

12/16/2025

APS Assessment 2

Coursework-Problem Analysis and
Solution Development

Analysing and Solving Problems

ACCA 4019

Student Name: Evgeni Veselinov Iliev

Student ID: 2424555

Group: 1

Lecturer's Name: Redhwan Alamri

Table of Contents

1. Introduction	2
2 . Problem Deconstruction	2
2.1 Breaking the scenario into components	2
Inputs & data.....	2
Decisions to make	2
Constraints & objectives	2
2.2 Variables involved	2
2.3 Requirements and constraints using MoSCoW	3
Must Have	3
Should Have	3
Could Have	3
Won't Have (this time)	3
3. Proposed Solution and Explanation	4
3.1 Overview of the proposed approach	4
Stage A — Build a feasible plan quickly (feasibility first):	4
Stage B — Improve the plan (efficiency second):	4
3.2 Flowchart	5
3.3 Pseudocode.....	6
3.4 Why this meets GrocerQuick's three goals.....	8
Goal 1: Efficient allocation of orders to routes.....	8
Goal 2: Minimised fuel consumption by reducing total distance	8
Goal 3: Timely delivery with no delivery more than 30 minutes late.....	8
3.5 Implementation note	8
4. Reflection and Justification	8
4.1 Problem-solving techniques used (and why they fit)	8
Abstraction.....	8
Decomposition	8
Constraint-first thinking.....	9
Systems thinking	9
4.2 Justifying key design choices.....	9
4.3 Assumptions and limitations (critical evaluation).....	9
References	10

1. Introduction

GrocerQuick is facing a practical delivery challenge: when routes are poorly planned, fuel costs rise and customers receive orders late. This isn't just a "map problem"—it's a decision problem where many small choices (which orders go together, in what visiting sequence, and at what times) combine into either a smooth day or a messy one.

This report analyses GrocerQuick's routing issue using abstract thinking and structured problem-solving. The aim is to design a logical, generic solution that achieves: (1) efficient allocation of orders to delivery routes, (2) reduced fuel consumption by reducing total travel distance, and (3) timely delivery where no order is more than **30 minutes late**. In operations research terms, the scenario closely matches the Vehicle Routing Problem (VRP) and its time-constrained version (VRPTW), which has a long history in logistics research. [1], [3] [pubsonline.informs.org+1](https://pubsonline.informs.org/)

2. Problem Deconstruction

2.1 Breaking the scenario into components

To make the problem manageable, it helps to split it into smaller components:

Inputs & data

1. Customer orders (location, required delivery time, size/weight).
2. Depot location(s).
3. Fleet data (number of drivers, vehicle capacity, shift start/end).
4. Travel information (distance/time between locations, ideally adjusted for traffic).

Decisions to make

1. **Allocation decision:** which orders belong to which driver/vehicle route.
2. **Sequencing decision:** the visiting order within each route.
3. **Timing decision:** when each stop should occur to avoid lateness.

Constraints & objectives

1. Capacity and working time limit how many orders a vehicle can carry and complete.
2. Time targets: the key rule is **no delivery is more than 30 minutes late**.
3. Objective: minimise total travel distance (a good proxy for fuel use).

This structure matters because it shows the problem is not one decision, but a chain of decisions that must stay consistent.

2.2 Variables involved

To design a generic solution, we represent each order as a set of variables:

1. i : order/customer index
2. loc_i : geographic coordinates or postcode

3. d_i : demand (weight/volume/number of crates)
4. $[a_i, b_i]$: delivery time window (earliest, latest acceptable time)
5. s_i : service time at stop (hand-over time)
6. k : vehicle index
7. Cap_k : vehicle capacity
8. $[Start_k, End_k]$: driver shift window
9. $T(u, v)$: travel time between points u and v
10. $Dist(u, v)$: travel distance between points u and v
11. $Late_i$: minutes late at customer i (must be ≤ 30)

Abstracting like this turns the messy real world into a clean model that can be solved logically—without losing what matters.

2.3 Requirements and constraints using MoSCoW

MoSCoW helps separate “non-negotiables” from “nice-to-haves,” reducing scope confusion. agilebusiness.org

Must Have

1. Every order is assigned to exactly one route (no duplicates, no missing).
2. All deliveries happen within the rule: **no delivery more than 30 minutes late.**
3. Capacity constraints are respected (do not overload vehicles).
4. The system outputs clear routes and estimated arrival times.

