Forestry Vehicle Routing Problem with Binary Capacity and A Priori Consignments

Explorations in mixed-integer linear programming, simulated annealing, and ant colony optimization

MATH 691Y Project for AY 2022-2023: Abdulrahman Alenezi, William Howe, Lindsay Knupp, and Johnny Rasnic

Problem description - Forestry

In forestry operations, consignments between forests and sawmills are fulfilled by log transporting trucks. The network of orders that trucks operate on usually include constraints on

- capacity of the trucks
- the timing of the pickup and delivery of the order
- the amount of a trucks a forest/sawmill can concurrently load/offload

The network the trucks travel on can be non-Euclidean. The challenge of generating optimal or near-optimal routes for the trucks to travel on is called the vehicle routing problem (VRP).

Goal for a vehicle routing problem

- Optimization means minimizing total distance driven
- Brute forcing optimal solution often infeasible/unreasonably even on relatively small networks due to combinatorial explosion
- Introduce techniques which aim to simplify the computation of optimal route finding
- Acceptable to generate a quick solution, with little guarantee of optimality
- The above is called a **heuristic**, and is used to construct a sensible suboptimal solution in a reasonable time

Goal for this project

- Constraints:
  - VRP where each truck is either entirely full or empty (binary capacity)
  - Consignments are given to us a priori
- Performance measure: Total unloaded distance

Question: How well do the heuristics perform in comparison to the linear programming problem?

1. Mixed Integer Linear Programming
2. Ant Colony Optimization (heuristic)
3. Simulated Annealing (heuristic)

Data

- Real world data from a Scottish forestry operation [5]
- The data included
  - **coordinate data for the forests and sawmills**
  - **a list of orders for fulfillment**
  - **number of delivery trucks**
  - time windows for each order pickup and delivery
  - constraints on how many trucks can be loaded/offloaded concurrently at each location

Problem Formulation

- All trucks start and end at depot
- Only capable of carrying one forest's load
- Allow for multiple trips to forests
- No time constraints
- No max order constraints

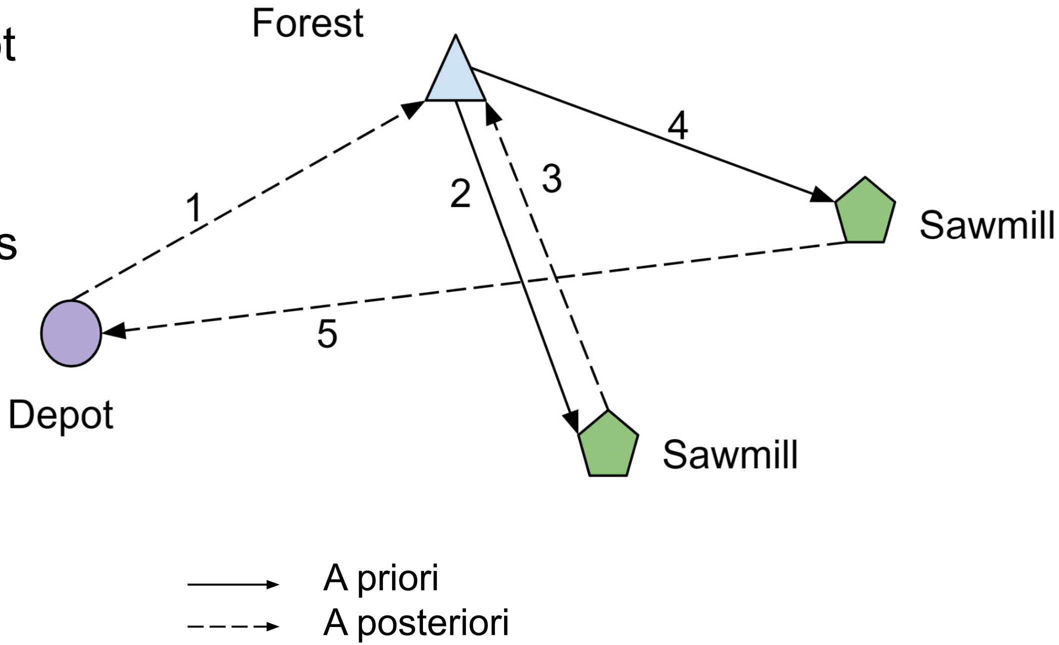
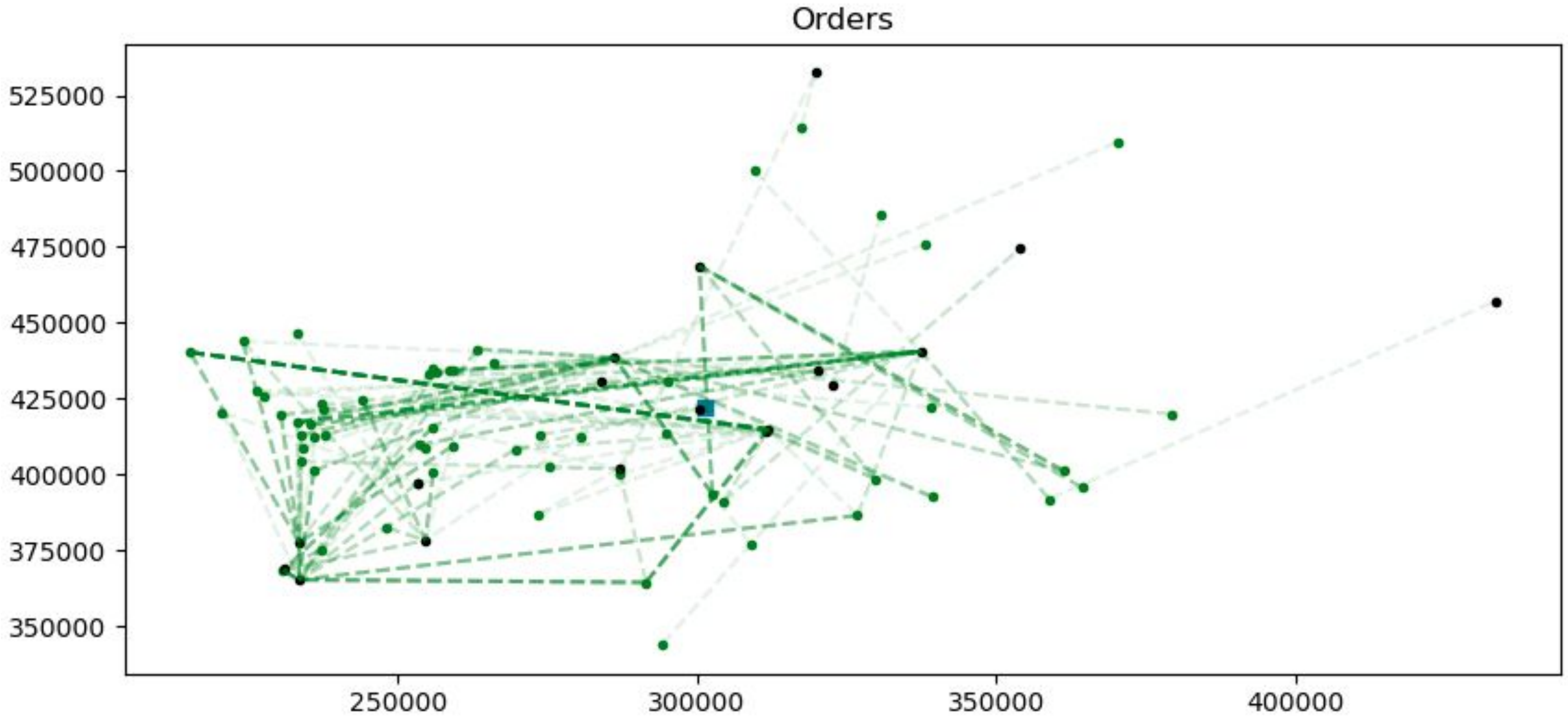


