MATH 691Y - Applied Math Group Project

Abdulrahman Alenezi
William Howe
Lindsay Knupp
Johnny Rasnic

February 10, 2024

1 Abstract

For this project we investigated the logistical problem of log transportation in forestry. This is a variant of the famous vehicle routing problem. We investigated models and methods in the literature and focus on a specific variant of the routing problem, the capacitated vehicle routing problem. We compared different methods on a preexisting data set. The data originates from a Scottish forestry company. The methods investigated are linear programming, simulated annealing, and ant colony optimization.

2 Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is a combinatorial problem that asks "If an organization has a fleet of vans/trucks/vehicles and wishes to visit a set of customers, what route minimizes the total distance travelled?". For the classic VRP, there is a single depot that all vehicles start and finish at, and the speed of travel is assumed to be consistent. Thus, there is no practical difference between minimizing time or distance. There are many variants of the VRP [17]. Some examples are:

- TSP: If there is only a single vehicle then the VRP simplifies to a Traveling Salesperson Problem (TSP) where one vehicle (or salesperson) visits all locations.
- TSPP: A variant of the TSP is the TSP with Prices (TSPP): each location has a price and the salesman must only visit a subset of the locations while maximizing profit (sum of prices of visits minus travel cost).
- CVRP: If the vehicles are delivering items to customers and each vehicle has a set amount of capacity, then the problem is the Capacitated Vehicle Routing Problem (CVRP). The CVRP simplifies to the VRP if each demand is 1 and the vehicle capacity exceeds the number of customers.
- VRPTW: In some instances, each customer can only be served within a time window. This is the Vehicle Routing Problem with Time Windows (VRPTW).
- AVRP: The Asymmetric VRP occurs when there is one or more pairs of locations where the cost from location i to location j is not equal to the cost from location j to location i. This causes the cost matrix to be asymmetric, thus the name.
- VRPB: For the VRP with Backhauls, customers exist in one of two groups: linehauls and backhauls. For any route, linehaul customers must all be visited before any backhaul customers. If backhaul customers are eliminated, then the VRPB simplifies to a VRP. A real life example of this is the grocery store industry. In that industry, each truck follow three ordered steps: (1) transport food from the depot to grocery stores (the linehaul customers, (2) pick up food from farmers (backhaul customers), and (3) return with newly-acquired food to the depot. The required order occurs because trucks are loaded/unloaded from the lack in a first-come last-out manner.
- TTVRP: The Timber Transport Vehicle Routing Problem (TTVRP) is a variant whereby timber orders are given a priori. These orders would specify the supply point, destination

point, and quantity. The objective wold be be route and schedule the vehicles with minimum distance travelled while respected the a priori orders.

Other variants exist such as a VRP with multiple depots or with a heterogeneous fleet (vehicles with different capacities) which, though interesting, are beyond the scope of this paper.

3 Vehicle Routing Problems and Forestry

In our project, we approach a VRP in a forestry context. All trucks start empty at a single depot, and must return to the depot empty at the end of a workday. Trucks travel to forests, where they pick up up a consignment of lumber to deliver to a mill. All trucks can only hold one consignment at a time. All consignments are from a specific forest to a specific sawmill. These consignments are provided a priori, so our task is to minimize the "deadhead", the distance accrued when a truck travels empty to the next forest. In other contexts, one can include constraints such as time windows of consignments, limitations on how many trucks can concurrently deliver orders to a location at a time, and constraints on the length of a work day for the truckers. For our project, our only constraints are on the capacity of the trucks.

4 Mixed Integer Linear Programming Formulations

4.1 VRP Two-Index Basic Formulation

This basic formulation uses a decision variable, $x_{i,j}$, to denote if a vehicle moves from location i to location j. Since the variable keeps track of the direction, the topology of the routing area can be understood as a directed graph. Since $x_{i,j}$ has two indexes, it is known as a two index formulation [1].