Should Have

1. Routes minimise total distance (reduce fuel).
2. Balanced workload between drivers (fairness and stability).
3. Robustness: include buffer time for traffic uncertainty.

Could Have

1. Live re-routing during the day if cancellations/traffic happen.
2. Customer notifications (ETA updates).
3. Learning from history (improving travel-time estimates over time).

Won't Have (this time)

1. Perfect “global optimal” routes for every scenario (too expensive to compute daily at scale).
2. Complex multi-depot or reverse-logistics features unless required.

This prioritisation supports a realistic deliverable: something accurate, usable, and scalable—not a theoretical perfection.

3. Proposed Solution and Explanation

3.1 Overview of the proposed approach

GrocerQuick needs a solution that is both **smart** and **practical**. VRPTW problems are known to be computationally hard at scale, so real systems usually combine modelling + heuristics + improvement steps rather than chasing perfect optimality. [3], [4] [pubsonline.informs.org+1](https://pubsonline.informs.org/doi/10.1287/orsc.1140.0920)

A strong generic approach is a **two-stage optimisation pipeline**:

Stage A — Build a feasible plan quickly (feasibility first):

1. Clean and validate inputs
2. Create initial route assignments (clustering/constructive heuristic)
3. Construct a route sequence that respects time windows and capacity

Stage B — Improve the plan (efficiency second):

1. Apply local search to reduce distance while keeping constraints satisfied
2. Evaluate KPIs and output final routes

This mirrors real logistics thinking: first make sure the day is *possible*, then make it *better*.

