The Cost of Same-Day Deliveries

Lars van der Meer

October 21, 2022

Contents

1 Summary **2**

2 Introduction **4**

3 Literature **5**

3.1 VRP Variations 5

3.2 Solving Techniques 6

4 The Problem **8**

4.1 Problem Description 8

4.2 Integer Linear Program Formulation 9

5 Data **12**

5.1 Random Instance Generation 12

5.2 Benchmark Instances 12

6 Methodology **14**

6.1 Same-Day Approach 14

6.2 Same-or-Next-Day Approach with Complete Information 14

6.3 Same-or-Next-Day Approach with Partial Information 17

7 Computational Experiments **19**

7.1 Heuristic Performance 19

7.2 Parameter Tuning 20

7.3 Outcome Analysis 21

8 Conclusion **24**

Bibliography **25**

A Benchmark Instances **26**

B Tuning Graphs **27**

Chapter 1 Summary

Sustainability is an increasingly more important topic. It also causes customers to be more critical of product and services they buy. This notion is supported by research done by the Descartes System Group, who state that the positive trend in customer's sustainability expectations of deliveries creates a huge opportunity for delivery companies. If these companies are able to take advantage of this opportunity by providing more sustainable delivery options, they can become more cost-effective and increase sustainability, which could attract a larger market share in the growing e-commerce industry.

This research explores a more sustainable delivery option. I investigate the effect of a more flexible delivery approach on the total distance travelled by delivery vehicles. The approach of interest is the same-or-next-day approach, in which deliveries are allowed to be delayed for a maximum of one day. This creates the opportunity to move an order to the next day if that gives more efficient routs. This approach and the same-day approach are compared over a planning period of five days.

This five day routing problem can be modeled as a sequence of five Capacitated Vehicle Routing Problems (CVRP). With the same-day approach, these CVRPs are independent and can be solved separately with existing heuristics. In case of the same-or-next-day approach, the sequence of CVRPs becomes interdependent as some nodes may be moved from one day to another, therefore, altering the CVRPs. For the purpose of deciding which nodes to move to the next day, I propose an algorithm. This is an Adaptive Large Neighbourhood Search algorithm, which utilizes several heuristics to choose which nodes to remove from a specific day and move to the next day. In each iteration, a heuristic is selected to move a node, after which the new solution is accepted according to a simulated annealing acceptance criterion. The heuristics is selected with a probability proportional to their weight. Weights of the heuristics are updated every time after a specified number of iterations according to their performance in the most recent iterations.

The algorithm is tested under two scenarios on instances created from the benchmark CVRP instances taken from literature. The instances are created by assigning one benchmark instance to each day in the planning period. Firstly, the algorithm is tested with complete information, meaning it is assumed that all orders within the planning period are known. Under complete information, all days in the planning period can be solved simultaneously and the same-day solution is always available, such that the same-or-next-day solution can never be worse. Secondly, the algorithm is tested under partial information, in which case the uncertainty of the future is incorporated. All days in the planning period have to solved sequentially as, while solving the first day, the orders on the other days are still unknown. To replace the unknown days, I use an approximation which bases its total demand on historic data. To limit the risk of moving too many nodes to the next day resulting in excess demand, I scale the demand in the approximation based on the risk-averse trimmed mean of the remaining capacity on days in the past.

Results of these tests show that, with complete information, an average cost reduction of 4.47% can be attained. With partial information, the average cost reduction is 3.40%. The modest decrease in average

cost reduction due to the lack of information, shows that the performance of the proposed algorithm is upheld under uncertainty. Aside from these promising results, the algorithm also shows clear paths to improvement. As running the algorithm multiple times on the same instance gives varying results, potentially as low as 11.83% and as high as 3.56% under partial information. This variation in solution quality shows that the proposed algorithm is not very consistent. Even though the algorithm is shown to have good performance on average, this does suggest an opportunity for significant improvements.

Nevertheless, this research did succeed to show that increasing flexibility in the delivery approach can lead to a reduction of the total distance travelled by the delivery vehicles. By applying the same-or-next-day approach, delivery companies can increase both their cost-effectiveness and their sustainability, which may attract a larger market share among the increasingly more critical customers with regard to sustainability.

