A Two-Stage Vehicle Routing Algorithm Applied to Disaster Relief Logistics after the 2015 Nepal Earthquake

Stephanie Allen, *SUNY Geneseo Mathematics Department* [*]
Advisor: Dr. Caroline Haddad, *SUNY Geneseo Mathematics Department* [†]

Abstract

After the April 2015 Nepal Earthquake, the Himalayan Disaster Relief Volunteer Group distributed supplies to affected areas. We model the organization's delivery of supplies as a vehicle routing problem using Fisher and Jaikumar's two-stage method, which allocates locations to vehicles via an integer program and then uses heuristics to route the vehicles. In the allocation stage, we use the assignment problem formulation to assign locations to vehicles. In the routing stage, we implement the greedy, sub-tour reversal, and simulated annealing heuristics for the sake of comparison. Our results illustrate the necessity of heuristics and the power that comes from using multiple heuristics for the vehicle routing problem.

1 Introduction

Operations research models can be used in disaster situations to improve efficiency as relief workers make critical decisions affecting the health and well-being of victims. In this paper, we model the operations of the Himalayan Disaster Relief Volunteer Group (HDRVG), which formed after the April 2015 Nepal Earthquake and delivered supplies to affected areas for 26 days. Using the group's detailed operational records, we seek to find the optimal assignment and routing decisions for the group's operations. We model the logistical system of the organization as a Vehicle Routing Problem (VRP) because HDRVG's base of operations was "a bed and breakfast" called Yellow House, and it is reasonable to assume that vehicles departed from and returned to this location. We choose to use Fisher and Jaikumar's two-stage VRP algorithm (described in Brandimarte and Zotteri's [4]) and provide background, theory, and results of the algorithm applied to HDRVG's operations. Section 2 examines potential models for disaster relief logistics. Section 3 presents the details and theory of the methods used for the two-stage algorithm, drawing upon multiple sources. Section 4 discusses data, model assumptions, and model specification for the HDRVG network. Section 5 discusses the results of the specified model for the HDRVG system, which reiterate the open nature of the VRP and the necessity of heuristics. Section 6 summarizes our findings and ideas for additional research.

2 Operations Research Models for Disaster Relief

2.1 Disaster Response Models

Luis E de la Torre et al.'s literature review identifies some of the challenges in modeling disaster relief logistics which include the following: establishing the trade off between equity and speed, incorporating the uncertainty in crisis situations into models, incorporating the presence and/or number of command centers into models, incorporating types of aid, the chance of robbery, & the loyalty/type of drivers into models, and using various types of objective functions [11]. Caunhye et al.'s literature review looked at "pre-disaster" and "short-term post-disaster" logistic models. In particular, for "post disaster planning," the models are "mostly multi-period," "contemplate single objective[s]," "rarely stochastic," and "construe relief distribution in terms of commodity

[*]Contact information: stephanie.a.allen95@gmail.com
[†]Contact information: haddad@geneseo.edu

flow or resource allocation," which means there are significant areas of open research for these models [7].

Our model specifically fits into Caunhye et al.'s category of "short-term post-disaster" models as we seek to deliver supplies to locations immediately after the April 2015 Nepal Earthquake. Both reviews demonstrate that there are many open research areas in disaster relief modeling.

2.2 General Network Models

All of the following definitions come directly from [8]. A directed graph is composed of a set of nodes/vertices V that are connected via a set of arcs/edges E which point in specified directions. Each edge k can be defined as an ordered pair of vertices, $E_k = (V_i, V_j)$, such that $i \neq j$. More formally, a directed graph has a vertex set V and an irreflexive relation E which is a subset of $V \times V$. A directed network is a function $G : E \to \mathbb{R}$, where the numeric result for our purposes represents the length of the arc/edge. A path from node V_1 to V_n in a graph G is a sequence of nodes and edges such that no nodes are repeated; a cycle only repeats the beginning and ending nodes — meaning $V_1 = V_n$. Finally, a *Hamiltonian Cycle* includes all the nodes of the graph G exactly once except for the beginning node which is equal to the ending node. A graph is *Hamiltonian* if it has a *Hamiltonian Cycle*.

Using this terminology, we define a general *transportation model* as a set of "supply" and "demand" nodes where we seek to minimize the costs of transporting goods from the supply to the demand nodes [18]. The *minimum cost flow model* accommodates intermediary nodes called transshipment nodes, which lie between the demand & supply nodes and create more potential routes for goods [18]. This is a powerful model because it can be modified to represent many transportation situations [18]. Finally, the *vehicle routing problem model* represents situations in which we need to reach n demand nodes using $m \geq 1$ vehicles such that the vehicles satisfy all of the demand nodes, do not overlap in their coverage of the nodes, and return to their mutual starting point (which we will refer to as the hub) [4, 6, 13].

Although all of these models can be used for disaster relief logistics, we choose the Vehicle Routing Problem Model because of the position of Yellow House as HDRVG's base of operations.

3 Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) has been approached in many different ways, but no one has thus far developed a comprehensive solution technique that will guarantee an optimal solution for n nodes [6]. Within the VRP literature, the Traveling Salesman Problem (TSP) (which is the VRP but with $m = 1$) has also been examined extensively, but no optimal algorithm has been devised. Nevertheless, there are many algorithms available which produce feasible solutions that differ in their distances from the true 'optimal' solution. First, there are heuristic algorithms which include modified minimum spanning tree algorithms such as Kruskal and Prim and algorithms specifically for Hamiltonian circuits such as the Sub-tour Reversal algorithm, Nearest Neighbor Algorithm, and Sorted Edge Algorithm [18, 14, 24, 34]. General metaheuristics also exist that can be customized to the TSP, including the Tabu Search, Simulated Annealing, and Genetic Algorithms [18]. There are also integer programming formulations of the TSP (some of which Pataki [28] illustrates in her article). Finally, Caric and Gold's book discusses more advanced formulations of the VRP [6].