3.2 Flowchart

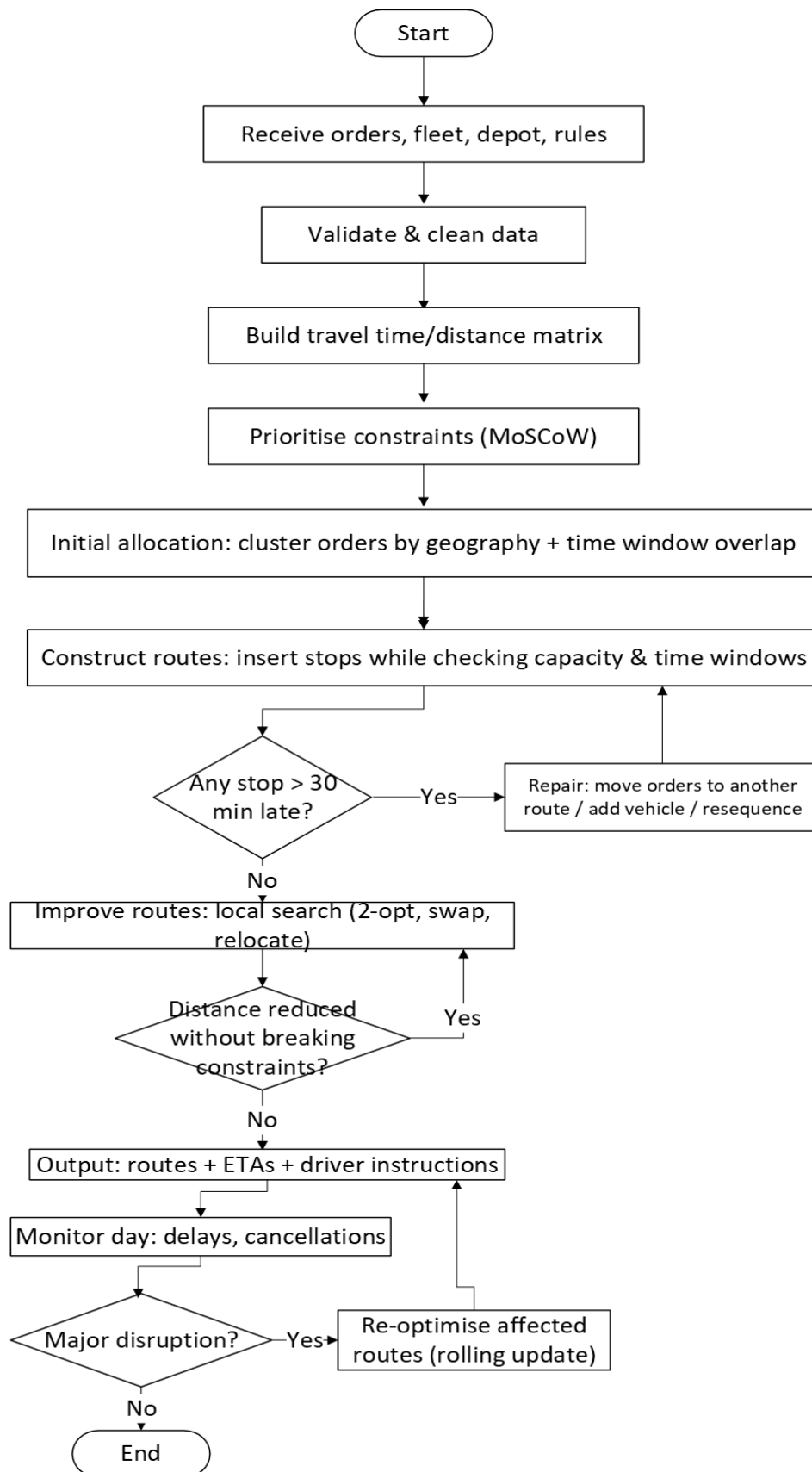


Figure 1. Flowchart (MS Visio)

3.3 Pseudocode

INPUT:

Orders O with (location, demand, timeWindow, serviceTime)

Fleet V with (capacity, shiftStart, shiftEnd)

Depot D

Rule: maxLate = 30 minutes

OUTPUT:

Set of routes R, each route is an ordered list of assigned orders with ETAs

ALGORITHM GrocerQuick_RouteOptimiser(O, V, D):

1. CleanData(O, V)

- remove duplicates, verify time windows, validate demands, geocode locations

2. ComputeMatrices()

- for all points p,q in {D ∪ O}:

time[p,q] = estimatedTravelTime(p,q)

dist[p,q] = estimatedDistance(p,q)

3. PrioritiseConstraints()

- Hard constraints: capacity, shift limits, maxLate
- Soft objective: minimise total distance (fuel proxy)

4. InitialClustering()

- group orders into clusters based on:
 - (a) geographic proximity
 - (b) similar delivery windows
- assign each cluster to a vehicle candidate

5. ConstructFeasibleRoutes()

For each vehicle v in V:

route[v] = [D]

remainingOrders = orders in v's cluster

while remainingOrders not empty:

 choose next order i using "best insertion" rule:

- minimal added distance
- AND respects capacity
- AND respects time windows with lateness $\leq$ maxLate

 if no feasible insertion exists:

 mark i as unassigned and continue

 else:

 insert i into route[v] at best position

6. RepairStep()

- if there are unassigned orders OR any lateness $>$ maxLate:

 try:

- (a) move an order from overloaded/tight route to another route
- (b) resequence problematic route (greedy by earliest deadline)
- (c) if still infeasible, add extra vehicle/driver or split workload

- repeat until all orders feasible OR report "needs more capacity"

7. ImproveRoutes_LocalSearch()

- repeat until no improvement:

 apply operators (within and between routes):

- 2-opt (reverse segment)
- relocate (move one stop)
- swap (exchange stops)

 accept changes only if:

- capacity respected
- shift limits respected
- lateness $\leq$ maxLate
- AND total distance decreases

8. OutputRoutes()

- compute ETAs for each stop

- export driver route plan + timing

9. RollingUpdate(optional)

- if disruption occurs (traffic spike/cancellation):
 - re-optimize only affected routes using steps 5–8

3.4 Why this meets GrocerQuick's three goals

Goal 1: Efficient allocation of orders to routes

The clustering step allocates orders using two real-world signals: geography and time-window similarity. This reduces “zigzagging” across the city and avoids assigning a morning-only order into a route full of late-afternoon drops. It’s abstract thinking in practice: grouping by shared features rather than handling each order as a one-off.

Goal 2: Minimised fuel consumption by reducing total distance

Fuel use is influenced by many factors (vehicle type, driving style, idling, stop-start traffic), but distance is still a strong and simple planning proxy. Reducing unnecessary miles is a direct operational win. The UK Department for Transport’s evidence review also links driving behaviours and planning choices (like avoiding inefficient patterns) to lower fuel consumption and emissions, supporting the idea that smarter routing contributes to efficiency. [7] [GOV.UK](#)

Goal 3: Timely delivery with no delivery more than 30 minutes late

This solution treats the 30-minute rule as a **hard constraint**. If any order breaches it, the algorithm triggers a repair step: move orders, resequence, or add capacity. This is critical thinking: it admits a truth that many plans ignore—sometimes the issue is not “bad routing,” but simply not enough capacity for the demand level. A good system must be honest about that.