Chapter 2 Introduction

E-commerce is growing and with that the customer's sustainability expectations. According to research by the company Descartes, these trends create a huge opportunity for retailers (des, 2022). In their research, Descartes conducted a survey with 8013 respondents across 9 countries in Europe, the US, and Canada. They found that today 23% think understanding the carbon footprint of their delivery is quite/very important. However, within five years, this percentage is expected to increase to 51%. Additionally, 54% indicated that they are willing to accept longer delivery times from more sustainable companies. Thus, by providing more sustainable delivery options, companies can become more cost-effective and increase customer satisfaction related to sustainability, possibly attracting a larger market share in this growing industry.

The central subject in this study is the cost of same-day deliveries. I aim to determine the reduction in total cost when allowing for next-day deliveries as well, such that it becomes a same-or-next-day delivery approach. Does the potential delay of deliveries lead to a significant decrease in total costs? Besides cost savings, more efficient routing also results in a decrease of CO2 emissions, helping companies to reach their sustainability goals. Moreover, the cost savings can be shared with clients to encourage sustainable choices.

To investigate the effect of a same-or-next-day delivery approach, I develop a Adaptive Large Neighbourhood Search algorithm which plans the delivery routes over a planning period of five days. With the same-day approach, this becomes a sequence of independent vehicle routing problems (VRP), for which there are very good existing solution methods. However, by adapting the same-or-next-day approach the sequence of routing problems becomes interdependent. The algorithm proposed in this report employs several heuristics to decide which deliveries to delay to the next day. This algorithm is tested on instances created from the benchmark capacitated VRP instances from Augerat et al. (1995), by assigning one benchmark instance to each day. The tests are performed under two scenarios. Firstly, with complete information, assuming all orders are known for the complete planning period, the days in the planning period are solved simultaneously. The proposed algorithm attained an average cost reduction of 4.47% with respect to the same-day approach. Secondly, with partial information, the days in the planning period are solved sequentially using approximations for the remaining days. Under partial information, the algorithm was still able to accomplish an average cost reduction of 3.40%.

The structure of the remainder of this report is as follows. Section 3 gives an overview of literature with similar problems and solution methods. I provide a detailed problem description as well as the corresponding integer linear program formulation in Section 4. Furthermore, I describe the data and methodology used throughout this study in Sections 5 and 6, respectively. Section 7 presents an analysis of the results and, lastly, Section 8 gives a summary of the main findings and contributions as well as recommendations for further research.

Chapter 3 Literature

Due to its practical importance, the vehicle routing problem is one of the most studied combinatorial optimization problems (Vidal, 2022). There are many studies on different variations of the VRP and solving techniques. In this section, I will discuss literature on some VRP variations in Section 3.1 and solving techniques relevant to this study in Section 3.2.

3.1 VRP Variations

The most classical VRP is the capacitated VRP (CVRP), as described by Mor and Speranza (2022). It deals with an extra constraint for the vehicle capacity such that the total demand on any route cannot exceed the vehicle capacity. In their survey on VRPs over time, Mor and Speranza (2022) structure the VRPs according to the decisions that have to be taken to solve them. For the classic CVRP those decisions are with which vehicle to serve each customer and, for each vehicle, in which order to serve the customers assigned to that vehicle. Another class of VRPs is characterized by the additional decision of when to start a route. This class is called VRPs over time.

One instance in this class of VRPs over time, is the Periodic VRP (PVRP). In the PVRP, each customer has several visiting options of when they can be served. The decision of when to visit a customer is taken jointly with the decisions about the assignment and sequencing of customers. The PVRP is characterized by a periodic nature in the customers to be server (Campbell and Wilson, 2014). Another example of a VRP over time, as discussed by Mor and Speranza (2022), is the VRP with release and due dates (VRPRD). As defined in Shelbourne et al. (2017), the VRPRD associates a release and due date for each customer which indicate the earliest time the order is available at the depot and the time by which the order should be delivered. Over time, a decision is made whether to serve known customers or to wait for more customers such that better routes can be planned.

The discussed VRPs over time do not quiet align with the problem discussed in this study. Parcel deliveries do not typically have a periodic nature as customers are visited just once. It is possible that customers place another order but there is no obvious regularity or periodicity. Generally, the property of the problem discussed in this report that the number of vehicles available, vehicle capacity, and the route duration limit can vary for each day complicates the implementation of any of these VRPs over time, as they assume one fleet of vehicles for the entire planning period.

