Team Information

Name	**BU ID**	**SCC Username**
Antonio Alonso Montealegre	U66345178	aalonso1
Yang Lu	U67163043	ylu149
Yimin Xu	U59820673	yxu168
Christopher Gough	U26336811	cwgough

Abstract

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem that involves finding the optimal route for a set of vehicles to serve a given set of customers. The objective of this problem is to minimize the total distance or travel time, while satisfying all specified constraints. This problem is extremely similar to the Traveling Salesman Problem, with the main difference being that multiple salesmen can exist but cannot visit the same node.

The Capacitated Vehicle Routing Problem (CVRP) is a variation of the Vehicle Routing Problem (VRP) that introduces an additional constraint of capacity. In this problem, each customer has a specific demand, and the vehicles have a limited carrying capacity. The objective is to optimize the routing while also accounting for the capacity constraint, which is similar to the knapsack problem. The CVRP is an NP-hard problem, which means that finding an exact optimal solution for large instances is computationally challenging. As a result, heuristic algorithms and approximation techniques are commonly used to find near-optimal solutions. There are many different approximate solutions available for the CVRP, including iterative heuristic algorithms and large-scale metaheuristics such as simulated annealing and tabu search.

As part of this project, two different approaches were analyzed to solve the CVRP. Specifically, a homebrewed insertion heuristic algorithm was evaluated alongside the use of the Google OR-Tools VRP solver.

Instructions for Running the Code

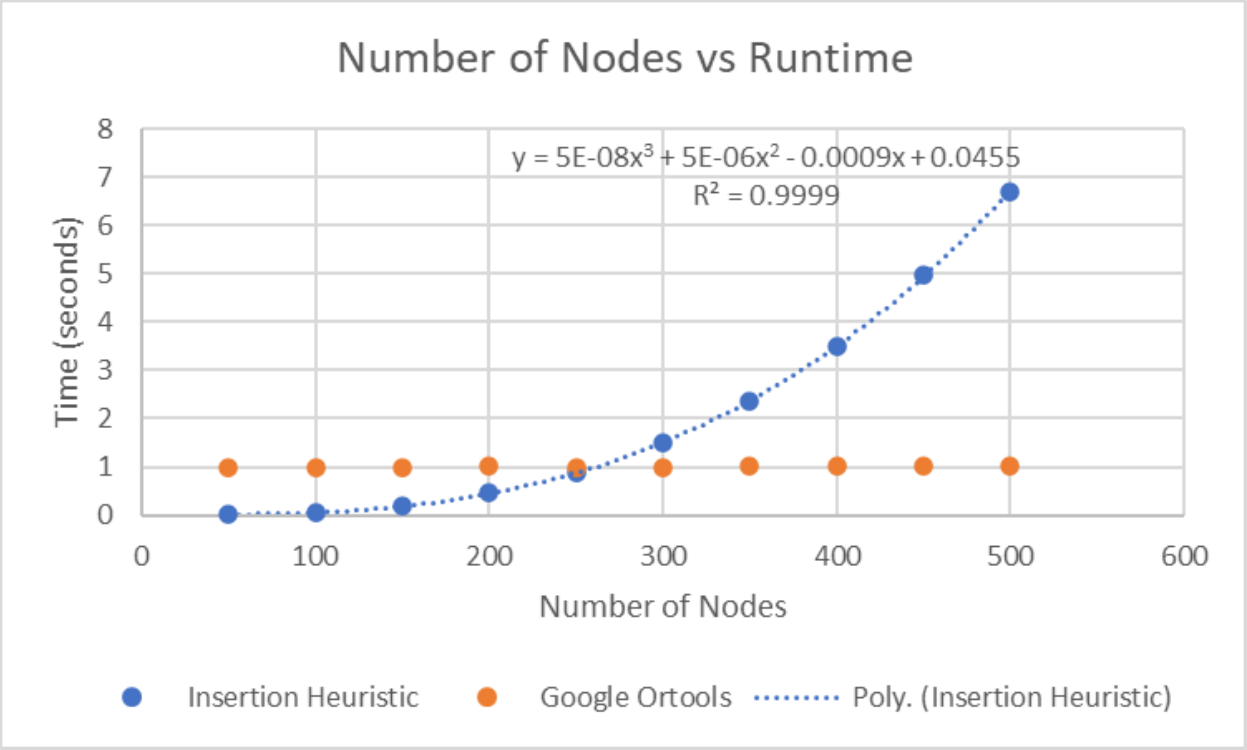
The Makefile is designed to run on a Linux distribution and was tested on the SCC. Once you've cloned the repository, run:

```
make all
```

