Optimizing Routes to Deliver Packages

Chen Jiang, Varsha Shukla, Ang Xiao

Introduction

Courier services have become crucial for transporting packages, documents, and other valuable items. Today's customers expect fast delivery, making it essential for courier companies to invest in fuel, equipment, maintenance, and wages. Efficient routing and allocation methods can significantly reduce delivery time, fuel consumption, carbon emissions, and improve asset utilization and customer service. Our team used VRP and TSP algorithms to experiment and simulate solutions for logistics companies facing these challenges. Even small improvements can have a significant impact, making efficient routing and allocation methods vital for courier services[1].

Our team has identified two sub-questions to address the challenges of package delivery:

1. How can delivery routes be split most effectively between multiple trucks, given the number of packages and the delivery center's coverage area? This is known as the Vehicle Routing Problem (VRP), and solving it optimally can result in significant improvements.

2. Once packages are in the truck and a set of delivery addresses has been provided, what is the most efficient order to visit them? This problem is called the Traveling Salesman Problem (TSP), and finding the optimal solution can help minimize travel time and costs.

By addressing these two questions, our team aims to improve the efficiency of package delivery, reduce costs, and enhance the overall customer experience.

Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a challenging optimization problem that requires finding the most efficient routes for a fleet of vehicles to visit a set of locations while satisfying specific constraints. The goal of VRP algorithms is to minimize the total distance traveled by the vehicles while meeting constraints such as vehicle capacity, time windows, and customer preferences. Various variants of VRP exist, such as the CVRP, VRPTW, PDP, MDVRP, and SDVRP. Each variant focuses on a specific aspect of the problem, such as vehicle capacity, time windows, or multi-depot visits.

Assumptions

In order to simplify the solution of the VRP problem, we have made the following assumptions:

- One-way travel: The vehicles only travel in one direction and perform either pure pickups or pure deliveries.

- Single distribution center: There is only one distribution center or depot from which the vehicles start and end their routes.
- Undividable demand: Each customer's demand can only be fulfilled by a single vehicle.
- Looping: The vehicles must return to the distribution center after completing their delivery tasks.
- Sufficient vehicles: There are no limitations on the number of vehicles available for delivery, meaning the demand for delivery vehicles can be fully met.
- Non-full load: The demand for any customer point is less than the maximum load capacity of the vehicle.

Clarke and Wright savings algorithm [1]

The Clarke and Wright savings algorithm is a heuristic algorithm commonly used in vehicle routing problems. The algorithm is based on the concept of savings, which is defined as the reduction in distance that can be achieved by merging two separate routes into one.

1. Calculate the savings for all possible pairs of customers. The savings represent the reduction in cost that can be achieved by combining the two customers into a single route.

2. Sort the savings in descending order and create a list of candidate routes, where each route initially consists of a single customer

3. For each pair of customers with the highest savings, check if they can be added to an existing route without violating any capacity constraints. If yes, merge the routes and update the savings list. If no, move to the next pair of customers with the highest savings. Repeat this step until no more customers can be added to existing routes.

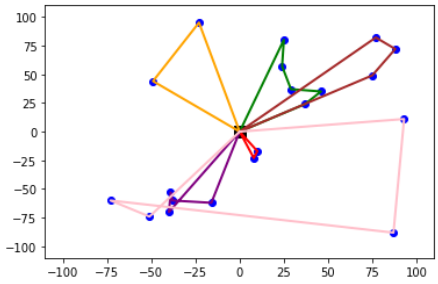
Time Complexity

To calculate the savings, it takes $O(n^2)$ time, since there are $n(n-1)/2$ possible pairs to compute. To sort the savings using Quicksort, it takes $O(n^2 \log n)$ time, since there are $n(n-1)/2$ pairs to sort. To merge the routes, $n(n-1)/2$ pairs need to be merged into k final routes. Therefore, the time complexity of merging routes is $O(n^2 - k + 1)$.

The total time complexity of the algorithm is $O(n^2) + O(n^2 \log n) + O(n^2)$, which simplifies to $O(n^2 \log n)$.

Results

The blue points in the following graph represent customers, while the different colored lines represent vehicles. Our goal is to cluster the customers into different groups based on their location, and assign each group to a specific vehicle. Once we have these clusters, we will use the Traveling Salesperson Problem (TSP) algorithm to find the best route for each vehicle to visit all the customers in their assigned cluster.