Another class of VRPs, as discussed by Oyola et al. (2018), are the stochastic VRPs (SVRP). VRPs in this class are characterized by stochastic elements such as stochastic demand, stochastic customer presence, or stochastic travel times. In solving stochastic problems, the solution must be at least partially decided before the exact values of all parameters are known. It is possible that a solution fails in the sense that after the realized data becomes known some constraint is violated. Two common ways of modeling stochastic problems are described. On one hand, as a chance constrained program (CCP). On

the other hand, as a stochastic program with recourse (SPR). In CCP, the problem is solved ensuring that the probability of failure is lower than a certain threshold. In SPR, failures are allowed but there is some recourse policy in place which repairs the solution. Although the delivery problem of this study does have stochastic elements, the amount of stochasticity is too much for an implementation of one of the SVRPs discussed in the literature review by Oyola et al. (2018). However, there are ways to utilize deterministic solution strategies while still accounting for stochastic elements.

Two approaches that accomplish this are studied by Toktas et al. (2006) in relation to capacity uncertainty in the assignment problem. First, they present an approximation approach in which approximations of the stochastic parameters are used in the deterministic solver. Several approximation functions are used and compared including single sample, average, median, most likely and three versions of trimmed mean. The risk-averse trimmed mean was found to perform the best. The risk-averse trimmed mean takes the average of a sample excluding the most optimistic observations, therefore, reducing the risk of overestimating the available capacity. On the other hand, Toktas et al. (2006) propose a solution construction approach which uses the deterministic solver to find solutions for possible realizations of the capacities and uses the information obtained from these solutions to construct a final solution. However, this approach requires substantially more calls to the deterministic solver which, therefore, requires more computation time. Thus, for the the purpose of this research, the solution construction approach is impractical.

Lastly, the dynamic VRP (DVRP) is a class of VRPs for which input on the problem is received and updated concurrently with the determination of the routes (Psaraftis et al., 2016). We observe some dynamic nature to the delivery problem of this study, as the realized data becomes available per day and routes for that day, consisting of customers that were moved there from other days, are reoptimized. However, one could also see the problem as a sequence of static problems. When the realized data of a new day is made available, the day before cannot be reoptimized and the solution to that new day has to be decided with the information available at that point and cannot be changed afterwards.

3.2 Solving Techniques

Many solving methods were developed for the classic CVRP. In their study, Vidal (2022) showed that the proposed hybrid genetic search algorithm, specialized to the CVRP, is the leading metaheuristic with regard to solution quality and convergence speed. This is an improved implementation of the algorithm proposed by Vidal et al. (2012).

A general heuristic for vehicle routing problems is presented by Pisinger and Ropke (2007). This algorithm is shown to be robust as it is not easily trapped in a local minimum. It uses an adaptive large neighbourhood search framework to solve the problems. In each iteration the current solution is destroyed by a chosen algorithm and repaired by a chosen algorithm. The new solution is accepted according to some acceptance criteria defined by the applied local search framework, for which simulated annealing is used.

As described by Gendreau et al. (2010), simulated annealing provides a mechanism to escape local minima by allowing moves that worsen the objective value. The simulated annealing algorithm provides an acceptance probability function that determines the probability of accepting a move in the current

iteration. In earlier iterations, moves that worsen the objective value have a higher probability to be accepted than in later iterations. This process is controlled by the temperature parameter with a corresponding cooling schedule. The cooling schedule determines how much the temperature decreases every iteration.

As Pisinger and Ropke (2007) proposed a general heuristic, it can quite easily be adapted to solve any VRP. Using different adaptations of this heuristic, Pisinger and Ropke (2007) found several new best solutions to standard benchmarks. An adaptation of this algorithm for the pickup and delivery problem with time windows by Ropke and Pisinger (2006), is also shown to be successful.

Chapter 4 The Problem

This chapter sketches the setting in which the problem of interest is applicable in Section 4.1. Furthermore, this chapter aims to provide a integer linear program formulation of the problem in Section 4.2. We do have to keep in mind that the integer linear program formulation can only practically be used to solve small instances that require only a limited computation time. Therefore, the integer linear program formulation mainly functions as a base for further developments discussed throughout this report and as an introduction of the notation relevant for this report.