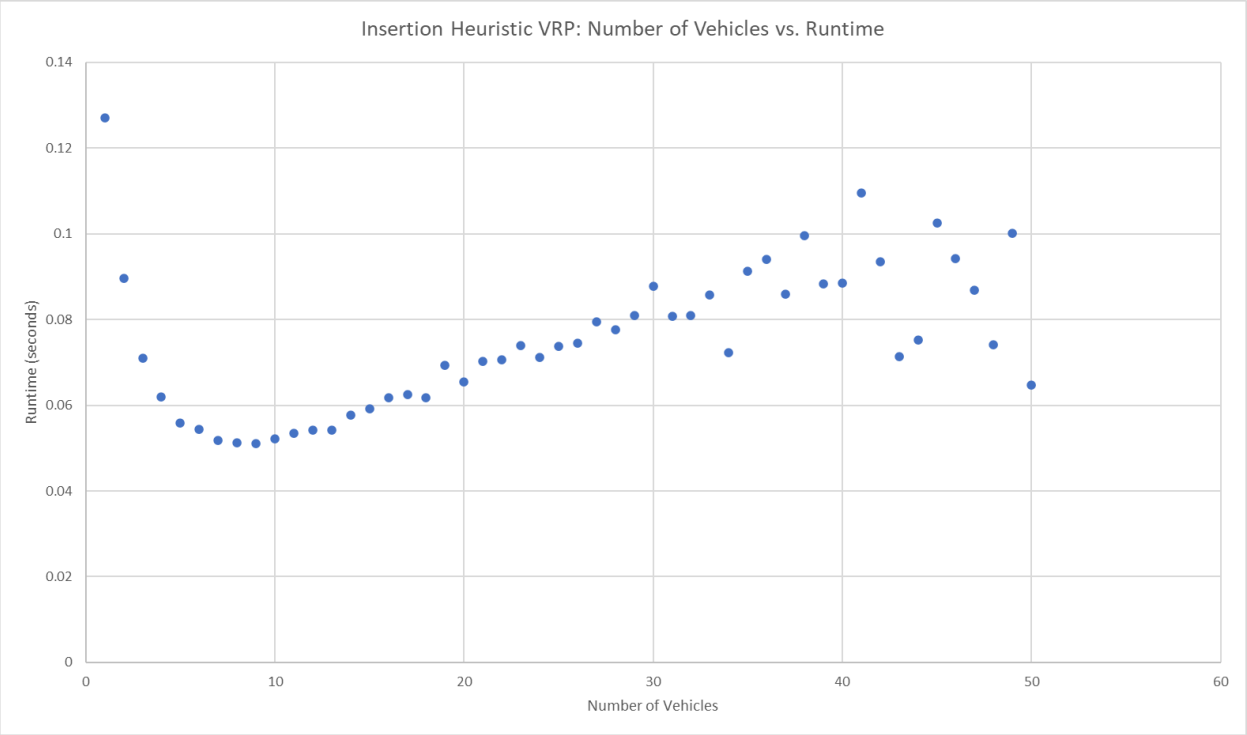
This will set up a virtual environment, install the necessary dependencies, and run the two programs. The visualized output graphs will be available as `.png` files in the root directory.