We use Fisher and Jaikumar's two-stage method to produce solutions for the formulation of the HDRVG network as a VRP (described in Brandimarte and Zotteri's [4]). Fisher and Jaikumar's algorithm uses both integer programming techniques and heuristics/metaheuristics, with the heuristics/metaheuristics drawing upon the TSP literature above.

3.1 Two Stage VRP Algorithm: Overview

In the two-stage method [4], we assign nodes to vehicles using an integer program and then use heuristics/metaheuristics to find the optimal path for each vehicle to its assigned nodes.

For the assignment stage, we define z_{ik} as the binary decision variable indicating whether or not node i goes to vehicle k, h_{ik} as the additional distance node i would add to vehicle k's route, q_i as the amount to be delivered to node i, and R_k as the capacity of vehicle k. There are m vehicles and n demand nodes, so we have nm decision variables. Therefore, we formulate the minimization problem:

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} h_{ik} z_{ik} \tag{1}$$

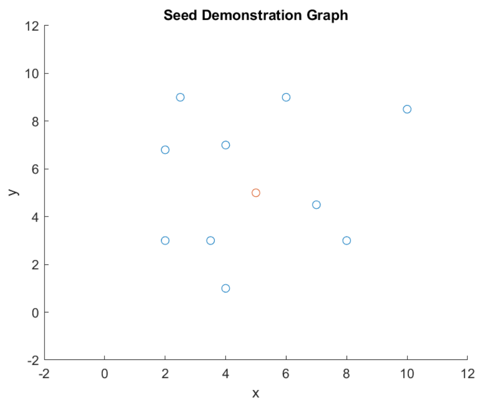
$$\sum_{k=1}^{m} z_{ik} = 1 \ \forall \ i = 1 \ldots n \tag{2}$$

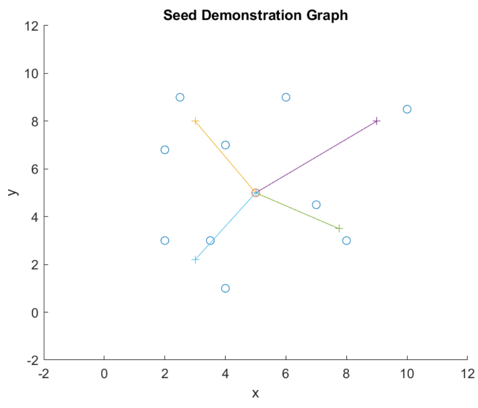
$$\sum_{i=1}^{n} q_i z_{ik} \leq R_k \ \forall \ k = 1 \ldots m \tag{3}$$

$$z_{ik} \in \{0, 1\} \tag{4}$$

The (2) constraints ensure coverage of each demand node by exactly one vehicle, and the (3) constraints enforce each vehicle's capacity. The (4) constraints ensure the z_{ik} variables are binary. We seek to minimize the objective function (1), which is the sum of the h_{ik} values. In order to calculate the h_{ik} values — the additional distance node i would add to vehicle k's route — we need to use an approximation because, during the integer program solution process, the node assignments are unknown. Therefore, we choose m "seed" nodes which are "far from the central location and far from each other" such that h_{ik} is the "extra mileage" of adding node i to vehicle k's route, with the route being represented by the line connecting the hub to the route's seed [4]. The seeds serve as the other ends of a series of lines that come out from the hub. Therefore we say $h_{ik} = d_{0i} + d_{i,\sigma_k} - d_{0,\sigma_k}$, with d_{ij} as the distance between nodes i and j and with σ_k as the kth seed.

To demonstrate this more concretely, we generated some sample data and graphs. On the left, we have an orange hub surrounded by blue circles, indicating the locations that need to be reached from the orange hub. The integer program expressed by (1)-(4) above will assign each of the blue circles to one of k vehicles. We will suppose there are four vehicles ($k = 4$) to be routed, so we place four seed nodes into the x-y plane, as illustrated by the graph on the right. The lines between the hub and the seed nodes act as artificial routes that allow us to cluster the nodes and thus assign them to the vehicles. The h_{ik} extra mileage measure uses these artificial routes to assign the nodes to the artificial routes.





In Section 4.3, we will explain in detail how we specified the parameters above, including how we determined the number of vehicles needed for each day (and thus the number of seeds) as well as the placement of the seeds among the nodes.

Once the integer program produces assignments of nodes to vehicles, we use heuristics and metaheuristics to route the vehicles with the aim of minimizing the distance traveled by each vehicle as it leaves the hub, services its assigned nodes, and returns to the hub [4]. Therefore, the second stage centers upon solving many small Traveling Salesman Problems (TSPs). Section 3.3 discusses our chosen heuristics/metaheuristics for solving these TSPs.

According to [4], "if there exists a feasible solution using m vehicles [for the vehicle routing problem], we will find" a feasible solution with the two-stage algorithm. We guarantee this statement by proving its contrapositive.

Theorem: Assume a feasible solution with m vehicles to the vehicle routing problem requires that the vehicles satisfy all n demand nodes, do not overlap in their coverage of the nodes, and return to their mutual starting point [4, 6, 13]. If the two-stage algorithm does not produce a feasible solution, there is no feasible solution for the vehicle routing problem with m vehicles.