Scalars and Indexes

$n \in \mathbb{N}$		number of customers
$V = 1..n$		Set of all customer indexes
$V_0 = \{0\} \cup V$		Set of all locations (also called nodes)
$i, j \in V_0$		index of locations: depot (0) and customers (1 to n)

Parameters

$c_{i,j} \geq 0$	$\forall i, j \in 0..n$	cost/distance from i to j
$Q > 0$		Capacity of each vehicle
$K \in \mathbb{N}$		Number of vehicles which can be used

Variables

$x_{i,j} \in \{0, 1\}$	$\forall i, j \in 0..n$	Is 1 if some vehicle moves from location i to location j, 0 otherwise

$$min_x \sum_{i,j \in 0..n} c_{i,j} x_{i,j} \tag{1}$$

$$\sum_{i \in V_0} x_{i,j} = 1 \forall j \in V \tag{2}$$

$$\sum_{j \in V_0} x_{i,j} = 1 \forall i \in V \tag{3}$$

$$\sum_{j \in V} x_{0,j} \leq K \tag{4}$$

$$\sum_{i,j \in S, i \neq j} x_{i,j} \leq |S| - 1 \forall S \subseteq V \tag{5}$$

Equation 1 is the objective which minimizes the length of the sum of all the tours. Constraints 2 and 3 ensure that each customer has one predecessor and one successor, respectively. Constraint 4 ensures that no more than K vehicle can depart from the depot. With only those constraints, meaning without constraint 5, the model would create subtours. A subtour is a route that is isolated from the depot. For example the tour $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ is an example of a subtour. Constraint 5 would eliminated the $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ subtour by asserting that the number of active arcs between members of subset $S = \{1, 2, 3\}$ is less than or equal to 2. This eliminates the subtour since $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \implies \sum_{i,j \in 1..3, i \neq j} x_{i,j} = 3$. This type of subtour elimination constraint is called the DFJ constraint, after the authors Dantzig, Fulkerson, and Johnson [3]. Note that the cardinality of S grows exponentially with the number of customers. In mixed integer linear programming, DFJ constraints are asserted only when an incumbent solution violates it. This kind of constraint is known as a lazy constraint.

4.2 Capacitated Vehicle Routing Problem

For the CVRP, each customer is associated with a demand and each vehicle travels delivering the demanded product to each customer on its route. The total demand over the customers of a route must not exceed a route. New notation for the CVRP is the following:

Q	Capacity of each vehicle (this assumes each vehicle is the same)
$d_i > 0 \forall i \in V$	Demand for each customer i

The DFJ subtour elimination constraints (equation 5) can be modified in order to assert capacity:

$$\sum_{(i,j): i \in S, j \notin S} x_{i,j} \geq \lceil \sum_{i \in S} d_i / Q \rceil \forall S \subset V \tag{6}$$

In effect, any subset of customers must have enough vehicles to supply that subset of customers. An alternative set of constraints that simultaneously assert subtour elimination and vehicle capacity are the MTZ (Miller-Tucker-Zemlin) constraints [13][3].

$u_i \in [0, Q - d_i] \forall i \in V_0$ keeps track of level of inventory the vehicle leaves node i with

$$u_j \leq u_i - d_i + Q(1 - x_{i,j}) \forall i \in V_0, j \in V \tag{7}$$

The MTZ constraints disallow subtours since for any route which includes customer i, $u_i < u_j < .. < u_k < u_i \implies u_i < u_i$ which is a contradiction. The MTZ constraints also sets a capacity for each vehicle as the upper bound of u which is Q.

4.3 CVRP with Time Windows

Integrating time windows requires new parameters, variables, and constraints. It also generally requires an updated representation of the depot. For this section, the depot is represented by two indexes: 0, and $n+1$. Practically, each vehicle departs from 0 and finally goes to $n+1$ [5].

Time Window Parameters

$V_{0,n+1}$	set of edges included the two edges representing the depot: 0 and n+1
$a_i \forall i \in V_{0,n+1}$	when window for edge i starts
$b_i \forall i \in V_{0,n+1}$	when window for edge i ends
$s_i \forall i \in V_{0,n+1}$	servace time for edge i
$t_{i,j} \forall i,j \in V_{0,n+1}$	time taken to travel from i to j, usually proportional to cost or distance

