Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Bachelor Econometrie & Operationele Research

Driver and Vehicle Routing Problem with Multiproduct demand and Perishable goods: a multistart heuristic approach

Milan van Puffelen (609824)

Supervisor:	Zhu, JH
Second assessor:	Huisman, D
Date final version:	25th June 2024

Abstract

This paper explores the Driver and Vehicle Routing Problem, a routing problem with two depots in which vehicles travel from one depot to the other, while drivers return to their starting depot. To enable this, all drivers and vehicles visit a single exchange location to switch vehicles and continue the route. Additionally, two extensions of the Driver and Vehicle Routing Problem are examined in this paper, which involve multiproduct demand and perishable goods. When goods are perished, there is no guarantee that all products can still be delivered to the customers as planned. A multistart heuristic, consisting of the construction of initial solutions and performing local search, is used to solve the original problem. Subsequently, this heuristic is adjusted in order to align with the framework of the two extensions. Our computational results on 81 test instances show that the multistart heuristic finds near-optimal solutions to the Driver and Vehicle Routing Problem for small data sets and handles larger ones effectively. The costs of solutions to the demand-related extensions are higher than the costs of solutions to the original Driver and Vehicle Routing Problem. Nevertheless, the adjusted-multistart heuristic successfully lowers the probability that a vehicle is not able to deliver everything.

1 Introduction

The Vehicle Routing Problem (VRP) is a fundamental problem in logistics and is extensively examined due to its significance as a benchmark for addressing real-world problems in logistics. Given a set of customers and a fleet of vehicles, the VRP aims to find a route for each vehicle such that all customers are served and a certain objective function is minimised. In this paper, we consider a version of the VRP that is called the Driver and Vehicle Routing Problem (DVRP) and is introduced by Domínguez-Martín, Rodríguez-Martín and Salazar-González (2018a). Instead of the benchmark version of the VRP that only has a single depot, we have two depots. In the classical VRP, it is often assumed that each vehicle is paired with a driver and vice versa, such that they traverse the same route. However, this paper examines the case where each driver has to return to the depot of departure and each vehicle travels from one depot to the other. For this to become feasible, there is a customer location that serves as an exchange location where drivers can switch vehicles.

This particular variant of the VRP has not received extensive attention in the literature thus far. Despite this, it has numerous practical applications, distinguishing itself with a unique relationship between driver and vehicle routes compared to traditional VRP investigations. For example, the DVRP applies to air transportation in the Canary Islands. In this application, a set of flights must be served by several pilots. There are two depots (Tenerife North and Las Palmas) and several other airports, which can be viewed as customers. Pilots return to the same depot they departed from, whereas the airplanes end at a different depot than the starting depot. This problem was originally investigated by Salazar-González (2014) and was the inspiration for the DVRP, introduced by Domínguez-Martín et al. (2018a).

An exact method can compute the optimal solution for relatively small instances, as done by Domínguez-Martín et al. (2018a). However, it does not suffice for larger instances due to the

running time, which becomes too large. Therefore, this paper uses a multistart heuristic. First, the method constructs driver routes, whereafter those routes are improved through local search. These procedures are integrated into a multistart loop.

In this paper, we also introduce two variants of the DVRP that are not covered by the literature yet. The Driver and Vehicle Routing Problem with Multiproduct demand (DVRPM) is similar to the Capacitated VRP (CVRP). The CVRP is an extension to the VRP where vehicles have a capacity for the products that need to be delivered to customers. The DVRPM extends the DVRP in a similar manner. However, there are two types of products. A customer's demand only consists of one type of product. One product type is delivered by vehicles from the first depot, whereas the other type is delivered by vehicles from the other depot.

The other variant we introduce is the Driver and Vehicle Routing Problem with Multiproduct demand and Perishable goods (DVRPMP). This extends the DVRPM by assuming the goods are perishable. Examples of such goods are fruits, flowers and medicines. At some point in time, the goods perish and cannot be delivered anymore. The goal is to minimise costs while maximising the demand that is met. For both variants, we use the multistart heuristic as a foundation. It needs to be adjusted in order to align with the new set-up and improve performance. Besides, we introduce a matching algorithm that deals with capacity constraints.

Our computational study on 81 instances shows that the multistart heuristic finds near-optimal solutions to the DVRP for small instances and effectively handles larger ones. The costs of solutions to the DVRPM are up to 38% higher than the costs of solutions to the DVRP. The delivery failure probability of the DVRPM functions as a benchmark for that of the DVRPMP. The DVRPMP-adjusted multistart heuristic generally results in solutions with higher costs and lower delivery failure probability than the DVRPM approach for small instances. This means there is a trade-off between lower costs and a lower delivery failure probability.

The remainder of the paper is structured as follows: Section 2 briefly discusses the existing literature on both the DVRP and its extensions. Hereafter, Section 3 introduces all versions of the DVRP that are considered in this paper. Section 4 then elaborates on the multistart heuristic in Section 4.2 and it introduces new and adjusted methods in Section 4.3 and Section 4.4 for the extensions to the DVRP. Subsequently, Section 5 examines the results and finally, Section 6 presents the important findings and conclusions.

2 Literature review

The VRP was first introduced by Dantzig and Ramser (1959). A few years later, Clarke and Wright (1964) proposed a savings heuristic, used to solve the VRP of Dantzig and Ramser (1959). The idea of that heuristic is to merge routes by connecting the end of a route with the beginning of another route, if that results in a cost decrease. Nowadays, that heuristic is widely used in both the literature and in practical situations. Several variants of the VRP have been considered ever since the introduction of the original VRP. Most of the fundamental

variants, such as the Capacitated VRP (CVRP) and VRP with Time Windows (VRPTW), are discussed by Elatar, Abouelmehdi and Riffi (2023). The CVRP comes with the extension that vehicles with certain capacities should deliver products to customers with fixed demand. In the VRPTW, customers can only be visited within prespecified time windows.

Another variant of the VRP is the DVRP that is introduced by Domínguez-Martín et al. (2018a). This paper gives a mathematical formulation for the problem and solves the problem to optimality for small instances (at most 30 locations). In the DVRP, the inclusion of two depots sets it apart as a subtype of the multi-depot VRP. Vehicle routes connect these depots, while driver routes originate and terminate at a single depot. This aligns the DVRP with the multi-depot VRP with inter-depot routes (MDVRPI) framework introduced by Crevier, Cordeau and Laporte (2007). Due to the complexity of the MDVRPI, much of the research devoted to this problem uses heuristic approaches, like in Karakatič and Podgorelec (2015). This complexity also applies to the DVRP. For that reason, Domínguez-Martín, Rodríguez-Martín and Salazar-González (2018b) and Domínguez-Martín, Rodríguez-Martín and Salazar-González (2023) focus on heuristic methods.

The paper Domínguez-Martín et al. (2023) uses a multistart heuristic in order to solve the DVRP. An iteration of a multistart algorithm consists of two phases, as explained by Martí (2003). The first phase is to generate an initial solution, after which this solution is improved in the second phase. The result will be the best solution that is found in all iterations of the multistart heuristic. In Domínguez-Martín et al. (2023), optimal or near-optimal solutions are found for instances with up to 1000 locations in just a couple of minutes. This indicates that the multistart heuristic is powerful for the DVRP.

The DVRPM extends the DVRP in a similar way the CVRP extends the VRP. The paper of Ralphs, Kopman, Pulleyblank and Trotter (2003) elucidates the CVRP and applies heuristics to solve the problem. In our problem, there are two depots, two types of products and two types of vehicles. The Inventory Routing Problem (IRP) discussed by Coelho and Laporte (2013) also contains the latter two characteristics. In general, an IRP can be viewed as a CVRP over multiple time periods.

The DVRPMP is the version of the DVRPM where the goods are perishable. In this case, the goal is both to minimise costs and to maximise delivery to customers. The paper by Sahraeian and Esmaeili (2018) considers a multi-objective for the CVRP with perishable goods. In this paper, the perishability of the products is captured by a probability density function of the expiration times. This is done in a similar manner by Ketzenberg, Gaukler and Salin (2018).

3 Problem description

In this section, we introduce the Driver and Vehicle Routing Problem (DVRP) and two extensions to this problem: Driver and Vehicle Routing Problem with Multiproduct demand (DVRPM) and Driver and Vehicle Routing Problem with Multiproduct demand and Perish-

able goods (DVRPMP).

