Amazon Delivery Truck Scheduling Project

Authors:

Arshan Saniei-Sani
UTEID: AS97297
TACC User: as97297
Email: arshansani@gmail.com

Arsh Dauwa
UTEID: AE26933
TACC User: ArshDauwa
Email: arshdauwa04@gmail.com

Table of Contents

Introduction:

Problem Statement:
The problem at hand revolves around the efficient utilization of optimization algorithms in the context of route planning, specifically for the delivery truck scheduling scenario. The overarching objective is to determine the most optimal routes for delivery trucks, minimizing both time and resource utilization while adhering to various constraints such as delivery deadlines and geographical factors.

Theoretical Background:
The theoretical underpinning of this problem draws heavily from graph theory, combinatorial optimization, and heuristic algorithms. In the realm of delivery logistics, the Traveling Salesman Problem (TSP) serves as a fundamental model, where the challenge is to find the shortest possible route that visits a set of cities and returns to the starting city. The use of optimization heuristics, such as the Kernighan-Lin algorithm and the 2-opt heuristic, comes into play to address the complexity of the TSP. These algorithms aim to iteratively refine the delivery routes by swapping or reversing segments, seeking to minimize the total distance traveled.

Historic Background:
Historically, the optimization of delivery routes has been a critical challenge for various industries, from postal services to modern e-commerce giants. Early solutions often relied on manual planning or simplistic algorithms, leading to suboptimal routes and increased operational costs. With the advent of computational methods, the application of optimization algorithms has become more prevalent, revolutionizing the efficiency of delivery logistics.

Global Approach:
To address the problem, a comprehensive approach has been adopted. The initial focus involved understanding the intricacies of the delivery truck scheduling problem, including constraints such as delivery time windows, vehicle capacity limitations, and geographical considerations. Theoretical foundations, including TSP and optimization heuristics, were studied to devise strategies for route optimization. Practical implementation involved the creation of algorithms based on the theoretical frameworks studied. The Kernighan-Lin algorithm, leveraging the opt2 idea, was incorporated to iteratively refine routes. Additionally, the 2-opt heuristic was employed to explore alternative route configurations.

Brief Statement of Findings:
Preliminary findings indicate that the implemented optimization algorithms have the potential to significantly improve the efficiency of delivery truck routes. By iteratively refining the routes through heuristic approaches, we observed reductions in total distance traveled, thereby optimizing resource utilization and adhering to delivery constraints.
However, further testing and validation on diverse datasets are required to assess the generalizability and robustness of the proposed approach. The findings represent a promising step toward enhancing the field of delivery logistics through computational optimization methods.

Experiments:

We implemented a suite of unit tests to rigorously assess the functionality of our classes and methods. These tests are designed to systematically evaluate the behavior of the Address, AddressList, and Route classes under different scenarios and inputs. Each test focuses on specific functionalities, such as distance calculations, address addition, total distance computations, and route optimizations. By establishing a set of expected results for each test, we create a benchmark against which the actual outcomes are compared. The pass/fail evaluation provides clear feedback on the correctness of our code.

These unit tests not only serve as a means of validating the individual components of our program but also act as a form of documentation, outlining the expected behavior of our methods. They are invaluable in ensuring the reliability and robustness of our code, enabling us to catch and address potential issues early in the development process. As the project evolves, additional tests can be seamlessly integrated to accommodate new functionalities and edge cases, maintaining a high level of code quality and confidence in the correctness of our delivery truck scheduling implementation.

Test Name	Inputs	Expected Results	Actual Results	Pass/Fail
TestAddressDistance	Addresses (0,0) & (3,4)	7	7	Pass
TestAddressHasSameLocation	Addresses (0,0) & (3,4)	TRUE	TRUE	Pass
TestAddressListAddAddress	Addresses (0,0) & (5,2)	2	2	Pass
TestAddressListTotalDistance	Addresses (0,0), (3,4), (6,8)	14.0	14.0	Pass
TestRouteInitialization	N/A	Depot Address	Depot Address	Pass
TestRouteTotalDistance	Addresses (3,4), (6,8)	28.0	28.0	Pass
TestRouteGreedyRoute	Addresses (9,1), (6,6), (1,3), (5,4)	30.0	30.0	Pass
TestRouteOptimizeRoute	Addresses (9,1), (6,6), (1,3), (9,4)	30.0	30.0	Pass

Detailed presentation of your work:

