Analytical Business Modelling
Assignment 3

Students:	*Student #:*
Louis Carnec	15204934
Vijay Katta	15202724
Adedayo Adelowokan	15204151

April 28, 2017

Louis Carnec
Vijay Katta
Adedayo Adelowokan

1 Introduction

The Vehicle Routing Problem (VRP) dates to the late 50s, although there has been nearly 60 years of research in this phenomenon, large-scale versions of this problem still pose a challenge for the scientific community.
The Vehicle Routing Problem tries to find a way to visit locations in a graph given a number of vehicles in a cost-effective way. The simplest definition states that every customer is visited by one vehicle, and each vehicle does one trip starting and ending at the depot. The problem involves finding in which order should customers be visited.

2 Application: Goods Delivery

The case study problem we have tackled is the delivery of goods throughout Irish cities and towns from a single depot location based in Dublin given a general time window constraint on delivery times. A single vehicle, with a given carrying capacity, must deliver goods to all towns/nodes in the graph and meet each town's demand for goods, that is deliver an amount of goods equal to that city's demand. Given the fact that each vehicle has a given capacity, the transportation vehicle can only take so many goods and must therefore revert back to the depot to pick up more goods to deliver. Further, deliveries must be conducted within a maximum number of working hours and the vehicle must return to the depot. Therefore, each 'trip', if forced by the time constraint on working hours available, is a working day. Here a 'trip' is a succession of edges connecting nodes that leaves and returns to the depot once.
Our problem involves finding the optimal route for the vehicle so that costs are minimised, where costs incurred stem from the the time spent travelling across the route, the sum of the 'trips'. This is a Vehicle Routing Problem (VRP), a variant of the Travelling Salesman Problem for which the *order* of nodes in a graph to be visited must be optimised to find the tour with lowest travelled distance.
In his chapter on vehicle routing, Cordeau warns that due to high variability of problems in practice, the objective function and constraints of VRP are highly variable and thus must be tailored to each problem [1].

We initially attempted to solve a slightly different version of the problem solved here and modelled in Section 3; the 'Capacitated Vehicle Routing Problem' [4]. In this problem multiple vehicles are used at the same time to deliver goods around the symmetric undirected graph. In our implementation we followed closely the implementation used in section 11.5 of 'Applications of optimization with Xpress-MP' [3] for planning flights. The model is similar to the one presented in this report, distance is minimised given that each node/city must only be visited once. To account for the fact that n trips can be made by n vehicles, an additional constraint is added to stipulate that the depot must have n incoming edges. Optimising the model, we were able to create the subtours for the graph to be 'broken' into n subtours. Having Dublin as the depot, the n closest cities to Dublin were used as outgoing/incoming nodes to be attached to the n subtours.

When it came to making subtour eliminiation procedure into n subtours, we were unable to create a function which would allow us to form the subtour containing the depot to then break the smaller subtours.

3 Related Work

Combinatorial optimization problems have been studied in detail. The Vehicle Routing Problem was initially introduced in 1959 [?]. The needs of the transportation industry were the motivation behind the problem, small improvements in efficiency could result in large economic gains. Routing problems always have an abundance of complex constraints and variables, this is evident in industry and our personal lives. Constraints specific to the industry include shift limits for drivers and specified arrival and delivery times. The widely-studied version of VRPs are often more simplistic, utilising fewer constraints. Simple versions with few nodes and few constraints still take relatively long to run.

3.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the oldest known routing problems. It presents a salesman that has a specified number of cities to visit and needs to know the optimal order to visit these cities to minimise the distance needed to travel.

3.2 Vehicle Routing Problem

VRP is also an NP-hard problem but it is safe to assume that the it iss a much harder problem to solve over TSP. VRP can be applied to various fields; logistics, communications, manufacturing, transportation, to list a few. In a VRP there are several vehicles that need to visit many customers. Vehicles starts at depot before visiting a section of the customers before returning to the starting depot. Capacitated Vehicle Routing Problem (CVRP) is a classic problem, this involves finding a solution to a transport problem where customers can be reached via multiple identical vehicles with different capacity restrictions. A special case of VRP is Vehicle Routing Problem with Time Windows (VRPTW). VRPTW comes with its own added complexity, each customer has a start and end time which indicates the time the vehicle should be servicing. An example of this application would be in case of a company that delivers heating oil. Although this constraint is hard, the vehicle does not need to arrive between that time, it can arrive before the start time but must remain inactive for a given period. Like TSP, VRP is an NP-hard problem [?] but it safe to assume that the VRP is a much harder problem to solve than TSP.