Proof: Given a set of nodes, we can, without loss of generality, write any ordering of the set of nodes, and this would be a feasible route. Therefore, if the first stage of the algorithm produces feasible assignments of nodes, the second stage will produce a feasible route for each vehicle because, at a minimum, we could route the vehicles by choosing a random ordering of nodes such that Hamiltonian Cycles are formed. Therefore, we must examine the first stage.

Consequently, if the two stage algorithm cannot produce a feasible solution, this means:

$$\exists i = 1, ..., n \text{ such that } \sum_{k=1}^{m} z_{ik} \neq 1 \tag{5}$$

or

$$\exists k = 1, ..., m \text{ such that } \sum_{i=1}^{n} q_i z_{ik} > R_k \tag{6}$$

For the vehicle routing problem, we must be able to assign each node to one and only one vehicle to satisfy the feasibility requirements for a solution. If (5) is true, then there is no feasible solution to the vehicle routing problem with m vehicles because at least one node was not assigned to a vehicle or because at least one node was assigned to multiple vehicles. We also know that the vehicle routing problem in a realistic setting requires that we satisfy the demand at each node. However, if (6) is true, then our assignment has produced, for at least one vehicle, too much cargo such that it cannot deliver some of the supplies and thus cannot satisfy the demand of at least one node. As a result, if (6) is true, there is no feasible solution to the vehicle routing problem given m vehicles. Consequently, we have shown that if we cannot find a feasible solution with the two-stage algorithm, we cannot find a feasible solution to the vehicle routing problem. $\square$

Note that for the overview of the two-stage method, for the "extra mileage" measure, the addition of another node into a route must either increase or have no effect on the length of the route [4]. The Euclidean distance metric satisfies this condition because, for any three points a, b, and $c \in \mathbb{R}^2$, $||a - b|| \leq ||a - c|| + ||c - b||$, as seen through the simple algebra of $||a - b|| = ||a - c + c - b|| \leq ||a - c|| + ||c - b||$ (due to the Euclidean distance triangle inequality) [29].

3.2 Stage One: Integer Programming for Node Assignment

In the previous section, we explained the set-up of the integer program used in the first stage of the algorithm. To solve this integer program, we utilized the *MATLAB* mixed-integer linear programming solver, which employs a six step process to find a solution to a mixed-integer linear program. We describe in detail this six step process in the arXiv version of this paper, which is available for any interested readers.[1]

3.3 Stage Two: Meta-heuristics for Vehicle Routing

Once the first stage is completed, we move into a series of Traveling Salesman Problems (TSP) because we must route each vehicle among its assigned nodes. We employ multiple algorithms to route the vehicles and compare the results. First, though, we must discuss the necessity of these algorithms, given that we could (in theory) check every permutation of the nodes through which a given vehicle could be routed. This would not have been feasible for some of the routes. For example, at maximum, it took 0.000068 seconds to calculate the length of a route and assign this length to a variable for Day 19 when the payload capacity was 2000kg. We will use this value to estimate computational upper bounds. Under a 2000kg payload capacity, Day 19's two routes consisted of 11 and 12 nodes. Therefore, with this maximum time, to check all permutation for these routes, this would take:

$$(11!/2)0.000068 = 1357.17 \text{ seconds which is equivalent to } 22.6 \text{ minutes}$$

$$(12!/2)0.000068 = 16286.1 \text{ seconds which is equivalent to } 271.434 \text{ minutes}$$

We divide by two because, due to the circular natures of routes, an ordering of $1, 2, 3, 4, 5, 1$ would be equivalent to $1, 5, 4, 3, 2, 1$. Even more extreme, Day 10 had 14 nodes for one route, and a timing of 0.000062, which meant that it would take 31.279 *days* to evaluate all of the permutations. These calculations do represent upper bounds on the computational time for checking these permutations, but even a fraction of this computational time would be unacceptable in a disaster relief situation in which feasible solutions must be generated quickly.

Therefore, as we can see from these calculations, we need heuristics and metaheuristics to save us time as we seek to find the best way of routing vehicles in disaster relief. For our problem, we utilize the Greedy Algorithm, the Sub-Tour Reversal Algorithm, and the Simulated Annealing Algorithm to generate feasible routes for our vehicles. In the next three paragraphs, we describe these three algorithms and discuss some of their attributes.

We first examine a simple *Greedy Algorithm* whereby we build a vehicle route by adding nodes such that each addition minimizes the cumulative distance of the path up to that node. The main disadvantage of this algorithm is that we do not consider the larger route when making our choices regarding which node to add into the route next. For instance, the algorithm does not consider the fact that we will need to go back to the starting point to finish the route [24, 34].

Next, we examine the *Sub-Tour Reversal Algorithm*, which iteratively "deletes exactly two links from [a] previous tour and replaces them by exactly two new links to form the new tour" [18]. This occurs by "selecting a subsequence of the cities and simply reversing the order in which that subsequence of cities is visited" [18]. Given a sequence of nodes that form a Hamiltonian Cycle, $\{1, 2, 3, 4, 5, 6, 1\}$, some possible sub-tour reversals would be:

$$\{1, 3, 2, 4, 5, 6, 1\}, \{1, 2, 3, 6, 5, 4, 1\}, \{1, 5, 4, 3, 2, 6, 1\}$$

The technique is well known and well documented [18, 4, 15, 31, 27]. It is known as a "local improvement procedure" or a "local search" algorithm because, although it improves a solution, it arrives at a local optimum rather than the global optimum [18, 4]. In other words, the algorithm only focuses on its "local neighborhood" [18]. The full algorithm proceeds as such [18]:

1. Establish an initial feasible solution. For each vehicle, we choose the initial feasible solution as the nodes in numeric order with the Yellow House bed-and-breakfast as the first and last node (to form the Hamiltonian Cycle).