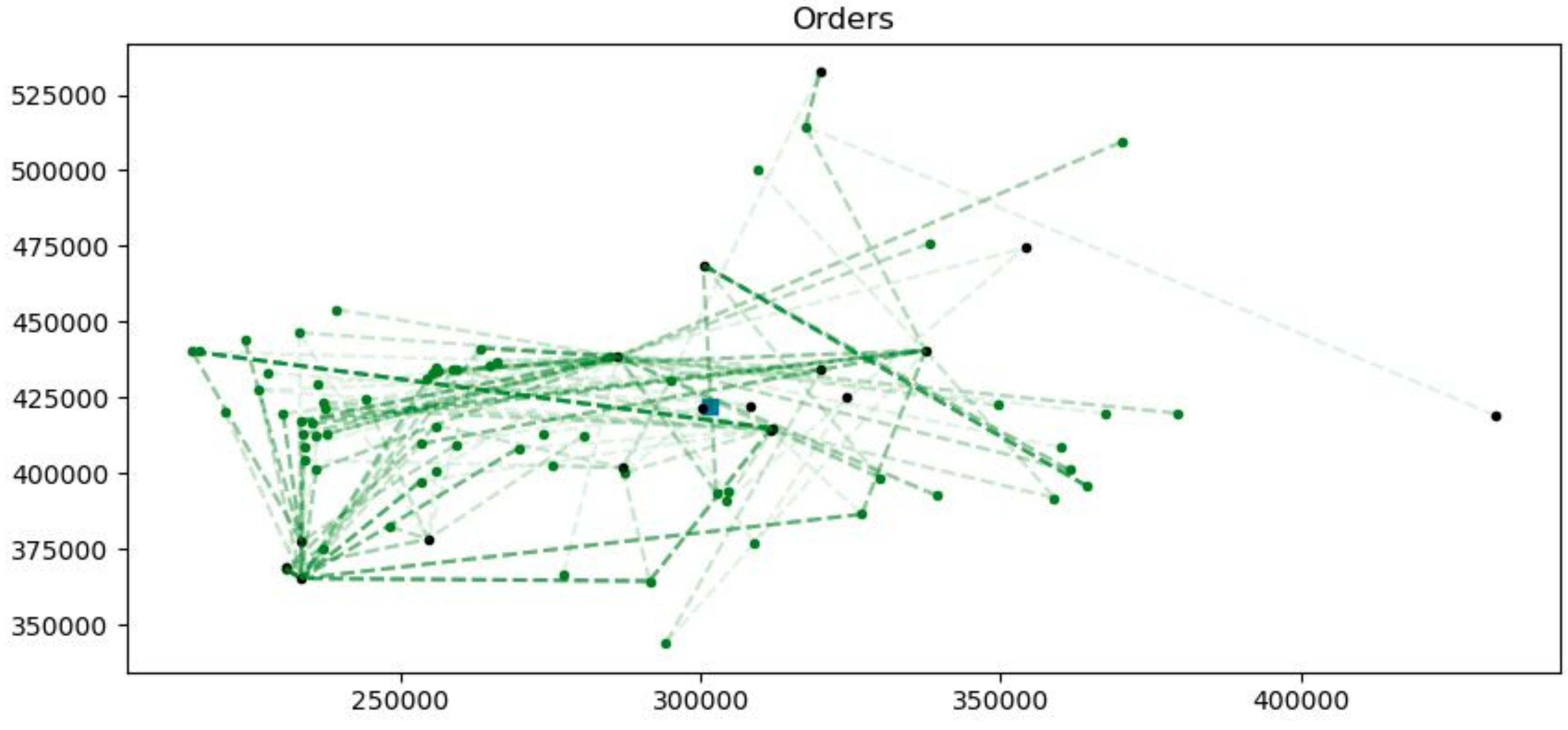
Figure 1: sample route for single truck fulfilling two orders