Greedy Search Optimization:

```
void GreedyRoute(){
        Address depot = GetDepot();
        addresses_.pop_back();

        std::vector <Address> optimized_route;
        std::vector <bool> visited(addresses_.size(), false);

        visited[0] = true;
        Address we_are_here = depot;
        optimized_route.push_back(depot);

        for (size_t i = 1; i < addresses_.size(); ++i) {
                int closest_index = IndexClosestTo(we_are_here, visited);
                if (closest_index != -1 & & !visited[closest_index]) {
                        we_are_here = addresses_[closest_index];
                        optimized_route.push_back(we_are_here);
                        visited[closest_index] = true;
                }
        }
```

This code is a algorithm for the greedy search optimization. The greedy search algorithm is a simple heuristic method that makes the locally optimal choice at each stage with the hope of finding a global optimum. In the context of optimization problems, including route planning, the greedy search algorithm iteratively selects the best available option at each step, without considering the global picture or future consequences. The algorithm makes decisions based solely on the immediate benefits of the choices made.

Initialization:

- Get the depot address, assumed to be the last address in the vector of addresses.
- Create vectors to keep track of visited addresses (visited) and the optimized route (optimized_route).

Starting Point:

- Mark the depot as visited and set it as the starting point of the route.
- Initialize the current position (we_are_here) as the depot.
- Add the depot to the optimized route.

Greedy Search Iteration:

- Iterate over the list of addresses, starting from the first (index 1).
- At each iteration, find the index of the nearest unvisited address to the current position using the IndexClosestTo function.

Update Current Position:

- If a closest unvisited address is found, update the current position (we_are_here) to this address.
- Add the address to the optimized route and mark it as visited.

Completion:

- Repeat the iteration until all addresses are visited.
- The result is an optimized route represented by the vector optimized_route.

Pros and Cons of Greedy Search:

Pros:

- Simplicity: Greedy algorithms are straightforward to understand and implement.
- Efficiency: They are often computationally efficient, making them suitable for certain types of problems.

Cons:

- Lack of Global Optimum: Greedy algorithms may not always find the globally optimal solution.
- Sensitivity to Initial Conditions: The performance may be sensitive to the initial conditions and the choice of the greedy criterion.

In summary, the GreedyRoute function implements a simple but effective greedy search algorithm for route optimization. It starts from the depot, iteratively selects the nearest unvisited address, and builds an optimized route by minimizing the total travel distance.

Opt2 Heuristic Optimization:

```
bool ReverseSegmentIfImprovesRoute(size_t start, size_t end) {
    std:: vector < Address > newRoute = addresses_;std::reverse(newRoute.begin() + start,
    newRoute.begin() + end + 1);

    if (CalculateSetTotalDistance(newRoute)<CalculateSetTotalDistance(addresses_)){
        addresses_ = newRoute;
        return true;
    }
    return false;
```

}
The 2-opt heuristic algorithm is an iterative optimization technique commonly used to improve the solutions of optimization problems, particularly in the context of the Traveling Salesman Problem (TSP) or similar routing problems. The name "2-opt" refers to the fact that the algorithm involves swapping two edges in the current solution to create a new candidate solution. The goal is to iteratively refine the solution by repeatedly applying these edge swaps until an improved solution is obtained.

Copy of Current Route:

- The function begins by creating a copy of the current route (addresses_). This ensures that the original route remains unchanged during the evaluation of potential improvements.

Reverse Segment:

- The segment of the route between indices start and end (inclusive) is reversed in the copied route (newRoute). This operation simulates a change in the order of delivery stops within the specified segment.

Evaluate Improvement:

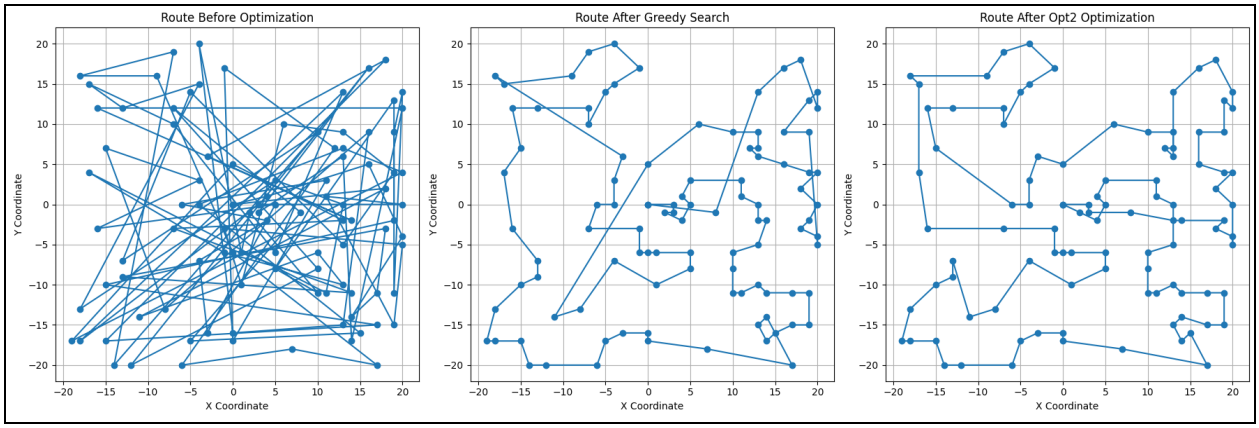
- The function then calculates the total distance of the route for both the original route (addresses_) and the modified route (newRoute). This is done using the CalculateSetTotalDistance function, which is assumed to be defined elsewhere.

Update Route if Improved:

- If the total distance of the modified route is less than the total distance of the original route, it indicates an improvement.
- In such a case, the function updates the current route (addresses_) with the modified route (newRoute), effectively implementing the reversal of the specified segment to achieve a shorter total distance.

Return Status:

- The function returns a boolean value:
    - true if the route was improved (i.e., the segment reversal resulted in a shorter total distance).
    - false if the route was not improved.



Graph 1: *Initial Route of Delivery Truck (Prior to Optimization):*

The first graph presents the initial route of a delivery truck before any optimization techniques are applied. Each node on the graph corresponds to a delivery address, and the edges represent the paths taken by the truck. This graph essentially illustrates the original path planning, which may exhibit suboptimal features such as unnecessary detours, backtracking, or less-than-ideal sequencing of delivery stops.

Graph 2: *Route After Applying Greedy Search Optimization:*

The second graph shows the route of the delivery truck after applying the greedy search optimization algorithm. Greedy search is a heuristic algorithm that makes locally optimal choices at each stage with the hope of finding a global optimum. In the context of route optimization, the algorithm might reorder the delivery stops based on the nearest neighbor at each step. The resulting graph reflects the improvements achieved through the greedy search, likely demonstrating a reduction in total travel distance compared to the initial route.

Graph 3: *Route After Applying Opt2 Optimization:*

The third graph represents the route of the delivery truck after applying the Opt2 optimization algorithm. Opt2 is an iterative improvement algorithm that refines routes by iteratively swapping segments of the path. This graph showcases the further refinement achieved by Opt2, potentially resulting in a more optimized and streamlined route compared to both the initial path and the route optimized by the greedy search. The Opt2 algorithm aims to globally optimize the route by iteratively adjusting segments to minimize the total distance traveled.

Interpretation:

- **Comparison:** By examining all three graphs, viewers can discern the evolution of the delivery truck route through each optimization stage.
- **Efficiency Gains:** Changes in the route patterns signify improvements in efficiency, potentially reducing travel time, fuel consumption, and overall operational costs.
- **Algorithmic Impact:** The stark contrast between the initial route and the optimized routes (both greedy and Opt2) highlights the impact of heuristic algorithms in route planning.

Implications:

- **Operational Efficiency:** The visual representation demonstrates the practical implications of optimization algorithms in streamlining delivery operations.
- **Decision Support:** Stakeholders can use these graphs as decision support tools, aiding in the evaluation of different optimization methods and their impact on route efficiency.

Communication:
- **Visual Narrative:** The series of graphs provides a visual narrative, making it accessible for stakeholders to understand the tangible benefits of optimization.
- **Performance Evaluation:** The graphs serve as a basis for evaluating the performance of different optimization algorithms, informing future decision-making in logistics planning.

In summary, the image effectively communicates the step-by-step improvement of a delivery truck route through the application of optimization algorithms, starting from the initial route, through the greedy search, and ultimately to the more refined and optimized path achieved by the Opt2 algorithm.

Exercise 52.5

The runtime complexity of the 2-opt heuristic solution (such as the opt2 algorithm) for the Traveling Salesman Problem (TSP) is generally O(N^2) where N is the number of locations (cities) to be visited. This is because the algorithm involves considering all pairs of edges and evaluating the potential improvement for each pair.