3.3 How similar your problem is well-known solved cases

4 Mathematical Programming Model Formulation

The formulation proposed in this report is an extension of the formulation proposed in the 'Applications of optimization with Xpress-MP' book in section 10.4 [3]. The formulation is extended by; adding a general time window which applies to all cities within the graph (as opposed to individualised time windows) and thus a constraint forcing the vehicle to return to the depot within the time window, an additional constraint to specify the number of times the vehicle can return to the depot and an probability parameter simulating the random probability of being stuck in traffic on a given leg of the trip.

4.1 Assumptions

Several assumptions are made in modelling this VRP problem.
We assume that the time taken to travel from one city to another is linearly proportional to the distance by air between the two points, as such in the mathematical formulation of the model distance, rather than time is minimised. It is also assumed that no time is lost at each node, that is the time taken for the route is equal to the time taken proportional to the distance. However, a fixed cost is incurred at each stop. This assumption means that for a route of length x, the time taken is the same whether there were 5 stops or 10 stops along the route. It is assumed that the distance covered over time does not affect cost. Cost is only related to time spent travelling, plus a fixed cost.

4.2 Objective Function

The objective function (1) we wish to minimise is the cost of delivering goods throughout the graph.

$$\text{Minimise} \quad Cost = \sum_{i \in cities} \sum_{j \in cities, i \neq j} c_{i,j} p_{i,j} \tag{1}$$

Where,

$$c_{i,j} = (d_{i,j}/kmph_{i,j}) * cph + \bar{c}$$

$c_{i,j}$ is the cost of travelling from node i to node j, which is given by the time taken to travel from i to j (distance / speed) times the cost per hour *cph* for the given leg journey, plus a fixed cost for each stop $\bar{c}$.

There are three decision variables in this model, $p_{i,j}$ which states whether node j is a successor to node i, the total quantity at node i (q_i), and the total time spent delivering at node i (t_i).

$$p_{i,j} \in \{0,1\} \quad , \forall i,j \in cities \quad , where \quad i \neq j \tag{2}$$

$p_{i,j}$ in (2) is a binary decision variable which states that the vehicle travels from node i to node j if $p_{i,j} = 1$. The total quantity delivered at i ensures that the quantity delivered is equal to demand for each node and that each trip in the tour is conducted while respecting the maximal capacity of the vehicle. The total time delivering decision variable is similar to the total quantity delivered variable and ensures that the route respects the fact that there is a general time window to be respected for delivery and that the vehicle must return to the depot within this time window. By solving for these variable, the model should return the shortest route which respect to the constraints presented below.

4.3 Constraints

Equations (3) and (4) state that each city (expect from the depot, which is always node 1) must be entered and departed exactly once, so that every city is delivered to but that the tour is not unnecessarily extended by visiting nodes which have already been visited. Note that $i, j \neq 1$ due to the fact that the depot can be entered and departed multiple times. The vehicle should return to the depot once it has delivered all goods within its capacity (C) or worked for a given number of hours (H, see Table 1).

$$\sum_{i \in cities} p_{i,j} = 1 \quad , \forall j \in cities \quad , j \neq 1 \tag{3}$$

$$\sum_{j \in cities} p_{i,j} = 1 \quad , \forall i \in cities \quad , i \neq 1 \tag{4}$$

4.3.1 Increasing Total Quantity

The following constraints are used in finding the optimal tour given a set of constraints while solving for the decision variable total quantity, the total quantity delivered at a given node. The total quantity delivered at a given node (q_i) must be less or equal to the maximal capacity (C) of the delivering vehicle (7) and less or equal to the demand (D_i) / or the amount to be delivered at that node (8). Constraint (7) forces the vehicle to take trips back to the depot if it must deliver a total demand greater then the total capacity of the vehicle.
Since total quantity delivered is a decision variable to be maximised within maximal capacity of the vehicle, constraint (5) ensures that the total quantity delivered if city i is the first city visited *in the trip* is equal to the demand at i, otherwise if i is not the first city delivered to, then the total quantity at i must be less or equal to the maximal capacity of the vehicle (while is already stated in (7)). That is (5), resets the total quantity variable at each trip to the demand of the first node visited within a new trip. Constraint (6) is a strictly increasing total quantity constraint. It states that if node j follows node i, the total quantity delivered at j must be the total quantity delivered at i plus the demand at j. This constraint ensures that at each node successor node the total quantity delivered increases and that each 'trip' is optimised.

