Strategic Upgrades

A vehicle routing overview and implementation for Nokia Siemens Network

Course Title: Operations Research
Course Number: ETM 540
Instructor: Dr. Tim Anderson
Term: Fall
Year: 2013
Author(s): Lukas Kessler, Chris Davis, Farzad Moshfegh, Jerrod Thomas

ETM OFFICE USE ONLY
Report No.:
Type: Student Project
Note:

Abstract

The Traveling Salesman Problem (TSP) is combinatorial optimization problem that has real-world synonyms for many organizations and problem types [1]. In this paper we discuss various implementations of the Vehicle Routing Problem (VRP), an extension of the traditional traveling salesman problem and explore a best-fit solution as well as a heuristically derived "good" solution. The best-fit solution utilizes a branch and bound technique but requires an exponential amount of computational resources to complete when applied to problems of marginal size. Some heuristic methodologies are elucidated to compare and contrast their benefits and drawbacks and an open source solution for capacitated vehicle routing that utilizes a tabu search heuristic is chosen for a pilot data. Next we compare the results of small batch runs between our best-fit and "good" solutions, exploring projections based on our real-world output. We conclude with our routing solution pilot data and recommendations for utilization by Nokia Siemens Networks as part of their ongoing equipment upgrade and deployment tool set.

Table of Contents

1 Introduction

In order to properly frame the requirements needed by Nokia Siemens Networks, we will review the basics of the traveling salesman problem, its formulation, and various techniques on solving this exigent problem in computer science and operational research. We begin with the background information and problem formulation.

1.1 Company Information

Nokia Siemens Networks (NSN) is one of the major vendors for telecommunication operations in Iran and within the last decade, it has been participating in various segments within the telecommunication industry. One of the main customers of NSN is MTN Irancell, the second largest mobile phone network in Iran. NSN provides infrastructure for Irancell and has the main share of the projects around the country; from acquisition and simple drive test projects to upgrading to the most sophisticated cellular equipment. Tehran, the capital of Iran, is where some of the main clusters of NSN projects are located, which are the most sensitive projects in the region. Because of this sensitivity, providing the best service in the minimum possible time is one of the primary objectives of NSN in Iran.

1.2 Problem Description

Nokia Siemens Networks in Tehran, Iran is looking for assistance in optimizing their base transceiver station upgrade project and would like a reproducible model for scheduling, deploying, and routing their upgrade crews and vehicles as efficiently as possible. The current method involves routing trucks based on ad-hoc decisions of location and type of installation, but the management has noticed significant errors in either over-scheduling or under scheduling

resulting in more slack and overall project timeline slippage than desired. They would like to have a system that can be used to input a large dataset of distances or travel times, equipment upgrade time estimates. Its output should be a route and schedule that they can assign to their crews each day. In many cases, even the best estimates will be faulty and the possibility of rerunning the remaining upgrade sites will be required as the project progresses. As such, any solution should be easy to modify and re-run at scale in a reasonable amount of time; i.e. in less than 30 minutes.

2 Literature Review

As exposed in the previous chapter, the concept of the planning the routes or more commonly "tours" is an important aspect of the task. For the planning of the tours one must first distribute the towers to each planned day and then the teams to each tour which determines the order of towers they visit within a day. These problems can be identified as classical problems of the tour from the field of Operations Research. Through a literature review in the following sections we will define a model of the Traveling Salesman Problem and the Vehicle Routing Problem which will be worked out.

2.1 Traveling Salesman Problem

The background of the Traveling Salesman Problem (TSP) is that a traveling agent needs to plan a trip that begins at a starting point and he is required to visit a number of "cities" or locations, exactly once each, and then return to the starting location (see Fig. 1).

The problem of the TSP was probably first mentioned in 1831 in an article by B. F. Voigt. The vernacular of the traveling salesman is not mentioned used, however the idea itself is described [2]. “Traveling Salesman Problem” was first mentioned in the 1930s, though it is unclear where the actual attribution should be given to [3]. The TSP is one of the most basic, important, and studied problems in combinatorial optimization [4], its importance mainly due to its application in solving higher-level, practical problems [5]. An example when the TSP model is used is to determine the routes of the individual vehicles in the vehicle routing problem. Another example is the sequence determination of orders that need to be done on a machine [6]. The TSP is a NP-complete[1] problem [7]. The fact that it is difficult to solve, and at the same time easy to formulate, is responsible for the numerous studies on the TSP [8].

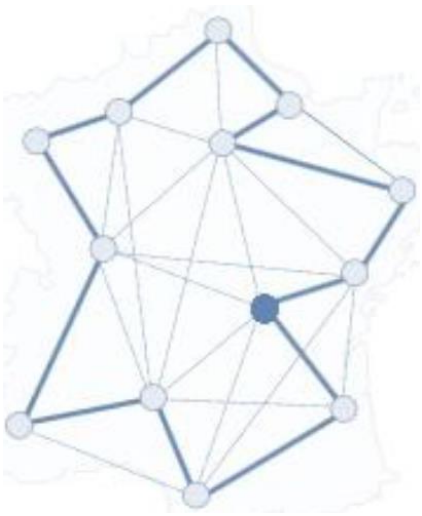

Fig. 1: Graphical representation of the TSP [34]

From a graph-theoretical point of view, the TSP can be described as a directed graph G = (V, A) [5], [9]. The node set V = {1, ..., m} is the set of all the places to visit with m as the number of places. The quantity of arcs A = {(i, j): i, j ε V and $i \neq j$} is the set of connections between the locations. With (c_{ij}), i, j ε V, $i \neq j$, it is referred to the distance matrix wherein each arc (i, j) is assigned to a distance c_{ij}. When solving the TSP, it is looked for a cycle which contains each node exactly once (so-called Hamiltonian cycle) [9]. The above applies to the so-called asymmetric TSP thus $c_{ij} \neq c_{ji}$. A special case of the asymmetric TSP is the symmetric one, thus $c_{ij} = c_{ji}$. Here the undirected graph G = (V, E) is considered.

[1] Attribute of (optimization) problems, such problems are probably not efficiently resolvable in deterministic polynomial time [7].

E is the set of edge $\{\{i, j\}: i, j \varepsilon V, j \neq i\}$ [5]. There are various models for the classical TSP[2], in the following the advanced mapping model will be introduced, as it is the most commonly used one. In the equations below $\Delta+(i)$ or $\Delta-(i)$ describe the set of all successors or predecessors of i. x_{ij} is the binary decision variable that is one if the place j is visited immediately after i [5].

The objective function of the model is shown in equation (1), in which the cumulative distance should be minimized.

$$min\, z = \sum_{(i,j)\in A} c_{ij} * x_{ij} \qquad (1)$$

Equation (2) and (3) guarantee that every place is exactly visited once and then left [5]:

$$\sum_{j \in \Delta+(i)} x_{ij} = 1 \qquad (2)$$

$$\sum_{i \in \Delta-(j)} x_{ij} = 1 \qquad (3)$$

The decision variables x_{ij} are binary [L5]:

$$x_{ij} \in \{0,1\} \qquad (4)$$

Moreover subtours[3] have to be forbidden for which different options exist [5], [10]:

1. Explicit banning subtour by subtour due to elimination conditions
2. The MTZ formulation by Miller, Tucker and Zemlin
3. The multi-commodity flow model

At this point, the MTZ formulation is exemplified. For this formulation the auxiliary variables u_i and u_j have to be introduced: $u_i, u_j \epsilon V \setminus \{1\}$ [5] [11]. According to the MTZ formulation (m-1)*(m-2) subtour elimination conditions are required.

$$u_i - u_j + m*x_{ij} \leq m-1 \;\forall\; (i,j) \in A,\; i, j \neq 1 \qquad (5)$$

[2] Examples are: mapping model, 2-Matching-model (only for the symmetrical TSP), 1-tree-model (only for the symmetrical TSP), square mapping model, 2-Warenfluß-modell, m-way model

[3] Subtours are tours which to not cover all places in the set of nodes V, thus they are not connected to the origin and form degenerate tours between intermediate nodes [35]

2.2 Vehicle Routing Problem

After the description of the TSP in the following the the so-called vehicle routing problem (VRP) will be discussed. The VRP was first studied by Dantzig and Ramser in 1959, to solve the problem of "Truck Dispatching" [12]. In the followong 50 years, particularly in the 1980s, the VRP was further developed and many variants emerged, which are tailored to specific problems [13]. With further development of technical innovations, such as the location via GPS, the VRP nowadays plays an important role in the logistic [14].