4.1 Problem Description

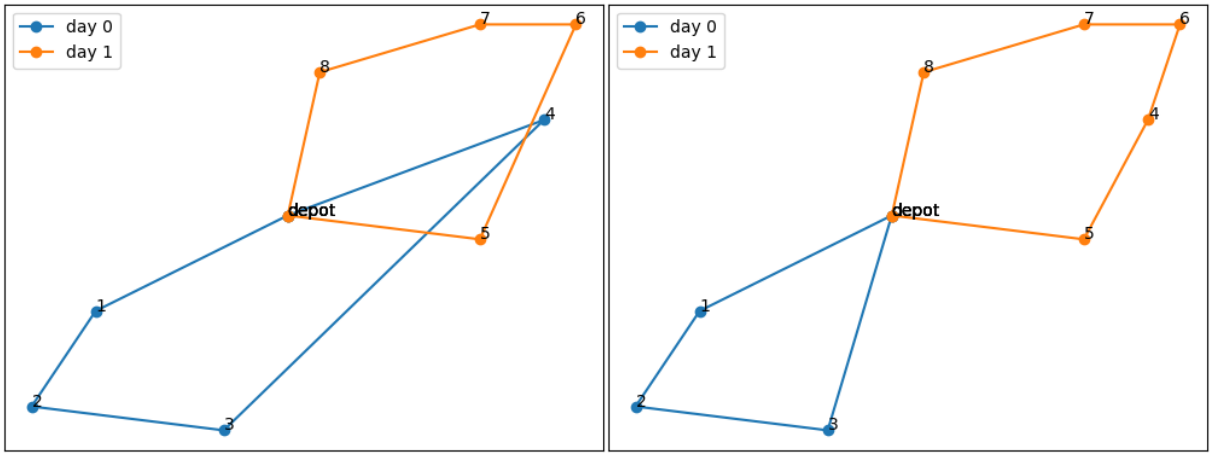
E-commerce is a fast growing industry in today's world. It is very convenient to order something online and have it delivered to your doorstep. It is even more convenient to have it delivered on the same day that you place the order. However, in many occasions, the level of urgency of the order is not that high, but the delivery is still performed on the same day.

Routes have to be planned for each day that deliver to all customers. All routes start and end at the depot. There are a limited number of vehicles available with a limited capacity. Lastly, the execution of a route can take a limited amount of time. It may be that the number of vehicles available, vehicle capacity, and maximum route duration differ per day. By employing a same-day constraint, the problem reduces to a sequence of independent Capacitated Vehicle Routing Problems (CVRP). However, if we allow for delays of deliveries, the CVRPs become interdependent as customers can be moved from one day to another. Consequently, same-or-next-day deliveries allow for many more solutions to the routing problem.

Having to deliver all parcels on the same day, is a very rigid constraint. Even though there is only one order in a certain neighbourhood today but more in that same neighbourhood tomorrow, trucks still have to drive to that neighbourhood twice as same-day deliveries are a requirement. However, it would be much more efficient to only drive to that neighbourhood once by delaying that one order until tomorrow. Thus, by relaxing the same-day constraint, more efficient routes can be planned.

Figure 4.1 illustrates this effect. In this instance, parcels have to be delivered to customers 1, 2, 3, and 4 on day 0 and to customers 5, 6, 7, and 8 on day 1. Figure 4.1a shows the shortest routes that will be performed in case all parcels have to be delivered on the same day as the order. Apparent is that customer 4 is very isolated from the other customers on day 0, while it lies much closer to customers that appear on day 1. Thus, by allowing for same-or-next-day deliveries, more efficient routes can be constructed as shown in Figure 4.1b by moving customer 4 to day 1. Clearly, the decrease in total distance of the route on day 0 is substantially larger than the increase in total distance of the route on day 1.

As the total distance travelled is closely related to the total cost of a company and the amount of emissions, relaxing the same-day constraint, to allow for later deliveries, may have a combination of



(a) Routes when delivering on the same day.

(b) Routes when allowing a next-day delivery.

Figure 4.1: Illustration of same-day versus same-or-next-day deliveries.

positive effects. On one hand it may reduce the total cost incurred. On the other hand it may reduce the total emissions, increasing sustainability.