3.1 Driver and Vehicle Routing Problem

The Driver and Vehicle Routing Problem (DVRP) assumes that there are several locations that are of a particular type. Two locations function as depots, whereas all other locations are customers that need to be visited by a driver and a vehicle. For this purpose, there are drivers and vehicles available at each depot. While vehicles start their routes at their base depot and finish at the other depot, drivers must complete their routes by returning to their base depot. A driver needs a vehicle to travel and a vehicle should always be led by a driver. However, it is allowed for a driver to travel as a passenger in a vehicle that is led by another driver.

A driver always has a different final destination (his base depot) than the vehicle he departs with (the other depot). To make this feasible, a designated customer acts as an exchange point where drivers can swap vehicles. Therefore, the exchange location may and should be visited by multiple vehicles. The duration of a driver's route is defined as the total time it takes to arrive at the base depot again, after departure. The travel time between two locations is fixed and known in advance. Each driver route is not allowed to exceed a certain duration limit. There are no time restrictions imposed on the vehicle routes. A vehicle swap is only possible if both the driver and his new vehicle are at the exchange location at the same time. This is ensured by adjusting the departure times of all drivers and vehicles accordingly. For one exchange location, this is always possible.

In addition to the travel time between two locations, a cost is associated with travelling between two locations. The costs of a route are defined as the accumulated costs of travelling between all locations of that route. The goal is to find a set of driver routes and vehicle routes that minimise the objective function, which equals the sum of the costs of all driver routes.

3.2 Driver and Vehicle Routing Problem with Multiproduct demand

The Driver and Vehicle Routing Problem with Multiproduct demand (DVRPM) is an extension of the DVRP. Both problems have two depots and a number of customers, with one designated as the exchange location. However, in the DVRPM, all customers have a known demand for a specific type of product. There are two types of products. One type is delivered by vehicles starting from the first depot and the other type is delivered by vehicles starting from the other depot. A customer's demand solely concerns one type of product. At each depot, there are equally many vehicles available. All vehicles have a capacity, which is the same for all vehicles. As in the DVRP, all customers should be visited exactly once, except for the exchange location, which must be visited by all vehicles. This implies that each customer, excluding the exchange location, can only be served by exactly one vehicle. The customer at the exchange location may be served by multiple vehicles. Similar to the DVRP, the goal of the DVRPM is to find the driver and vehicle routes that minimise the total costs of the driver routes.

3.3 Driver and Vehicle Routing Problem with Multiproduct demand and Perishable goods

The Driver and Vehicle Routing Problem with Multiproduct demand and Perishable goods (DVRPMP) is an extension to the DVRPM. This problem assumes the goods are perishable, which means that not all products that were initially in a vehicle can be delivered. We assume that the fraction of products that are perished follows a known time-dependent distribution. As in the DVRPM, the goal is to construct driver and vehicle routes that minimise the total costs of the driver routes. However, the additional objective now is to serve as many customers as possible. In the case a vehicle does not have enough products left to deliver to a customer, we assume for simplicity that the vehicle still visits the remaining customers and finishes its route.

4 Methodology

In this section, we introduce some notation, after which we describe a full multistart heuristic as in Domínguez-Martín et al. (2023), to solve the DVRP with a single exchange location. Furthermore, we adjust the multistart heuristic in order to align with the set-up of both the DVRPM and the DVRPMP.

4.1 Notation

Given is a set of customers $V_c = \{1, ..., n\}$ and a set of depots $D = \{0, n+1\}$. It follows that the set of all locations is given by $V = \{0, ..., n+1\}$. In this paper, we consider only one exchange location $e \in V_c$. At an exchange location, drivers can switch vehicles in order to return to their respective depot of departure. The exchange location must be visited by all vehicles, whereas all other customer locations should be visited by exactly one vehicle. Define the set of arcs as $A = \{(i, j) : i, j \in V, i \neq j\}$. The sets of drivers and vehicles starting at depot $d \in D$ are given by K_d and L_d, respectively. It takes t_{ij} time to travel from i to j, $\forall (i, j) \in A$. Each route of a driver cannot exceed a given time limit T. Each arc has a cost c_{ij}.

For each depot $d \in D$ and driver $k \in K_d$, the driver route is defined by $S_{d,k}$, which is a sequence of locations. Both the first and last location of this sequence should be the depot d. Moreover, each customer can be visited at most once, whereas the exchange location should always be visited. Each driver route can be split into two parts $S^1_{d,k}$ and $S^2_{d,k}$, such that $S_{d,k} = S^1_{d,k} \cup S^2_{d,k}$. Here, $S^1_{d,k}$ is the subroute from depot d to the exchange location e (inclusive) and $S^2_{d,k}$ is the subroute from the exchange location e (exclusive) to the depot d.

The vehicle route is given by $R_{d,l}, \forall d \in D, l \in L_d$. The first location is the depot d, whereas the last location is the other depot $d' \in D \setminus \{d\}$. Again, each other customer can be visited at most once, whereas the exchange location should always be visited. The vehicle route is composed of two subroutes of driver routes. In particular, $\exists k_1 \in K_d, k_2 \in K_{d'}$, such that $R_{d,l} = S^1_{d,k_1} \cup S^2_{d',k_2}, \forall d \in D, l \in L_d$.

The extensions of the DVRP examined in this paper consider customer demand. For every

customer $i \in V_c$, this is given by q_i. Moreover, let Q denote the capacity of each vehicle. For a given (sub)route S, the total quantity delivered to customers (excluding the exchange location) on that route is defined by:

$$q(S) = \sum_{i \in S \cap V_c : i \neq e} q_i. \tag{1}$$

The time of the last delivery is denoted by $\tau(S)$ for all routes S. In the DVRPMP, each product will perish at some point in time. For each vehicle, denote the expiration time of the Q products by $T_1, T_2, ..., T_Q$. Product i can only be delivered to a customer at time t if $T_i > t$.

4.2 Multistart heuristic

The multistart heuristic consists of several parts. Algorithm 1 shows the different steps of this method. After initialisations of the best objective value, driver routes, vehicle routes and the number of drivers, the first multistart loop starts. In this loop, a greedy construction algorithm creates an initial solution that is feasible. Subsequently, this initial solution is improved by local search. From Domínguez-Martín et al. (2023), we know that both inter-route search and 2-opt search perform quite well for the DVRP. For that reason, the local search in this algorithm first improves the solution through inter-route search, whereafter 2-opt search is applied. If applicable, the best solution is updated to the new solution. This process is repeated until a certain time limit or maximum number of iterations is reached.

If the multistart loop does not result in a feasible solution, the number of drivers at each depot is increased by one and a new multistart loop will be executed. This continues until a certain time limit is reached or when the number of drivers exceeds the maximum number of drivers fixed at the start. If the preceding steps led to a feasible solution, the corresponding vehicle routes are constructed using a specific algorithm. The remainder of this section elucidates the different parts of the multistart heuristic.

4.2.1 Construction initial solution

The first step within the multistart loop is to construct driver routes S that form an initial solution for a given number of drivers at each depot k. First, the driver routes $S_{d,k}$ are initialised for every depot $d \in D$ and driver $k \in K_d$, such that the route visits the exchange location e and starts and ends at d. Hereafter, unvisited customers are inserted into a route using the cheapest insertion strategy. For every route, this strategy computes the cost increase of each place (index in the route) customer $i \in V_c \setminus \{e\}$ can be inserted. This cost increase is defined as $c_{ui} + c_{iv} - c_{uv}$ for all consecutive nodes u, v in the current route. The best route and place to insert a customer are where the cost increase is minimal. If the duration bound T will not be exceeded after insertion, the customer is inserted. Otherwise, it is inserted using the cheapest insertion strategy into the route that currently has the minimum duration. When all customers are included in a route, the set of all driver routes is given by $S = \bigcup_{d \in D} \bigcup_{k=1}^{|K_d|} S_{d,k}$. The pseudocode for this procedure can be found in Appendix A.