Order set 1

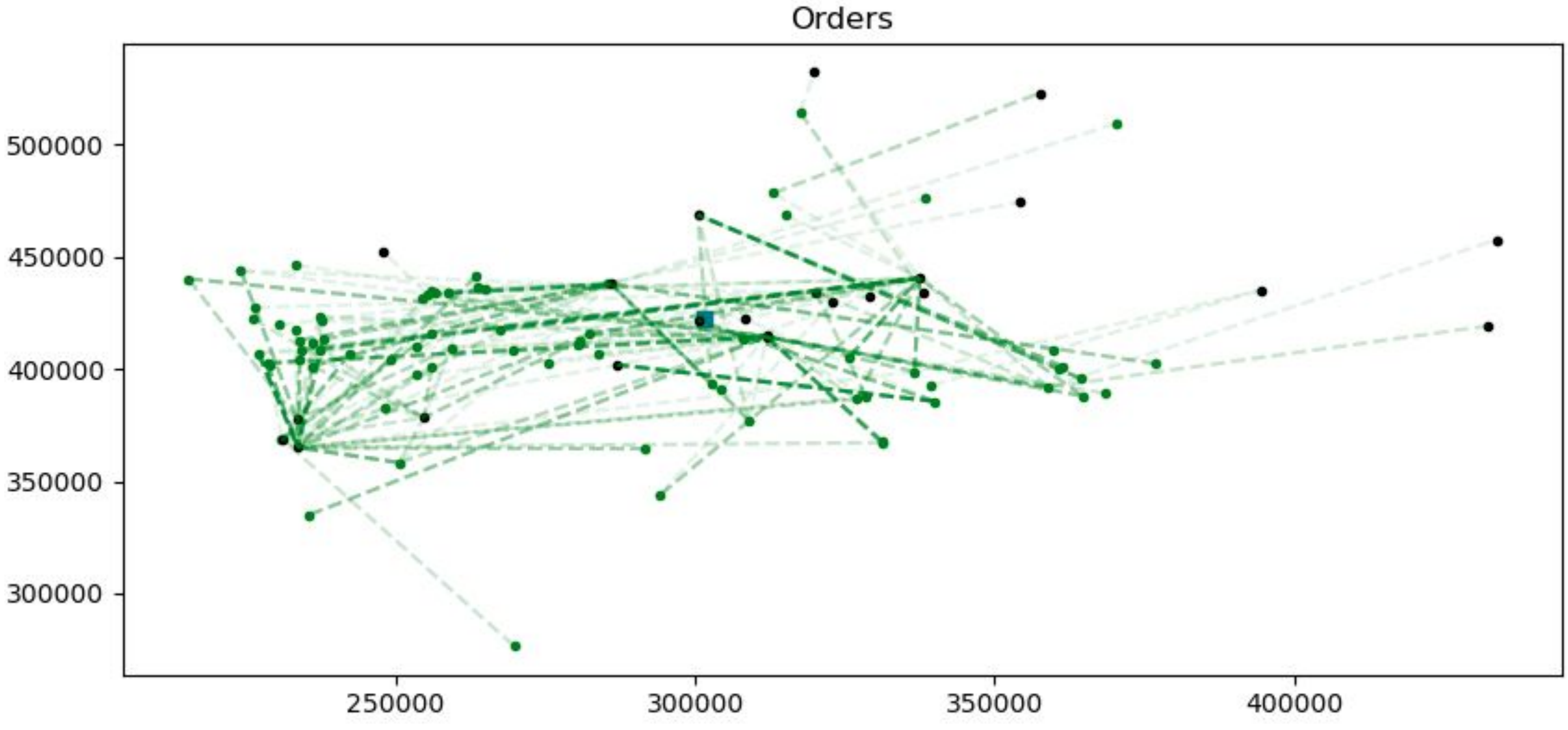


Orders
525000
500000
475000
450000
425000
400000
375000
350000
250000
300000
350000
400000

Order set 2

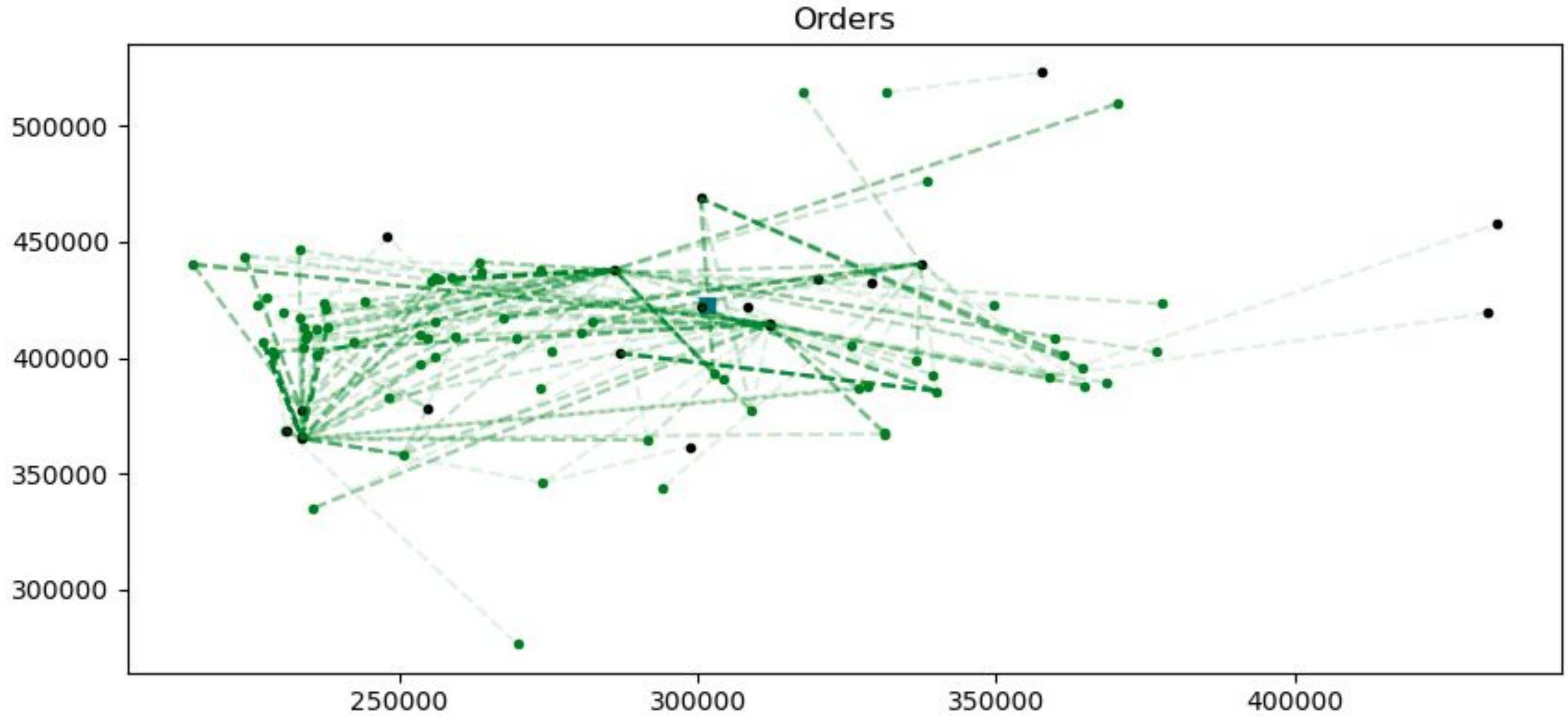

Orders
525000
500000
475000
450000
425000
400000
375000
350000
250000
300000
350000
400000