[1] See `https://arxiv.org/abs/1709.00162` for this additional section.

2. Create all possible sub-tour reversals using the current feasible solution by inverting all subsequences in the solution which will generate new routes (when compared to the current route). Choose the new route that has the smallest distance as the new feasible solution.

3. Repeat Step 2 until sub-tour reversals do not result in smaller distances (when compared to the current feasible solution).

Another disadvantage of this algorithm is the potential for computational issues as the number of nodes increases. Given enough nodes, it may become too costly to check all feasible subsequences during multiple iterations.

Finally, we look at the *Simulated Annealing Algorithm* which, as opposed to the Sub-Tour Reversal algorithm, goes beyond just its local neighborhood to try to find the global optimum [18, 27, 31]. The algorithm follows the following procedure (from [18]):

1. Select a "trial solution" (which could also be termed a "trial route") whose objective function value is z_c. For our application, our objective is to minimize the distance traveled by the vehicles, and the objective function value is the length of the route. Also, the initial trial solution is the same as the initial trial solution for the Sub-Tour Reversal Algorithm.

2. Utilizing the concept of a Sub-Tour Reversal, randomly choose the beginning and end of a subsequence to invert. The beginning cannot be the first, the last, or the second to last node, the end cannot be the last node, and the subsequence itself cannot include both the second *and* the second to last node. All of these distinctions assume a route with the hub at the beginning and at the end of the route (or, in other words, the sequence of nodes).

3. Find the length of the route generated from this Sub-Tour Reversal; call this length z_n.

   (a) If $z_n \leq z_c$, then the route just generated from the Sub-Tour reversal becomes the new trial solution, and Step 2 repeats.

   (b) If $z_n > z_c$, the new route (generated from the Sub-Tour reversal) becomes the new trial solution if, given the parameter T (called a "temperature") and a randomly chosen number w from a uniform distribution with end points of 0 and 1, $e^{\frac{zc-zn}{T}} > w$. If this test fails, the original trial solution (before the Sub-Tour reversal) is the 'new' trial solution, and we return to Step 2.

4. We continue Steps 2-3 based upon a chosen "schedule" of Ts whereby we iterate a set number of times per T [18].

We select large T values at the beginning to generate a higher chance that we will utilize the new routes generated by the algorithm because "the early emphasis is on taking steps in random directions...in order to explore as much of the feasible region as possible," which means the algorithm, with these larger T values, will be more likely to "accept" solutions that have greater distances [18]. The advantage to this strategy is that the algorithm will have a greater chance of "escaping a local optimum" [18]. However, as time passes, we want to close in upon a "good" solution, so we decrease the T values in order to prevent fewer "worse" solutions from being chosen [18]. We are not guaranteed to find the global optimum, but this algorithm increases our chances of "escaping a local optimum."

4 Data, Model Assumptions, & Model Specification

4.1 Data

After providing supplies for 26 days to victims of the April 2015 Nepal Earthquake and its aftershocks, the Himalayan Disaster Relief Volunteer Group (HDRVG) publicly released its data via the Tableau Public platform [25]. The data set contains information regarding the date, location (District and Village Development Committee (VDC) information including latitude and longitude information), mission number, type of product, and amount of product for each mission. In the dataset, a mission can span multiple rows because each row represents one product that was delivered during a particular mission. We utilized *R* Version 3.2.0 with the dplyr and stringr packages to produce CSV data files for each of the 26 days, containing the delivery VDC locations for each day.

4.2 Model Assumptions

We had to make some assumptions regarding our model in order to produce results during the time alloted to this research. Therefore, we make the following assumptions/set the following parameters:

- We seek to minimize the distance traveled by the vehicles.
- We assume all vehicles begin their journeys at the Yellow House Bed and Breakfast and return to Yellow House after making their routes. These are reasonable assumptions because the organization refers to Yellow House as the central hub of activity for the group.
- We treat each day independent from the other days, so we re-run the model each day using that day's data.
- For each day, we assume all supplies are available at the start of the day, that there is no difference between products, and that the products are distributed evenly across the locations. These are not entirely realistic assumptions because a relief organization might be receiving supplies throughout the day, affected locations would need a variety of products (and thus would care about the difference between products), and some areas might need more goods than others (based on the degree to which different areas were affected). Nevertheless, these assumptions enable us to formulate the problem.
- We use the Euclidean Distance Metric to calculate the distances between nodes. This does not take into consideration the road infrastructure of Nepal, which would be important with trucks, but it does ensure that we do not violate the triangle inequality, which is important for our model. In practice, our routes could be achieved with helicopters or drones.
- We took the delivery locations established by the organization as given. Therefore, we do not decide where to deliver supplies; we simply want to find the best way to route vehicles to the locations established by the organization.

4.3 Model Specification

We translated the two-stage method into code in *MATLAB R2014b* with the Optimization Toolbox (Version 7.1) (which provided the integer program solver). Our script executes the two-stage algorithm for all of the days of HDRVG's operations. We had to select several parameter values for the two-stage method, which will be discussed in this section. To begin with the integer program for the first stage, we discussed its formulation in Section 3.1 as follows:

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} h_{ik} z_{ik}$$

$$\sum_{k=1}^{m} z_{ik} = 1 \ \forall \ i = 1 \ldots n$$

$$\sum_{i=1}^{n} q_i z_{ik} \leq R_k \ \forall \ k = 1...m$$

$$z_{ik} \in \{0, 1\}$$

First, we specified the R_k value(s), which refer to the payload capacities of the vehicles.[2] According to [12], light-duty pickup trucks have a payload capacity of about 900-1400 kilograms. Upon examining images of HDRVG's trucks, we chose to test two R_k values, 1500 kg and 2000 kg [30]. For the q_i coefficients, we decided to assume each demand node in a given day would receive the same amount of supplies. Consequently, for each day, we divided the total amount of supplies by the number of demand nodes to calculate the q_i values for the day. Finally, to determine the number of vehicles for a given day, we used the formula: $ceiling((n)/(payload/q))$.