Algorithm 1: Multistart heuristic for the DVRP

Input : Instance data and parameters `timeLim`, `maxDrivers` and `maxIter`
Output: Driver routes S^*, vehicle routes R^* and solution value f^*

```
1  f* ← ∞, S* ← ∅, R* ← ∅, k ← 1
2  while time ≤ timeLim, k ≤ maxDrivers and S* not feasible do
3  |  nIter ← 1
4  |  while time ≤ timeLim and nIter ≤ maxIter do
5  |  |  // Multistart loop
6  |  |  S ← constructGreedySol (k)
7  |  |  S ← localSearch (S)
8  |  |  if S is feasible and f(S) < f* then
9  |  |  |  S* ← S
10 |  |  |  f* ← f(S)
11 |  |  end
12 |  |  nIter ← nIter + 1
13 |  end
14 |  if S* is not feasible then
15 |  |  k ← k + 1
16 |  end
17 end
18 if S* ≠ ∅ then
19 |  R* ← constructVehicleRoutes (S*)
20 end
```

4.2.2 Local search

The local search that is applied to the initial solution consists of inter-route search, followed by 2-opt search. In each iteration of inter-route search, the algorithm evaluates all driver routes. For every customer in a driver's route, it attempts to enhance the solution by removing the customer from its current route and examining all possible insertions into other routes. This means the cheapest insertion strategy is used over all other routes. If the relocation is feasible and improves the solution, it is executed. While improvements are found, the method continues. The pseudocode for the inter-route search can be found in Appendix A.

After termination of the inter-route search, 2-opt search is used to look for further improvements to the solution. In 2-opt search, changes can only be made within driver routes. These changes, known as 2-opt swaps, involve selecting two locations in the route, removing one arc connected to each location, and then reconnecting the route by adding two new arcs. For every pair of locations in a given route, we check whether a 2-opt swap leads to a cost decrease. If so, we change the route accordingly. We proceed with 2-opt search for each route until no improvements are found for all location pairs. Ultimately, this results in the final solution for the driver routes. As for inter-route search, the pseudocode for the 2-opt search can be found in Appendix A.

4.2.3 Construction vehicle routes

First, consider the vehicles that depart from depot 0. Choose drivers $k_1 \in K_0$ and $k_2 \in K_{n+1}$ and construct the first vehicle route $R_{0,1} = S^1_{0,k_1} \cup S^2_{n+1,k_2}$. Subsequently, choose drivers $k'_1 \in K_0$ and $k'_2 \in K_{n+1}$ that have not been chosen before and construct the next vehicle route $R_{0,2} = S^1_{0,k'_1} \cup S^2_{n+1,k'_2}$. Repeat this process until all drivers in K_0 have been chosen. Note that if $|K_0| > |K_{n+1}|$, it is allowed to choose drivers from K_{n+1} multiple times and vice versa. In this case, a driver travels as a passenger in a vehicle. The same procedure can be applied to find the vehicle routes that depart from depot $(n+1)$. The vehicle routes are now constructed as follows: $R_{n+1,l} = S^1_{n+1,k_1} \cup S^2_{0,k_2}$ with $k_1 \in K_{n+1}$, $k_2 \in K_0$ and $l \in L_{n+1}$.

4.3 DVRPM-adjusted multistart heuristic

Now, we consider the DVRPM framework. Before we discuss the adjustments to the multistart heuristic, we first show a visualisation of the situation in Figure 1. D1 and D2 are the depots from where the vehicles depart that deliver the type 1 and type 2 products, respectively. The exchange location is represented by E [1]. Note that driver routes are circular, whereas vehicle routes start at one depot and end at the other depot. The blue and black products are the type 1 and type 2 products, respectively. Observe that from a depot to E, the driver delivers products of its starting depot. From E to a depot, the opposite holds. From Figure 1, it becomes clear that customers on subroutes from D1 to E and from E to D2 are of type 1, whereas customers on subroutes from D2 to E and E to D1 are of type 2.

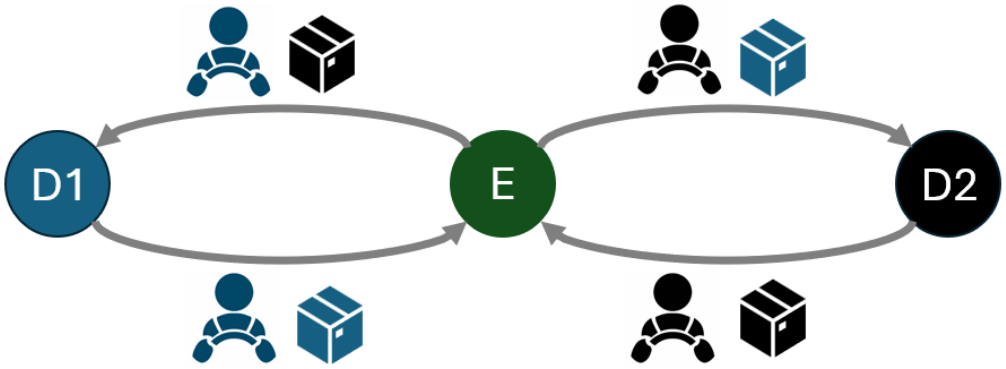

Figure 1: Visualisation of driver routes and product delivery in the DVRPM.

Now, we discuss the multistart heuristic to align with the DVRPM framework. For the initial construction of driver routes, the insertion strategy will change. The cheapest insertion strategy is still used to find the best insertion place. Nonetheless, customers with type 1 demand can only be inserted at places (spots in the route) where type 1 products are delivered. Specifically, insertion can take place between depot 0 and the exchange location e for driver routes starting from depot 0 and between e and depot $(n+1)$ for driver routes from depot $(n+1)$. Customers with type 2 demand can only be inserted at places between depot $(n+1)$ and e for driver routes starting from depot $(n+1)$ and between e and depot 0 for driver routes

[1] D1 can be viewed as depot 0, D2 as depot $(n+1)$ and E as the exchange location e.

from depot 0. Furthermore, the duration bound T cannot be exceeded after insertion, as in the original algorithm. However, now it also has to be checked whether this insertion is possible, considering capacity constraints. For the reason that the vehicle routes are not yet constructed, it cannot be directly assessed whether the capacity constraints are met after insertion. To overcome this problem, observe that vehicle routes are nothing else than a matching between two driver subroutes. Therefore, it suffices to ensure the existence of feasible matchings, rather than constructing vehicle routes in advance.

Algorithm 2 returns whether the existence of feasible matchings is guaranteed. It considers two subroute sets $\mathcal{G}_1$ and $\mathcal{G}_2$ that are compatible. This means that either $\mathcal{G}_1$ consists of subroutes from depot 0 to the exchange location e and $\mathcal{G}_2$ consists of subroutes from e to the depot $(n+1)$, or $\mathcal{G}_1$ consists of subroutes from depot $(n+1)$ to the exchange location e and $\mathcal{G}_2$ consists of subroutes from e to the depot 0. While there are unmatched subroutes left, the unmatched subroute in $\mathcal{G}_1$ with the highest total delivery is matched to the unmatched subroute in $\mathcal{G}_2$ with the highest total delivery, such that the capacity constraint of the vehicle route (the union of the subroutes) is satisfied. We call this the 'highest-delivery strategy' for finding a matching. The following theorem implies the correctness of Algorithm 2:

Theorem. There exists a feasible matching if and only if the highest-delivery strategy finds a feasible matching.

Proof.

'$\Leftarrow$': This is trivial, as there exists a feasible matching if the highest-delivery strategy finds a feasible matching.

'$\Rightarrow$': Given is that there exists a feasible matching. Let M be one such matching. We will modify M by rematching the subroutes using the highest-delivery strategy and show it remains feasible. Suppose that the unmatched subroute with the highest total delivery is $g_1 \in \mathcal{G}_1$. The unmatched subroute with the highest delivery to which g_1 can be matched is denoted by $g_2 \in \mathcal{G}_2$. Let $g_1' \in \mathcal{G}_1$ be the subroute matched to g_2 in M and $g_2' \in \mathcal{G}_2$ be the subroute matched to g_1 in M. Suppose that $g_1' = g_1$ and $g_2' = g_2$. This means that g_1 is matched to g_2 in M. Therefore, there still exists a feasible matching if g_1 is matched to g_2, namely M, so match g_1 to g_2. Suppose now that $g_1' \neq g_1$ and $g_2' \neq g_2$. By construction, it is possible to match g_1 to g_2. Moreover,

$$q(g_1') + q(g_2') \leq q(g_1) + q(g_2') \leq q(g_1) + q(g_2) \leq Q. \tag{2}$$

It follows that it is also possible to match g_1' to g_2'. Now, change M by replacing the matchings g_1-g_2' and g_1'-g_2 by g_1-g_2 and g_1'-g_2'. By construction, M remains a feasible matching. Repeat the process until there are no unmatched subroutes left. The final matching M is the result of the highest-delivery strategy and is feasible because each iteration did not change its feasibility. This concludes the proof. ■

If the highest-delivery strategy does not result in a feasible matching, it follows by the theorem, using the law of contraposition, that there does not exist a feasible matching.