Similar to the TSP the VRP deals with the problem that a given amount of places, hereinafter referred to as customers, has to be visited from a depot. However, in the VRP the customers are not served in a single tour. Every customer has non-negative demand of d_i. However, the vehicle k (it is also possible that a number of vehicles is available; $k \in V = \{1,..., k\}$) has a limited capacity of q. Thus, the cumulative demand of all customers may exceed the capacity of a vehicle. Consequently, several tours have to be assigned. Thus, customers have to be assigned to routes and the order of every single route in which they should be served has to be determined [15], [16]. Figure 2 points out the idea of the VRP.

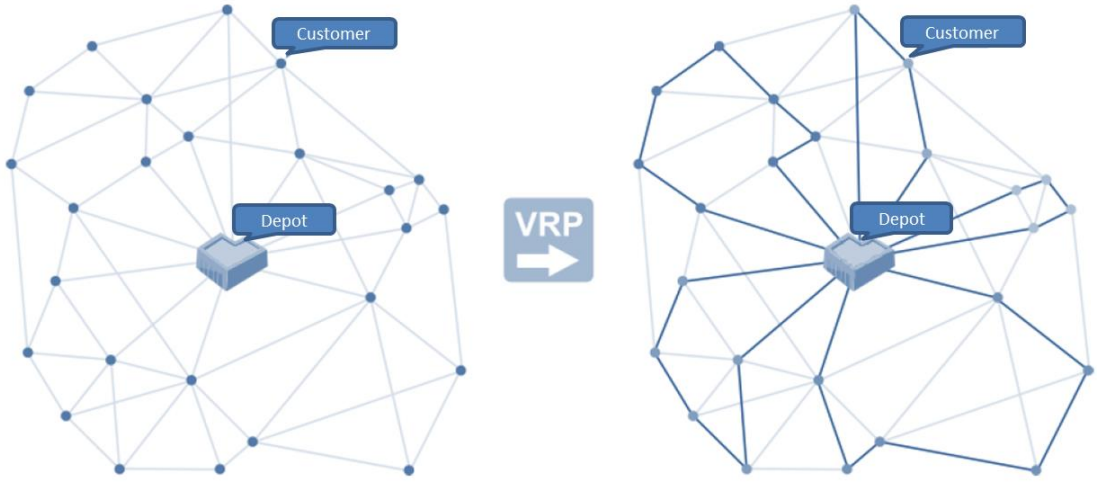

Fig. 2: Graphical representation of the VRP [11]

Like the TSP the VRP can be described as a directed graph G = (N, A). N = {0} ∪ C ∪ {n+1} represents the set of customers, while 0 and n+1 stand for the depot [17]. Thus, another set of customers is needed: C = {1,…,n}, which represents the set of customers without the depot [18]. Like in the TSP the arcs are represented by A (A ⊆ NxN). Also c_{ij}; i,j ∈ N; i ≠ j represent travel costs for each arc. The objective is to minimize the total distance [5], [16], [17], [18]:

$$min\ z = \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} c_{ij} * x_{ijk} \qquad (6)$$

x_{ijk} a binary decision variable that is one if vehicle k drives from customer i to customer j ($x_{ijk} \in$ {0,1}, ∀ (i,j) ∈ A, ∀ k ∈ V). Various constraints are needed to complete the model. First, it has to be ensured that all customers will be served:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1,\ \forall i \in C \qquad (7)$$

Second, an equation is needed that the sum of the demand of the customers cannot exceed the capacity of the vehicle:

$$\sum_{i=C} \sum_{j=C} d_i * x_{ijk} \leq q,\ \forall k \in V;\ i \neq j \qquad (8)$$

Additionally, only one route is possible for one vehicle and it starts at the depot. It is possible that the tour is empty, thus the vehicle drives from 0 to n+1.

$$\sum_{j \in N \setminus \{0\}} x_{0jk} = 1,\ \forall k \in V \qquad (9)$$

Moreover, the decision variables have to be binary:

$$x_{ijk} \in \{0,1\},\ \forall (i,j) \in A,\ \forall k \in V \qquad (10)$$

The tours have to end at the depot:

$$\sum_{i \in N} x_{ijk} = 1,\ j = n+1,\ \forall k \in V \qquad (11)$$

Next, the vehicles arrive and depart from the customers:

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0,\ \forall h \in C;\ \forall k \in V \qquad (12)$$

In analogy to the TSP subtours also have to be eliminated, n is the maximum number of nodes any vehicle k can visit:

$$u_i - u_j + n*x_{ijk} \leq n-1 \quad \forall (i,j) \in A, i, j \neq 1, \forall k \in V \quad (13)$$

3 Techniques to solve the problem

There are a few different methods that will help to find a solution for vehicle routing. The method will be selected based on its flexibility, simplicity, accuracy and speed. The primary objective that needs to be analyzed, is to minimize the amount of time it takes for a tower maintenance team to upgrade as many towers in a specific region. In the following one will first shed light on exact solution techniques and second on heuristics.

3.1 Exact Solutions

Basically, for these problems an optimal route can be found. However, such a solution proves to very extensive and complex for larger problems [5]. In the following the basic ideas of two methods, namely, the branch-and-bound and the branch-and-cut method are presented. The VRP is a NP-hard problem, which consists of determining optimal routes that covers the demands of the customer without using too many vehicles [19]. The branch and bound method is a way of finding an exact solution to the VRP. The branch and cut method is another way of determining the optimal solution.

3.1.1 Branch and Bound Technique

The branch and bound method of solving VRP is a common method to use when dealing with a larger VRP sometimes with multiple depots. A branch and bound algorithm uses a divide and conquer method to partition the solution space into sub-problems and then optimizes individually

over each sub-problem [20]. To continue the algorithm for this method one of the candidate sub-problems will be selected and processed. There are four possible results. In the processing or bounding phase the problem set is relaxed. When that is done solutions that are not in the feasible region are admitted. Solving the relaxation will yield a lower bound on the value of an optimal solution. If a feasible solution is better than the previous one found then the new solution will replace that solution and the process continues. If there is a point where the sub-problem can no longer be discarded or pruned, then the problem is branched and the children of the sub-problem is added to the list of active candidates. This process is continued until the list of active candidate sub-problems is empty; at which point the current best solution is optimal.

3.1.2 Branch and cut Technique

Branch and cut method is a combination of cutting plane algorithm and branch and bound procedure. Branch, cut and price (BCP) is an LP-based branch and bound technique for solving large-scale discrete optimization problems. In BCP, both cuts and variables can be generated dynamically throughout the search tree. A well-known example of application of cutting plane method has been developed in 1958 by Gomory [21]. The starting situation is generally the dissolved LP relaxation of the original problem, followed by a separation problem [22]. Thus new inequalities are generated that the feasible region of the original problem is reduced (by non-integer and therefore not allowable solutions are forbidden). This is followed by a re-optimization of the LP relaxation using the dual simplex method [23]. This procedure is repeated until either the relaxation of the original problem is admissible or no-injured section planes longer exist. The branch-and-cut algorithm combines the two mentioned methods in a way that is every node of the branch-and-bound method the feasible region is narrowed down with the cutting plane method. Thus it can efficiently find the optimal solution.

Besides the above Branch techniques, it should finally be mentioned that explicit enumeration (i.e. finding a solution due to “try-and-error”) can provide an exact solution for routing problems.

3.2 Heuristic Solutions

Heuristic methods perform a relatively limited exploration of the search space and typically produce good quality solutions within modest computing times. The heuristics, which have been created for VRP, are general modifications of "traveling salesman problem" heuristics [24]. They are of 4 different types: 1) tour building heuristics, 2) tour improvement heuristics, 3) two-phase methods, and 4) incomplete optimization methods.

The incomplete optimization method is the more commonly used method and will be the method that is most closely resembled for the purpose of this project in assigning the entire population of the cell tower improvement fleet to their respective routing maps using distance as the overall objective to optimize. Incomplete optimization methods apply some optimization algorithm, such as branch and bound. Over the years heuristic methods that are applied to the VRP have been modified and new ones have been created. The basic methods mentioned above are still the more commonly used methods. The quest for faster heuristics has been going on since the beginning of computerized solutions to VRPs, but developments are still taking place and will continue to evolve over time to create the fastest most effective method to solving the VRP [25].

