

Analysing and Solving Problems (ACCA 4019_LON)

Module Code: ACCA 4019

Student Name : Murat Yalcin

Student ID :2428668

Lecturer's Name: REDHWAN AL-AMIR



UNIVERSITY OF WALES
Trinity Saint David
London

Table of Contents

Table of Contents	1
Analysing and Solving Problems	2
Introduction	2
Problem overview	2
MoSCoW requirements	2
The relevant variables.....	3
The proposed solution.....	3
Flowchart	5
Reflection and Justification.....	7
Conclusion.....	8
References	8

Analysing and Solving Problems

Introduction

This document goes through Grocerquick's delivery route planning issues as customers demand rises. It outlines and offers a straightforward way to increase delivery effectiveness, lower fuel expenses, and guarantee on-time delivery.

Problem overview

Grocerquick, an online grocery delivery service, is having trouble because more people are ordering from them. The company has trouble organising delivery routes as the demands grow. Because of this customers are less happy, delivery times will be a lot longer and fuel costs are higher because of this. As the number of orders grows the biggest problem is figuring out how to better manage and organise each delivery.

There is three main parts to the problem. First, the customer orders must be assigned to delivery routes. Second, routes should be planned to cut down on the total distance travelled. Third, every delivery must be made on time. These areas are very connected, so they need to be thought about together to make sure the delivery process doesn't run into any issues.

MoSCoW requirements

The MoSCoW method will be used to prioritise the system requirements [1]:

“Must Have requirements” is the most important for the system to work properly. Every order from a customer needs to be put on a delivery route. For the delivery drivers to use less fuel they should travel as little as possible. Also, no delivery should be more than 30 minutes late. This time limit is very strict and must always be met.

“Should Have requirements” should be about making things a lot more efficient. The drivers should not have to make too many deliveries, so routes should be balanced. These requirements would make things work better overall, but they are not necessary for basic operation.

“Could Have requirements” could include extra features that would make the system better but isn't really needed. For example, some deliveries might be more important than others. These features are not necessary to solve the main problem they are mostly optional.

“Won't Have requirements” will define what is not needed. The solution doesn't include hiring more drivers, changing the promised delivery times or even using real-time traffic data. This helps keep the focus on planning routes instead of changes in the business.

The relevant variables

There are multiple variables involved in this problem such as order number, a delivery address, and the delivery time are all included in each order. A delivery route is made up of a group of delivery locations and the total distance travelled, and the total time it takes to make the delivery. Drivers are defined by how available they are and how many deliveries they can make. The estimated travel time between places, the cost of fuel for each route, and the maximum delivery delay of 30 minutes are some of the other factors. These factors are used to figure out if a route is good and useful.

So, to summarize, the problem is figuring out how to assign customer orders to delivery routes in a way that makes sure deliveries are on time. The problem is easier to understand and gives a starting point for coming up with a solution when you clearly define all requirements.

The proposed solution

A simple and useful way to optimise can be used that doesn't need advanced tools or complicated systems [2]. The solution is to make delivery routes step by step while always keeping an eye on two main things. The first thing to think about is how much fuel is used. Fuel can be saved by keeping the total distance travelled as short as possible. The second thing is managing time, which means that the solution makes sure that no delivery is more than 30 minutes late.

There are three steps in the method. During the first step, all the necessary information is gathered. It involves getting information about the order, the driver, and figuring out how long and how far it will take to get from each location. In the second step, delivery routes are made by logically assigning customer orders to drivers. Orders are put into routes that work best for them, while still meeting deadlines. And finally in the third step the finished routes are improved by changing the order of delivery stops to make the trip even shorter without any delays being made.

This method makes sure that every customer order is given a delivery route that all routes stay within each limit and no time is wasted traveling a longer distance.

The process of building the route starts after the preparation stage. it only takes care of one order at a time so that good choices can be made. The system first tries to add an

order to an existing route when it is being given. At this point the new order changes the total distance of the route and the delivery times. This is important because adding one delivery can change when the others on the route will arrive.

If any delivery is more than 30 minutes late an action will be made. If this happens the system will not accept the route option and will try a different route or a different position within the route. This stops the solution from going on with routes that don't meet the time limit. The route is valid and can be used if the order can be added without causing any delays.