Returning to the adjustment of the insertion strategy in the construction of driver routes, if the

Algorithm 2: Matching of driver subroutes

Input : Instance data, subroute sets $\mathcal{G}_1, \mathcal{G}_2$
Output: Whether a feasible matching exists

while $\mathcal{G}_1 \neq \emptyset$ **do**
  Determine the subroute $g_1 = \arg\max_{g \in \mathcal{G}_1} q(g)$
  if $\{g \in \mathcal{G}_2 : q(g_1 \cup g) \leq Q\} = \emptyset$ **then**
    return false
  end
  Match g_1 to the subroute $g_2 = \arg\max_{g \in \mathcal{G}_2 : q(g_1 \cup g) \leq Q} q(g)$
  $\mathcal{G}_1 \leftarrow \mathcal{G}_1 \setminus \{g_1\}$
  $\mathcal{G}_2 \leftarrow \mathcal{G}_2 \setminus \{g_2\}$
end
return true

insertion place is between the starting depot $d \in D$ and the exchange location e, $\mathcal{G}_1$ consists of subroutes from d to e and $\mathcal{G}_2$ consists of subroutes from e to $d' \in D \setminus \{d\}$. If the insertion place is after e, $\mathcal{G}_1$ consists of subroutes from $d' \in D \setminus \{d\}$ to e and $\mathcal{G}_2$ consists of subroutes from e to d. The existence of a feasible matching, indicated by the output of Algorithm 2, is a necessary condition for the feasibility of a solution of driver routes. This requirement is incorporated into the feasibility check within the multistart loop of Algorithm 1.

Besides the initial solution algorithm, the local search algorithms also need to be changed compared to the original version of the DVRP without customer demand. We first consider the inter-route search algorithm. An inter-route move consists of the removal of a customer from its current route and the insertion of that same customer into another route. Again, customers can only be inserted at places where their product type is being delivered. As opposed to the original version of this insertion, the capacity constraints should hold now as well. To check this feasibility, we utilise Algorithm 2.

Now, we look at the 2-opt search algorithm. It is not possible to connect two customers from different subroutes, as the customers have different types of demand and would be served by the same vehicle. Because the change in the route only affects one subroute of the route, feasibility is still ensured when the previous solution is feasible. This is because the total delivery does not change for each subroute.

As for the construction of the vehicle routes, we use Algorithm 2. However, instead of checking if a feasible matching exists, we obtain vehicle routes that are a result of the matchings in the algorithm. This way, it is guaranteed that the capacity constraints are satisfied.

Note that the DVRPM-adjusted multistart heuristic has not taken the deliveries to the exchange location into account yet. That is because we make the assumption that the sum of capacities of all vehicles that deliver type 2 products (which is the type of the exchange loca-

tion) is at least equal to the sum of all demand of type 2. This ensures that any vehicle with remaining products can deliver them to the exchange location, as all vehicles are scheduled to visit this location.

4.4 DVRPMP-adjusted heuristic

4.4.1 Integer linear program model for matchings

We model the expiration times $T_1, T_2, ..., T_Q$ of the Q products in a vehicle as random variables that are independently and identically distributed. In this paper, $T_i \sim Exp(\mu), \forall i \in \{1, ..., Q\}$, where μ is the mean. Assume that μ is the same for the expiration time of both type 1 and type 2 products. The order statistics are denoted by $T_{(1)} \leq T_{(2)} \leq ... \leq T_{(Q)}$. Let S be a vehicle route. If $Q - q(S) + 1$ products expire before the time of the last delivery $\tau(S)$, then not all products could be delivered. The probability that this happens is:

$$P\big(T_{(Q-q(S)+1)} \leq \tau(S)\big) = \sum_{j=Q-q(S)+1}^{Q} \binom{Q}{j} \Big(1 - e^{-\frac{\tau(S)}{\mu}}\Big)^j \Big(e^{-\frac{\tau(S)}{\mu}}\Big)^{Q-j}. \tag{3}$$

This probability follows from the cumulative distribution function of the order statistics. From Equation (3), we observe that both the total delivery and the time of the last delivery have a positive influence on the probability of not being able to deliver everything, which we refer to as 'failure probability'. This makes sense because fewer spare products are available in case of high total delivery. Moreover, the later the last delivery time, the larger the probability that a product expires. Preferably, this failure probability should be as low as possible. Therefore, the construction of vehicle routes needs to minimise the average failure probability of the routes. Instead of using Algorithm 2, we use an integer linear programming model.

Given a solution of driver routes, let $\mathcal{G}_1^l$ be the set of all subroutes from the depot to the exchange location for type $l \in \{1, 2\}$. Similarly, let $\mathcal{G}_2^l$ be the set of all subroutes from the exchange location to the depot for type $l \in \{1, 2\}$. For every $i \in \mathcal{G}_1^l, j \in \mathcal{G}_2^l$, the decision variable x_{ij} equals 1 if subroutes i and j are matched and 0 otherwise. Parameters q_{ij} and p_{ij} are the total delivery and the failure probability of the vehicle route composed by subroutes $i \in \mathcal{G}_1^l$ and $j \in \mathcal{G}_2^l$, respectively. The model for type $l \in \{1, 2\}$ is as follows:

$$\min \sum_{i \in \mathcal{G}_1^l, j \in \mathcal{G}_2^l} p_{ij} x_{ij}, \qquad \text{s.t.} \tag{4}$$

$$\sum_{i \in \mathcal{G}_1^l} x_{ij} = 1 \qquad \forall j \in \mathcal{G}_2^l, \tag{5}$$

$$\sum_{j \in \mathcal{G}_2^l} x_{ij} = 1 \qquad \forall i \in \mathcal{G}_1^l, \tag{6}$$

$$q_{ij} x_{ij} \leq Q \qquad \forall i \in \mathcal{G}_1^l, j \in \mathcal{G}_2^l, \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{G}_1^l, j \in \mathcal{G}_2^l. \tag{8}$$

Equation (4) minimises the sum of failure probabilities, which is equivalent to minimising the average failure probability, as the number of vehicle routes is fixed. To ensure that every subroute from the depot to the exchange location is matched to a subroute from the exchange location to the depot and vice versa, Equation (5) and Equation (6) are included. Furthermore, Equation (7) ensures that the total delivery of every vehicle route does not exceed the capacity Q. Finally, Equation (8) gives the bounds of the binary decision variables.

4.4.2 DVRPMP-adjusted multistart heuristic

It is possible to adjust the multistart heuristic in order to decrease the average failure probabilities. For this, the key observation is that routes with the highest delivery are most likely to have high probabilities of not being able to deliver every product. The integer linear programming problem is more likely to find better solutions if the parameters p_{ij} and q_{ij} are more balanced, leading to more matching possibilities. To establish this, we adjust the multistart heuristic. In the initial solution algorithm, instead of going through all the routes to insert a certain customer, we only consider the $\alpha \cdot 100\%$ of the routes with the lowest delivery with $0 \leq \alpha \leq 1$. Note that when $\alpha = 0$ or close to 0, there would be no routes to choose from. For this reason, the minimal number of routes to choose from is set to 1. This way, the distribution of the delivery will be more balanced among the routes. Because of the positive influence of the delivery volume on the failure probability, this probability will also be more balanced. As for the inter-route search, the same method is applied, such that customers are only inserted into the lowest delivery routes. The 2-opt search algorithm does not need to change because changing the visiting order does not change the total delivery.

We apply the DVRPMP-adjusted multistart heuristic to find the driver routes. To check for feasibility in this heuristic, we utilise Algorithm 2 as in the DVRPM-adjusted multistart heuristic. However, we solve the integer linear problem to find the vehicle routes from the final solution of driver routes.

5 Results

In this section, the data we use for our computational analysis is specified first. Subsequently, the results of the DVRP are examined, after which the results of both the DVRPM and the DVRPMP are presented. All methods and algorithms are run on a Windows computer that has 16GB RAM with 4 cores at 2.9 GHz. We use the programming language Java for coding. In addition, we use Gurobi to solve the ILP, described in Section 4.4.1.