$$q_i \leq C + (D_i - C) \times p_{1,i} \quad , \forall i \in cities \quad , i \neq 1 \tag{5}$$

$$q_j \geq q_i + D_i - C + C * p_{i,j} + (C - D_j - D_i) * p_{j,i} \quad , \forall i \in cities \quad , i, j \neq 1 \quad , i \neq j \tag{6}$$

$$q_i <= C \quad , \forall i \quad , i \neq 1 \tag{7}$$

$$q_i >= D_i \quad , \forall i \quad , i \neq 1 \tag{8}$$

4.3.2 Increasing Total Time Constraint

The concept behind the total time of delivery constraints (9), (10) and (11) is the same as that of the (5) and (6) constraints for total quantity delivered. (9) ensures that the total time delivering at i, if i is a successor to the depot (node 1), is equal to the time taken to deliver to that node. Alternatively if i is not a successor to the depot node, the total time of delivery at i is less or equal to the hours available for delivering in a day. Constraint (11) is therefore made redundant in this case. Both constraints ensure that the vehicle returns to the depot within the delivering time limit allocated (before the end of the working day). The strictly increasing delivering time constraint (10) ensures that the total time of delivery at j is equal to the total time of delivery at j, plus the time taken to travel from i to j.

$$T_i \leq H + (t_{1,i} - H) \times p_{1,i} \quad , \forall i \in cities \quad , i \neq 1 \tag{9}$$

$$T_j \geq T_i + t_{i,j} - H + H * p_{i,j} + (H - t_{i,j})p_{i,j} \quad , \forall i \in cities \quad , i, j \neq 1 \quad , i \neq j \tag{10}$$

$$q_i <= H \quad , \forall i \quad , i \neq 1 \tag{11}$$

4.3.3 Number of times through the depot

Lastly, we present an optional constraint which forces the model to return a route solution with a given number of trips, or returns to the depot.

$$\sum_{i \in cities} p_{i,1} = n \tag{12}$$

5 Mosel Model

The Mosel model used in this report is an application of the mathematical model presented above. It was run using both real and generated data. A probabilistic component was added to the travel between nodes/cities.
The city location and population data used to create the symmetric undirected graph of Irish cities was obtained from Tageo[1]. The data was cleaned in `python` and an

[1]`http://www.tageo.com/index-e-ei-cities-IE.htm`

Table 1: List of Variables

Decision Variables	Symbol	Description
Precedes	$p_{i,j}$	Binary Variable - 1 if i precedes j
Total Quantity	q_i	Total Quantity Delivered at j
Total Time	T_i	Total Time Delivering at j

Variables	Symbol	Description
Distance	$d_{i,j}$	Distance from i to j
Speed	$kmph_{i,j}$	Average Constant Speed of Vehicle from i to j
Cost per hour	cph	Cost per hour of delivering
Fixed cost	$\bar{c}$	Fixed Cost per stop
Capacity	C	Capacity of vehicle
Demand	D_i	Demand / Quantity ordered at city i
Hours	H	Maximal Delivering Hours
Travel Time	$t_{i,j}$	Travel time from i to j
Trips	n	Number of Trips

adjacency matrix of the distance ‘by flight’ between cities was calculated using the *vincenty*[2] distance (great-circle distance) from the `geopy` library.
Demand for each city was calculated within `Xpress Mosel` as being proportional to the population of each city with an added random component.
To model the fact that travel time between two cities may be affected by traffic, we added a function which would assign different average speeds to different legs of the route. A node was given an 20% chance of having an average speed of 60km/h and an 80% chance of having a 100km/h average speed.
A printing procedure was created which would output the total distance covered, quantity delivered and number of stop for each ‘trip’.

5.1 The model development process; what facilities/ features does the development environment have to aid model development and solution of your problem?

Xpress Mosel, the development environment used in conducting this report, allowed us, the users, to easily check the status of our model which was beneficial in developing and finding a solution to our model.
The info bar, when an error occurs, outputs a range of errors in the code, this allowed us to quickly and easily debug our code. By right clicking on a variable within the text and showing values assigned to the variable, we were able to quickly check that each variable was correctly assigned, this was useful when working with a dummy data while building the model. The ‘Run Bar’ feature, to the right of the screen allowed us to check

[2]`https://geopy.readthedocs.io/en/1.10.0/`

the solutions being produced when running the model by navigating to the ‘Solutions’ tab. The ‘Stats’ tab enabled us to check the status of the program without having to print it out. Finally the ‘Row View’ feature within the ‘Matrix’ tab allowed to test the effect of different vehicle capacity and time capacity parameters on the slack variables for each constraint.