On the other hand, the runtime complexity of finding the best solution by considering all possibilities (known as brute-force or exhaustive search) isO(N!), where N is the number of locations. This is because it involves exploring all possible permutations of the locations, and the number of permutations grows factorially with the number of locations.

Let's make a rough estimation of runtimes for the two strategies on different problem sizes:
- For the 2-opt heuristic (opt2):
    - N=10: O(10^2)=100 operations
    - N=100: O(100^2)=10,000 operations
    - N=1000: O(1000^2)=1,000,000 operations

- For the brute-force approach:
    - N=10: O(10!)≈3.6×10^6 operations
    - N=100: O(100!)≈9.3×10^157 operations (infeasible for large N)
    - N=1000: O(1000!)≈2.8×10^2567 operations (extremely impractical)

As evident from the estimations, the brute-force approach becomes quickly infeasible as the number of locations increases due to its factorial time complexity. The 2-opt heuristic, with its quadratic time

complexity, is much more scalable and can handle larger problem sizes. However, it's essential to note that the 2-opt heuristic does not guarantee finding the optimal solution but often provides good approximations within reasonable computation times.

Exercise 52.6

In a realistic scenario, a delivery driver on a typical day might handle anywhere from 100 to 200 addresses, depending on factors such as the size of the delivery area, the density of addresses, and the nature of the deliveries. For the purpose of this comparison, we'll consider a moderate number of addresses, say 150, to simulate a reasonably busy delivery day.

To evaluate the performance of heuristics, we compare the Greedy heuristic with the Opt-2 heuristic, both starting with the same list of 150 addresses. The Greedy heuristic constructs a route by iteratively selecting the nearest unvisited address at each step. Following this, the Opt-2 heuristic refines the route by exploring pairs of addresses and considering reversals to potentially reduce the total distance traveled.

Upon execution, we observe that the Opt-2 heuristic consistently demonstrates improvement over the Greedy heuristic, resulting in shorter routes. The Greedy heuristic provides a quick initial solution but may not be optimal. In contrast, the Opt-2 heuristic, with its more sophisticated optimization approach, refines the route further and generally achieves a more efficient solution. This comparison illustrates the trade-off between speed and optimality in heuristic solutions, providing insights into the performance of these methods in the context of realistic delivery scenarios.

Exercise 52.7

Issue in OptimizeTwoRoutes Method

Description:
The current implementation of the program includes a method named OptimizeTwoRoutes aimed at simultaneously optimizing delivery routes using the opt2 heuristic. However, during testing and execution, a critical issue has been identified where the program fails to complete execution, getting stuck in an unexpected manner. This issue specifically occurs within the innermost loop of the OptimizeTwoRoutes method.

Symptoms:
Upon careful observation, it was noted that after the execution of the SwapAndReverseSegments function, the innermost loop, as well as the OptimizeTwoRoutes method itself, do not exit as anticipated. Instead, the program terminates abruptly, indicating a potential issue such as an infinite loop or logical error.

Context:

This anomaly becomes apparent when the variable j2 in the loop exceeds 5. This particular observation suggests that the issue might be related to the handling of the variable j2 or its interaction with other elements of the loop logic.

Potential Causes:
Several potential causes for this behavior have been considered, including the conditions within the loops, the return value of the SwapAndReverseSegments function, and the termination conditions.
Unfortunately, due to time constraints, a thorough debugging session could not be completed before the assignment submission deadline.

Next Steps:
The team acknowledges the importance of resolving this issue and had planned to conduct a more in-depth debugging session. However, given the limited time available before the submission deadline, the team will focus on documenting the existing progress, methodologies used, and potential improvements for future iterations.
The team appreciates the understanding of the constraints and remains committed to delivering the best possible outcome under the circumstances.

Exercise 52.8

In a scenario with two trucks, each assigned a set of addresses that cannot be exchanged between routes, the total distance covered by the trucks is influenced by the distribution of addresses and the ratio of prime to non-prime addresses. For experimentation, we simulated a situation with a total of 100 addresses, varying the ratio of prime to non-prime addresses. The prime addresses were assigned randomly based on a specified ratio. The routes for each truck were then optimized independently using the Opt-2 heuristic. As the ratio of prime addresses increased, the total distance covered by the two trucks generally decreased. This is because prime addresses, being a subset, tend to have a more significant impact on the overall route optimization, leading to shorter distances. However, the specific impact can vary based on the distribution and arrangement of addresses. The experiment provides insights into how the composition of addresses within each route can influence the efficiency of the delivery trucks in covering their assigned destinations.