5.1 Data

The data sets that are used to test the methods are from Domínguez-Martín et al. (2023). There are 81 instances in total, divided into class 1, class 2 and class 3 instances. Class 1 instances are introduced by Domínguez-Martín et al. (2018a) and have $n + 2 \in \{10, 15, 20, 25, 30\}$ locations, where n is the number of customers. The coordinates of the locations are randomly chosen on the Cartesian plane in the square $[0, 100] \times [0, 100]$. Locations 0 and $n+1$ function as the depots,

whereas location n is chosen to be the exchange location. There are three vehicles and three drivers available at each depot, such that `maxDrivers` $= 3$ in Algorithm 1. The costs c_{ij} for travelling between locations i and j are defined to be the Euclidean distance of those locations. The travel time between locations i and j is then defined by $t_{ij} = \frac{c_{ij}}{60} + 0.5$. For each data set, there are four parameter values for the time limit T, namely T_A, T_B, T_C and T_D. These values are in increasing order, where T_A generally is the tightest parameter value that leads to a feasible solution. There are five instances for each number of locations, resulting in 25 class 1 instances.

The class 2 instances are still randomly generated, but these data sets follow a certain structure. There are 16 instances with 50 locations, first introduced by Domínguez-Martín et al. (2018b). One depot is located somewhere in the rectangle $[0, 20] \times [0, 100]$, whereas the other depot is located in $[80, 100] \times [0, 100]$. The coordinates of the exchange location are inside $[40, 60] \times [0, 100]$ and the remaining customers are inside the $[0, 100] \times [0, 100]$ square. This setup aims to represent a common scenario in transportation, where depots are located far apart, and an exchange point is centrally situated between them. The parameter value of the time limit is set to $T = 18$ for all instances. The other parameters are the same as before.

The class 3 instances have $n + 2 \in \{100, 200, 300, 400, 500, 600, 800, 1000\}$ locations and are introduced by Domínguez-Martín et al. (2023). These data sets are generated in the same way as the class 2 instances. For each instance, the time limit T has four different increasing values: T_A, T_B, T_C and T_D. The other parameters are the same, except for the number of drivers and vehicles available at each depot, which is `maxDrivers` $= 3$ now. There are again five data sets for each problem size, resulting in 81 instances in total for the three classes. The access link to the 81 instances is stated in Domínguez-Martín et al. (2023).

The time limit in Algorithm 1 is `timeLim` $=$ 600s, as in Domínguez-Martín et al. (2018b) and Domínguez-Martín et al. (2023). Furthermore, Domínguez-Martín et al. (2023) performed preliminary tests on the instances that contain 100 locations. It followed that using 100,000 iterations led to a decent balance between solution quality and computation time, so we use $maxIter = 100,000$.

Both the DVRPM and the DVRPMP deal with demand. However, the data sets, as described above, do not contain information about demand. Therefore, we generated the data ourselves. For each problem size of the three classes, we only consider the first instance, resulting in 14 data sets in total. The demand of each customer is randomly chosen from a uniform distribution ranging between 1 and 100. Customers $1, ..., \lfloor \frac{n}{2} \rfloor$ have a demand for type 1 products and customers $\lfloor \frac{n}{2} \rfloor + 1, ..., n$ have a demand for type 2 products. The capacity of all vehicles is given by $Q = \max\{100, 1.5 \cdot \frac{\sum_{i=1}^{n} q_i}{|K_0|+|K_{n+1}|}\}$. If $Q > 100$, then the sum of all capacities is 50% more than the sum of all demand. We enforce Q to be at least 100, as customers' demand can be 100. This way, it is guaranteed that each customer can be served by a vehicle.

5.2 Results DVRP

First, we run Algorithm 1 for all class 1 instances. We compare the results with the solutions found by an exact method in Domínguez-Martín et al. (2018a). Table 1 shows the results for the parameter values T_A and T_C. The instances have the following format: nx-y, where x is the number of locations and y is the number of the instance. The objective values found using an exact method by Domínguez-Martín et al. (2018a) are in the column **Sol. E**. Our results after applying the multistart heuristic are shown in the column **Sol. H**. The number of drivers in our heuristic solution is given by **k**. Finally, the **Time (s)** column gives the running time in seconds for each instance. The results for the parameter values T_B and T_D can be found in Appendix B.

Table 1: Results of exact method and multistart heuristic for class 1 instances for T_A and T_C.

Instance	T_A	**Sol. E**	**Sol. H**	**k**	**Time (s)**	T_C	**Sol. E**	**Sol. H**	**k**	**Time (s)**
n10-1	6	652	654	2	1	8	410	411	1	1
n10-2	5	486	486	2	1	7	292	293	1	1
n10-3	5	987	987	3	2	7	390	390	1	1
n10-4	5	610	611	2	1	7	384	384	1	1
n10-5	5	595	594	2	1	7	356	355	1	1
n15-1	6	454	456	2	1	9	349	349	1	1
n15-2	6	746	746	2	1	9	406	405	1	1
n15-3	6	660	660	2	1	9	388	387	1	1
n15-4	6	1094	1093	3	3	9	497	498	1	1
n15-5	6	787	788	2	1	9	471	472	1	1
n20-1	7	1285	1263	3	5	10	557	559	1	1
n20-2	7	666	667	2	2	10	438	441	1	1
n20-3	7	845	846	2	2	10	540	540	1	1
n20-4	7	600	602	2	3	10	447	448	1	1
n20-5	7	647	650	2	3	10	432	434	1	1
n25-1	8	718	719	2	4	11	499	501	1	1
n25-2	8	825	796	2	3	11	501	510	1	1
n25-3	8	603	601	2	5	11	439	436	1	1
n25-4	8	683	686	2	5	11	469	470	1	1
n25-5	8	721	724	2	5	11	491	494	1	1
n30-1	9	877	879	2	8	13	615	628	1	2
n30-2	9	927	851	2	10	13	560	561	1	2
n30-3	9	812	812	2	11	13	506	507	1	3
n30-4	9	756	721	2	11	13	536	536	1	2
n30-5	9	756	750	2	7	13	493	495	1	2

It appears that most of the times, the heuristic solution is approximately the same as the exact solution. For 4 of the 25 instances (with T_A), the multistart heuristic substantially outperforms the exact approach, resulting in solutions with costs reduced by 20 or more. Frequently, two drivers are used for instances with time limit T_A, whereas only one driver is used for instances with T_C. The running time seems to be increasing in the problem size. Moreover, the tighter the maximum duration of a route, the longer the algorithm runs. This is presumably because there are no feasible solutions with 1 driver, such that the algorithm is searching for multiple values of k. In less than 15 seconds, heuristic solutions that are comparable to or even better than the exact solutions are found.

There are a few outliers, which are n10-3, n15-4 and n20-1 (T_A). For these instances, all of the objective value, the number of drivers and the running time are larger than for other comparable data sets. The number of drivers positively influences the objective value, as there are more driver routes that come with costs. In all cases, the solution becomes better when the duration bound is loosened. This is reasonable, as a larger time limit allows for more routes, thereby increasing the feasible region.

Table 2 presents the results for the 16 class 2 data sets. This time, there is only one value of T. Besides, we make a distinction between the number of drivers used in the exact solution (**k E**) and in the heuristic solution (**k H**), as this often differs. Other than that, the columns are the same.

Table 2: Results of exact method and multistart heuristic for class 2 instances for $T = 18$.

Instance	**T**	**Sol. E**	**k E**	**Sol. H**	**k H**	**Time (s)**
n50-1	18	606	1	608	1	8
n50-2	18	-	-	591	1	8
n50-3	18	581	1	588	1	9
n50-4	18	593	1	597	1	10
n50-5	18	933	2	757	2	26
n50-6	18	584	1	585	1	8
n50-7	18	548	1	550	1	8
n50-8	18	588	1	591	1	9
n50-9	18	603	1	607	1	8
n50-10	18	597	1	788	2	20
n50-11	18	607	1	603	1	8
n50-12	18	1269	2	628	1	9
n50-13	18	-	-	786	2	28
n50-14	18	-	-	618	1	9
n50-15	18	867	2	813	2	21
n50-16	18	613	1	615	1	9

Observe that the exact model failed to find a feasible solution for the instances n50-2, n50-13 and n50-14, whereas the heuristic solution does find solutions for these instances. Furthermore, the heuristic solution outperforms the exact solution for the data sets n50-5, n50-11, n50-12 and n50-15. Nonetheless, the exact solution is better in all other cases. An interesting observation is that the heuristic approach for n50-10 fails to find a solution where only one driver is used, although such a solution exists. For n50-12, it is the other way around. Moreover, the multistart heuristic always yields better results when the exact solution contains two drivers. A possible explanation for this could be that an increment in the number of drivers adds a layer of complexity, which the heuristic can manage more effectively than the exact method. The running times vary, but are generally larger than the running times in Table 1.