Time Window Variables

$w_i \forall i \in V_{0,n+1}$	lower bound of time of arrival at edge i

Two Index VRPTW

$$min_x \sum_{i,j \in V_{0,n+1}} c_{i,j} x_{i,j} \tag{8}$$

$$\sum_{i \in V_{0,n+1}} x_{i,j} = 1 \forall j \in V \tag{9}$$

$$\sum_{j \in V_{0,n+1}} x_{i,j} = 1 \forall i \in V \tag{10}$$

$$\sum_{j \in V} x_{0,j} \leq K \tag{11}$$

$$a_i \leq w_i \leq b_i \forall i \in V_{0,n+1} \tag{12}$$

$$w_j \geq w_i - b_i + a_j + x_{i,j}(b_i - a_j + s_i + t_{i,j}) \forall i \in V_0, j \in V_{n+1} \tag{13}$$

The biggest difference with VRPTW is the inclusion of the variable w_i denoting when a vehicle visiting location i leaves location i. Constraint 12 ensures that the time window of each location is respected. Among the locations, the depot is special since each vehicle must (1) depart after the start of the shift and (2) arrive before the end of the shift. This necessitates the representation of the depot using 2 nodes: 0 and $n+1$. Constraint 13 is like the previously referenced MTZ constraint but for time instead of capacity. The Capacitated VRPTW (CVRPTW) can be constructed by simply asserting vehicle capacities and capacity constraints in the VRPTW. For some instances of the VRPTW, all locations are open over the same period of time and so the time constraint can be simplified to merely forcing each vehicle to depart and arrive at the depot within the shift. Under this simplification, a practitioner could forgo asserting constraint 12 for all customer locations; only asserting it for the depot.

4.4 Three Index Formulation

In the preceding two-index formulations the set of all routes was determined indirectly by location specific information. The third index in a three index formulation designates the vehicle used. The

three index formulation allows for a wide range of vehicle specific idiosyncrasies [5].

3-Index Formulation Variables

$x_{i,j,k} \forall i \in 0..n, j \in 1..n+1, k \in 1..K$	if vehicle k travels from location i to location j
$w_{i,k} \forall i \in 0..n+1, k \in 1..K$	time when vehicle k leaves location i

$$min_x \sum_{\substack{i \in 0..n \\ j \in 1..n+1 \\ k \in 1..K}} c_{i,j} x_{i,j,k} \tag{14}$$

$$\sum_{\substack{i \in 0..n \\ k \in 1..K}} x_{i,j,k} = 1 \forall j \in 1..n \tag{15}$$

$$\sum_{i \in 0..n} x_{i,j,k} = \sum_{i' \in 1..n+1} x_{j,i',k} \forall j \in 1..n, k \in 1..K \tag{16}$$

$$w_{i,k} + s_i + t_{i,j} - w_{j,k} \leq (1 - x_{i,j,k}) M_{i,j} \tag{17}$$

$$a_i * (\sum_{j \in 0..n+1} x_{i,j,k}) \leq w_{i,k} \leq b_i * (\sum_{j \in 0..n+1} x_{i,j,k}) \forall k \in 1..K, i \in 1..n \tag{18}$$

$$a_i \leq w_{i,k} \leq b_i \forall k \in 1..K \tag{19}$$

Constraint 15 ensures that each customer is visited once. Constraint 16 requires that if vehicle k arrives at a customer, that vehicle also leaves that customer. Constraint 18 works like constraint 12 except that $w_{i,k}$ is zeroed out if location i is not visited by vehicle k.
In the form shown above, there is a symmetry between all of the vehicles: every solution exists in a set of $K!$ permutations since permuting the vehicle index has no practical effect on the solution. Symmetry will usually slow the solver because it exponentially increases the size of the search space. It is good practice to eliminate symmetry through a symmetry defeating constraint [14]. A symmetry defeating constraint is asserting that the distance travelled by vehicle j is greater than the distance travelled by vehicle $j+1$. This is shown in constraint 20.

$$\sum_{\substack{i \in 0..n \\ j \in 1..n+1}} c_{i,j,k} (x_{i,j,k} - x_{i,j,k+1}) \geq 0 \forall k \in 1..K-1 \tag{20}$$

4.5 Set Covering and Column Generation

4.5.1 Overview

A strategy related to the three-index formulation is set covering. In a set covering formulation, each possible route is enumerated and assigned a decision variable. The formulation picks a subset of the routes that minimizes the cost while covering the entire set of customers. A route is represented by a column of binary variables with a length equal to the number of customers. A 1 in space i represents that that route visits customer i. A set covering method is infeasible due to the exponentially large number of possible routes. The column generation method circumvents this problem [4].
There are two problems in the column generation algorithm that are repeatedly solved: the restricted master problem (RMP) and the subproblem (SP). The subproblem could also be called

the pricing problem. The RMP takes a set of routes and chooses which combination of routes to use. At first, the integral constraints of RMP are relaxed so the RMP can be solved as an LP. The subproblem creates a new route which minimizes the reduced cost of the RMP. The algorithm iteratively optimizes the RMP and adds a new route option using the SP. When no new route with a negative reduced cost can be added, the RMP is solved as an IP which results in the ultimate solution.