3.2.1 Evolutionary or Genetic Algorithms

The evolutionary or genetic algorithm (GA) is an adaptive heuristic search method based on population genetics [26]. The creation of a new generation within the algorithm involves four fundamental steps or phases: representation, selection, recombination, and mutation. The four

phases can be described as such: representation is the population of the decision variable that will be evaluated in the algorithm; then selection is a way of evaluating each solution in the population and selecting one of the parent solutions. Recombination is a way of applying crossover and mutation operators to parent solutions to generate offspring solutions and the mutation phase is replacing the old population with the new population of offspring solutions [27]. This process is repeated for a number of iterations or until the system does not improve anymore. A simple GA can be created as follows:

1. Create an initial population of P solutions. In the context of this project for upgrading cell phone towers the population will be created with sub-groups of the overall population of cell towers, divided into smaller districts that each upgrading team will have on their route.
2. Evaluate each solution.
3. Repeat for a fixed number of iterations:

    3.1. Repeat until P offspring solutions are created:

        3.1.1. Select two parent solutions in the population (with replacement) using a randomized selection procedure based on the solution values.

        3.1.2. Apply crossover to the two parent solutions (with certain probability) to create two offspring solutions. If crossover is not applied, the offspring are identical to the parents.

        3.1.3. Apply mutation (with certain probability) to each offspring.

        3.1.4. Insert the two offspring in the new population.

    3.2. Evaluate each offspring in the new population.
4. Return the best solution found.

The initial population for step 1 can be created randomly, but it is better to use construction heuristics to create at minimum a small fraction of the population.

3.2.2 Greedy insertion

Greedy insertion or greedy algorithms in general are simplified techniques for finding the optimal solution. In a since, the GRG Nonlinear solver is a greedy solver in that it always picks the best answer adjacent to its present location. Though this solution technique is generally quick for locating solutions, it also tends to get caught with local optima, or to make missteps in node or adjacency maps. We talk about it briefly here, as most heuristic and meta-heuristic techniques require a starting model to create a solution to begin the exploration of the problem area. A simplified illustration of a greedy search is shown in Fig. 3 that demonstrates how a greedy algorithm can easily incorrectly assess a maximization problem attempting addition over a node structure.

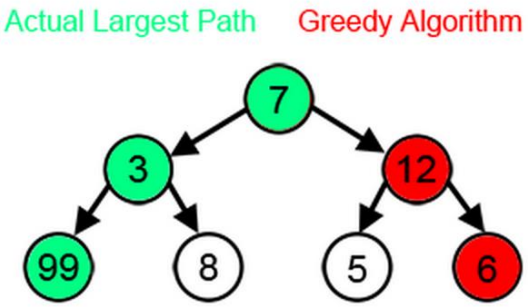

Fig. 3: Greedy algorithm vs. actual largest path (maximization error) [36]

3.2.3 Tabu Search

The Tabu search is a meta-heuristic class of search techniques that uses adaptive memory and responsive exploration. These search techniques "are capable of searching the solution space economically and effectively" [1] and it is because of this that many researchers have tested and adapted the technique with some success; particularly for solving various VRP formulations [28], [29]. The tabu search method for solving a vehicle routing problem cannot guarantee an optimal solution, but does take an efficiency approach to looking for optimal, likely local optima if the

problem set is large. Because it uses a memory-based methodology, tabu search can utilize a strategy-based technique with the "supposition that a bad strategic choice can yield more information than a good random choice [1]. The algorithm considers a sequence of adjacent solutions obtained by repeatedly removing a vertex from its current route and reinserting it into another route [29]. To avoid cycling solutions throughout tabu search's iterative process, some attributes of recently explored solutions are temporarily declared tabu or forbidden. The duration that an attribute remains tabu is called a tabu-tenure. It can vary over different intervals of time and the tabu status can be overridden if certain conditions are met. Various techniques are often employed to diversify or intensify the search process; this relates to intensifying the search around the "elite" solutions, or diversification to invoke a solution that might break out of situations where the search appears to be stuck at a local optima.

4 Methodology

The project team quickly realized that implementing of a solution within Excel or utilizing any technique described within the materials of our course was infeasible for the size of our pilot data set: 35 nodes or base transceiver stations, which have to be upgraded. We began exploring various options available and ultimately came across an open-source implementation of a tabu search algorithm called Open-VRP written in Common LISP.

With our initial formulation of the VRP model we received some invaluable aid from our graduate teaching assistant, Dong-Joon Lim–an expert in operation research and a proficient programmer in The R Project statistical programming environment. He was able to implement the model in R Project and began running tests of our pilot data with approximately four weeks remaining in our ten-week term. This provided us two avenues to test our VRP and look for a feasible solution. In this section we will discuss the two implementations and their outcomes.

4.1 Implementation in The R Project for Statistical Computing

R is a free programming language for statistical computing and statistical graphics. R is part of the GNU Project and is available on many platforms. It is increasingly seen as the statistical standard language in both the commercial as well as in the scientific field and was developed in 1992 by Ihaka and Gentleman at the University of Auckland [30]. Due to the implementation of the provided model in section 2.2 the problem could be solved with a Branch and Bound algorithm (see sec. 2.2). Thus, in our case "n" represents the set of towers, which can be thought of as the customers. The travel costs c_{ij} are measured in minutes between the towers, hence not the distance is minimized, but instead it is the time needed to serve all towers and the travel time needed. Moreover, d_i represents the demand from each tower. Thus, the time needed to update the tower. The capacity, or available time for each team is also measured in minutes totaling 480 minutes, or one eight-hour day. The result would be optimal routes shown as different clusters of each tour for each team, for each day. In our initial runs, Dong-Joon tested the code utilizing 3-5 tower locations. After successfully defining a route, he then started the algorithm at work on our full problem set of 35 nodes.

Table 1: Actual Runtimes of R Model. Average of five runs for n = {3-12}.

n	*Average runtime (sec)*
3	0.002
4	0.006
5	0.012
6	0.018
7	0.036
8	0.084
9	3.244
10	19.966
11	23.734
12	867.502

After a week, we still did not have an answer. After some back-of-the-napkin calculations we started to realize that it was unlikely that our full suite of locations could be calculated in a reasonable time using our formulation and we decided to estimate how long it would take to calculate the optimal solution by calculating the average running times with lower-values of n. Dong-Joon then moved

the full problem set to a much faster research-designed compute server as well as running timing cases with values for n = {3…12} as shown in Table 1. With this information in hand, we decided to model the trend of the average runtime.

The model takes the form:

$$y = a + b * x^c. \quad (14)$$

Whereas d_n is defined as:

$$d_n(a,b,c) = (a + b * {x_n}^c) - y_n \quad (15)$$

To measure how good the model fits the sum of squared differences will be employed [L19]:

$$\min z = \sum_n [d_n(a,b,c)]^2 \quad (16)$$

No further constraints are needed, and the model can be solved with the GRG Nonlinear Solving method provided by Microsoft Excel. However, as the model is non-linear and the GRG-algorithm does not necessarily provides a global optimum so we enhanced the model with the following constraints to make it possible to use multiple starting points:

$$-100 \leq a \leq 1000 \quad (17)$$

$$-10 \leq b \leq 10 \quad (18)$$

$$0 \leq c \leq 20 \quad (19)$$

With the usage of 10000 starting points, the message from solver stating “likely converged to an global optima” and the knowledge that the objective function is convex and should be minimized, it can be stated that the GRG-algorithm has found a global optimum [31]. The solution is as followed: a = -37,2374949894347; b = $2{,}67449368825694 * 10^{-8}$; c = 9,65842295272406. Figure 3 compares the function of the predicted results with the actual ones:

Also it can be argued that the estimation is a very conservative one, as for n = 13 there is no y-value there after several weeks of calculation time. With this result the computing time for n = 35 can be calculated as:

$$y_n = y_{35} = (a + b * {x_{35}}^c) \text{ [sec]}$$

$$= (-37{,}2374949894347 + 2{,}67449368825694 * 10 - 8 * 35)^{9{,}65842295272406} \text{ [sec]}$$

$$= 21902973{,}57 \text{ [sec]} \triangleq 253.5 \text{ [days]}$$

Therefore, the usage of the branch and bound algorithm within The R Project as an everyday tool to allocate routes is not feasible.