Class 3 instances consist of a lot more locations, which makes it burdensome to find an exact solution in a reasonable amount of time. Therefore, Table 3 only shows the results of the multistart heuristic. The instances in Table 3 have up to 500 locations. These results are for

the parameter values T_A and T_C. The outcomes of the heuristic for the parameter values T_B and T_D can be found in Appendix B.

Table 3: Results of multistart heuristic for class 3 instances for T_A and T_C.

Instance	T_A	**Sol. H**	**k**	**Time (s)**	T_C	**Sol. H**	**k**	**Time (s)**
n100-1	15	1096	3	83	30	885	2	51
n100-2	15	1233	3	82	30	960	2	59
n100-3	15	1057	3	82	30	848	2	63
n100-4	15	1177	3	68	30	933	2	53
n100-5	15	1163	3	83	30	915	2	74
n200-1	25	1466	3	507	50	1191	2	401
n200-2	25	1523	3	514	50	1213	2	401
n200-3	25	1418	3	506	50	1186	2	386
n200-4	25	1421	3	449	50	1211	2	286
n200-5	25	1393	3	504	50	1195	2	353
n300-1	35	1657	3	600	70	1441	2	600
n300-2	35	1800	3	600	70	1422	2	600
n300-3	35	1727	3	600	70	1425	2	600
n300-4	35	1723	3	600	70	1459	2	600
n300-5	35	1647	3	600	70	1370	2	600
n400-1	45	1902	3	600	90	1607	2	600
n400-2	45	1874	3	600	90	1594	2	600
n400-3	45	1882	3	600	90	1633	2	600
n400-4	45	2078	3	600	90	1687	2	600
n400-5	45	2007	3	600	90	1686	2	600
n500-1	55	2073	3	600	110	1807	2	600
n500-2	55	2148	3	600	110	1816	2	600
n500-3	55	2223	3	600	110	1837	2	600
n500-4	55	2302	3	600	110	1935	2	600
n500-5	55	2040	3	600	110	1817	2	600
n600-1	65	2301	3	600	155	1992	2	600
n600-2	65	2393	3	600	155	1987	2	600
n600-3	65	2330	3	600	155	1991	2	600
n600-4	65	2232	3	600	155	1942	2	600
n600-5	65	2349	3	600	155	2018	2	600
n800-1	85	2483	3	600	175	2274	2	600
n800-2	85	2762	3	600	175	2316	2	600
n800-3	85	2460	3	600	175	2288	2	600
n800-4	85	2456	3	600	175	2282	2	600
n800-5	85	2646	3	600	175	2265	2	600
n1000-1	105	2854	3	600	205	2536	2	600
n1000-2	105	2974	3	600	205	2588	2	600
n1000-3	105	2902	3	600	205	2518	2	600
n1000-4	105	2800	3	600	205	2504	2	600
n1000-5	105	2805	3	600	205	2526	2	600

The total costs tend to increase as the size of the instances increases. Remarkable is that the number of drivers does not increment as the number of locations grows for both T_A- and T_C-values. For all instances, the objective value is lower for the value T_C of the maximum duration than for the stricter value T_A. This is logical because the feasibility region expands as the driver

route duration constraint is less restrictive. Furthermore, it can be seen that the objective values of data sets of the same size for the same duration parameter value are fairly comparable. The only exception to this seems to be for data sets with 800 locations (T_A), as n800-2 and n800-5 have solution costs of two to three hundred more than those of the other instances. The running time is between 1 and 7 minutes for instances with 100 or 200 locations. Nevertheless, the running time for all other instances is the maximum of 10 minutes (600 seconds).

5.3 Results DVRPM

Now, we present the results of the DVRPM-adjusted multistart heuristic that is discussed in Section 4.3. For each number of locations, the first instance is used to test the methodology. Information about customer demand is now also included in these data sets, as explained in Section 5.1. Denote the original multistart heuristic to the DVRP in Algorithm 1 by H1. The DVRPM-adjusted multistart heuristic is referred to as H2. Table 4 shows the results for H2. The results of H1 are discussed in Section 5.2. The maximum duration T_A is used for each run and is given in column **T**. The objective values of H2 are in the column **Sol. H2**. Similarly, the number of drivers used in the final solution of H2 can be found in the column **k H2**. Lastly, the running times of H2 are stated in column **Time (s)**.

Table 4: Results of H2 for instances of multiple sizes.

Instance	**T**	**Sol. H2**	**k H2**	**Time (s)**
n10-1	6	654	2	1
n15-1	6	456	2	1
n20-1	7	1276	3	3
n25-1	8	719	2	3
n30-1	9	912	2	5
n50-1	18	778	2	7
n100-1	15	1219	3	78
n200-1	25	1784	3	600
n300-1	35	2029	3	600
n400-1	45	2327	3	600
n500-1	55	2602	3	600
n600-1	65	3168	3	600
n800-1	85	3273	3	600
n1000-1	105	3786	3	600

It appears that H2 is able to find feasible solutions to the DVRPM using 2 or 3 drivers at each depot within 10 minutes of running time for all instances. In general, the costs increase as the number of locations of the data sets grows. For all runs, the costs of the H2 solution are equal to or higher than the costs of the H1 solution to the DVRP. This is expected, as H2 also takes into account the capacity and delivery constraints. The maximum increase is about 38% of the initial solution, occurring in the case of n600-1. Furthermore, for n50-1, the number of drivers used in the H2 solution is one more than the number of drivers used in the solution found by H1. For all other instances, the number of drivers is the same. The running time is relatively low for instances n10-1 up until n100-1. However, it is equal to the maximum running time of 600 seconds for all other instances.

5.4 Results DVRPMP

Finally, we consider the case where the goods are perishable. First of all, we run the DVRPMP-adjusted multistart heuristic, explained in Section 4.4, for different values of α. Subsequently, we simulate the expiration times of all products 10,000 times for both the solution found by H2 and the solution found by H3 (which denotes the heuristic from Section 4.4). From these simulations, the average failure probability is computed for both solutions. Recall that the failure probability is the probability that a vehicle is not able to deliver everything it is scheduled to. Table 5 presents the results for instances with up to 200 locations and Table 6 shows the results for instances with 300 locations or more. Again, the maximum duration T_A is used for each run and is given in column **T**. The values of parameter μ of the exponential distribution the expiration times follow are given in column μ. For each instance, the value of μ is chosen such that the methodology can be tested. Too low values lead to failure probabilities of 1, whereas too high values lead to failure probabilities of 0. Hence, the value is chosen in between. We run the algorithm for $\alpha \in \{0.5, 0.25, 0\}$. The objective values of H2 and H3 are in the columns **Sol. H2** and **Sol. H3**, respectively. Similarly, the average failure probabilities of the final solution of H2 and H3 are stated in the columns **Fail % H2** and **Fail % H3**.

Table 5: Results of H2 and H3 with average fail percentages for instances of multiple sizes.

Instance	**T**	μ	α	**Sol. H2**	**Fail % H2**	**Sol. H3**	**Fail % H3**
n10-1	6	16	0.5	654	43.455	660	25.000
n10-1	6	16	0.25	654	43.455	724	5.333
n10-1	6	16	0	654	43.455	724	5.295
n15-1	6	20	0.5	456	49.235	456	24.275
n15-1	6	20	0.25	456	49.235	526	0.213
n15-1	6	20	0	456	49.235	526	0.185
n20-1	7	27	0.5	1276	49.915	1287	43.573
n20-1	7	27	0.25	1276	49.915	1360	0.238
n20-1	7	27	0	1276	49.915	1390	2.407
n25-1	8	30	0.5	719	25.050	754	25.000
n25-1	8	30	0.25	719	25.050	911	17.068
n25-1	8	30	0	719	25.050	942	11.275
n30-1	9	30	0.5	912	50.000	1044	43.093
n30-1	9	30	0.25	912	50.000	1226	0.000
n30-1	9	30	0	912	50.000	1238	0.000
n50-1	18	22	0.5	778	49.988	893	0.000
n50-1	18	22	0.25	778	49.988	994	0.000
n50-1	18	22	0	778	49.988	986	0.000
n100-1	15	30	0.5	1219	50.000	1346	33.333
n100-1	15	30	0.25	1219	50.000	1574	0.814
n100-1	15	30	0	1219	50.000	1686	0.000
n200-1	25	35	0.5	1784	50.000	2075	33.069
n200-1	25	35	0.25	1784	50.000	2168	25.000
n200-1	25	35	0	1784	50.000	2364	12.533

Table 6: Results of H2 and H3 with average fail percentages for instances of multiple sizes.