4.5.2 Definitions

Scalars and Indexes

$n \in \mathbb{N}$		number of customers
$i, j \in 1..n$		index of locations: depot (0) and customers (1 to n)
$r \in 1..\|R\|$		index of routes

Parameters

$d_{i,j} \geq 0$	$\forall i, j \in 0..n$	distance from i to j
$A_{ir} \in \{0, 1\}$	$\forall i \in 1..n, r \in 1..\|R\|$	if customer i is visited on route r
$c_r \geq 0$	$r \in 1..\|R\|$	cost/distance of route r
R		set of all routes

Variables for Restricted Master Problem (RMP)

$y_r \geq 0$	$\forall r \in 1..\|R\|$	if route r is used, may be continuous or discrete

Variables for Subproblem (SP)

$x_{i,j} \in \{0, 1\}$	$\forall i, j \in 0..n$	if a van travels from i to j
$a_i \in 0, 1$	$\forall i \in 0..n$	if location i gets visited in the subproblem route (a_0 is always 1)
$u_i \in (0, C)$	$\forall i \in 0..n$	for MTZ subtour elimination constraints

4.5.3 Restricted Master Problem (RMP)

RMP

$$min \sum_{r \in 1..|R|} c_r y_r \tag{21}$$

$$\sum_{r \in 1..|R|} A_{ir} y_r \geq 1 \forall i \in 1..n \rightarrow \pi_i \forall i \in 1..n \tag{22}$$

The objective, equation 21, minimizes the total travel distance of the chosen routes. Constraint 22 ensures that each customer is visited at least once. The π is the dual variable for equation 22. There is a π_i for each customer.

4.5.4 Subproblem (SP)

$$min(\sum_{i,j \in 0..n, i \neq j} d_{i,j} x_{i,j}) - \sum_{i \in 1..n} a_i \pi_i \tag{23}$$

$$u_j \leq u_i - x_{i,j} + (C + 1)(1 - x_{i,j}) \forall i \in 0..n, j \in 1..n, i \neq j \tag{24}$$

$$\sum_{j \in 0..n} x_{i,j} = a_i \forall i \in 0..n \tag{25}$$

$$\sum_{i \in 0..n} x_{i,j} = a_i \forall j \in 0..n \tag{26}$$

$$x_{i,i} = 0 \forall i \in 0..n \tag{27}$$

The objective is the reduced cost of the RMP. Constraint 24 eliminates subtours using MTZ style elimination. Constraints 25 and 26 ensures each visited customer has one successor and one predecessor. Equation 27 could be replaced with just defining $x_{i,j}$ for cases where $i \neq j$.

4.5.5 Algorithm

1: Set A ▷ create a feasible solution
2: Solve RMP as an LP
3: Solve SP
4: **if** objective of SP is negative **then** ▷ check if reduced cost < 0
5: $R = R + 1$
6: Set S_R ▷ Add the new route into the set of routes
7: Set a_R ▷ new column
8: $A_{:,R} = a_R$
9: go to step 2
10: **end if**
11: Solve RMP as IP
12: Set solution to chosen routes

4.6 Minimum Cost Flow Problem

For this paper, the constraints simplify the problem into a minimum cost flow problem with upper and lower bounds. Due to the total unimodularity of the constraint matrix, the basic feasible solutions are integer and so the integer program collapses into a linear program [2]. For each a edge directed from a logging forest to a mill, the upper and lower bounds were set to the number of orders for that pair.