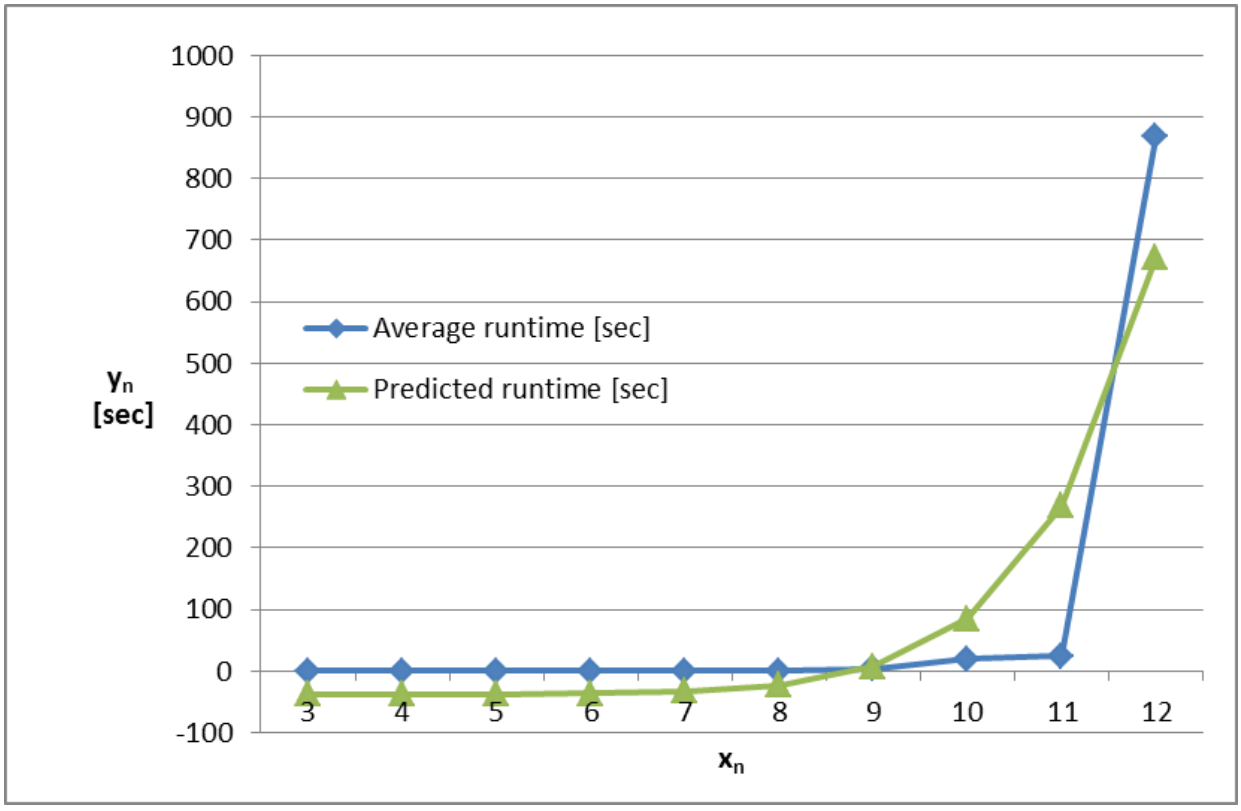

Figure 4: Comparison of the predicted and average runtime

4.2 Tabu Search using Open VRP

With none of our team members computer scientists, and not wishing to impose more requests upon our generous and supportive teaching assistant we located an already compiled solution for running an interesting heuristic approach for the Vehicle Routing Problem, Open-VRP. The software's author designed the implementation in LISP to be a framework that others could use to implement any algorithm of their choosing. The project has shown some positive feedback, mostly from students who appear to be utilizing it in similar endeavors for academic pursuits. The author uses a tabu search implementation that uses first the Greedy-best insertion technique to quickly locate a feasible solution. It then iterates through a very basic neighborhood structure using a function titled "best-insertion." The author notes that this minimalistic neighborhood structure is "works pretty well in most times, but might not be strong enough in some cases to overcome local optima." [32] Recommendations to use multi-run, force additional iterations, modify the tabu-tenure, or to design your own adaptive routines for dealing with apparently stuck optimality have to be considered.

Open-VRP does implement the namesake tabu list of tabu moves, a table of elite neighborhoods for faster computational comparison, and the aspiration feature noted as a means to clear best-yet solutions from the tabu list.

4.3 Pilot data set

Data for this project is quite basic and was collected by one of our team members with contacts from Nokia Siemens Networks. Our initial data set consisted of a set of latitude and longitude coordinates detailing the exact locations of the 35 base transceiver stations servicing the northern area of Tehran along with the location of the service depot, which operates as the dispatch point

for all upgrade crews. We were also given average time estimates for specific upgrades on specific towers. We normalized this data to be in units of time equal to minutes. As a capacitated VRP, with all requirements in time, we can schedule and plot our tours based on minutes-per-truck equal to an 8-hour day, or 480 minutes per truck. This accounts for the travel time and the time-to-upgrade each tower (see Appendix A). Data for this table was gathered with Google Earth using the "Directions" functionality from each node to every other node. Only half of the table was required to be filled out and the data mirrored, thus it is a symmetric problem (see sec. 2.1).

This data could be used by R but once we had investigated the Open-VRP tool further we discovered that issues importing a TSPLIB95 [33] file that utilized a time or distance-based matrix. To resolve this we massaged our data by converting latitude/longitude values to UTM, which provided relative offsets, measured in meters for all coordinates. From here, we normalized the values to the lowest common denominator for each east and north value and input the normalized X, Y meter coordinates for the Cartesian straight-line distance calculation provided by Open-VRP. This is sub-optimal but after significant trial and error, was the only method we could produce using our pilot test results. Given that the distances are relatively short in this pilot phase, the main concern is scheduling the proper tower assignment or capacity and secondarily the efficiency of the vehicle routes; the Open-VRP solutions generated fulfilled these criteria.

5 Analysis

After our initial implementations, we know enough to make a few recommendations about our two options and how Nokia Siemens Network might utilize them in planning system upgrades around the Tehran region.

5.1 Comparing criteria

As noted previously, we were looking for solutions that met several criteria: flexibility, simplicity, accuracy and speed.

5.1.1 Flexibility

Flexibility-wise, Open-VRP has a lot of promise as it was initially designed as a framework for solving many VRP types of scenarios. It is capable of handling two common files types for VRP problem sets, it can operate on symmetric and asymmetric VRP problem, and can account for simple VRP, capacitated VRP, time-window VRP, and capacitated time-window VRP. With these myriad of options, it is possible to leverage the tool in quite a few ways and offers many options in tuning the output. You can adjust tabu-tenure to your problem set, do multi-start runs, and choose to iterate more if you are unhappy with a given solution. If Cartesian math is offered, it also provides a nice plotting function to provide a graphical representation of the routed solution.

Unfortunately, it also appears to have been a project without a following. Marc Kuo made some significant progress and got the code to a stable release but few modifications have been made to it since last year. The last commit against the code base was minor and 2 months ago; all other commits were over a year ago. Compared to the branch and bound implementation by Dong-Joon Lim, the Open-VRP solution has the potential to be more flexible, but its complexity belies its potential power. With more time and effort, a solution based on R Project would likely be more flexible, though that is outside the scope of our discussion and capabilities.

5.1.2 Simplicity

The Open-VRP solution is fairly simple. It took a little work to find a suitable Common LISP environment and get the proper libraries installed but operating the software once all the pieces were in place was not challenging. The use of branch and bound in R was simple, but without an expert in the R language available, modifications will be difficult. With the current R code, you feed the software the proper data inputs and it churns away until it provides an answer as a grid of tours. Because there are many more options with the Open-VRP solution, and more pitfalls in working out what format it can and can not accept, the R implementation is easier to work with overall. The caveat to this lies in the fact that an expert was available who hand-coded the implementation and assisted with its operation. It is also designed as a single-solution implementation whereas Open-VRP was designed as a framework that additional algorithms can be implemented and compared.

5.1.3 Accuracy

Accuracy of a simplex abd branch and bound model, if no errors are introduced, will produce the global optimal solution. Because of this, accuracy cannot be exceeded and any heuristic or meta-heuristic technique can only get lucky in finding a global optima.

Open-VRP on the other hand does not always provide the global optima. The provided results are generally good but after running it many times, it has a tendency to require a multiple runs to find some level of agreement on what a best solution might be. With a deeper investigation into the platform, and if we could have figured out the multi-start feature, we expect that the average runtime would have gone up but that we would have also seen an improvement on solutions with less fiddling and re-running of our problem set.

5.1.4 Speed

The counterpoint of this is that the time to find an optimal solution in an NP-hard problem such as the traveling salesman problem is that it can take an unrealistic time to compute this solution (see sec. 4.1). In our case, the optimal solution is still being calculated and our model used to estimate the time to completion would take us from Winter 2013 into late summer 2014 waiting for an answer.

With the tabu search implementation in Open-VRP, answers are delivered quickly. The longest run we experienced was just over 30 seconds with most runs lasting 1-5 seconds. With this kind of speed, the requirement of being able to generate new route plans as needed is most certainly fulfilled.