Instance	**T**	μ	α	**Sol. H2**	**Fail % H2**	**Sol. H3**	**Fail % H3**
n300-1	35	65	0.5	2029	50.000	2413	12.500
n300-1	35	65	0.25	2029	50.000	2534	0.000
n300-1	35	65	0	2029	50.000	2708	0.000
n400-1	45	65	0.5	2327	63.730	2795	13.108
n400-1	45	65	0.25	2327	63.730	3068	11.189
n400-1	45	65	0	2327	63.730	3258	3.801
n500-1	55	75	0.5	2602	83.333	3106	4.584
n500-1	55	75	0.25	2602	83.333	3438	1.863
n500-1	55	75	0	2602	83.333	3605	0.000
n600-1	65	95	0.5	3168	66.667	3454	25.000
n600-1	65	95	0.25	3168	66.667	3733	12.500
n600-1	65	95	0	3168	66.667	3987	0.000
n800-1	85	135	0.5	3273	66.667	3881	0.000
n800-1	85	135	0.25	3273	66.667	4189	0.000
n800-1	85	135	0	3273	66.667	4510	0.000
n1000-1	105	135	0.5	3786	66.667	4408	12.500
n1000-1	105	135	0.25	3786	66.667	4858	0.018
n1000-1	105	135	0	3786	66.667	5138	0.000

H2 focuses on the minimisation of costs only, whereas H3 simultaneously tries to decrease the failure probability by constructing balanced routes, as explained in Section 4.4. Observe that H3 often leads to higher costs than those of H2. The growth becomes larger as the number of locations increases. In all cases, a smaller value of α results in solutions with higher costs. This can be explained by the fact that during the insertion phase of the algorithm, a smaller value of α results in fewer routes to choose from. This potentially reduces cheaper insertion options, inducing a final solution with higher costs.

In general, H3 successfully lowers the failure probability of the solutions compared to that of H2 solutions. For n25-1, $\alpha = 0.5$ does not suffice in order to lower the failure probability from the H2 solution. To a lesser extent, this is also the case for n10-1, n15-1, n20-1, n30-1, n100-1 and n200-1. Nevertheless, $\alpha = 0.25$ and $\alpha = 0$ often force this probability close to 0. Only for n25-1, the failure probability for $\alpha = 0$ is higher than 10%. The failure probability of the solution for instance n20-1 and $\alpha = 0$ is not lower than for $\alpha = 0.25$, but this can be caused by the randomness in the model. In all other cases, there is a positive relation between the value of α and the failure probability of the established solution.

In brief, there is a trade-off between lower costs with a higher failure probability and higher costs with a lower failure probability for all instances. Nonetheless, this trade-off is not perfectly linear, as the increase in costs required to achieve a 1% decrease in the failure probability varies among the different instances and different values of α. Hence, the choice of solution depends on one's willingness to incur additional costs to decrease the failure probability.

6 Conclusion

In this paper, we employ a multistart heuristic to solve the Driver and Vehicle Routing Problem (DVRP) with a single exchange location. This problem is devoted to matching vehicles and drivers that traverse different types of routes while minimising costs. Furthermore, we adapt this method to solve two extensions of this problem: Driver and Vehicle Routing Problem with Multiproduct demand (DVRPM) and Driver and Vehicle Routing Problem with Multiproduct demand and Perishable goods (DVRPMP). The multistart heuristic first constructs driver routes and then improves those routes using local search. These steps are incorporated into a multistart loop. Hereafter, we adjust both the driver routes construction algorithm and the local search algorithms to align with the frameworks of the DVRPM and DVRPMP, which include deliveries to customers. Finally, we modify the heuristic once more to additionally lower the probability that a vehicle cannot deliver to every customer, next to minimising the costs.

The model for the DVRP is tested on 81 instances with a number of locations ranging from 10 to 1000. For smaller instances, exact solutions are readily accessible in existing literature. The multistart heuristic finds solutions with approximately the same objective values as those found by the exact method in less than 15 seconds. Furthermore, the algorithm successfully handles instances containing up to 1000 locations within 10 minutes.

The performance of the adjusted versions of the multistart heuristic is evaluated using 14 instances, one of each size. The costs of solutions to the DVRPM are up to 38% higher than the costs of solutions to the DVRP. The additional constraints concerning capacity and delivery possibilities explain this increase. The heuristic for the DVRPMP leads to an increase in costs compared to the solution to the DVRPM for all instances. Moreover, the DVRPMP approach finds on average lower probabilities that a vehicle is not able to deliver to every customer than the heuristic for the DVRPM. Most often, these probabilities can become close to 0. However, this will come with an increase in costs, resulting in a trade-off between lower costs and lower delivery failure probabilities.

Suggestions for future research include considering multiple exchange locations. This complicates the problem because the arrival and departure times of different drivers and vehicles have to align. Moreover, a time constraint can be imposed on the duration of vehicle routes to simulate real-world scenarios with limited battery or fuel capacity. A recommendation for future research on the DVRPM and DVRPMP is to examine the case where there are more than two types of products. Besides, it could be interesting to explore the situation where vehicles have heterogeneous capacities. Lastly, it may be possible that modifying the routes of a solution to the DVRPMP during the time horizon improves the solution, as more information about the expiration times of the products becomes available. This provides an intriguing opportunity for analysis.

References

Clarke, G. & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, *12*(4), 568–581.

Coelho, L. C. & Laporte, G. (2013). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, *51*(23-24), 7156–7169.

Crevier, B., Cordeau, J.-F. & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European journal of operational research*, *176*(2), 756–773.

Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, *6*(1), 80–91.

Domínguez-Martín, B., Rodríguez-Martín, I. & Salazar-González, J.-J. (2018a). The driver and vehicle routing problem. *Computers & Operations Research*, *92*, 56–64.

Domínguez-Martín, B., Rodríguez-Martín, I. & Salazar-González, J.-J. (2018b). A heuristic approach to the driver and vehicle routing problem. In *International conference on computational logistics* (pp. 295–305).

Domínguez-Martín, B., Rodríguez-Martín, I. & Salazar-González, J.-J. (2023). An efficient multistart heuristic for the driver and vehicle routing problem. *Computers & Operations Research*, *150*, 106076.

Elatar, S., Abouelmehdi, K. & Riffi, M. E. (2023). The vehicle routing problem in the last decade: variants, taxonomy and metaheuristics. *Procedia Computer Science*, *220*, 398–404.

Karakatič, S. & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, *27*, 519–532.

Ketzenberg, M., Gaukler, G. & Salin, V. (2018). Expiration dates and order quantities for perishables. *European Journal of Operational Research*, *266*(2), 569–584.

Martí, R. (2003). Multi-start methods. *Handbook of metaheuristics*, 355–368.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R. & Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical programming*, *94*, 343–359.

Sahraeian, R. & Esmaeili, M. (2018). A multi-objective two-echelon capacitated vehicle routing problem for perishable products. *Journal of Industrial and Systems Engineering*, *11*(2), 62–84.

Salazar-González, J.-J. (2014). Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, *43*, 71–82.

A Pseudocodes multistart heuristic

In Section 4.2.1, we illustrate the method to construct the driver routes of the initial solution. Algorithm 3 demonstrates the pseudocode for this construction.

Algorithm 3: Construction of driver routes

Input : Instance data and parameter k
Output: Driver routes S

$S_{d,k} \leftarrow \{d, e, d\}, \forall d \in D, k \in \{1, ..., |K_d|\}$
while *there are unvisited customers* **do**
  Randomly choose unvisited customer $i \in V_c \setminus \{e\}$
  Use cheapest insertion strategy to find best insertion route $S_{d,k} \in S$ and place
  if *duration bound T is not exceeded by inserting i into $S_{d,k}$* **then**
    Insert i into $S_{d,k}$
  end
  else
    Determine d^* and k^* s.t. S_{d^*,k^*} has minimum duration
    Insert i into S_{d^*,k^*} using cheapest insertion strategy
  end
end
$S \leftarrow \bigcup_{d \in D} \bigcup_{k=1}^{|K_d|} S_{d,k}$
return S

The pseudocode of the inter-route search that we describe in Section 4.2.2 is given by Algorithm 4.