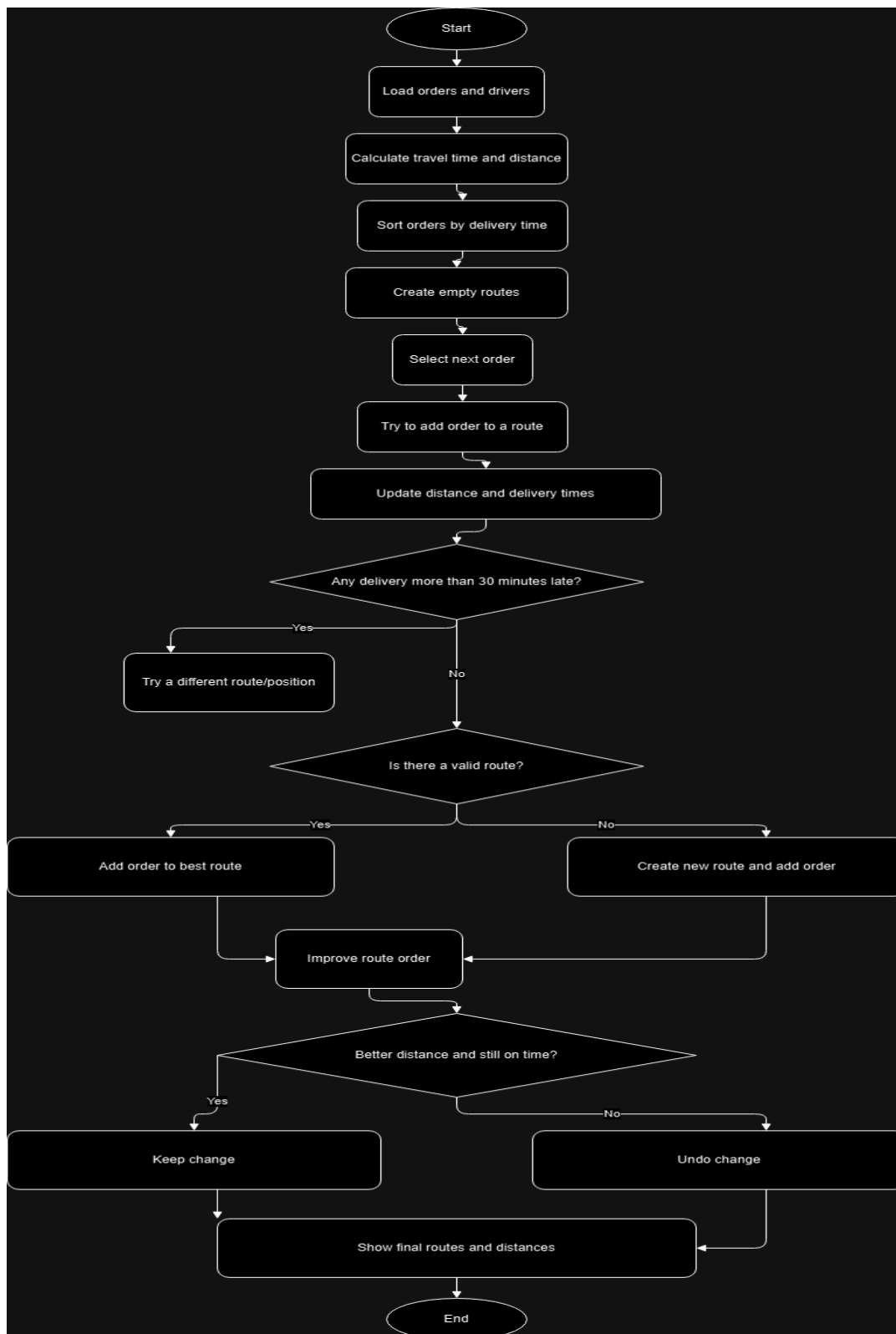
The solution checks the valid options and then decides if there is at least one good route available. If there is a proper route the order is added to the one that adds the least amount of distance to the total. This helps save gas by keeping people from travelling when they don't need to [5]. If a valid route can't be found, a new one is made, and the order is given to it. This makes sure that all orders are delivered, even if the current routes can't handle them.

The solution moves on to the route improvement stage after orders have been given out. At this point, the order of the delivery stops is changed to see if the route can be made shorter. They test out small changes and then figure out the new total distance. After that, another choice is made to see if the change makes the distance shorter while still getting all the deliveries on time. The change stays if it works. If not the route goes back to the way it was before.

This process is made clear by the flowchart [4]. It helps you understand how the solution goes from one step to the next and how choices affect the result. The flowchart's loops show where the system tries different route options again, and the decision diamonds show how late deliveries and route improvements are handled. This makes it easier to understand the whole logic.

So overall this method gives a clear and organised way to improve deliveries. It makes sure that orders are given out correctly, deadlines are met, and fuel use is kept to a minimum. The answer is still simple, but it will still solve the main problems that the delivery company is having.