Sets and Scalars

E	set of all edges
V	set of all vertices
d	depot
T	number of trucks

Parameters and Decision Variables

x_e	amount of flow across edge e
c_e	cost/distance for edge e
u_e	upper bound of flow for edge e
l_e	lower bound of flow for edge e

$$\min_x \sum_{e \in E} c_e x_e \tag{28}$$

$$u_e \leq x_e \leq l_e \forall e \in E \tag{29}$$

$$\sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e \forall v \in V \setminus \{d\} \tag{30}$$

$$\sum_{e \in \delta^+(d)} x_e = \sum_{e \in \delta^-(d)} x_e = T \tag{31}$$

5 Simulated Annealing (SA)

Simulated Annealing is a meta-heuristic optimization method that is modeled after the process of slowly cooling metal. The process transforms the metal into its strongest and most workable form. Analogously, the simulated annealing algorithm utilizes a slowly decreasing temperature parameter that allows for the discovery of the optimal solution [16].
We used the model formulated in Haridass et al. [10] to base our own models after. From this, a fitness function was developed to account for the costs associated with the total distance travelled and unloaded miles of a particular truck route. Next, a cooling schedule has to be developed that includes the initial temperature, the final "cooled" temperature, the cooling parameter, and the number of iterations at each temperature [11]. These parameters are usually experimented with by authors until a satisfactory solution is discovered.
Then, we can use both the temperature and the fitness function as probabilistic parameters that determine whether a worse solution is accepted in our algorithm. For example, if the temperature is still relatively high, then a worse solution is more likely to be accepted than if the temperature is nearing its "cooled" temperature [16]. This allows the algorithm to escape any local minimum it may have found in search for the global minimum.
Various authors propose different probability thresholds. For example, in Haridass et al. [10], a worse solution will be accepted if

$$\text{Random}(0,1) < \frac{T+\delta}{T}$$

where T is the current temperature and δ = (current solution fitness) − (new solution fitness).
In Tan et al. [16] and Harmanani et al. [11], a worse solution will be accepted if

$$\text{Random}(0,1) \leq e^{(-\frac{\delta}{T})}$$

where T is again the current temperature and δ = (new solution fitness) − (current solution fitness).
After the newly generated solution is accepted or rejected according to a particular scheme, the algorithm repeats the process for the specified number of iterations before multiplying the current temperature by the cooling parameter until the final temperature is reached.
Our own model was developed in Python and our fitness function only included the total unloaded miles. We implemented specific operators present in Haridass [10] such as swapping or transferring routes between trucks to generate new solutions; swapping routes maintains the total number of orders given to a set of trucks while transferring does not. Further, trucks could be given a route of any length as we had no time constraints.

6 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a swarm intelligence based metaheuristic based on the metaphor of ants and pheromones. This metaheuristic was first developed by Dorigo in his 1992 PhD dissertation and continually developed throughout the 90s [6][8][9]. "Ants" wander around randomly on our graph, depositing pheromone either after each edge traversal, or after a full solution to our problem is found. This pheromone increases the probability the path will be chosen in the future by any ant. This increase in probability is usually scaled inversely to the length of the edge. All pheromone also evaporates at a certain rate after each step or solution finding iteration, decreasing the probability that an ant will pick that edge. We also have a parameter to control edge selection a priori, such as preferring short paths over long ones. Metaphorically, we can think of this as an ant being able to "see" the length of an edge and changing the likelihood of picking a path based on this observed length. We can represent the probability that the kth ant takes a path as

$$p_{xy}^k = \frac{\left(\tau_{xy}^{\alpha}\right)\left(\eta_{xy}^{\beta}\right)}{\sum_{\{\text{allowed } z\}}\left(\tau_{xz}^{\alpha}\right)\left(\eta_{xz}^{\beta}\right)}$$

where τ_{xy}^{α} is the amount of pheromone on the edge xy subject to a parameter α, and where η_{xy}^{β} is the a priori adjustment to the path subject to the parameter β. For a VRP, we will restrict which edges an ant can pick at each step, those edges are represented in the equation above as $\{\text{allowed } z\}$. We can write the pheromone update rule as follows