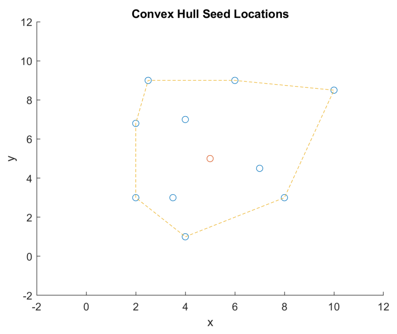
Next, as we discussed in Section 3.1, in order to calculate h_{ik}, we needed to use an approximation. Therefore, we chose m "seed" nodes which were "far from the central location and far from each other" such that h_{ik} was the "extra mileage" of adding node i to the route with the kth seed. Consequently, we used the formula, $h_{ik} = d_{0i} + d_{i,\sigma_k} - d_{0,\sigma_k}$, where d_{ij} was the distance between nodes i and j and σ_k was the kth seed ($k = 1...m$) [4]. The challenging part of this "extra mileage" formula was the determination of the "seeds." We established a hierarchy of steps in the *MATLAB* script to determine the seed locations for each day, with the number of seeds equal to the number of vehicles:

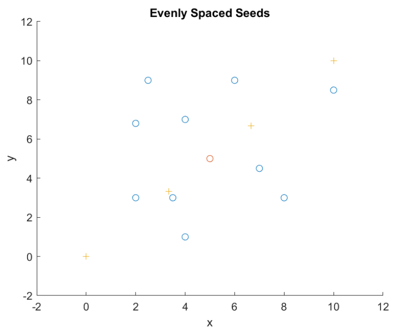
1. If the number of demand nodes is greater than 1, compute the convex hull of the nodes (including the hub) for the given day. If the number of corners of the convex hull exceeds the number of vehicles v for the day, choose v corners.

   (a) Otherwise, if the number of corners of the convex hull is less than the number of vehicles, take v equally spaced points inclusively between the minimum and maximum latitude and longitude values of the entire set of nodes across all of the days.

2. If there is only one demand node, that node is the seed.

Once we established the seed nodes for a given day, we calculated all of the h_{ik} values for the objective function for that day. Below, using our sample data from Section 3.1, we demonstrate the convex hull and the evenly-spaced points methods for choosing the seeds. The yellow dotted line on the left illustrates the convex hull, and the yellow plus signs on the right indicate four evenly spaced seeds between the minimum and maximum x and y values. For the convex hull, assuming as we did in Section 3.1 that we needed four vehicles ($k = 4$), we would choose four corners.





[2]The payload capacity is "the total weight of people and equipment...[a vehicle]...can carry without overloading" [5].

For the heuristics/metaheuristics in stage two of the algorithm, only the simulated annealing algorithm had parameters to be specified. The schedule of T values — with z_0 as the initial route distance — was set according to [18] as: $T_1 = 0.2z_0, T_2 = 0.2T_1, T_3 = 0.2T_2, T_4 = 0.2T_3, T_5 = 0.2T_4$. We also ran the algorithm for 15 iterations per temperature T_i for the 1500kg payload capacity and ran the algorithm for 20 iterations per temperature T_i for the 2000kg payload capacity. As a result of multiple trials, we discovered 20 iterations yielded significantly better results for the 2000kg payload capacity as opposed to 15 iterations.

5 Results

For each of the 26 days, we produced six sets of results, one set for each of the heuristic and payload capacity (1500 kg and 2000 kg) combinations. In the assignment stage, we observe variability when we compare the assignments under the two payload capacities, demonstrated below in a sample of four days. Note, markers of the same color and shape have been assigned to the same vehicle.