Flowchart



The pseudocode below shows the same steps as the flowchart. It explains how orders are checked and how routes are picked, and how time limits are kept.

```
INPUT:
  Orders (order_id, location, promised_time)
  Drivers (driver_id, capacity)
  MaxLate = 30 minutes

PREPARE:
  Calculate travel_time and travel_distance between all locations
  Sort Orders by promised_time (earliest first)
  Create an empty route for each driver

FUNCTION CheckRoute(route):
  Calculate arrival times for each stop in the route
  If any arrival_time > promised_time + MaxLate, return false
  If route has more orders than driver capacity, return false
  Return true

FOR each order in Orders:
  best_route = NONE
  smallest_extra_distance = INFINITY

  FOR each route in Routes:
    Try placing order into the route (test possible positions)
    Update the route distance and delivery times

    IF CheckRoute(trial_route) is true:
      extra_distance = trial_route.distance - route.distance
      IF extra_distance < smallest_extra_distance:
        smallest_extra_distance = extra_distance
        best_route = trial_route

  IF best_route exists:
    Save best_route (order is added to the best valid route)
  ELSE:
    Create a new route and assign the order to it

ROUTE IMPROVEMENT:
  FOR each route in Routes:
    Try small changes to stop order
    IF distance becomes smaller AND CheckRoute(route) is true:
      Keep the new stop order
    ELSE:
      Undo the change

OUTPUT:
  Final routes for each driver
  Total distance travelled
  Check that no delivery is more than 30 minutes late
```

The first part of the pseudocode is about the preparation process. It can enter orders and deliveries and set the max lateness to 30. Then, the times and distances it takes to get from one place to another are figured out. This step is important because it needs to know how long it will take to get from one stop to the next to make route decisions. It also sorts orders by delivery time so that it can take care of the most important ones first. This lowers the risk of missing early deadlines because of decisions made later in the route.

Also, the main route-building loop goes through each order one at a time. The system checks existing routes for each order and tries to add the order in a way that keeps the route valid. A route is only valid if it doesn't go over the time limit and doesn't go over the driver's capacity. A different function called CheckRoute checks this. This function makes the solution easier to understand because it puts the time-limit logic all in one place instead of repeating it in different parts of the pseudocode.

The main step in building a route only works on one order at a time. The system will check the existing routes for each order and tries to add the order without breaking any rules. A route is only okay if it can be done in time and by the number of drivers.

If there is more than one valid route the one that adds the least distance will be chosen. This helps save gas by not going places that aren't necessary. If there isn't a valid route a new one is made for the order. This makes sure that every order is in the delivery plan.

Finally in last stage of route improvement. To cut down on distance small changes to the order of stops will be tested. A change is only kept if it makes the route shorter and still gets all the all the orders there on time. This stops late deliveries from happing by saving fuel.

Reflection and Justification

The main way to solve this problem was to break it down into smaller sections. Instead of trying to plan all the deliveries at once it was broken up into simple steps like getting the data ready, making routes and making routes better. This made the problem clearer and helped keep the solution organised.

The actual delivery process was broken down into important parts like orders, drivers, routes, distances, and time limits. Most important parts were used such as information which kept things from getting too complicated. This made it easier to understand and follow the solution.

The flowchart and pseudocode were also very important. The flowchart helped show how the system goes from one step to the next and how decisions are made when a route isn't right. The pseudocode made the same logic clear in writing. Both the pseudocode and the flow chart all worked together to make sure the solution made sense and was consistent.

This method works because it focusses on the main issues. Every order is given a route, and the delivery time limits are always checked. Choosing shorter routes also cuts down on fuel use[2]. The step-by-step method also lets you quickly reject bad route choices, which stops deliveries from being late.

But the solution does have some problems. It depends on estimated travel times and distances, which might not always be right in real life. Stuff like road closures or traffic jams are not considered. The solution also will also assume that drivers and deliveries are the same, which may not always be the case.

Overall, this method is good for the problem because it gives a clear and useful way to plan deliveries. It keeps the solution simple while still meeting the delivery service's main needs.

Conclusion

In conclusion this work will give Grocerquick a clear and easy way to make their delivery process better. The shown methods help lower fuel costs and cut down on late deliveries by shortening the distance travelled and checking delivery times.

References

- [1] D. Clegg and R. Barker, Case Method Fast-Track: A RAD Approach, Addison-Wesley, 1994. [Online]. Available:
<https://www.sciencedirect.com/topics/computer-science/moscow-method>
- [2] P. Toth and D. Vigo, Vehicle Routing: Problems, Methods, and Applications, SIAM, 2014. [Online]. Available:
<https://epubs.siam.org/doi/book/10.1137/1.9781611973594>
- [3] P. Checkland, Systems Thinking, Systems Practice, Wiley, 1999. [Online]. Available:
<https://onlinelibrary.wiley.com/doi/book/10.1002/9780470687414>
- [4] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 8th ed., McGraw-Hill, 2015. [Online]. Available:
<https://www.mheducation.com/highered/product/software-engineering-practitioner-s-approach-pressman/M9780078022128.html>
- [5] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," European Journal of Operational Research, vol. 59, no. 3, pp. 345–358, 1992. [Online]. Available:
<https://www.sciencedirect.com/science/article/pii/037722179290251Y>