5.2 Implementation of pilot data

Given the implementation in R Project has a time-horizon of nearly a year, it is easy to describe the results as: still running. This fails so miserably at one of our primary criteria of speed that it is going to be impossible to recommend this solution for any organization needing route-planning assistance unless the number of nodes is quite small. In our case, NSN is looking to pilot a test routing solution that could be used for hundreds or possibly thousands of base transceiver sites over time. Most likely they will work with smaller pockets at a time, prioritizing their routes and re-running as the capacity calculations for time-to-upgrade are basic estimates, but they do plan to utilize the tool to ascertain how many vehicles (crews) it will require to compete segments of the project within a given timeframe.

After adjusting the Open-VRP installation and providing it a workable X, Y coordinate dataset, we were able to run, re-run, and tune for a variety of outputs. Since the solution is based on an iterative approach with a random, "Greedy" initial insertion; almost all answer will be different. Again, with a problem such as this, it is not reasonable to search out *the* optimal solution but rather locate a good solution in a reasonable amount of time. When using a Cartesian coordinate system, the Open-VRP package can also use a plotting subsystem to output the results in a nice graphical representation as shown in Fig. 5.

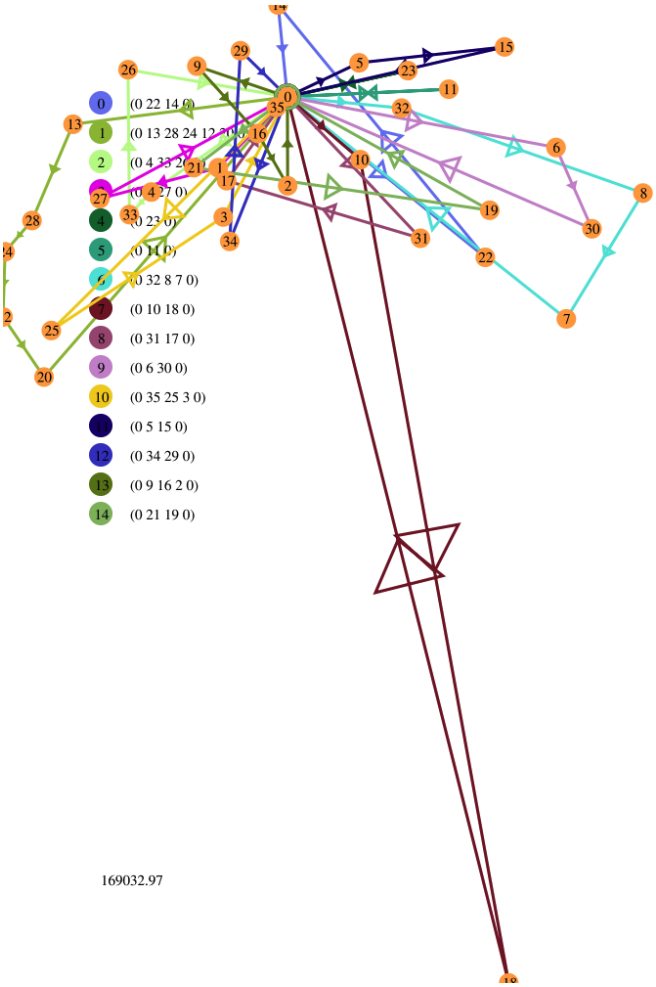

Fig. 5: Graphical output of proposed Open-VRP solution

Since each run is most likely going to provide different results, our experience showed that running the tool several times and grabbing the best solution from the mix was the easiest course of action. There are some batch tools to run the algorithm a number of times, but some bugs in the application caused issues with this technique overwriting files from previous runs if the results were generated faster than the file output could be written.

Final results were created by running the Open-VRP package on our TSPLIB95 [33] file (see Appendix C) until we received a trend of outputs that trended lower than most. We then picked from the lowest "fitness"–or distance-traveled–output from these runs and present this as our "good" solution for routing vehicles for NSN for their pilot data test. In playing with the various adjustments, a slightly larger tabu-tenure (20 vs. the default 15) was used as this tended to run the algorithm longer before reaching a local optima, and in some cases breaking out of this local optima and locating an improved output. In our chosen final output the optimal answer was reached on the 13th iteration of a run capped at 1000 iterations. The algorithm continued to look for improvements in its neighborhood until it gave up because of the lack of decrease in fitness at 74 iterations in. The Open-VRP package allows for further modification to continue iterating to look for a better solution as well as disabling the stopping conditions but results did not provide significant improvements and implementing these changes in the tool were not well documented and somewhat elusive. Another nicety of the application is the detailed log file that is output for each run. This provides insight into the operations of the

```
* (solve-prob etm540 (make-instance 'tabu-
search :iterations 1000 :tabu-tenure 20))
.....
.
Stopping condition met.
No more iterations left.
Run took a total of 0 seconds.
Final solution of run with TABU-SEARCH
on ETM540 was found on iteration 987
---------------
Fitness: 169032.97
---------------
[ 0]: (0 22 14 0)
[ 1]: (0 13 28 24 12 20 0)
[ 2]: (0 4 33 26 0)
[ 3]: (0 1 27 0)
[ 4]: (0 23 0)
[ 5]: (0 11 0)
[ 6]: (0 32 8 7 0)
[ 7]: (0 10 18 0)
[ 8]: (0 31 17 0)
[ 9]: (0 6 30 0)
[10]: (0 35 25 3 0)
[11]: (0 5 15 0)
[12]: (0 34 29 0)
[13]: (0 9 16 2 0)
[14]: (0 21 19 0)
---------------
#<TABU-SEARCH {1004FCF243}>
```

Fig. 6: Best answer from cluster of runs to ETM540 problem set

algorithm and can assist in tuning the available variables for increased performance. A truncated copy including only the beginning and end of the run log of the final solution is shown in Appendix B.

6 Conclusion

Initially we were interested in comparing the relative effectiveness of a heuristic technique to that of an exact technique and we had hoped to achieve a solution providing the ability to do a comparative analysis of these two solutions. Unfortunately we learned through our implementation very specifically why this is such a large area of research in computer science and combinatorial optimization: the problem sets quickly become so large and computationally intensive that exact solutions are outside of the realm of solving within realistic time-horizons [1]. By utilizing a freely-available open source solution called Open-VRP, we were able to experiment and provide a realistic set of vehicle routes for each upgrade crew that included a set of towers to be upgraded, allowed enough time for each vehicle to service each tower within the route within one single day, and minimized the distances traveled, though based on straight-line distances and not a travel-time matrix. This solution details 14 specific routes that consistent of one to five upgraded base transceiver stations per route. Nokia Siemens Networks can utilize this information to plan their work crews to upgrade as many segments of the network as they have resources available and know that the planned approach should minimize the distance traveled and not exceed their crew capacities to service each location in a given day. As is inevitable in real-world deployment scenarios, any adjustments or time-overages in the upgrade process can easily be accounted for by modifying the list of remaining towers to be upgraded and re-running the Open-VRP solver to generate new routes as needed within several minutes time.

With more time and resources, more advanced solver platforms or algorithms might be introduced, stricter, more exact capacity rating systems and constraints for each vehicle might be accounted for, or possibly even the integration of both upgrade duties and typical service calls could be incorporated to more efficiently service, upgrade, and maintain the NSN cellular system in Tehran, Iran.

7 Acknowledgements

First we would like to thank Dong-Joon Lim for his exceptional support in developing our branch and bound model (see Appendix C) in The R Project for Statistical Computing. Without his assistance we would not have realized the impact NP-hard solutions have on today's computing hardware. He also provided all the test case data for calculating runtime averages across a random set of data to elucidate the time-to-compute model in section 4.1. Dong-Joon also acted as a resource and advisor with our model preparation, review, and feedback throughout. Thank you Dong-Joon.

Second we would like to thank Marc Kuo (or "mck-" on Github) for his excellent package Open-VRP. His development allowed us to use a VRP solver implementation with real-world data utilizing a basic tabu search model to come up with a reasonable solution for this report. Without his freely available software, our team would have had to take a more theoretical approach and the hands on experience with two implementations was a useful learning exercise.

8 References

[1] Fred Glover and Manuel Laguna, *Tabu Search*. Boston, MA: Kluwer Academic Publishers, 1997.

[2] B. F. Voigt, "Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und (The Travelling Salesman, how he should be and what he should do to obtain Commissions and be successful in his Affairs. By a veteran Travelling Salesman, Ilmenau)," *Von einem alten Commis-Voyageur, Ilmenau*, 1831.