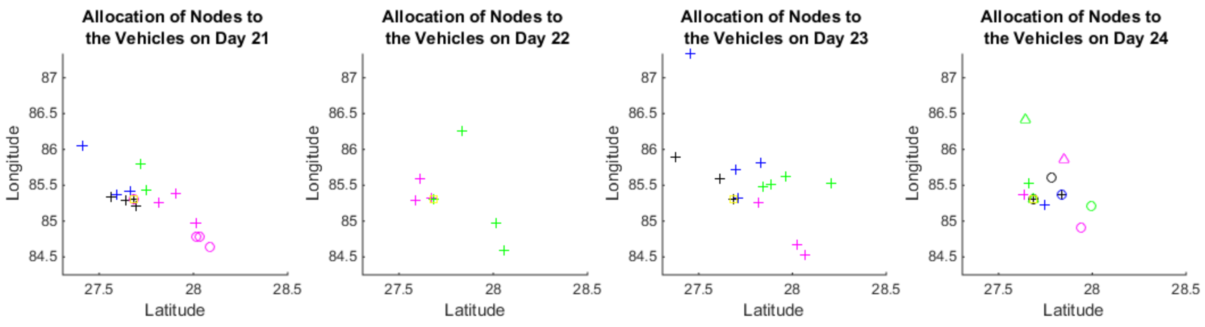


Figure 1: Sample of Node Assignments for Four Days with Payload = 1500kg

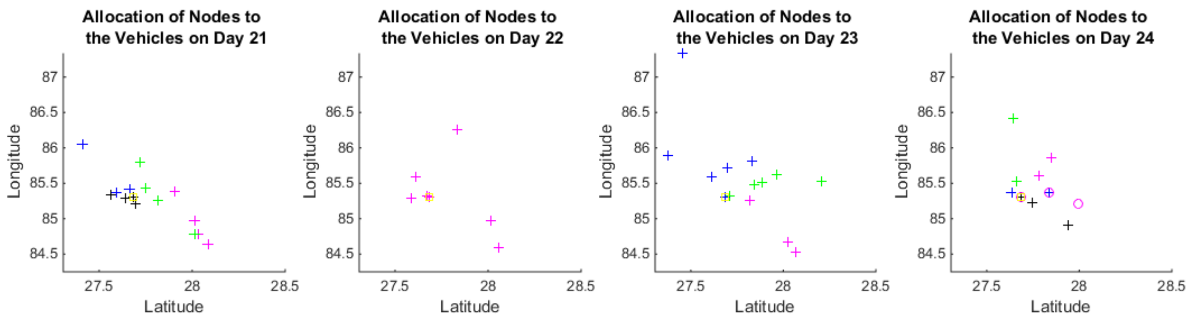


Figure 2: Sample of Node Assignments for Four Days with Payload = 2000kg

These figures demonstrate that, as the payload capacity increased, the number of vehicles needed to cover the nodes decreased. For Day 21, the increase in payload capacity eliminated the pink circles. For Day 22, a payload capacity of 1500kg necessitated two vehicles to cover the nodes, while a capacity of 2000kg necessitated only one vehicle. Similarly, Day 23 saw a reduction in vehicles, from four to three. Finally, Day 24 best illustrates the impact of an increase in payload capacity because, before the increase, virtually no clusters existed but, after the increase, clusters formed.

For stage 2, to understand the differences between the three routing algorithms, it is useful to examine the routing of vehicles on Day 19 under each of the three algorithms for each of the two chosen payload capacities (which the graphs below illustrate).

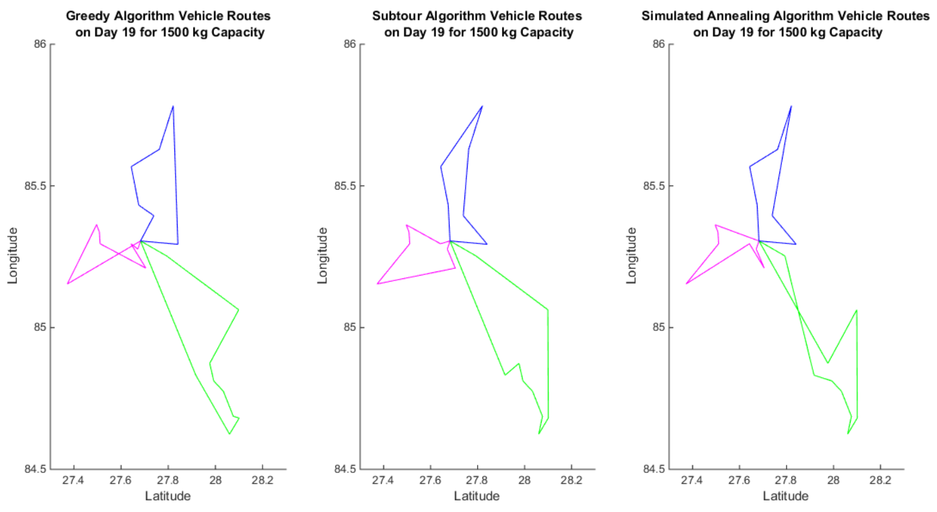


Figure 3: Day 19 Routing under Three Algorithms with Payload = 1500kg

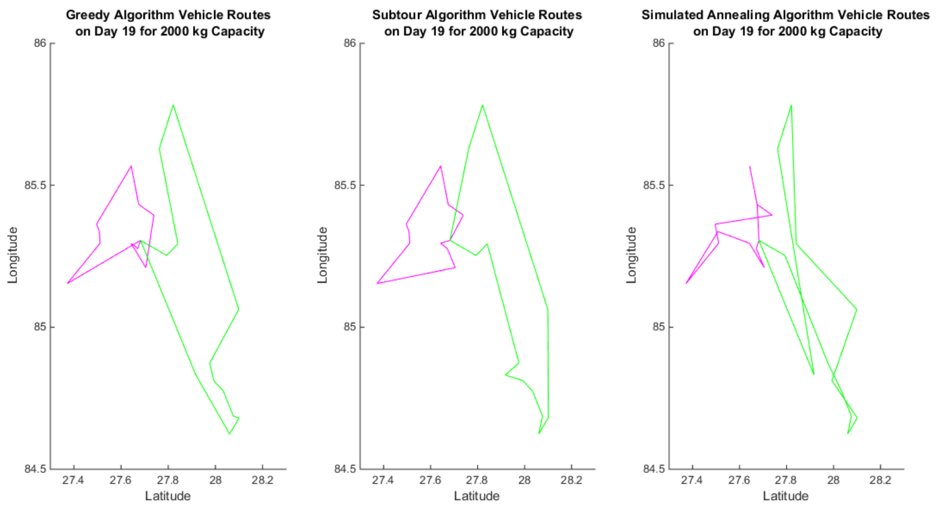


Figure 4: Day 19 Routing under Three Algorithms with Payload = 2000kg

For each payload capacity, we see there are distinct differences in the shapes of the paths among each of the three algorithms. The crossings in some of the routes indicate inefficiencies, especially in the Simulated Annealing graph under the 2000kg payload capacity.