Sample Results with Discussion

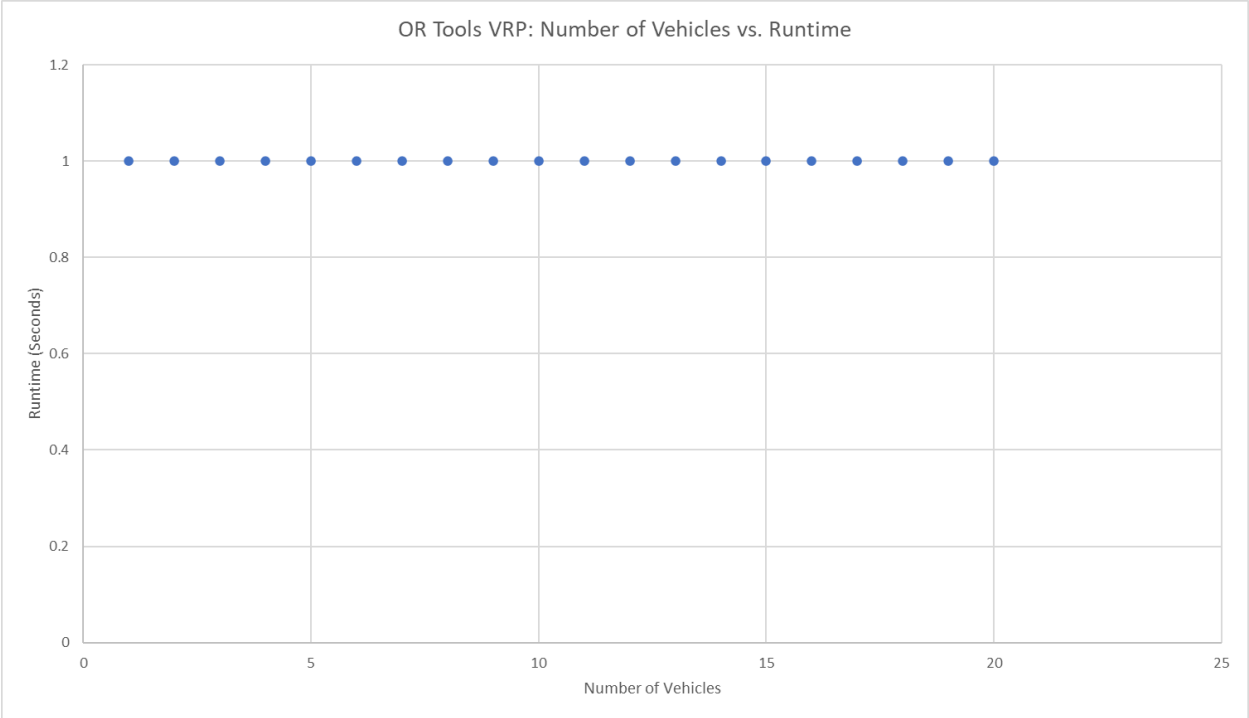
Note all results in this section were generated on a i7 12700k processor