Order set 3

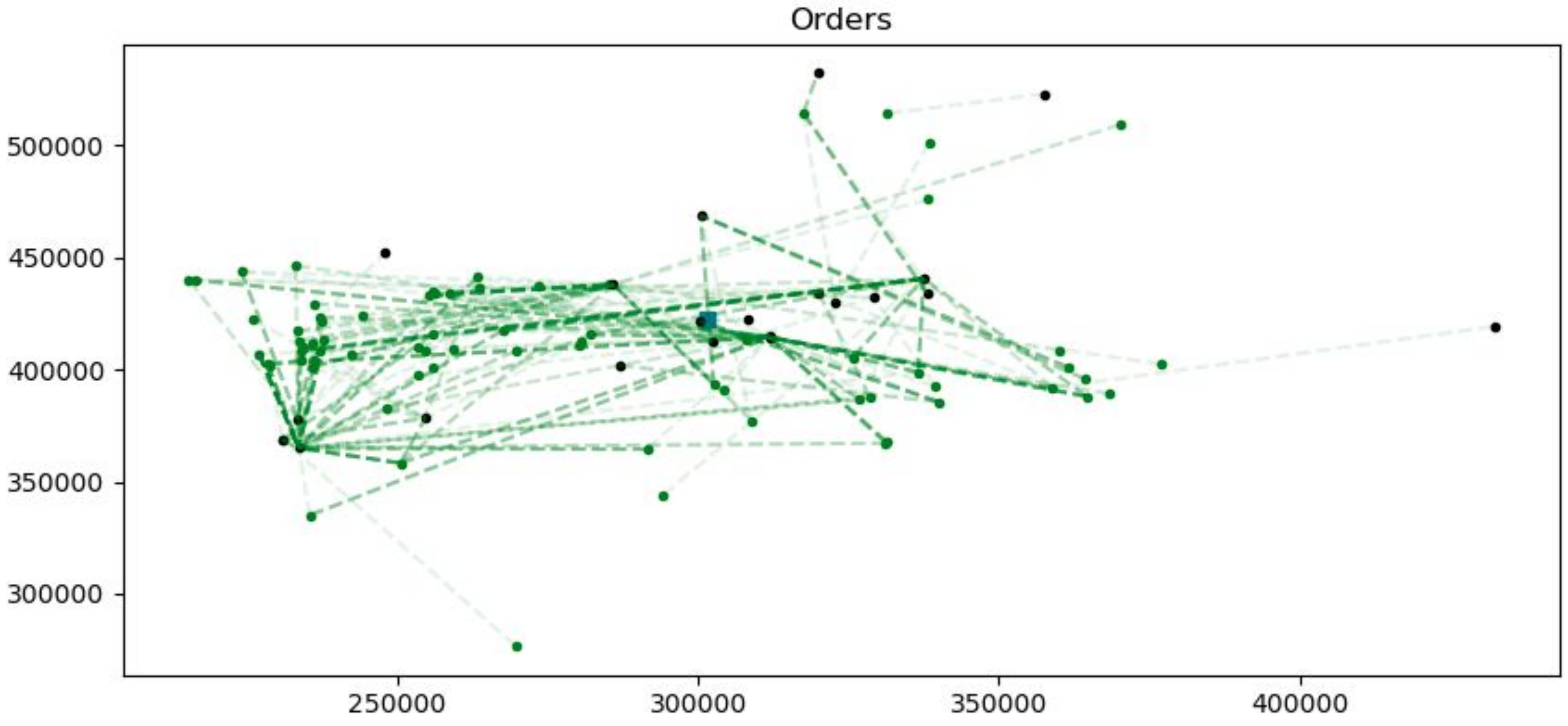


Orders
500000
450000
400000
350000
300000
250000
300000
350000
400000

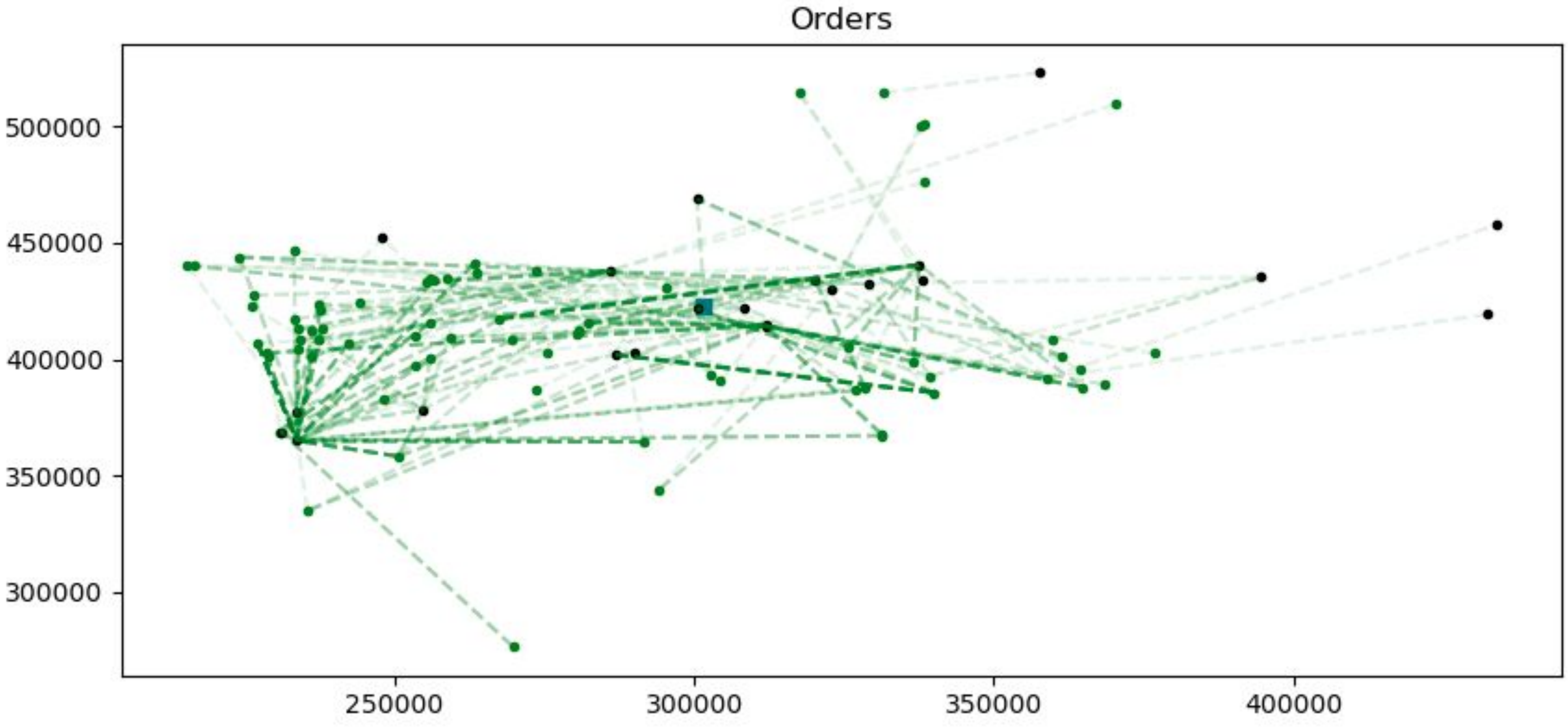
Order set 4


Orders
500000
450000
400000
350000
300000
250000
300000
350000
400000

Order set 5


Orders
500000
450000
400000
350000
300000
250000
300000
350000
400000

Order set 6


Orders
500000
450000
400000
350000
300000
250000
300000
350000
400000

Mixed Integer Linear Programming

- Linear Programming → a linear objective function is maximized/ minimized subject to constraints
- Mixed Integer Linear Programming → some variables are constrained to integers, others are not
- Common Elements
  - Destinations are modeled as vertices while a path is an edge
  - In two index formulations each binary variable represents the use of an arc by any vehicle
  - In three index formulations each possible arc truck pairing is represented by a binary variable
  - Vehicles are commonly assumed to travel at a constant speed → minimizing distance & time can be equivalent

Flow Problem

- In our forestry problem, can think of network as a flow problem
- A flow problem, in general, has the following characteristics
  - A thing being transported, in this case truck loads (x)
  - A source node where flow originates, which in this case is the depot (d)
  - A sink node where flow exits, which in this case is also the depot
  - Intermediary nodes where the incoming flow is equal to the outgoing flow