[3] Jan Karel Lenstra, A.H.G. Rinnooy Kan, and David B. Shmoys, "The traveling salesman problem: a guided tour of combinatorial optimization.," vol. 3, 1985.

[4] Rainer E. Burkard, Vladamir G. Deĭneko, René van Dal, Jack A. A. van der Veen, and Gerhard J. Woeginger, "Well-Solvable Special Cases of the Traveling Salesman Problem: A Survey," *SIAM Review*, vol. 40, no. 3, pp. 496-546, 1998. [Online]. http://www.jstor.org/stable/2653230

[5] T. Grünert and S. Irnich, *Optimierung im Transport, Band II: Wege und Touren*. Aachen: Shaker Verlag, 2005, vol. 2.

[6] R. S. Garfnikel, "Motivation and modeling," in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Chichester: Wiley, 1985, vol. 3, pp. 17-36.

[7] F. Gurski, I. Rothe, J. Rothe, and E. Wanke, "Exakte Algorithmen für schwere Graphenprobleme".

[8] A. J. Hoffman and P. Wolfe, "History," in *The traveling salesman problem: a guided tour of combinatorial optimization*. Chichester: Wiley, 1985, vol. 3, pp. 1-16.

[9] M. Gendreau, A. Hertz, and G. Laporte, "New Insertion and Postoptimization Procedures for the Traveling Salesman Problem," *Operations Research*, vol. 40, no. 6, pp. 1086-1094, November/December 1992.

[10] G. Pataki, "Teaching Integer Programming Formulations Using The Traveling Salesman Problem," *SIAM Review*, vol. 45, no. 1, pp. 116-123, March 2003.

[11] NEO Networking and Emerging Optimization. [Online]. http://neo.lcc.uma.es/vrp/wp-content/uploads/vrp.png

[12] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80-91, 1959.

[13] M. Desrochers, J. K. Lenstra, M. W. Savelsbergh, and F. Soumis, "Vehicle routing with time windows: optimization and approximation," *Vehicle routing: Methods and studies*, vol. 16, pp. 65-84, 1988.

[14] Bruce L., Raghavan, Subramanian Golden and Edward A. Wasil, Eds., *The Vehicle Routing Problem, Latest advances and new challenges*.: Springer, 2007, vol. 43.

[15] M. Desrochers, J. K. Lenstra, and M. W. Savelsbergh, "A Classification Scheme for Vehicle Routing and Scheduling Problems," *European Journal of Operations Research*, vol. 46, pp. 322-332, 1990.

[16] M. M. Solomon and J. & Desrosiers, "Time Window Constrained Routing and Scheduling Problems," *Transportation Science*, vol. 22, no. 1, pp. 1-13, 1988.

[17] M. Dror, G. Laporte, and P. Trudeau, "Vehicle Routing with Split Deliveries," *Discrete Applied Mathematics*, vol. 50, pp. 239-254, 1994.

[18] Geir Hasle and Oddvar Kloster, *Industrial vehicle routing*. Heidelberg: Springer Berlin, 2007.

[19] A. G. Qureshi, E. Taniguchi, and Tadashi Yamada, "An exact solution approach for vehicle routing and scheduling problems with soft time windows," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46.5, pp. 960-977, 2009.

[20] Laszlo Ladányi, Ted K. Ralphs, and Leslie E. Trotter Jr., "Branch, cut, and price: Sequential and parallel," *Computational combinatorial optimization*, pp. 223-260, 2001.

[21] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American mathematical society*, vol. 64, no. 5, pp. 275-278, 1958.

[22] Gilbert Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59.3, pp. 345-358, 1992.

[23] S. Vajda, "An outline of linear programming," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 165-172, 1955.

[24] Marshall L. Fisher and Ramchandran Jaikumar, "A Generalized Assignment Heuristic For Vehicle Routing," *Networks*, vol. 11.2, pp. 109-124, 1981.

[25] Stefan Ropke, Heuristics and Exact Algorithms for Vehicle Routing Problems, 2005.

[26] Olli Bräysy and Michel Gendreau, "Genetic algorithms for the vehicle routing problem with time windows," *Arpakannus*, vol. 1, pp. 33-38, 2001.

[27] Jean-Yves Potvin, "State-of-the art review—evolutionary algorithms for vehicle routing," *INFORMS Journal on Computing*, vol. 21.4, pp. 518-548, 2009.

[28] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30.2, pp. 105-119, 1997.

[29] Michel Gendreau, Alain Hertz, and Gilbert Laporte, "A tabu search heuristic for the vehicle routing problem," *Management science*, vol. 40.10, pp. 1276-1290, 1994.

[30] Ross Ihaka and Robert Gentleman, "R: A language for data analysis and graphics," *Journal of computational and graphical statistics*, vol. 5.3, pp. 299-314, 1996.

[31] Kenneth R. Baker, *Optimization Modeling with Spreadsheets*. Hoboken, NJ: Wiley, 2011.

[32] Mark Kuo. (2012) Open-VRP. [Online]. https://github.com/mck-/Open-VRP

[33] Gerhard Reinelt. (1995) Tsplib95.

[34] Johann Dréo. (2006, May) Wikimedia. [Online]. http://upload.wikimedia.org/wikipedia/commons/2/2a/Aco_TSP.svg

[35] Tolga Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34.3, pp. 209-219, 2006.

[36] Swfung8. (2011, April) Wikimedia. [Online]. http://en.wikipedia.org/wiki/File:Greedy-search-path-example.gif