To quantify our results, we found the total distance in degrees traveled by the vehicles each day under each heuristic/metaheuristic for each payload capacity. We multiplied all of these degree measurements by 61.8 miles, which is the average of the degree-to-mile equivalences for latitude and longitude measures (1 degree latitude = 69 miles and 1 degree longitude = 54.6 miles) [33]. We then summed these mile measurements to find the total number of miles traveled under each of the algorithms for each payload capacity. The table below displays the results.

Day	**Greedy 1500kg**	**Sub-Tour 1500kg**	**Sim Annealing 1500kg**	**Greedy 2000kg**	**Sub-Tour 2000kg**	**Sim Annealing 2000kg**
Day 1	418.674	403.544	406.017	397.533	371.44	376.537
Day 2	166.363	151.22	151.22	166.363	151.22	151.22
Day 3	114.632	113.243	113.243	114.632	113.243	113.243
Day 4	18.775	18.775	18.775	18.775	18.775	18.775
Day 5	3.624	3.624	3.624	3.624	3.624	3.624
Day 6	233.889	221.439	231.188	233.889	221.439	232.367
Day 7	317.358	307.802	311.522	317.358	307.802	307.802
Day 8	402.981	366.401	366.401	337.32	314.915	314.915
Day 9	486.708	479.716	479.716	422.827	397.894	397.894
Day 10	247.38	216.98	228.604	244.518	211.304	278.387
Day 11	169.994	169.329	169.329	169.994	169.329	169.329
Day 12	448.04	439.901	439.901	314.667	286.738	284.332
Day 13	269.981	269.981	269.981	269.981	269.981	269.981
Day 14	370.145	351.383	351.383	320.647	298.729	304.594
Day 15	98.21	98.21	98.21	84.725	84.725	84.725
Day 16	170.376	170.376	170.376	138.169	136.221	136.221
Day 17	63.419	63.419	63.419	63.419	63.419	63.419
Day 18	28.664	28.434	28.434	28.664	28.434	28.434
Day 19	254.088	245.995	265.911	261.207	250.058	324.296
Day 20	103.035	97.635	97.635	93.588	92.668	99.706
Day 21	362.719	355.001	355.001	387.431	369.632	369.632
Day 22	257.254	254.058	254.058	221.124	221.124	221.124
Day 23	529.043	525.782	525.782	468.185	450.042	450.042
Day 24	440.434	440.434	440.434	340.179	340.179	340.179
Day 25	753.973	741.107	798.328	753.973	741.107	791.937
Day 26	136.692	135.352	135.352	136.692	135.352	138.595
Totals	6866.451	6669.142	6773.845	6309.484	6049.395	6271.31

Table 1: Total Distance Traveled by Vehicles Each Day Under the Three Routing Algorithms for 1500kg and 2000kg Payload Capacities

For both payload capacities, the Sub-tour Reversal Algorithm is the best algorithm overall, followed by the Simulated Annealing Algorithm and then the Greedy Algorithm. We expected the Simulated Annealing Algorithm to be the superior algorithm because it has a better chance of "escaping a local minimum"; indeed, the algorithm is designed to be able to search a solution space [18]. These results reconfirm that the Traveling Salesman Problem (TSP) has not been solved; as can be seen from the table, for any given day, there is not a guarantee regarding which algorithm will perform the best. There are several days in the 2000 kg payload capacity chart in which the Greedy Algorithm produces a better solution (in terms of miles) than the Simulated Annealing Algorithm!

As we discussed in Section 3.3, it is not computationally feasible to examine all of the possible solutions when routing vehicles through several nodes. These were the approximate run times for both stages of the algorithm under each of the heuristics and metaheuristics (for each of the capacities):

	Greedy	**Sub-Tour**	**Sim Annealing**
Run Time 1500kg (secs)	1.650	2.066	2.670
Run Time 2000kg (secs)	2.153	2.542	2.615

Table 2: Computation Time for Two Stage Algorithm for all 26 Days Under Each Routing Algorithm

The speed of these algorithms when compared to the speed of checking permutations lends credibility to using these algorithms for vehicle routing problems. In Section 3.3, we saw, at maximum, it takes 22.6 minutes to check 11 nodes, 271.434 minutes to check 12 nodes, and 31.279 days to check 14 nodes.

If required to recommend one algorithm, we would recommend the Sub-Tour Reversal algorithm for the HDRVG's work because the Sub-Tour Reversal algorithm produces the best results under both payload capacities. Our recommendation might change in the case of a significant increase in the number of nodes because the number of subsequences would increase with the number of nodes. With the Simulated Annealing Algorithm, we would be able to control the number of iterations directly, which would be useful in a potential situation in which the Sub-Tour Reversal Algorithm would be computationally infeasible. However, given the computation time for these three algorithms, we would *overall* recommend running all three algorithms because they would take 10 seconds or less to run (for a single payload capacity) and, as demonstrated by the charts above, one algorithm is not consistently superior for all of the days.

6 Conclusion and Future Work

This project demonstrates that operations research is a rich mathematical subfield filled with powerful theory and techniques that can allow us to model logistics scientifically. Our goal was to minimize the distance traveled by vehicles as they brought supplies to affected communities. We used a two-stage vehicle routing problem algorithm, which allowed us to explore integer programming techniques and heuristics/metaheuristics. Our results reaffirmed the open nature of the vehicle routing problem and demonstrated that, until an optimal algorithm is found, it is necessary to use a variety of techniques and to then compare the output of these techniques to find the best solution.