  - Directed edges (E) which connect nodes each having upper (u) ,lower (l) flow limits and cost (c)
  - An a priori order is equivalent to setting the upper bound (u) equal to the lower bound (l)

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

$$u_e \leq x_e \leq l_e \forall e \in E$$

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

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

Ant Colony Optimization

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 [1] and continually developed throughout the 90s [2][3]. “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.

Ant Path Selection

We can represent the probability that the *k*th ant takes a path *xy* by the following formula

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

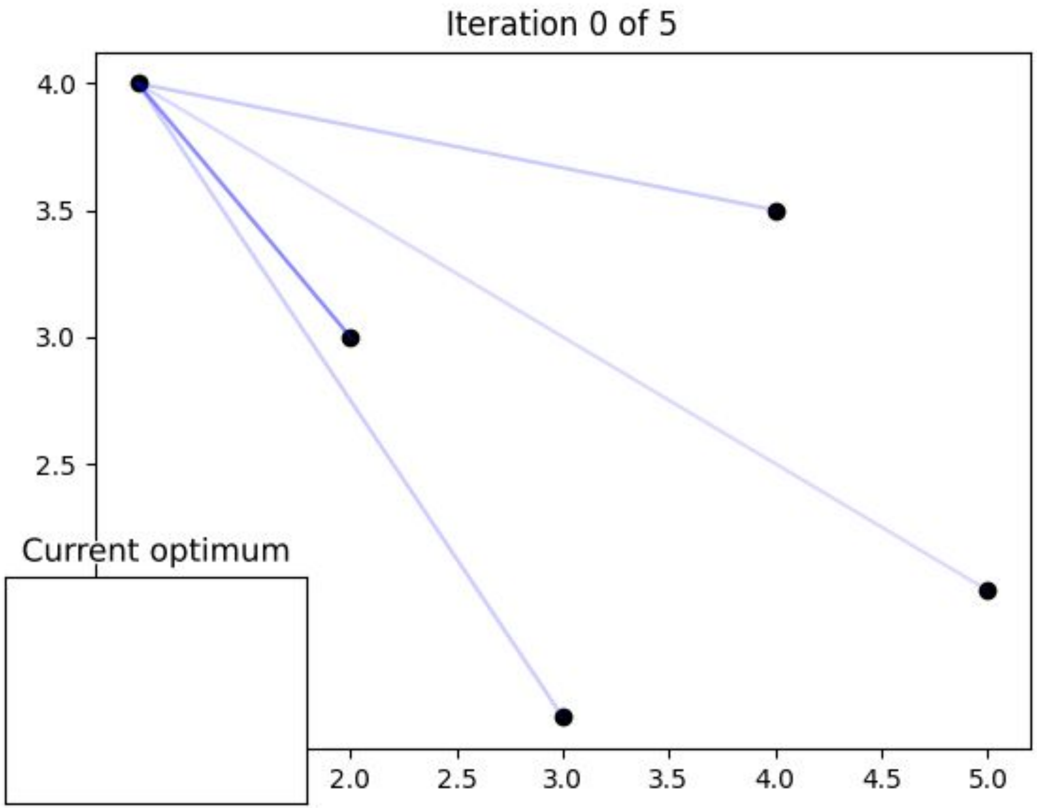
where tau denotes the pheromone weight, eta denotes the a priori adjustment term (usually the distance of *xy*), and alpha and beta are parameters that we can adjust to give more weight to the pheromone or a priori adjustment, respectively.

Pheromone Update Rule

Depending on the implementation of ACO, the pheromone on each edge may be updated after each traversal, or after a full solution is obtained. The update rule is as follows

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

where rho is the evaporation rate (between 0 and 1), and delta tau is the pheromone function, which is usually defined to be some constant divided by the length of the edge *xy*. The sum occurs over all instances of an ant taking the path *xy*.