9 Appendices

9.1 Appendix A: VRP Data Matrix – Time from Location A to Location B

Table X.XX: Time Matrix calculated in minutes.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	Depot
1	0	6	7	13	10	6	16	19	20	7	9	11	16	12	5	15	5	5	31	15	19	7	15	9	15	16	13	11	14	6	21	13	10	15	12	1
2	6	0	8	10	6	11	17	19	20	7	13	14	12	7	12	18	6	3	28	13	17	2	16	13	12	13	8	6	11	8	20	13	13	12	8	3
3	7	8	0	14	10	12	14	16	17	11	8	12	14	14	14	16	7	6	27	14	19	9	12	10	14	16	15	13	15	10	17	10	11	13	12	5
4	13	10	14	0	6	12	14	16	16	9	12	14	12	10	12	17	6	3	24	17	15	7	13	13	12	12	16	13	13	15	17	10	14	12	5	11
5	10	6	10	6	0	13	14	16	17	11	12	14	10	8	14	18	8	5	25	13	11	5	13	15	10	7	8	6	6	12	17	10	15	5	5	7
6	6	11	12	12	13	0	13	19	18	11	7	7	20	16	7	11	9	9	31	14	24	11	16	6	20	21	18	16	19	11	18	14	6	20	16	6
7	16	17	14	14	14	13	0	7	5	18	8	7	19	18	19	8	16	13	19	5	24	15	7	10	18	21	22	20	20	19	6	12	8	18	17	14
8	19	19	16	16	16	19	7	0	8	22	17	18	21	21	25	17	20	16	17	9	20	18	9	19	20	20	24	22	22	22	9	13	17	16	19	17
9	20	20	17	16	17	18	5	8	0	22	14	13	21	21	23	14	19	16	20	9	22	18	11	16	21	22	25	23	23	23	7	16	14	20	20	18
10	7	7	11	9	11	11	18	22	22	0	14	15	17	11	13	19	9	8	34	18	21	7	20	14	16	18	8	10	15	2	26	18	14	16	14	4
11	9	13	8	12	12	7	8	17	14	14	0	7	14	13	11	12	9	9	25	9	22	11	8	6	13	19	20	18	18	13	15	7	5	13	12	8
12	11	14	12	14	14	7	7	18	13	15	7	0	18	19	14	6	13	13	26	9	24	14	11	4	18	21	21	19	20	14	13	12	2	17	16	9
13	16	12	14	12	10	20	19	21	21	17	14	18	0	13	23	27	17	14	29	17	8	13	22	24	8	4	21	12	7	18	13	12	2	17	16	13
14	12	7	14	10	8	16	18	21	21	11	13	19	13	0	17	24	12	10	32	17	16	5	20	18	13	13	7	5	9	13	24	17	19	13	12	9
15	5	12	14	12	14	7	19	25	23	13	11	14	23	17	0	19	9	9	35	20	24	11	19	13	20	21	18	16	20	11	24	17	19	13	12	6
16	15	18	16	17	18	11	8	17	14	19	12	6	27	24	19	0	15	15	26	13	28	16	10	7	21	25	25	23	24	18	26	17	14	19	16	13
17	5	6	7	6	8	9	16	20	19	9	9	13	17	12	9	15	0	5	31	15	19	6	15	9	15	12	13	19	14	8	12	15	7	21	20	3
18	5	3	6	3	5	9	13	16	16	8	9	13	14	10	9	15	5	0	26	12	15	5	15	10	10	12	11	8	11	7	22	14	10	15	12	3
19	31	28	27	24	25	31	19	17	20	34	25	26	29	32	35	26	31	26	0	20	28	28	23	29	30	28	33	31	30	33	19	12	11	10	7	14
20	15	13	14	17	13	14	5	9	9	18	9	9	17	17	20	13	15	12	20	0	23	14	4	12	17	20	21	19	19	19	20	27	26	26	29	14
21	19	17	19	15	11	24	24	20	22	21	22	24	8	16	24	28	19	15	28	23	0	15	21	24	10	12	17	14	10	20	8	6	9	17	16	16
22	7	2	9	7	5	11	15	18	18	7	11	14	13	5	11	16	6	5	28	14	15	0	14	13	10	12	6	4	9	8	20	20	24	10	14	4
23	15	16	12	13	13	16	7	9	11	20	8	11	22	20	19	10	15	15	23	4	21	14	0	11	17	20	21	19	21	18	21	14	14	10	9	13
24	9	13	10	13	15	6	10	19	16	14	6	4	24	18	13	7	9	10	29	12	24	13	11	0	21	6	19	17	21	12	13	13	3	20	17	7
25	15	12	14	12	10	20	18	20	21	16	13	18	8	13	20	21	15	10	30	17	10	10	17	21	0	6	14	11	6	18	13	13	3	20	17	13
26	16	13	16	12	7	21	21	20	22	18	19	21	4	13	21	25	12	12	28	20	12	12	20	6	6	0	11	9	5	15	13	13	3	20	17	10
27	13	8	15	16	8	18	22	24	25	8	20	21	21	7	18	25	13	11	33	21	17	6	21	19	14	11	0	7	12	9	20	17	22	7	10	10
28	11	6	13	13	6	16	20	22	23	10	18	19	12	5	16	23	19	8	31	19	14	4	19	17	11	9	7	0	6	13	25	18	20	13	13	9
29	14	11	15	13	6	19	20	22	23	15	18	20	7	9	20	24	14	11	30	19	10	9	21	21	6	5	12	6	0	16	22	16	21	5	11	11
30	6	8	10	15	12	11	19	22	23	2	13	14	18	13	11	18	8	7	33	19	20	8	18	12	18	15	9	13	16	0	25	16	13	17	14	5
31	21	20	17	17	17	18	6	9	7	26	15	13	13	24	24	26	12	22	19	20	8	20	21	13	13	13	20	25	22	25	0	13	14	17	17	20
32	13	13	10	10	10	14	12	13	16	18	7	12	12	17	17	17	15	14	12	27	6	20	14	13	13	13	17	18	16	16	13	0	15	16	15	11
33	10	13	11	14	15	6	8	17	14	14	5	2	2	19	19	14	7	10	11	26	9	24	14	3	3	3	22	20	21	13	14	15	0	16	15	8
34	15	12	13	12	5	20	18	16	20	16	13	17	17	13	13	19	21	15	10	26	17	10	10	20	20	20	7	13	5	17	17	16	16	0	6	13
35	12	8	12	5	5	16	17	19	20	14	12	16	16	12	12	16	20	12	7	29	16	14	9	17	17	17	10	13	11	14	17	15	15	6	0	9
Depot	1	3	5	11	7	6	14	17	18	4	8	9	13	9	6	13	3	3	14	14	16	4	13	7	13	10	10	9	11	5	20	11	8	13	9	0

9.2 Appendix B: ETM540.vrp file for problem setup

```
NAME: ETM540
BEST_KNOWN: 835.26
COMMENT: 835.260000
DIMENSION: 36
CAPACITY: 480
EDGE_WEIGHT_FORMAT: FUNCTION
EDGE_WEIGHT_TYPE: EXPLICIT
NODE_COORD_SECTION
1 5067 15784
2 3835 14521
3 5072 14200
4 3924 13644
5 2642 14088
6 6344 16390
7 9855 14882
8 10038 11829
9 11403 14070
10 3454 16331
11 6367 14665
12 7943 15920
13 0 11863
14 1247 15296
15 4902 17419
16 8938 16673
17 4541 15131
18 3984 14291
19 9011 0
20 8673 13764
21 730 10793
22 3401 14523
23 8594 12925
24 7213 16252
25 36 13024
26 860 11622
27 2227 16268
28 1722 13970
29 516 13581
30 4230 16606
31 10495 13434
32 7438 13265
33 7101 15583
34 2252 13686
35 4039 13210
36 4870 15598
DEMAND_SECTION
1 60
2 120
3 180
4 240
5 60
6 120
7 180
```

```
8 240
9 60
10 120
11 180
12 240
13 60
14 120
15 180
16 240
17 60
18 120
19 180
20 240
21 60
22 120
23 180
24 240
25 60
26 120
27 180
28 240
29 60
30 120
31 180
32 240
33 60
34 120
35 180
36 0
DEPOT_SECTION
1
-1
EOF
```