3.5 Implementation note

A practical way to implement the above is using an optimisation toolkit such as Google OR-Tools, which explicitly supports VRP/VRPTW concepts like time windows and routing constraints. This matters because it turns the design into something testable rather than just theoretical. [6] [Google for Developers](#)

4. Reflection and Justification

4.1 Problem-solving techniques used (and why they fit)

This task required more than “finding an answer”—it required designing a **generic solution**. The key techniques used were:

Abstraction

I converted the real-world scenario into a structured model: customers become nodes, routes become sequences, and requirements become constraints and objectives. This is aligned with classic routing research, where the original “truck dispatching” formulation is treated as an optimisation problem with measurable inputs and outputs. [1] [pubsonline.informs.org](#)

Decomposition

Instead of trying to solve everything at once, I split the problem into: allocation → construction →

repair → improvement → monitoring. This reduces complexity and mirrors how routing is handled in practice, because VRPTW-scale problems are difficult to solve perfectly in one step. [3], [4] pubsonline.informs.org+1

Constraint-first thinking

A common mistake in planning is focusing on cost first and then trying to “fit” constraints later. Here, the “no more than 30 minutes late” rule is treated as non-negotiable. That reflects the MoSCoW mindset: Must Haves protect the viability of the solution, while Should/Could items improve it if possible. [5] agilebusiness.org

Systems thinking

Routes do not exist in isolation: late deliveries cause complaints, refunds, and workload stress; stressed drivers make errors; errors create more delays the next day. Systems thinking highlights feedback loops and helps justify why the solution includes monitoring and rolling updates rather than a “set-and-forget” schedule. [8] [Florida Tech Research](https://floridatechresearch.com/)

4.2 Justifying key design choices

I chose a hybrid approach (construct + improve) rather than a single “perfect” optimiser because it balances quality with runtime. In a real grocery business, routing must run daily (sometimes multiple times per day), so a solution that produces good routes consistently is more valuable than one that sometimes produces slightly better routes but fails under time pressure. This is consistent with established VRPTW research, where heuristic and approximation methods are often preferred for practical problem sizes. [3] pubsonline.informs.org

The solution also explicitly includes a repair step. This is important because time windows create brittleness: one tight customer can break an entire route. A repair step reflects realism and reduces the risk of producing a plan that looks good on paper but collapses in execution.

4.3 Assumptions and limitations (critical evaluation)

1. **Travel time uncertainty:** The algorithm depends on estimated travel times. Traffic variability can cause lateness even when the plan is “correct.” A stronger version would use time-dependent travel times or buffers based on historical patterns.
2. **Fuel is not only distance:** Distance is a useful proxy, but real fuel use also depends on driving behaviour, idling, and stop-start patterns. The DfT evidence review shows fuel efficiency is influenced by behavioural and operational factors, so routing should be combined with driver guidance and policies (e.g., reducing idling). [7] [GOV.UK](https://gov.uk/)
3. **Capacity shortfalls:** If orders exceed fleet capacity within time windows, no algorithm can “solve” it without adding resources. The system should flag this early.
4. **Data quality risks:** Wrong postcodes, missing service times, or unrealistic time windows can create false “lateness” issues. Data validation is therefore essential.

Overall, the proposed design is appropriate because it is structured, generic, realistic to implement, and it clearly prioritises the Must Have delivery constraint while still improving fuel efficiency through distance reduction.

References

- [1] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959, doi: 10.1287/mnsc.6.1.80. pubsonline.informs.org
- [2] G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964, doi: 10.1287/opre.12.4.568. pubsonline.informs.org
- [3] M. M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987, doi: 10.1287/opre.35.2.254. pubsonline.informs.org
- [4] M. Gendreau *et al.*, "Solving Large-Scale Vehicle Routing Problems with Time Windows," CIRRELT, Working Paper, 2010. [Cirrelt](https://www.cirrelt.ca/)
- [5] Agile Business Consortium, "MoSCoW Prioritisation," *DSDM Project Framework Handbook*. [agilebusiness.org](https://www.agilebusiness.org/)
- [6] Google for Developers, "Vehicle Routing Problem with Time Windows (VRPTW) — OR-Tools Documentation," 2024. [Google for Developers](https://developers.google.com/optimization/routing/)
- [7] UK Department for Transport, "Efficient driving: a rapid evidence assessment," 2016. [GOV.UK+1](https://gov.uk+1)
- [8] D. H. Meadows, *Thinking in Systems: A Primer*, 2008. [Florida Tech Research](https://www.floridatechresearch.com/)