FINAL REPORT: TRAVELING SALESMAN PROBLEM

KATRINA MATWICHUK, SAMIHA SIMRAN, JOHN HOSKING

Contents

1. Introduction

1.1. **Overview.** Our group investigated the vehicle routing problem under varying conditions. The goal of our study is to find the shortest path(s) for a set of vehicles to deliver to a set of locations.

1.2. **Specific Questions in this Line of Research.** The first type of question that we are interested in is the relationship between the objective value and the number of trucks used in the deliver rout (K), for $K \geq 1$. We want to investigate whether the relationship is a decreasing function conditional on our cost structure (no fixed costs for vehicles). We would also like to examine the result if we relax our assumption of a complete graph, and consider a connected incomplete graph. Additionally, we will consider the effect of using the traditional ILP optimization methods as opposed to Heuristic methods on run-time efficiency.

2. Models

2.1. **Basic Model.** Our basic model is the Traveling Salesman Problem (TSP), which we have based off of the models from Danzig et al. and Gurobi Systems [3] [4] .

Given a set of customers $\mathbf{C} = \{1, ..., n\}$ we build a complete graph $\mathbf{G} = \mathbf{G}(\mathbf{V}, \mathbf{E})$, with $\mathbf{V} = \{0\} \cup \mathbf{C}$ where node $\mathbf{V}_0$ is understood to be the depot. We weight edges with a distance metric, d_{ij}. Our decision variables are

$$x_{ij} = \begin{cases} 1 & \text{if the tour takes the path from location } i \text{ to location } j \\ 0 & \text{otherwise.} \end{cases}$$

for $i, j \in \mathbf{V}$. In addition, $x_{i,j} = x_{j,i}$ since we are not considering a directional graph, thus $d_{ij} = d_{ji}$.
We wish to minimize the distance traveled by the vehicle, thus our utility function is:

$$U(x_{ij}) = \sum_{i,j \in \mathbf{V}} d_{ij} x_{ij}$$

subjected to degree and sub-tour constraints. Our first constraint:

$$\sum_{j \in \mathbf{V}} x_{ij} = 2, \quad \text{for all } i \in \mathbf{V} \tag{1}$$

represents that for every node i there exists only one exit and one entrance. However, with this constraint alone, we could end up with multiple sub-tours in our solution. To eliminate subt-tours we use the constraint:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq 2(|S| - 1); \quad \text{for all } S \subset V \backslash 0; S \notin \emptyset \tag{2}$$

The above constraint forces the number of edges in any sub-tour S to be at most two times the number of nodes in S minus one. This would break the sub-tour and force it to connect with nodes in the rest of the graph. Since we could end up utilizing (2) up to $2^{|\mathbf{V}|}$ times, we employ (2) as a lazy constraint.

2.2. **More Refined Modeling for a K Fleet Greater than 1.** A more advanced model that we considered is the K vehicle Vehicle Routing Problem (VRP), for $K \geq 2$. To do this we used the same model as before, but relax constraint (1) on the depot only, which results in adding the constraint:

$$\sum_{j \in \mathbf{V}} x_{0j} = 2K \tag{3}$$

This relaxed constraint ensures that our "depot" node will only have the number of entrances and the number of exits each equal to K, since $x_{0j} = x_{j0}$.
After some difficulties with coding, we introduced the dummy variable u_i where $1 \leq u_i \leq n$ to track the order of the tour. We then replaced (2) with the constraint

$$u_i - u_j + n(x_{ij}) \leq n - 1 \tag{4}$$

which ensures that there only one connected tour of all nodes.

2.3. **An Incomplete Graph.** Another refinement of our basic model addresses the question concerning the effects of an incomplete graph on our framework.

To create the incomplete graph, using construction data, we systematically constrained paths x_{ij} to equal 0. To do this we superimposed a graph of road closures in Vancouver over a graph of our solution to our basic model. We then eliminated any edges in our solution that crossed construction, by way of the constraint below. Then we ran our more advanced model and checked if any of the paths in the new solution crossed construction. This process was iterated until we obtained a route where no path crosses a road that is under construction. The constraint we added is:

$$\sum_{i,j \in I} x_{ij} = 0 \tag{5}$$

Where I is the list of pairs (r, s) that coincide with edges x_{rs} that have been removed due to construction.

3. Data

Using the City of Vancouvers Open Data Catalogue, we found a dataset of schools with their latitude and longitude (City of Vancouver). We built our distance metric via the Haversine distance of two coordinate-defined points. The Haversine formula [1] is

$$H_{ij} = 2R \arcsin(\sqrt{\sin^2(\frac{Lat_i - Lat_j}{2}) \cos(Lat_i) \cos(Lat_j) \sin^2(\frac{Long_i - Long_j}{2})})$$

For our more advanced model we gathered the locations of road closures in Vancouver ("Road Closures and Construction") and created a data table (see appendix). From the data, we plotted the points of construction alongside our basic optimal solution.