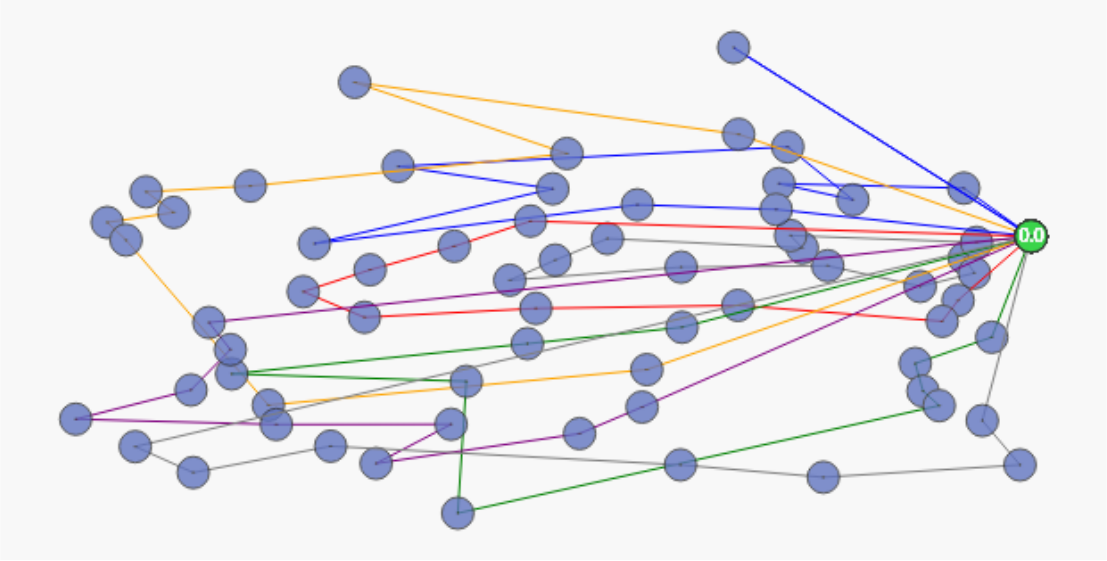
As shown in the graph above, the insertion heuristic runs much faster than the Google Ortools at least until the number of nodes exceeds approximately 270. After 270 nodes, the Google Ortools nearly dominates in terms of the average runtime as it stays almost linear. This effect really highlights the advantages of utilizing a metaheuristic approach to solving the CVRP.



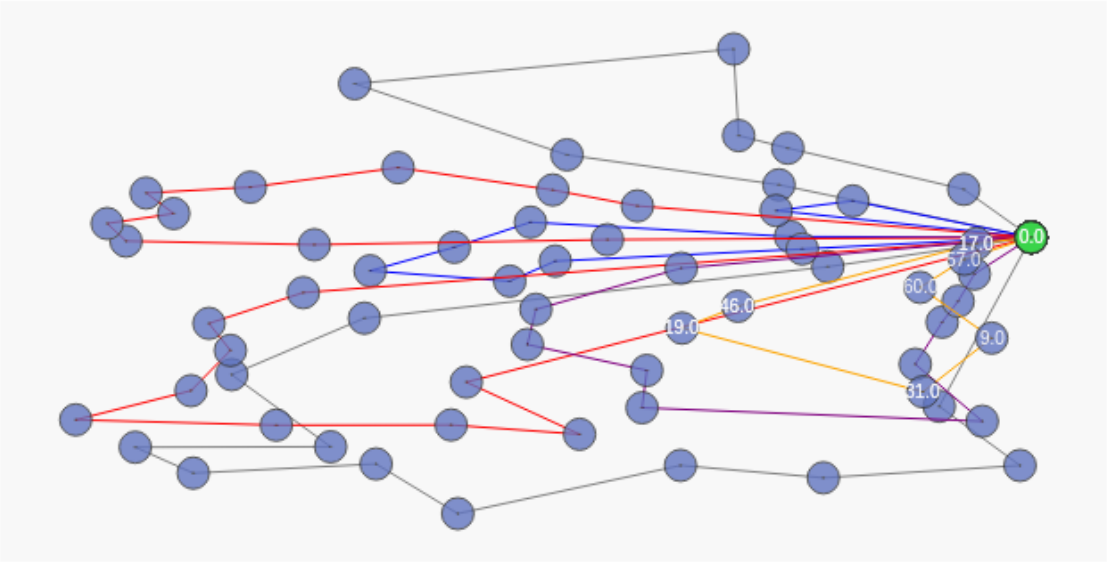
When examining the relationship between the number of vehicles and the runtime of the insertion heuristic, a distinct pattern emerges. Specifically, there appears to be an optimal number of vehicles—around 8 in this case—beyond which the runtime starts to increase again and becomes less predictable. This phenomenon can likely be attributed to the fact that, once the number of vehicles exceeds a certain threshold, some vehicles may end up with no work to do, leading to inefficiencies and longer processing times. Note this test was conducted for 100 total number of customers.