With regard to future work, there are numerous complexities we would seek to integrate into the model. We would seek to relax or even eliminate some of the assumptions present in this model, including the homogeneity of supplies & distribution and the necessity of the Euclidean distance metric. Instead of the Euclidean distance metric, we would aim to utilize road distances provided by Google Maps. Integrating stochastic events into the model would also be extremely useful, especially given that the conditions right after natural disasters are often quite unclear. Furthermore, we would research additional temperature scales for the Simulated Annealing algorithm to see if a different scale would improve the results of the algorithm for the HDRVG system.

7 Acknowledgements

We would like to take this opportunity to thank Dr. Caroline Haddad, our advisor at SUNY Geneseo, for all of her advice, help, and encouragement during this project. We became interested in operations research because of her, and we appreciate all of the support she provided during our undergraduate career. Without her, we wouldn't be in graduate school for an Applied Mathematics PhD.

References

[1] Alper Atamtürk and Martin W. P. Savelsbergh. Integer–programming software systems. *http : //www2.isye.gatech.edu/ ms79/publications/ipsoftware4.1 − kluwer.pdf*, July 2004.

[2] Djamel Berkoune, Jacques Renaud, Monia Rekik, and Angel Ruiz. Transportation in disaster response operations. *Socio-Economic Planning Sciences*, 46(1):23–32, 2012.

[3] Hax Bradley and Magnant. *Applied Mathematical Programming.* Addison-Wesley, 1977.

[4] Paolo Brandimarte and Giulio Zotteri. *Introduction to Distribution Logistics.* John Wiley & Sons, Inc, Hoboken, NJ, 2007.

[5] John Cadogan. Towing and load limits for suvs and utes. *http : //autoexpert.com.au/ buying − a − car/understanding − towing − and − load − limits − for − suvs − and − utes*, 2016.

[6] Tonci Caric and Hrvoje Gold. *Vehicle Routing Problem.* In-Tech, Croatia, 2008.

[7] Aakil M Caunhye, Xiaofeng Nie, and Shaligram Pokharel. Optimization models in emergency logistics: A literature review. *Socio-economic planning sciences*, 46(1):4–13, 2012.

[8] Gary Chartrand. *Introductory Graph Theory.* Dover Publications Inc, Mineola, NY, 1977.

[9] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming.* Springer International Publishing Switzerland, Switzerland, 2014.

[10] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[11] Luis E. de la Torre, Irina S Dolinskaya, and Karen R Smilowitz. Disaster relief routing: Integrating research and practice. *Socio-economic planning sciences*, 46(1):88–97, 2012.

[12] Doug DeMuro. Top 7 light-duty pickup trucks by payload capacity. *http : //www.autotrader.com/best − cars/top − 7 − light − duty − pickup − trucks − by − payload − capacity −* 241420, 2015.

[13] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[14] David Eppstein. Minimum spanning trees. *https : //www.ics.uci.edu/ eppstein/*161/960206*.html*, 1996.

[15] Felix Fischer. Heuristic algorithms. *http : //dcs.gla.ac.uk/ fischerf/teaching/mor/ notes/notes14.pdf*, 2014.

[16] George D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.

[17] Himalayan Disaster Relief Volunteer Group. Himalayan disaster. *http : //www.himalayandisaster.org/.*

[18] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* McGraw Hill, New York City, 2010.

[19] Bernard Kolman and Robert E. Beck. *Elementary Linear Programming with Applications.* Academic Press, San Diego, CA, 1995.

[20] Jeff Linderoth. Ie418: Integer programming. *https : //homepages.cae.wisc.edu/ linderot/classes/ie*418*/lecture*4*.pdf*, February 2005.

[21] Jeff Linderoth. Ie418: Integer programming. *https : //homepages.cae.wisc.edu/ linderot/classes/ie*418*/lecture*3*.pdf*, January 2005.

[22] MathWorks. intlinprog, 2014.

[23] MathWorks. Mixed-integer linear programming algorithms, 2014.

[24] Gabriele E. Meyer. Hamiltonian circuits. *http : //www.math.wisc.edu/ meyer/math*141*/graphs*2*.html*, NA.

[25] Himalayan Disaster Relief Network. Himalayan disaster relief volunteer group - operations report. *https : //public.tableau.com/profile/publish/ YellowHouseVolunteerGroupReliefOperationsv*2*/DashbHorizontal*#*/publish* − *confirm*, 2016.

[26] Martin J. Osborne. using bibtex: a short guide. *https : //www.economics .utoronto.ca/osborne/latex/BIBTEX.HTM*, 2015.

[27] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993.

[28] Gábor Pataki. Teaching integer programming formulations using the traveling salesman problem. *SIAM review*, 45(1):116–123, 2003.

[29] David Poole. *Linear Algebra: A Modern Introduction.* Cengage Learning, Stamford, CT, 2015.

[30] Himalayan Disaster Relief. Himalayan disaster relief volunteer group. *https : //www.facebook.com/pg/hdrvg/about/?ref* = $page_internal$.

[31] Kevin Ross. Ism206: Metaheuristics. *https : //classes.soe.ucsc.edu/ ism*206*/Fall*10*/Lecture*12*.pdf*, 2010.

[32] Martin WP Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6(4):445–454, 1994.

[33] US Geological Survey. Usgs faqs. *https : //www*2*.usgs.gov/faq/categories/* 9794/3022, November 2016.

[34] Leonardo Zambito. Traveling-salesman problem algorithm. *http : //www.cse.yorku.ca/ aaw/Zambito/*TSP_L*/Web/TSPAlg.html*, 2006.