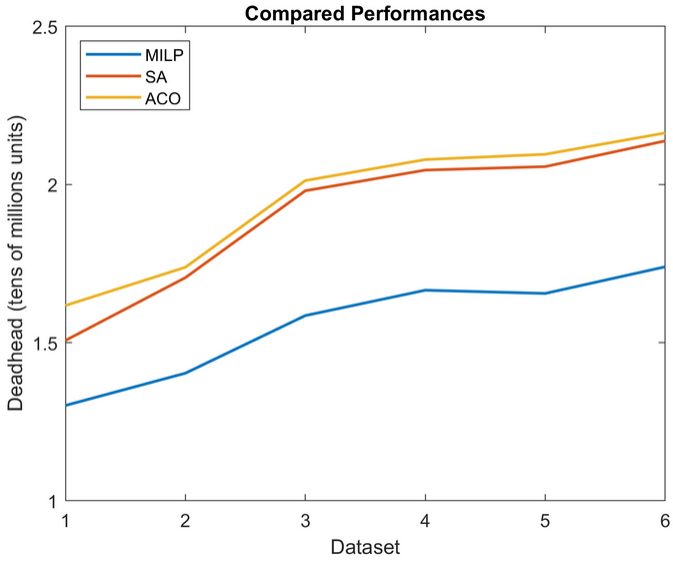
$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^k$$

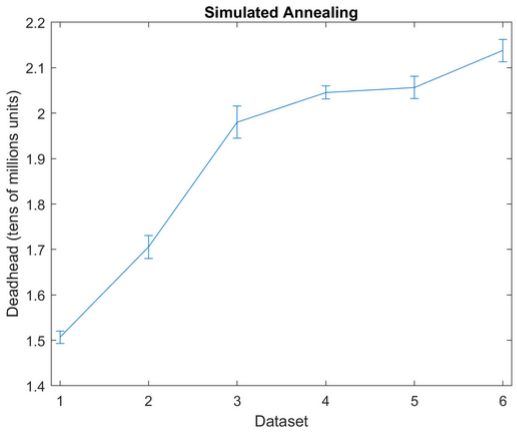
where ρ is the evaporation parameter, and $\Delta\tau_{xy}^k$ is the pheromone strength on an edge xy if it used by the kth ant. This amount is almost always scaled inversely to the length of the edge xy, so we may further define $\Delta\tau_{xy}^k = \frac{Q}{L_{xy}}$, where Q is a pheromone parameter we choose to represent the constant, absolute amount of pheromone deposited by a given ant, and L_{xy} represents the length of the edge xy. Thus, the key idea is that, with a properly chosen pheromone constant, and evaporation rate, the ants will converge on a near-optimal solution to our VRP, and much quicker than a brute force search.

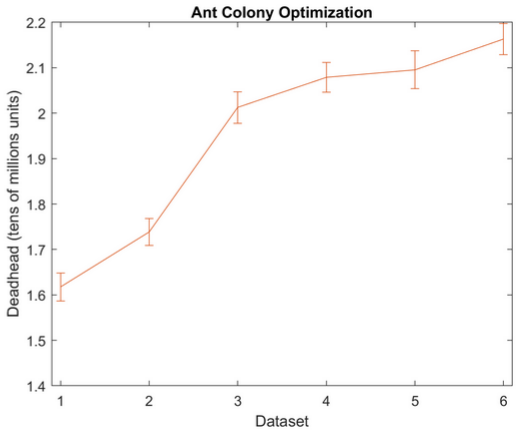
7 Data

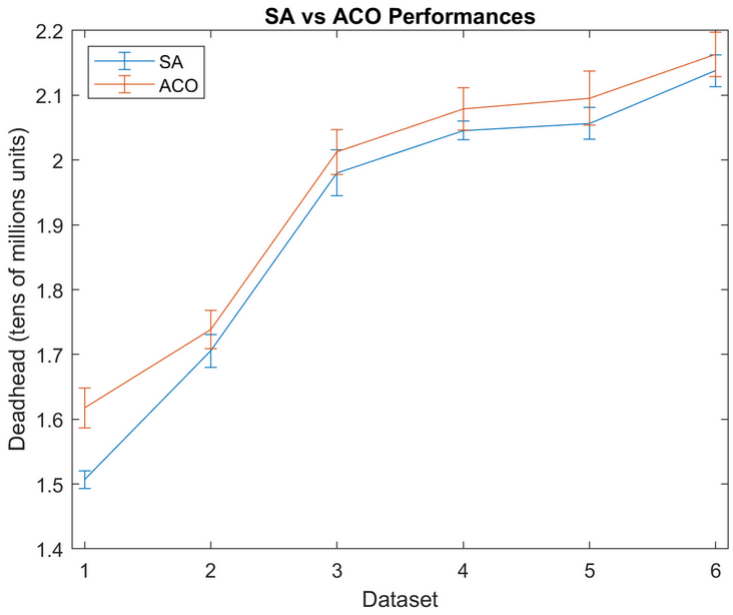
We use the data sets found in Kent et al. (2014) [12]. There were six data sets, each of which includes coordinate, location type, loading bay constraint data for each location, and consignments with associated time windows. For our project, we used the coordinate, location type, and consignment data without the associated time windows. Between the data sets, a range of 300 to 440 orders had to be fulfilled by 40 trucks. Units were not presented in the paper; all of our results are left as the raw numbers.