For the Google Ortools run, the number of vehicles does not seem to have any affect on the runtime. It seems to maintain the same performance regardless of the number of vehicles. Note this test was conducted for 100 total number of customers.



Output graph created by the insertion heuristic. The graph represents the optimal routes taken by six vehicles in Puerto Rico.



Output graph created by Google's Ortools given the same scenario.

In our Puerto Rico example, the number of nodes is equivalent to the number of cities we found data for, which is 66. The insertion heuristic therefore runs considerably faster for this small-scale scenario. However, the visualized output of both algorithms reveals that Ortools creates a significantly more efficient set of routes. The total edge weight outputted by the insertion heuristic is 16,842 meters, while the total edge weight from the Ortools implementation is 13,087 meters. In the case of Puerto Rico, Google OR-Tools generated a route that was 28.7% shorter than the insertion heuristic.

General Context of the Project

The insertion heuristic algorithm is a commonly used approach to solve the Capacitated Vehicle Routing Problem (CVRP). Its main idea is to iteratively insert customers into a route while minimizing the cost added per customer.

The algorithm starts with an empty solution, with no customers assigned to any route. To find the optimal route for each customer, the algorithm selects an unassigned customer and chooses the route that will incur the least additional cost to insert the selected customer. The selected customer is then inserted into the chosen route at the best position. This process is repeated for all customers until all customers have been assigned to a route.

The time complexity of such an algorithm for solving CVRP is usually $\Theta(n^2)$, where n is the number of customers in the problem instance. However, in the worst case scenario, when the algorithm needs to repeatedly analyze every single route, the time complexity can be closer to $O(n^3)$.

The pseudocode for generating such an algorithm is shown below:

Algorithm 1 Insertion Heuristic

```
1. N = set of unassigned customers
2. R = set of routes; always contains the empty route; initially contains
   only the empty route
3. while N ≠ ∅ do
4.    p* = −∞
5.    for j ∈ N do
6.       for r ∈ R do
7.          for (i − 1, i) ∈ r do
8.             if Feasible(i, j) and Profit(i, j) > p* then
9.                r* = r
10.               i* = i
11.               j* = j
12.               p* = Profit(i, j)
13.            end if
14.         end for
15.      end for
16.   end for
17.   Insert(i*, j*)
18.   N = N \ j*
19.   Update(r*)
20. end while
```

(Campbell and Savelsbergh, 2004)

Google ORtools on the other hand utilizes built in metaheuristic algorithms to perform a solve for the CVRP given an initial solution. Currently the initial solution is obtained via a greedy algorithm that simply chooses the nearest node in reference to the last added node of a route. ignoring capacity constraints.

Once an initial solution is obtained, the built in metaheuristic algorithms such as greedy descent, guided local search, simulated annealing, and tabu search are then utilized to solve and optimize the solution.

The greedy descent algorithm is a greedy algorithm that iteratively improves the solution by making re-assignments that optimize distance. The guided local search algorithm is an algorithm that gets to a local minimum solution by performing swaps between routes. Once a

local minimum is achieved, random changes are then introduced to better optimize the solution. Simulated annealing is a metaheuristic algorithm that can be used to escape local minima by accepting solutions that are worse than the current solution with a certain probability, in order to avoid getting stuck in a suboptimal solution. Tabu search is another metaheuristic algorithm that prevents revisiting previous solutions or neighborhoods, in order to encourage exploration of new, potentially better solutions. These techniques are commonly used in data analysis to improve the accuracy of optimization algorithms.

Overall, it is expected that the metaheuristic algorithms used by Google ORtools to outperform the homebrewed insertion heuristic algorithm at a high number of nodes.

References

Campbell, Ann Melissa, and Martin Savelsbergh. “Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems.” Transportation Science, vol. 38, no. 3, 2004, pp. 369–378., https://doi.org/10.1287/trsc.1030.0046.

“Vehicle Routing Problem | or-Tools | Google Developers.” Google, Google, https://developers.google.com/optimization/routing/vrp?hl=zh-cn.