4. Our Algorithms

Our models are all optimized using the Gurobi libraries in Python. Further analysis using Heuristic methods will be implemented in Python as well.

5. Our Main Results

5.1. **Models.** We first tested our basic model using the first five nodes in our data set and found an optimal solution of $18,489.9$m. We then ran our basic model and obtained an optimal solution of $103,476.55$m.

For our advanced model, we found that the total distance increases linearly with K (see Table 1, appendix). We first tested 10 nodes for 2,3 and 4 vehicles, with objective values of $31,043$m, $32,516$m and $36,524$m respectively. The linear relationship is more apparent for higher values of K: the change in objective value for K in $(1, \cdots, 20)$ is almost constant.

The linear relationship is made obvious when inspecting the solution: increasing K beyond 1 leads to additional tours servicing a single customer node. This is due to the lack of a vehicle capacity constraint and customer node demand. In hindsight, it's obvious that if there exists an optimal single tour through a set of points then demanding multiple tours, all originating from a single depot node,

would increase the sum of all tour lengths.

After implementing the additional constraints from the advanced model we found an optimal solution of 105, 394.74m, which is an obvious increase from our basic model. Gurobi took 24 seconds to calculate the solution to our more advanced model. In comparison to the 20 seconds that Gurobi took to calculate the original optimal solution, the advanced model took longer.

5.2. **Heuristics.** For large datasets, using exact solution methods like Branch and Cut to solve the TSP is impractical. Instead, using heuristic algorithms we can find approximate solutions that are computationally feasible. We considered 'Simulated Annealing', a hill-climbing algorithm that uses a stochastic element to 'escape' from local optima. We accomplish this by generating a new solution by randomly switching edges and accepting the new solution with probability P:

$$P = \begin{cases} 1 & \text{if} z_{new} > z_{best} \\ exp(-abs(z_{new} - z_{best})/T) & \text{if} z_{new} < z_{best} \end{cases}$$

With $T = \alpha \cdot t \cdot T_{start}$, where $\alpha \in (0, 1)$, t is the iteration, and T_{start} is the initial temperature. Pseudocode and a python script are attached in appendices, although the solution is not satisfactory.

6. Obstacles Encountered So Far

We had issues with implementing our model, primarily the recursive sub-tour constraints, into python using the Gurobi Library. After some work we were able to get our code to run for our basic model and toy problems, but we still had trouble with our advanced models. This was solved by using the U-variable constraint as opposed to the traditional lazy constraint. However, the U-variable constraint does not scale nearly as well; we have only tested a subset of data (60 nodes) as we have had significant run-time issues for $K \geq 2$ with our entire dataset.

7. Questions for Future Research

We feel that it beyond the scope of our project to implement a graph structure directly from OpenStreetMap (OSM). We also suggest a comparison of various sub-tour elimination constraints (Set-Partition and MTZ). Since the length of route increases when K increases, we suggest that the next step would be to find the exact distance travelled by each vehicle to see if that changes when$K$ changes. We would like to consider different move operators and cooling schedules to fine tune the heuristic algorithm.

References

[1] Anisya and Ganda Yoga Swara "Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route" *2017 IOP Conf. Ser.: Earth Environ. Sci.* 97 012004 http://iopscience.iop.org/article/10.1088/1755-1315/97/1/012004/pdf

[2] City of Vancouver. "Schools Data." *Open Data*. https://data.vancouver.ca/datacatalogue/schools.htm. Accessed Sept 29, 2018.

[3] Dantzig, G., et al. "Solution of a Large-Scale Traveling-Salesman Problem." Journal of the Operations Research Society of America, vol. 2, no. 4, 1954, pp. 393410. JSTOR, JSTOR, www.jstor.org/stable/166695.

[4] "The Traveling Salesman Problem with integer programming and Gurobi." *Gurobi Optimization*. http://examples.gurobi.com/traveling-salesman-problem/

[5] "Road Closures and Construction." City of Vancouver. https://vancouver.ca/streets-transportation/roadwork.aspx Accessed Nov 6, 2018.

[6] von Lindenberg, Kiel (2014) "Comparative Analysis of GPS Data," Undergraduate Journal of Mathematical Modeling: One + Two: Vol. 5: Iss. 2, Article 1. https://scholarcommons.usf.edu/cgi/viewcontent.cgi?referer=https://www.google.com-httpsredir=1article=4855context=ujmm Oct 13, 2018

[7] https://www.mathworks.com/help/gads/how-simulated-annealing-works.html Accessed October 25, 2018