9.3 Appendix B: Open-VRP log file of best solution (truncated)

```
It is now 19:53:21 of Monday, 12/09/2013 (GMT-8)
Commencing run with Tabu Search on ETM540

--------------#<STANDARD-CLASS CVRP> object details:--------------

Slot: NAME               Value: ETM540
Slot: DESC               Value: Capacitated Vehicle Routing Problem
Slot: TO-DEPOT           Value: T
Slot: DRAWER             Value: #S(DRAWER
                                  :MIN-COORD 0
                                  :MAX-COORD 17419
                                  :X-POS 0
                                  :Y-POS 0
                                  :MAX-PIX 1000
                                  :LEGENDP T
                                  :LEGEND-X 100
                                  :LEGEND-Y 900
                                  :FILENAME /Users/jerrodt/Open-VRP/plots/ETM540.png
                                  :PLOTP T)
Slot: LOG-FILE           Value: /Users/jerrodt/Open-VRP/run-logs/ETM540/ETM540.txt
```

```
Slot: LOG-MODE           Value: 1
-----------------------------------

--------------#<STANDARD-CLASS TABU-SEARCH> object details:--------------

Slot: NAME               Value: Tabu Search
Slot: DESC               Value: A local search that escapes local optima by means of
declaring certain moves tabu.
Slot: BEST-SOL           Value: NIL
Slot: BEST-FITNESS       Value: NIL
Slot: BEST-ITERATION     Value: 0
Slot: CURRENT-SOL        Value: NIL
Slot: ITERATIONS         Value: 1000
Slot: ANIMATEP           Value: NIL
Slot: MOVE-TYPE          Value: TS-BEST-INSERTION-MOVE
Slot: INIT-HEUR          Value: GREEDY-BEST-INSERTION
Slot: ASPIRATIONP        Value: T
Slot: ELITE-LISTP        Value: T
Slot: TABU-LIST          Value: NIL
Slot: TABU-TENURE        Value: 20
Slot: TABU-PARAMETER-F   Value: #<FUNCTION OPEN-VRP.ALGO::TS-PARS-N>
Slot: CANDIDATE-LIST     Value: NIL
Slot: STOPPING-CONDITION Value: #<FUNCTION OPEN-VRP.ALGO::STOPPING-CONDITIONP>
-----------------------------------

Performing INSERTION-MOVE with Node 14 and Vehicle 0 and Index 1
Performing INSERTION-MOVE with Node 22 and Vehicle 0 and Index 1
Performing INSERTION-MOVE with Node 20 and Vehicle 1 and Index 1
Performing INSERTION-MOVE with Node 12 and Vehicle 1 and Index 1
Performing INSERTION-MOVE with Node 8 and Vehicle 1 and Index 3
Performing INSERTION-MOVE with Node 13 and Vehicle 1 and Index 1
Performing INSERTION-MOVE with Node 16 and Vehicle 1 and Index 1
Performing INSERTION-MOVE with Node 26 and Vehicle 2 and Index 1
Performing INSERTION-MOVE with Node 5 and Vehicle 2 and Index 1
Performing INSERTION-MOVE with Node 27 and Vehicle 3 and Index 1
Performing INSERTION-MOVE with Node 23 and Vehicle 4 and Index 1
Performing INSERTION-MOVE with Node 1 and Vehicle 3 and Index 1
Performing INSERTION-MOVE with Node 33 and Vehicle 4 and Index 1
Performing INSERTION-MOVE with Node 24 and Vehicle 2 and Index 3
Performing INSERTION-MOVE with Node 21 and Vehicle 5 and Index 1
Performing INSERTION-MOVE with Node 11 and Vehicle 5 and Index 1
Performing INSERTION-MOVE with Node 7 and Vehicle 6 and Index 1
Performing INSERTION-MOVE with Node 4 and Vehicle 6 and Index 1
Performing INSERTION-MOVE with Node 18 and Vehicle 7 and Index 1
Performing INSERTION-MOVE with Node 10 and Vehicle 7 and Index 1
Performing INSERTION-MOVE with Node 17 and Vehicle 8 and Index 1
Performing INSERTION-MOVE with Node 31 and Vehicle 8 and Index 1
Performing INSERTION-MOVE with Node 30 and Vehicle 9 and Index 1
Performing INSERTION-MOVE with Node 3 and Vehicle 10 and Index 1
Performing INSERTION-MOVE with Node 15 and Vehicle 11 and Index 1
Performing INSERTION-MOVE with Node 6 and Vehicle 9 and Index 1
Performing INSERTION-MOVE with Node 25 and Vehicle 10 and Index 1
Performing INSERTION-MOVE with Node 32 and Vehicle 11 and Index 1
Performing INSERTION-MOVE with Node 29 and Vehicle 12 and Index 1
Performing INSERTION-MOVE with Node 34 and Vehicle 12 and Index 1
Performing INSERTION-MOVE with Node 35 and Vehicle 10 and Index 1
Performing INSERTION-MOVE with Node 28 and Vehicle 6 and Index 2
```

```
Performing INSERTION-MOVE with Node 2 and Vehicle 13 and Index 1
Performing INSERTION-MOVE with Node 9 and Vehicle 13 and Index 1
Performing INSERTION-MOVE with Node 19 and Vehicle 14 and Index 1
Run took a total of 0 seconds.
Final solution of run with GREEDY-BEST-INSERTION on ETM540 was found on iteration 0
---------------
Fitness: 199848.77
---------------
[ 0]: (0 22 14 0)
[ 1]: (0 16 13 12 20 8 0)
[ 2]: (0 5 26 24 0)
[ 3]: (0 1 27 0)
[ 4]: (0 33 23 0)
[ 5]: (0 11 21 0)
[ 6]: (0 4 28 7 0)
[ 7]: (0 10 18 0)
[ 8]: (0 31 17 0)
[ 9]: (0 6 30 0)
[10]: (0 35 25 3 0)
[11]: (0 32 15 0)
[12]: (0 34 29 0)
[13]: (0 9 2 0)
[14]: (0 19 0)
---------------
Performing INSERTION-MOVE with Node 8 and Vehicle 14 and Index 1
Performing TS-BEST-INSERTION-MOVE with Node 8 and Vehicle 14 and Index NIL
Iterations to go: 999
---------------
Fitness: 193909.97
---------------
[ 0]: (0 22 14 0)
[ 1]: (0 16 13 12 20 0)
[ 2]: (0 5 26 24 0)
[ 3]: (0 1 27 0)
[ 4]: (0 33 23 0)
[ 5]: (0 11 21 0)
[ 6]: (0 4 28 7 0)
[ 7]: (0 10 18 0)
[ 8]: (0 31 17 0)
[ 9]: (0 6 30 0)
[10]: (0 35 25 3 0)
[11]: (0 32 15 0)
[12]: (0 34 29 0)
[13]: (0 9 2 0)
[14]: (0 8 19 0)
---------------
<SKIP TO END OF FILE>
Performing INSERTION-MOVE with Node 31 and Vehicle 8 and Index 1
Performing TS-BEST-INSERTION-MOVE with Node 31 and Vehicle 8 and Index NIL
Iterations to go: 926
---------------
Fitness: 169064.33
---------------
[ 0]: (0 22 14 0)
[ 1]: (0 13 28 24 12 20 0)
[ 2]: (0 4 33 26 0)
[ 3]: (0 35 1 27 0)
```

```
[ 4]: (0 23 0)
[ 5]: (0 11 32 0)
[ 6]: (0 8 7 0)
[ 7]: (0 10 18 0)
[ 8]: (0 31 17 0)
[ 9]: (0 6 30 0)
[10]: (0 25 3 0)
[11]: (0 5 15 0)
[12]: (0 29 34 0)
[13]: (0 9 16 2 0)
[14]: (0 21 19 0)
---------------
Stopping condition met.
Run took a total of 0 seconds.
Final solution of run with TABU-SEARCH on ETM540 was found on iteration 987
---------------
Fitness: 169032.97
---------------
[ 0]: (0 22 14 0)
[ 1]: (0 13 28 24 12 20 0)
[ 2]: (0 4 33 26 0)
[ 3]: (0 1 27 0)
[ 4]: (0 23 0)
[ 5]: (0 11 0)
[ 6]: (0 32 8 7 0)
[ 7]: (0 10 18 0)
[ 8]: (0 31 17 0)
[ 9]: (0 6 30 0)
[10]: (0 35 25 3 0)
[11]: (0 5 15 0)
[12]: (0 34 29 0)
[13]: (0 9 16 2 0)
[14]: (0 21 19 0)
---------------
```

9.4 Appendix C: R implementation of model from section 2.2

```
# Author: Dong-Joon Lim, 2013
# Dataset
dist<-matrix(c(0,1,1,1,1,1,0,1,2,1,1,1,0,1,2,1,2,1,0,2,1,1,2,2,0),nrow=5) # testbed1
dist<-matrix(c(0,1,1,1,1,3,0,1,2,1,1,1,0,1,2,1,2,3,0,2,1,1,2,2,0),nrow=5) # testbed2
source("http://dl.dropbox.com/u/12900679/Datasets/salesman.txt") # your data, be careful,
 will take forever

# Function
salesman<-function(dist,m){ #assume that first row and column is depot

  library(lpSolveAPI)
  n<-nrow(dist)
  for(i in 1:n){if(i==1){d<-dist[1,]}else{d <- cbind(d,dist[i,])}}
  dim(d)<-c(1,n^2)

  results.r<-matrix(rep(-1.0, n^2), nrow=n, ncol=n)
  results.u<-matrix(rep(-1.0, n), nrow=1, ncol=n)
```

```
  sm<-make.lp(0,(n^2+n))
  set.objfn(sm, c(d,rep(0,n)))

  for(i in 1:n){
    if(i==1){
      add.constraint(sm, c(rep(1,n),rep(0,n^2)),"=", m) #r1_sum=m
      add.constraint(sm, c(rep(c(1,rep(0,(n-1))),n),rep(0,n)),"=", m) #c1_sum=m
      add.constraint(sm, c(1,rep(0,n^2+n-1)),"=", 0) #x11=0
      add.constraint(sm, c(rep(0,n^2),1,rep(0,(n-1))),"=", 1) #u1=1
    }else{
      add.constraint(sm, c(rep(0,((i-1)*n)),rep(1,n),rep(0,n^2+n-n*i)),"=", 1) #rs_sum=1
      add.constraint(sm, c(rep(c(rep(0,(i-1)),1,rep(0,(n-i))),n),rep(0,n)),"=", 1)
#csum=1
      add.constraint(sm, c(rep(0,(n*(i-1)+i-1)),1,rep(0,((n^2+n)-(n*(i-1)+i)))),"=", 0)
#xii=0
    }

    for(j in 2:n){
      if(i==j){next}
      else if(i<j){add.constraint(sm, c(rep(0,n*(i-1)),rep(0,(j-1)),n,rep(0,n^2-n*(i-1)-
(j-1)-1),rep(0,(i-1)),1,rep(0,(j-i-1)),-1,rep(0,(n-j))),"<=", n-1)}
      else if(i>j){add.constraint(sm, c(rep(0,n*(i-1)),rep(0,(j-1)),n,rep(0,n^2-n*(i-1)-
(j-1)-1),rep(0,(j-1)),-1,rep(0,(i-j-1)),1,rep(0,(n-i))),"<=", n-1)}
    }
  }

  set.bounds (sm, lower = rep(0, n^2+n))
  set.type (sm,1:n^2,"binary")
  set.type (sm,(n^2+1):(n^2+n),"integer")

  solve.lpExtPtr (sm)

  for(i in 1:n){results.r[i,]<-get.variables(sm)[(n*(i-1)+1):((n*(i-1)+1)+n-1)]}
  results.u<-get.variables(sm)[(n^2+1):((n^2+1)+n-1)]

  list(r=results.r,u=results.u)
}

# Run
salesman(dist,1)
salesman(x,1)
```