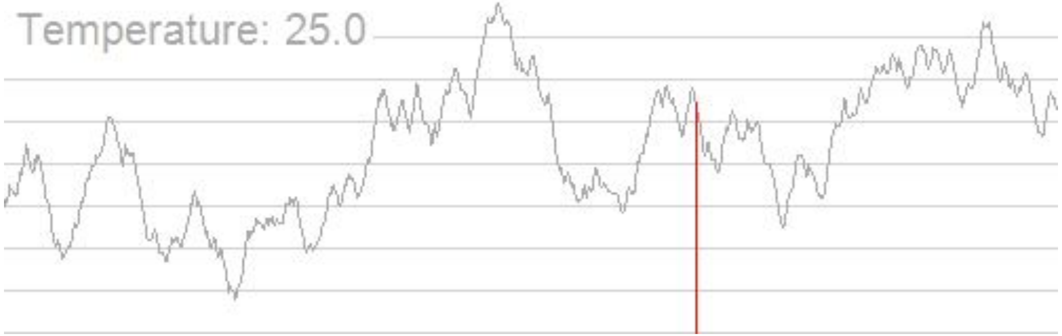

Iteration 0 of 5
4.0
3.5
3.0
2.5
Current optimum
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Ant Colony Optimization Idea

- We get to set the pheromone constant and evaporation rate parameters, as well as the level of influence of the pheromone and the *a priori* adjustment for a given path.
- The main idea behind ACO is that it can generate solutions very quickly, but will need some fine tuning of the parameters to improve the performance of the algorithm.

Simulated Annealing

- Heuristic that's modeled after slowly cooling metal
  - Meta-heuristic to solve complex large problems with a large solution space
  - Slowly decreasing temperature parameter that allows for the acceptance of possibly worse solution in search of the best solution → allows algorithm to escape local minima [1]

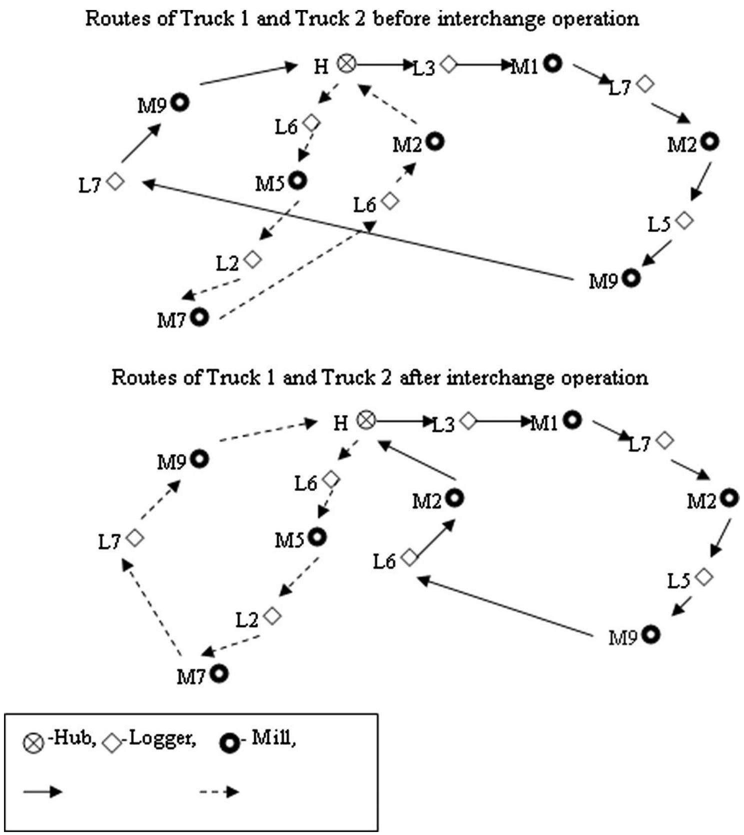


Generating solutions

- Only generate feasible solutions, so don’t need to worry about feasibility operations
- Operations modeled after Haridass et al

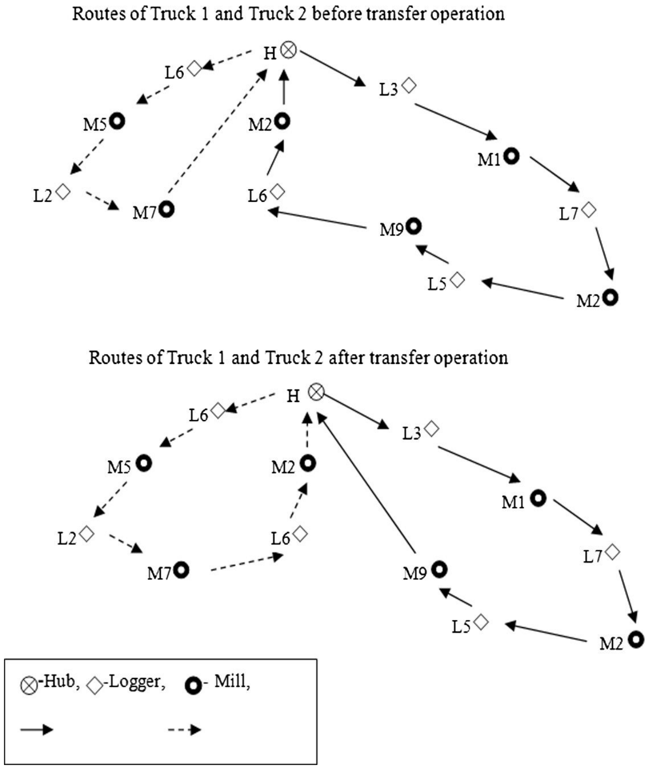
1. Swap operation -

    a - randomly select two trucks (could be the same truck), randomly select two orders and swap them



2 - Transfer operation

a. Randomly select two trucks (i and j), randomly select order from truck i and give to truck j



- When running simulations, swap actually proved to be the better operation for finding better solutions

Acceptance Probabilities

- Different authors use different probability thresholds for accepting worse solutions
- We used the threshold defined by Haridass et al and a fitness function based on total route distance

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

T - temperature

δ - difference in fitness function

Results

Orders’ distances:

Set 1: `18456286.3830882`

Set 2: `20355906.03452`

Set 3: `23649180.86164891`

Set 4: `24221688.901223615`

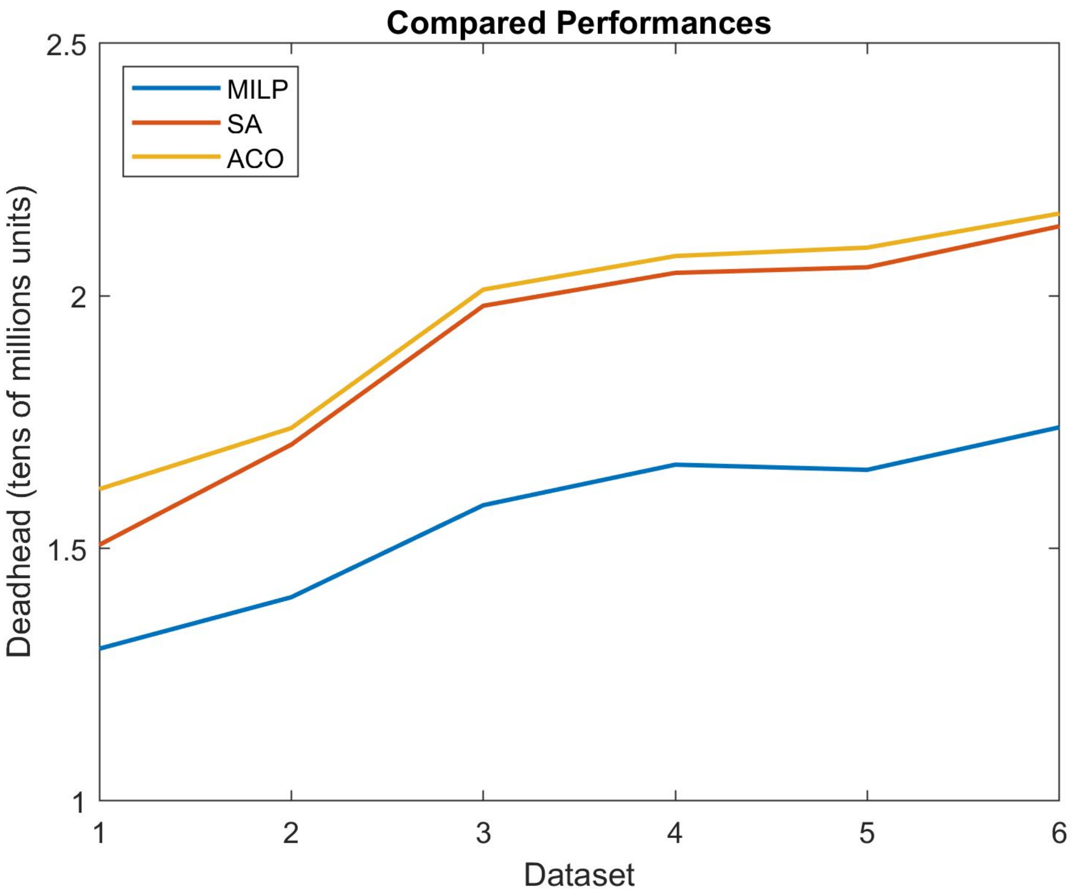
Set 5: `24050988.778515525`

Set: 6: `24780468.959954925`

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

Results



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

Deadhead Standard Deviation results (lower is better)

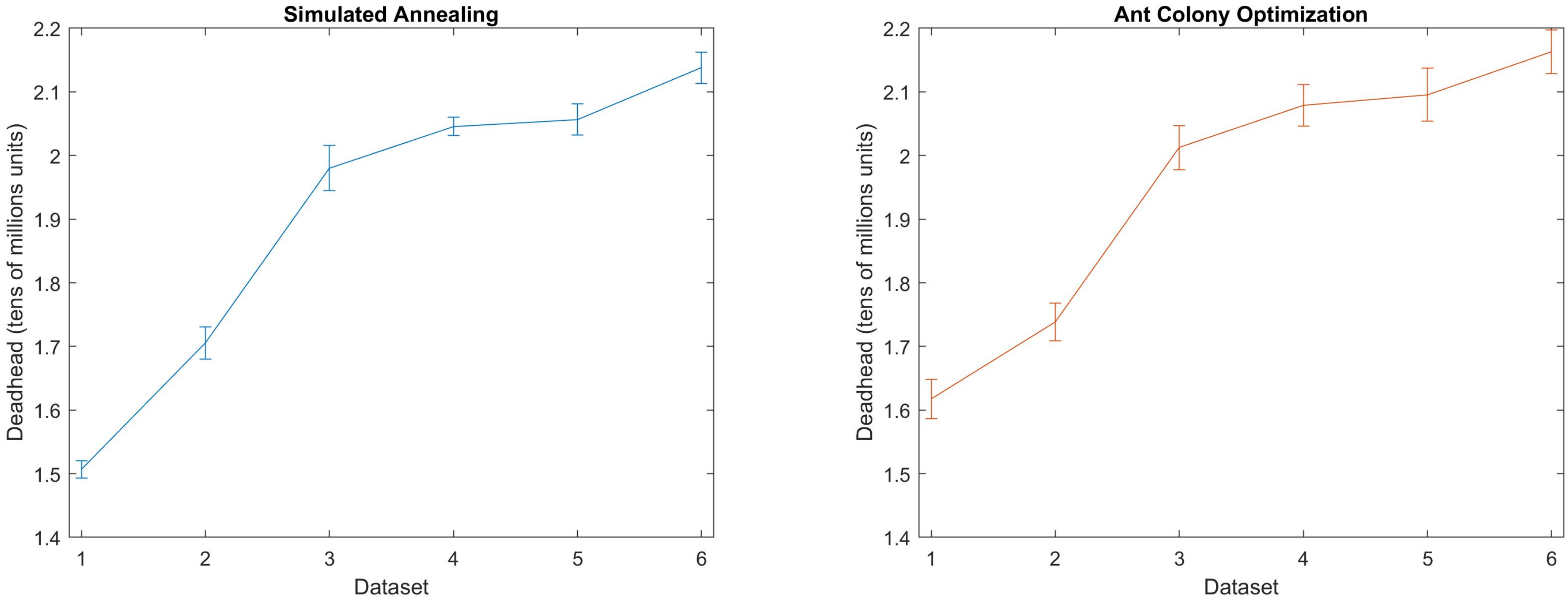
Dataset	SA	ACO
1	1.3608e05	3.0677e05
2	2.5119e05	2.9667e05
3	3.5220e05	3.4633e05
4	1.4263e05	3.2796e05
5	2.4417e05	4.1370e05
6	2.4624e05	3.4209e05

Differences between two random based heuristics?