5.2 How easy is it to verify correctness of the model and to separate the problem and its data

Verifying the correctness of the model was relatively simple. We built the model in `Xpress Mosel` using a dummy dataset containing a 5-by-5 adjacency matrix. Examining the solutions in ‘solution’ tab within Xpress, we were able to verify that each of the model’s constraints were being respected and that indeed for the small dummy dataset the result of the minimised objective function was as expected.
Applying the Mosel model to our application of choice, after initialising the datafile containing the adjacency matrix for the distances of Irish cities, although the model itself remained the same, some of the model’s parameters had to be altered to fit the new data in order for the model to output a result, otherwise the model would keep running. Constraints (5, 6, 9 and 10), depend on the capacity parameter and on the time available parameter of the vehicle. For some values of these parameters with a given dataset, the model will be infeasible. Additionally, the optional constraint for the number of times the vehicle must pass by the depot must be feasible. For example, if demand at city 1 is 400 units and the capacity of the vehicle is 300 units, the problem will be infeasible and Mosel will not find a solution. The capacity of the vehicle needs to be increased for the model to function. Likewise, if the speed of the vehicle is 20 km/h, the working hours of delivery are of 8 hours, the maximum distance covered in the route is 160 km.
In order to make the program more versatile in implementing with different datasets, we would have like to have created if-loops which would raise an error in case the parameters provided do not allow for a feasible solution to be found. If the program did not stop, this was generally due to a parameter making the problem infeasible. In this case we had to use the ‘Row View’ tab within the environment to inspect each constraint in order to debug the program.
This versatility enables us to solve different problems. For example, we could solve a VRP, for picking up broken Dublin City Bikes from the various stations around Dublin. We would download the locations of the stations, create an adjacency matrix using the same `python` program used in creating the Irish cities adjacency matrix and solve for the quickest route.

5.3 How easy is it to either perform sensitivity analysis on the defined problem or to amend/extend the problem?

Amending and/or extending the problem presented here is feasible depending on the changes/extensions the user seeks to make. We have demonstrated that the model can be extended easily, by stipulating some additional constraints such as the number of times the vehicle must return to the depot and respecting time windows are easy. This is possible by altering/adding some constraints.

Further, the model's parameters can easily be altered; such as the maximal capacity of the vehicle, the number of hours in the working day, the average speed of the vehicle, the probability of the time to go from one node to another increasing or decreasing.

However, for some changes/extensions, we would need a different model. If we wanted to change the model to use a number of n vehicles to find n tour for each vehicle so that each city is visited once, we would need to change the subtour elimination constraint completely.

Further, if we were to extend the model to a case study with many more nodes, we would need to adopt a different formulation such as tree search or a heuristic approach [3]. The model we are currently using would work for instances of up to 30 nodes only. Therefore, the Dublin Bikes case study proposed ion the previous section would be infeasible as there are 102 stations.

5.4 What theoretical principles are demonstrated in your application?

The theoretical principles demonstrated in our application come from *network optimisation* and *stochastic optimization.* The distance between cities, produced in the form of an adjacency matrix, represent a graph of cities where the nodes are cities and edge weights are the distances between them. Using combinatorial optimisation, the optimal set of nodes in a specific order are returned while respecting a set of constraints. At the same time this application is a stochastic optimisation problem as there is the presence of randomness in both the demand and speed variables, that is the result of the optimisation, the total cost of the route, should vary across runs of the model as it is affected by randomness. Using a stochastic optimisation model of this type allows us to more realistically *simulate* the cost of a route. Running the model a number of times and getting a set of statistics which represent the results would enable us, the analysts, in making recommendations on best route to choose to minimise cost.

6 Results

random increase in cost

6.1 Tour length: mean, median,std

6.2 Time to find for n=?: mean, median, std

7 Conlusion/Recommendations

7.1 The dependence on Software

Whether you agree with the statement above : In OR practice and research, software is fundamental. The dependence of OR on software implies that the ways in which software is developed, managed, and distributed can have a significant impact on the field.

Louis Carnec
Vijay Katta
Adedayo Adelowokan

Appendix - Results

Authorship

Adedayo Adelowokan Contribution: 1, 3, 7

Louis Carnec Contribution: Data Generation in python, 50% of Mosel Implementation, sections; 2,4,5

Vijay Katta Contribution: 50% of Mosel implementation, generating results, sections; 6

Louis Carnec
Vijay Katta
Adedayo Adelowokan

References

[1] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. Vehicle routing. *Handbooks in operations research and management science 14* (2007), 367–428.

[2] El-Sherbeny, N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University-Science 22*, 3 (2010), 123–131.

[3] Guéret, C., Prins, C., and Sevaux, M. Applications of optimization with xpress-mp. *contract* (1999), 00034.

[4] Toth, P., and Vigo, D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics 123*, 1 (2002), 487–512.