Algorithm 4: Inter-route search

Input : Instance data and current driver routes S
Output: Driver routes S

`improvementFound` $\leftarrow$ *true* `// inter-route search needs to be initiated`
while `improvementFound` **do**
  `improvementFound` $\leftarrow$ *false*
  for *all driver routes $S_{d,k} \in S$* **do**
    for *all customers $i \in S_{d,k}$* **do**
      Use cheapest insertion strategy over all other routes to find best insertion place
      if *duration bound T is not exceeded by inserting i* **then**
        if *relocation decreases total costs* **then**
          Remove i from its current route
          Insert i into best insertion place
          `improvementFound` $\leftarrow$ *true*
        end
      end
    end
  end
end

In Section 4.2.2, we explained the approach of 2-opt search. Algorithm 5 demonstrates the pseudocode for this local search method.

Algorithm 5: 2-opt search

Input : Instance data and current driver routes S
Output: Driver routes S

```
1  for all driver routes S_{d,k} ∈ S do
2      improvementFound ← true                  // 2-opt search needs to be initiated
3      if |S_{d,k}| ≥ 4                             // 2-opt swaps should be possible
4       then
5          while improvementFound do
6              improvementFound ← false
7              for all location pairs (i, j) ∈ S_{d,k} × S_{d,k} do
8                  if duration bound T is not exceeded by 2-opt swap then
9                      if 2-opt swap decreases total costs then
10                         Perform 2-opt swap for location pair (i, j)
11                         improvementFound ← true
12                     end
13                 end
14             end
15         end
16     end
17 end
```

B Results DVRP

Table 7 shows the results of the multistart heuristic for class 1 instances for the parameter values T_B and T_D. Likewise, Table 8 shows the results of the multistart heuristic for class 3 instances for the parameter values T_B and T_D. Section 5.2 explains the structure of the tables and the interpretation of the values.

Table 7: Results of exact method and multistart heuristic for class 1 instances for T_B and T_D.

Instance	T_B	**Sol. E**	**Sol. H**	**k**	**Time (s)**	T_D	**Sol.E**	**Sol. H**	**k**	**Time (s)**
n10-1	7	442	443	1	1	10	369	371	1	1
n10-2	6	292	293	1	1	10	292	293	1	1
n10-3	6	646	646	2	1	10	383	383	1	1
n10-4	6	534	533	2	1	10	383	383	1	1
n10-5	6	365	366	1	1	10	350	351	1	1
n15-1	8	349	349	1	1	12	302	302	1	1
n15-2	8	414	412	1	1	12	406	405	1	1
n15-3	8	442	441	1	1	12	388	387	1	1
n15-4	8	715	715	2	2	12	460	460	1	1
n15-5	8	751	749	2	1	12	469	469	1	1
n20-1	9	846	844	2	5	14	520	521	1	2
n20-2	9	450	613	2	4	14	399	402	1	1
n20-3	9	757	757	2	5	14	507	508	1	1
n20-4	9	580	582	2	4	14	415	417	1	1
n20-5	9	538	538	2	3	14	409	409	1	2
n25-1	10	711	708	2	5	16	482	483	1	2
n25-2	10	716	699	2	6	16	483	486	1	3
n25-3	10	555	553	2	4	16	405	402	1	2
n25-4	10	680	681	2	4	16	454	456	1	2
n25-5	10	703	692	2	5	16	451	454	1	2
n30-1	12	864	816	2	8	18	581	582	1	5
n30-2	12	863	827	2	7	18	552	553	1	4
n30-3	12	818	797	2	7	18	485	487	1	4
n30-4	12	676	663	2	10	18	495	495	1	7
n30-5	12	739	732	2	8	18	490	492	1	4

Table 8: Results of multistart heuristic for class 3 instances for T_B and T_D.

Instance	T_B	**Sol. H**	**k**	**Time (s)**	T_D	**Sol. H**	**k**	**Time (s)**
n100-1	20	886	2	62	40	741	1	92
n100-2	20	968	2	112	40	784	1	97
n100-3	20	886	2	225	40	767	1	97
n100-4	20	939	2	200	40	763	1	94
n100-5	20	956	2	109	40	786	1	106
n200-1	35	1250	2	427	60	1144	1	316
n200-2	35	1308	2	600	60	1142	1	318
n200-3	35	1250	2	600	60	1186	1	600
n200-4	35	1222	2	600	60	1127	1	318
n200-5	35	1258	2	600	60	1145	1	317
n300-1	45	1600	3	600	90	1377	1	600
n300-2	45	1781	3	600	90	1335	1	600
n300-3	45	1502	2	360	90	1334	1	600
n300-4	45	1624	3	600	90	1411	1	600
n300-5	45	1527	3	600	90	1344	1	600
n400-1	55	1817	3	600	115	1561	1	600
n400-2	55	1789	3	600	115	1555	1	600
n400-3	55	1782	3	600	115	1581	1	600
n400-4	55	2090	3	600	115	1580	1	600
n400-5	55	1963	3	600	115	1609	1	600
n500-1	65	2031	3	600	140	1806	2	600
n500-2	65	2065	3	600	140	1821	2	600
n500-3	65	2200	3	600	140	1815	2	600
n500-4	65	2246	3	600	140	1831	2	600
n500-5	65	1971	3	600	140	1814	2	600
n600-1	115	1998	2	600	205	1952	1	600
n600-2	115	2026	2	600	205	1928	1	600
n600-3	115	1995	2	600	205	1948	1	600
n600-4	115	1943	2	600	205	1902	1	600
n600-5	115	2050	2	600	205	1978	1	600
n800-1	135	2299	2	600	225	2265	1	600
n800-2	135	2480	2	600	225	2267	1	600
n800-3	135	2283	2	600	225	2259	1	600
n800-4	135	2288	2	600	225	2271	1	600
n800-5	135	2359	2	600	225	2235	1	600
n1000-1	155	2646	2	600	275	2547	1	600
n1000-2	155	2744	2	600	275	2516	1	600
n1000-3	155	2661	2	600	275	2481	1	600
n1000-4	155	2527	2	600	275	2487	1	600
n1000-5	155	2532	2	600	275	2486	1	600

C Programming code

We program the multistart heuristic and its adjustments in Java. Our Java project includes the following classes:

`Parameters`: this class is used to store the parameters. A `Parameters` object for the DVRP contains the number of customers, a two-dimensional array for the costs of each arc, a two-dimensional array for the travel time of each arc, the exchange location (integer) and the maximum duration of a driver route (double). A `Parameters` object for the DVRPM and DVRPMP additionally contains the value of the capacity and an array for the demand of each customer. These instance variables are made public in order to allow for easy access.

`ParameterReader`: this class is used to read the parameters from a particular data file. An object of this class stores a `Parameters` object. The class contains methods to read the parameters for both the DVRP and its versions, DVRPM and DVRPMP. It also has a method to write the data files for the demand variants of the DVRP. It copies the data files without demand and adds customer demand and vehicle capacity to the file.

`Solution`: this class is used to store all the defining characteristics of a solution. An object of this class stores a `Parameters` object, a list of driver routes and a list of vehicle routes. The `Route` class is an inner class of the `Solution` class. An object of this class contains a list of locations (integers) that form the route and a number of the driver and/or vehicle. It is possible to change the values of the instance variables of both `Solution` and `Route` objects. Furthermore, methods are included to check the feasibility of the solution.

`Solver`: this class is used to run the multistart heuristic for the DVRP, DVRPM and DVRPMP and retrieve the results. It is a utility class, in the sense that it does not have instance variables and all methods it includes are static. The `Solver` has methods for the general multistart heuristic, the driver routes construction for the initial solution, inter-route search and 2-opt search. These methods are specifically adapted for the different versions of the problem: DVRP (part 1), DVRPM (part 2) and DVRPMP (part 3). The class also contains methods to make matchings of subroutes both by using a heuristic and by solving an ILP. Gurobi is used to solve the ILP that is used to construct matchings of subroutes in the DVRPMP framework. In addition to the aforementioned fundamental methods, several helper methods are included in the `Solver` class. These methods are used to obtain the necessary input required by the main methods, such as total delivery or a list of subroutes.