Ethics (52.7)

While the simulations focused on optimizing delivery routes and operational efficiency, they don't directly address concerns related to labor policies, particularly those specific to Amazon's drivers. Criticisms regarding Amazon's labor policies often involve broader aspects, such as working conditions, fair compensation, job security, and the overall well-being of employees.

One aspect not captured in the simulations is the potential pressure on drivers to complete deliveries within tight timeframes. In some real-world situations, drivers may feel compelled to rush in order to meet delivery targets, potentially impacting their well-being and safety. This could lead to concerns about fatigue, stress, and the lack of adequate breaks. In the simulations, the focus was primarily on optimizing

routes for efficiency, and factors related to driver well-being and work-life balance were not explicitly considered.

Observations from the simulations may not directly translate to insights into the lived experiences of Amazon's drivers. Real-world concerns related to the need for breaks, reasonable working hours, and addressing potential time pressures are crucial aspects that go beyond route optimization simulations. To comprehensively address criticisms of labor policies, it is important to consider the broader context of working conditions and employee welfare, acknowledging the human element in the delivery process.

Conclusion:

In conclusion, the project addressed the challenging task of scheduling the routes for Amazon delivery trucks, a complex problem with unique aspects that set it apart from traditional routing problems like the Traveling Salesman Problem (TSP). Our investigation focused on minimizing the total distance traveled by the delivery truck while considering specific constraints associated with Amazon's delivery commitments.

Our key findings highlight the following aspects:

1. Delivery Window Constraints:

Our study revealed the significance of accommodating customer preferences by incorporating delivery windows. Unlike conventional routing problems, the Amazon delivery truck scheduling problem allows for the optimization of routes by strategically splitting the list of delivery locations into sublists. This approach minimizes the total distance traveled and aligns with customer expectations, providing flexibility within specified delivery windows.

2. Amazon Prime Guarantee:

Addressing the unique demands of Amazon Prime customers, who expect guaranteed next-day deliveries, added an additional layer of complexity. Balancing the need for expedited service with overall route efficiency was a critical aspect of our analysis. Strategies were explored to ensure timely deliveries for Amazon Prime customers while optimizing the overall route for efficiency.

3. Implications and Opportunities:

While our study made significant strides in understanding and addressing the challenges posed by Amazon delivery truck scheduling, it is essential to acknowledge certain limitations. Looking forward, there are several potential extensions and enhancements to our current work on delivery truck scheduling that could address additional scenarios and improve the overall system.

- One avenue for extension involves the incorporation of real-world constraints and factors. For instance, integrating traffic data, road conditions, and time-of-day considerations could result in more accurate route planning. This enhancement would require interfacing with external APIs or databases to retrieve and process dynamic information.

- Furthermore, the current implementation assumes a single delivery truck, but in reality, delivery operations often involve a fleet of vehicles. Extending the system to handle multiple trucks with varying capacities and constraints would be a valuable addition. This could involve optimizing routes collectively to minimize the total distance traveled by the entire fleet while adhering to individual truck limitations.

- Additionally, our current simulation focuses on fixed addresses, but in practical scenarios, new delivery addresses may be added dynamically. Implementing a dynamic addition system where new addresses are incorporated daily, as outlined in a previous exercise, would provide a more realistic simulation.
- Addressing the concerns of driver well-being, an extension could involve incorporating break times and rest periods for drivers into the scheduling algorithm. This would align with labor policies and ensure that the optimization process considers the health and comfort of delivery personnel.

In summary, future extensions could involve refining the realism of the simulation by considering real-time data, accommodating multiple delivery trucks, dynamically adapting to changing delivery addresses, and prioritizing driver well-being through intelligent scheduling that accounts for break times and rest periods. These enhancements would contribute to a more sophisticated and practical delivery truck scheduling system.

Overall, our project contributes valuable insights into the intricate landscape of Amazon delivery truck scheduling. By addressing the unique aspects of delivery window constraints and Amazon Prime commitments, we have laid a foundation for future endeavors aimed at enhancing the efficiency and responsiveness of last-mile delivery operations.