The Travelling Salesman Problem

The traveling salesman problem (TSP) asks the following question: Given a list of nodes and the distances between each pair of nodes, what is the shortest possible route that visits each node exactly once and returns to the origin node? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Bellman-Held-Karp algorithm [2]

The state space of the algorithm is represented by a matrix that stores the subproblem solutions for all possible subsets of nodes in the graph. The base case of the algorithm is the solution for the shortest path between the starting node and a single destination node. The recurrence relation of the algorithm is used to compute the optimal solutions for the subproblems.

The Bellman-Held-Karp algorithm has the following steps: initialize the matrix with the base case solutions, compute the optimal solution for each subset of nodes using the recurrence relation, and finally compute the shortest path that visits all the nodes and returns to the starting node. The time complexity of the algorithm is O(n^2 * 2^n), where n is the number of nodes in the graph.

In summary, the dynamic programming algorithm for TSP is a popular approach for finding the optimal Hamiltonian cycle in a complete graph. It breaks the main problem into a set of subproblems and uses a recurrence relation to compute the optimal solutions for the subproblems. Despite its high time complexity, the dynamic programming algorithm is widely used due to its efficiency and accuracy in solving TSP instances.

Annealing Algorithm [3]

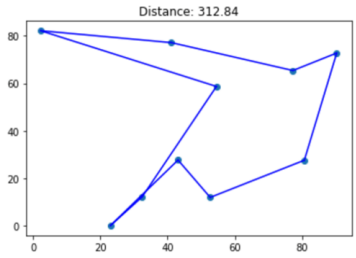
Simulated Annealing (SA) is a general optimization algorithm that mimics the annealing process of solid materials with a heating, isothermal, and cooling process. It is probabilistic and aims to find global optimal solutions by disrupting the original order and avoiding local optimal solutions. The algorithm's main advantage is its ability to find global optimal solutions, but it can be time-consuming for solving high-dimension questions and may only deliver an approximate optimal solution.

Simulation:

First, a "cities" variable is defined, which contains the coordinates of all the cities. Then, an "euclidean_distance" function is defined to calculate the Euclidean distance between two cities. Next, a "calculate_path_length" function is defined to calculate the length of the path that passes through all cities for a given path.

Then, a "simulated_annealing" function is defined, which is the main part of the simulated annealing algorithm. The parameters of this function include city coordinates, initial temperature, cooling factor, and maximum number of iterations. The function initializes the current path as a randomly ordered city list and calculates the current path length. Then, a loop is used to continuously iterate, generating a new path and calculating the length of the new path each time. If the length of the new path is shorter than the current path length, the new path is accepted. Otherwise, the new path is accepted with a certain probability. The temperature is reduced at each iteration, controlling the probability of accepting a new path to gradually decrease. The algorithm ends when the temperature drops to a certain level or the maximum number of iterations is reached. Finally, the current path and path length are returned.

Finally, the simulation repeats the simulated annealing function 1000 times in a loop and records the shortest path and its length. Finally, the shortest path and path length are output.



The output of the Annealing Algorithm simulation

Conclusion

In conclusion, our team explored the feasibility of utilizing the Clarke and Wright savings algorithm for assigning delivery points to vehicles in logistics companies and attained promising initial results. The experiments revealed that this algorithm can effectively optimize the delivery process and maximize efficiency. In addition, we have explored the use of two distinct algorithms for planning delivery routes for each vehicle's assigned delivery points. After comparing the simulated annealing algorithm and dynamic programming algorithm, our preliminary simulation experiments indicated that the simulated annealing algorithm is better suited for managing larger city businesses with more destinations, as it is more tolerant of uncertainty. Conversely, the Bellman-Held-Karp algorithm is better suited for smaller city businesses due to its ability to find the best solution, although it may be slower for larger datasets. Algouth these algorithms have certain limitations and are somewhat idealized, our findings suggest that improving and employing them based on our experiments can assist logistics companies in optimizing their delivery processes and enhancing their efficiency.

References

[1] Clarke, Geoff, and John W. Wright. "Scheduling of vehicles from a central depot to a number of delivery points." Operations research 12, no. 4 (1964): 568-581.
[2] Bellman, Richard. "Dynamic programming treatment of the traveling salesman problem." Journal of the ACM (JACM) 9, no. 1 (1962): 61-63.
[3] Van Laarhoven, Peter JM, Emile HL Aarts, Peter JM van Laarhoven, and Emile HL Aarts. Simulated annealing. Springer Netherlands, 1987.