing

The program initially creates a list of customers. From that list, it randomly selects a customer and tries to add it to the current route. If it is

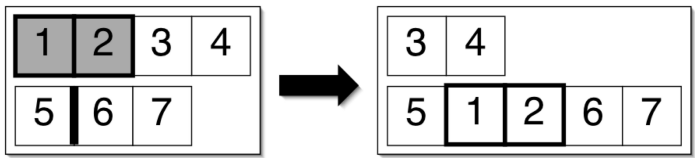

Figure 1: Displacement operation

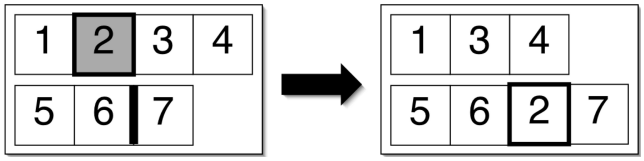

Figure 2: Insertion operation

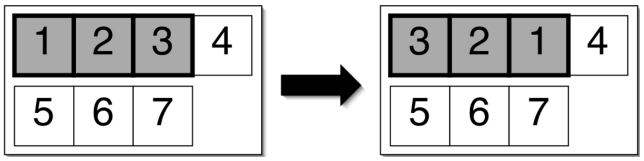

Figure 3: Inversion operation

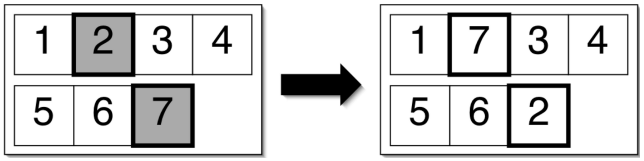

Figure 4: Swap operation

successful, then it proceeds to the next customer. If it is not, it then creates a new route and assigns the next vehicle. It then repeats this process until all customers are added. It then starts the process again to create a new Solution object, creating new Solution objects until it has hit a certain limit.

3.3 Heuristics

This program attempts to replicate a person's job. Therefore, it has been given certain guidelines in terms of how to evaluate results intelligently. The route finder estimates total distance by adding the length of the current path to the estimated distance to the target. The distance is estimated through the distance formula as stated below:

$$distance = \sqrt{(\Delta latitude)^2 + (\Delta longitude)^2} \tag{1}$$

where

$$\Delta latitude = (latitude_{current} - latitude_{destination}) * 69.17 \tag{2}$$

and

$$\Delta longitude = (longitude_{current} - longitude_{destination}) * \cos(latitude_{current} * 0.017) * 69.17 \tag{3}$$

The vertical distance, i.e. the difference in latitude, is found simply by subtracting the two latitudes and multiplying by 69.17 to find the distance in miles. The horizontal difference, i.e. the difference in longitude, is found by subtracting the two longitudes and then multiplying by the cosine of the starting latitude, and then by multiplying by 69.17.

4 Testing

The only method to test the final project will be to compare it against what a human could do. The program's result in a form such as product delivered over distance traveled would be compared against a human's result in the same form. Also factored in would be the time required for the program and the human to create their results. The means of testing will depend on how "efficiency" is defined. It will likely involve taking amount

of product delivered over distance traveled, with the goal being to maximize that number. Therefore, for a fixed amount of product, most efficient would mean that solution that had the shortest routes.

5 Results

The goal of this project is to create a program to assist volunteer groups and other organizations that need to deliver items but do not maintain standarized fleets of vehicles. Success will mean that it will be easier for these groups to have events that require the delivery of items. From a programming standpoint, success will mean the successful implementation of multiple heuristics and the integration of multiple programs.

References

[1] B. Yu, Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routingnext term problem ", *European Journal of Operational Research*, pp. 171-176, 1 July 2009.

[2] D. Coltorti and A. E. Rizzoli, "Ant Colony Optimization for Real-World Vehicle Routing Problems", *SIGEVOlution*, pp. 2-9, Summer 2007.

[3] X. Gao and L. J. Schulman, "On A Capacitated Multivehicle Routing Problem", *PODC '08*, pp. 175-184, 2008.

[4] M. Geiger, "Genetic Algorithms for multiple objective vehicle routing", *MIC'2001*, pp. 349-352, 2001.

[5] B. A. Julstrom, "Greedy, Genetic, and Greedy Genetic Algorithms for the Quadritic Knapsack Problem", *GECCO '05*, pp. 607-614, 2005.

[6] G. Lamont and M. Russell, "A Genetic Algorithm for Unmanned Aerial Vehicle Routing", *GEECO'05*, pp. 1523-1530, 2005

[7] H. W. Leong and M. Liu, "A Multi-Agent Algorithm for Vehicle Routing Problem with Time Window", *SAC '06*, pp. 106-111, 2006.

[8] J. Tavares, P. Machado, F. Pereira, and E. Costa, "On the Influence of GVR in Vehicle Routing", *SAC 2003*, pp. 753-758, 2003.

[9] S. R. Thangiah, O. Shmygelska, and W. Mennell, "An Agent Architecture for Vehicle Routing Problem", *SAC 2001*, pp. 517-521, 2001.