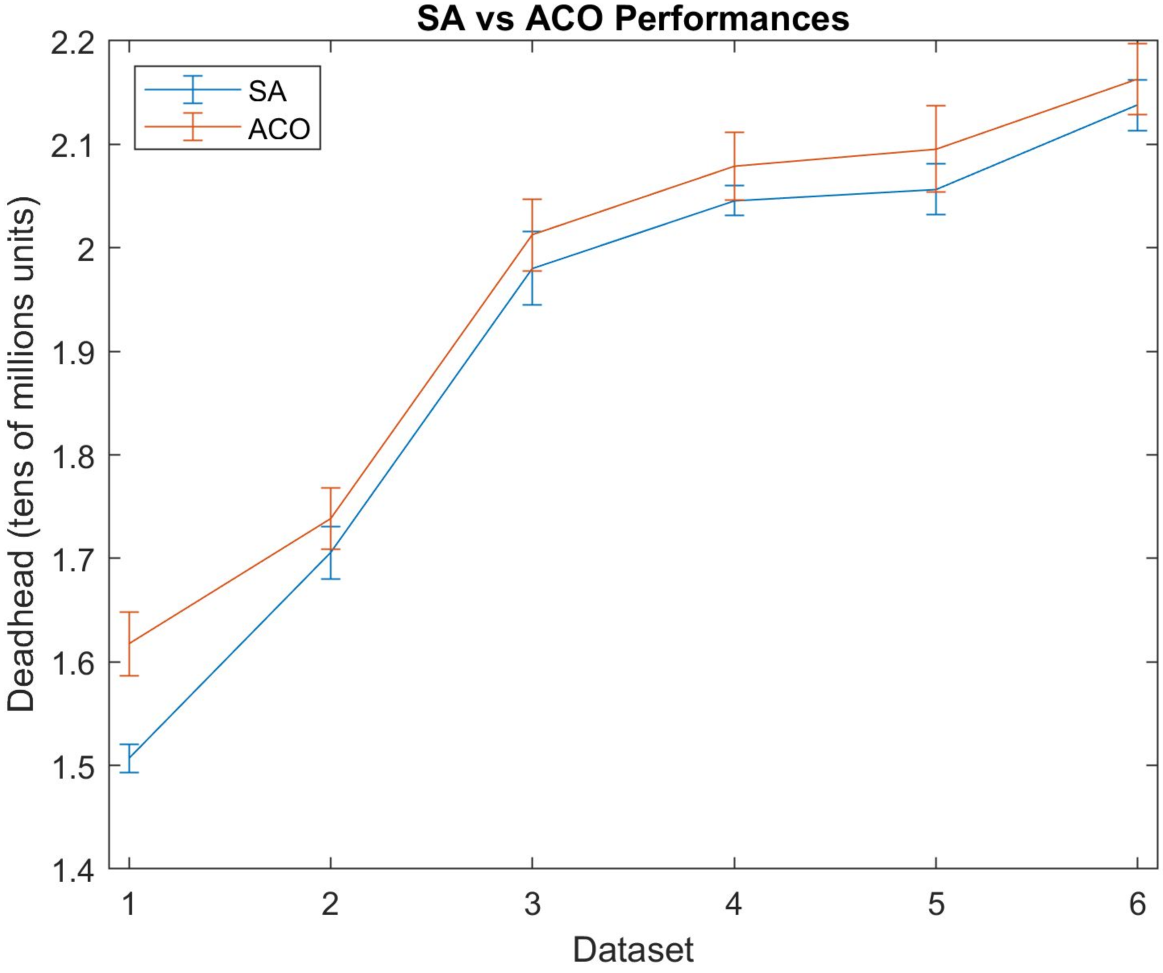
- ACO ran 100 trials
- SA ran 10 trials, w/ 549 temperature steps w/ 5 iterations per step

↓

Still relatively high variability



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



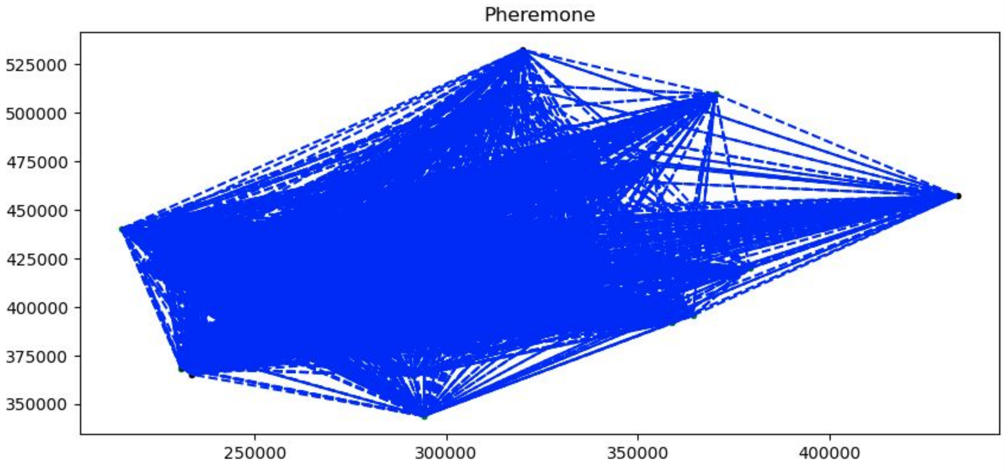
SA vs ACO Performances
SA
ACO
Deadhead (tens of millions units)
Dataset
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1.4
1
2
3
4
5
6

Discussion

- On average
  - MILP approach did ~3.3 million units better than SA
  - MILP approach did ~3.9 million units better than ACO
- Initially, longer orders led to SA performing worse, but difference eventually evened out
- Run more independent simulations to determine true difference

Discussion

- For ACO, it seems that the constraint of the trucks having nearly the same number of orders fulfilled “cancels out” the effect of pheromone and evaporation rate. Different evaporation rates seemed to have no effect on the solution.



Discussion

A possible extension to ACO that could cause trucks to have significant differences in order fulfillment is adding in some sort of probability that the ant could wait in place during solution finding, or returning home before all orders are fulfilled. This would involve creating another parameter to control how likely an ant is to do these actions.

References

[1] Marco Dorigo. “Optimization, Learning and Natural Algorithms”. In: (Jan. 1992).

[2] 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. 2002), pp. 29–41.

[3] 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).

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

References

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

[6] 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.

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

[8] 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.