8 Results


Compared Performances
MILP
SA
ACO
Deadhead (tens of millions units)
Dataset
2.5
2
1.5
1
1
2
3
4
5
6


Simulated Annealing
Deadhead (tens of millions units)
Dataset
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1.4
1
2
3
4
5
6


Ant Colony Optimization
Deadhead (tens of millions units)
Dataset
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1.4
1
2
3
4
5
6



Deadhead results (lower is better)

Dataset	MILP	SA (average case)	ACO (average case)
1	1.3014e07	1.5070e07	1.6174e07
2	1.4034e07	1.7055e07	1.7383e07
3	1.5856e07	1.9803e07	2.0125e07
4	1.6658e07	2.0456e07	2.0789e07
5	1.6556e07	2.0566e07	2.0956e07
6	1.7398e07	2.1378e07	2.1630e07

As we can see, the MILP implementation performed far better than our heuristics. This is to be expected, as MILP solved to optimality. Given the small size of our data, all methods took roughly

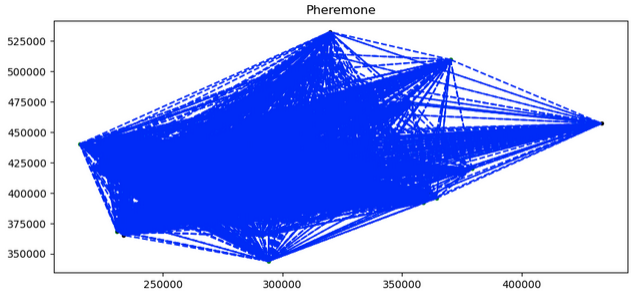
Approximate Standard Deviations of the heuristics

Dataset	SA	ACO
1	1.3608e05	3.0677e05
2	2.5119e05	2.9667e05
3	3.5220e05	3.4633e05
4	1.4263e05	3.2796e05
5	2.4417e05	4.1370e05
6	2.4624e05	3.4209e05

the same amount of time to run. Since our heuristics involve randomness, we sampled the methods many times to approximate the mean and variance of the performance of the heuristics. For the ACO method, an evaporation rate of 0.1 was used, and an a priori adjustment of the reciprocal of the distance to the fourth power was used. This was found through many manual adjustments to give the best results, although this is not ideal (discussed later). For the SA method, an initial temperature of 100, a final temperature of 1, a cooling parameter of 0.99, and an iteration number of 5 were used. These were found through experimentation to try and give the algorithm ample time to settle on a better solution with each temperature step. We can see that the SA and ACO methods performed similarly, with SA performing better both on average and with smaller variance. However, both methods had rather high variance, and SA is not guaranteed to outperform ACO in any given run of the heuristics.

9 Discussion

For the ACO method, we noticed that no matter what evaporation rate was used, the pheromone network did not form in any effective patterns, and always settled into a uniform network. This is visualized below by looking at the pheromone network after the final solution is returned. The opacity of the lines denotes the strength of the pheromone on that path.



Ideally, we would expect clear paths to emerge as better options for finding a solution, and not a network where almost all paths have equal likelihood of being chosen. This may be a limitation of the baseline ACO algorithm being applied to a system of ants rather than a singular ant. In the ACO implementation for this project, each ant must keep fulfilling orders until all the orders are fulfilled. While this would minimize the time taken to deliver the orders, it actually increases the distance covered in total. There are different ways of encouraging an ant who has already covered a lot of distance to "head home early", in order to minimize distance. One of those is to introduce a new parameter that increases in value as the total distance traveled by an individual ant increases. This would add another dimension to the parameter space, and so add some additional computation time in order to find a more optimal solution. In its baseline formulation, the ACO method seems to need some extension in order to tackle this problem more effectively. There are many extensions in the literature, such as the Ant Colony System (ACS) approach by Dorigo and Gambardella,[7] and the Max-Min Ant System (MMAS) approach by Stützle and Hoos[15].

For the SA method, we noticed that the swap operator, on average, produced better solutions than the transfer operator. This may be from the fact that the swap operator allowed for orders to be switched within a single truck. This type of operation is important as the order in which a singular truck fulfills its route can greatly affect the total distance at the end. Further, because of the low number of repetitions and the distribution of the logger and mill locations, it is reasonable to assume that our choice of an initial solution affected our found optimal solution. Our initial solution was chosen by assigning a roughly equal amount of orders to each truck. But, because there were a few orders scattered around the perimeter of the possible locations, it would be more feasible for one or two trucks to complete fewer long orders and the rest to complete many shorter orders. A solution to this problem would be to implement both the swap and transfer operators that allowed for significantly different order length and took into account how many orders a truck was already fulfilling.

In general, one may wonder why use a heuristic at all. In the presence of more constraints, the MILP formulation may totally fail to generate any solution, while the heuristics can still generate some solution. Additionally, heuristics often scale better for larger data sets in terms of computational expense, whereas a strict solver like MILP may take an infeasible amount of time to find the optimal solution. Small scale comparisons like this give us some idea of the gap between the optimal solution and the heuristic solution, although it is far from definitive.

10 References

[1] Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation". In: Operations research 52.5 (2004), pp. 723–738.

[2] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. Linear programming and network flows. John Wiley & Sons, 2011. Chap. 9.

[3] Soufia Benhida and Ahmed Mir. "Generating subtour elimination constraints for the Traveling Salesman Problem". In: IOSR Journal of Engineering 8.7 (2018), pp. 17–21.

[4] Julien Bramel and David Simchi-Levi. "Set-covering-based algorithms for the capacitated VRP". In: The vehicle routing problem. SIAM, 2002, pp. 85–108.

[5] Guy Desaulniers, Oli BG Madsen, and Stefan Ropke. "Chapter 5: The vehicle routing problem with time windows". In: Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, 2014, pp. 119–159.

[6] Marco Dorigo. "Optimization, Learning and Natural Algorithms". In: (Jan. 1992).

[7] Marco Dorigo and Luca Maria Gambardella. "Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem". In: IEEE Transactions on Evolutionary Computation 1 (1997), pp. 53–66.

[8] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: Optimization by a colony of cooperating agents". In: IEEE Trans. Syst., Man, and Bybern, Part B 26 (Jan. 1996), pp. 29–41.

[9] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Positive Feedback as a Search Strategy". In: Tech rep., 91-016, Dip Elettronica, Politecnico di Milano, Italy (Apr. 1999).

[10] Karunakaran Haridass et al. "Scheduling a log transport system using simulated annealing". In: Information Sciences (2014), pp. 302–316.

[11] Haidar M Harmanani et al. "A Simulated Annealing Algorithm for the Capacitated Vehicle Routing Problem." In: CATA. 2011, pp. 96–101.

[12] Edward Kent, Jason Atkin, and Rong Qu. "Vehicle Routing in a Forestry Commissioning Operation Using Ant Colony Optimisation". In: Lecture Notes in Computer Science 8890 (Dec. 2014), p. 95. DOI: 10.1007/978-3-319-13749-0_9.

[13] Clair E Miller, Albert W Tucker, and Richard A Zemlin. "Integer programming formulation of traveling salesman problems". In: Journal of the ACM (JACM) 7.4 (1960), pp. 326–329.

[14] Hanif D Sherali and J Cole Smith. "Improving discrete model representations via symmetry considerations". In: Management Science 47.10 (2001), pp. 1396–1407.

[15] Thomas Stützle and Holger H. Hoos. "MAX MIN Ant System". In: Future Generation Computer Systems 16 (2000), pp. 889–914.

[16] Kay Chen Tan et al. "Heuristic methods for vehicle routing problem with time windows". In: Artificial intelligence in Engineering 15.3 (2001), pp. 281–295.

[17] Paolo Toth and Daniele Vigo. Vehicle routing: problems, methods, and applications. SIAM, 2014.