Finding the optimal solution to this problem comes with some complexities. Firstly, as you use larger instances, it becomes infeasible to solve for the exact optimal solution due to increasing computation time. Thus, we need some heuristic to decide for each customer on each day, whether to deliver to them today or tomorrow. A second complexity is the uncertainty of what customers need to be served tomorrow. This means that without the knowledge of what customers have to be served tomorrow, we have to decide for each customer today whether it is more efficient to deliver to them today or to postpone their delivery until tomorrow. Uncertainty also occurs in the demand of tomorrow's customers. It is unknown how much of tomorrow's capacity is free and, thus, how many deliveries can be delayed until tomorrow. In case there is too little capacity for the total demand of a day, the delivery company must come up with an emergency solution. Such a solution naturally comes with extra costs, for example for additional vehicles and personnel or for longer delays than allowed by the customer. Thus, a penalty must be implemented for excess demand.

For the purpose of this research, I assume that every customer allows for a one-day delay in their delivery. However, this assumption is not strictly necessary for the proposed formulations. The formulations and implementations do not change in case this assumption is lifted. In practice, in could be that some customers do not allow for a delay or that some customers allow for an even longer delay.

4.2 Integer Linear Program Formulation

The starting point of the integer linear program formulation is that of a single CVRP. I use the strengthened single commodity flow formulation, as discussed in Letchford and Salazar-González (2015), which is an adaptation of the two-index formulation. To accommodate for moving customers to a later day, I made another adaptation to the strengthened single commodity flow formulation by implementing a third index for time in days as well as adding several constraints. I will continue by introducing the relevant notation.

Consider a complete directed graph $G = (N, A)$ with a set of nodes $N = \{0, 1, 2, ..., n\}$, in which node

0 represents the depot and $N_c = \{1, 2, ..., n\}$ is the set of customers. A is the set of arcs (i, j) between all nodes $i, j \in N$. I define the set $T = \{0, 1, 2, ..., \tau\}$ as the planning period of $\tau + 1$ days. The distance between two nodes is defined by d_{ij} for $i, j \in N$. The cost of traversing the arc $(i, j) \in A$ is c_{ij}. Each node $i \in N_c$ has a demand of q_i, a release date of rd_i, and a due date of dd_i. The release and due dates represent the day of the order and the latest possible day the order is allowed to be delivered respectively. Lastly, I define k_t, Q_t, and dl_t by the maximum number of vehicles, vehicle capacity and route duration limit on day $t \in T$. To handle the case where there is too much demand for the capacity on a given day, I define P as the penalty per unit of excess demand.

There are four sets of decision variables. Firstly, x_{ij}^t is a binary variable that is equal to one if and only if arc $(i, j) \in A$ is used within a route on day $t \in T$, and zero otherwise. In addition, x_{ij}^t can only be equal to one if and only if t falls within the release and due dates of both nodes i and j. Secondly, f_{ij}^t is a continuous variable that represents the capacity flow across arc $(i, j) \in A$ on day $t \in T$. Furthermore, in order to keep track of the route duration, I implemented a variable similar to the capacity flow, g_{ij}^t. This is another continuous variable that represents the distance that is still to be travelled on the route that uses arc $(i, j) \in A$ on day $t \in T$, including arc (i, j), until arrival back at the depot. Lastly, I define z_t to be the excess demand on day $t \in T$. The integer linear program formulation is as follows:

$$\min \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} x_{ij}^t + \sum_{t \in T} z_t P \tag{4.1}$$

$$s.t. \sum_{t \in T} \sum_{j \in N} x_{ij}^t = 1 \qquad \forall i \in N_c \tag{4.2}$$

$$\sum_{t \in T} \sum_{i \in N} x_{ij}^t = 1 \qquad \forall j \in N_c \tag{4.3}$$

$$\sum_{t \in T} \sum_{j \in N} (f_{ji} - f_{ij}) = q_i \qquad \forall i \in N_c \tag{4.4}$$

$$q_j x_{ij}^t \leq f_{ij}^t \leq (\sum_{i \in N} q_i - q_i) x_{ij}^t \qquad \forall (i, j) \in A,\ \forall t \in T \tag{4.5}$$

$$f_{0j}^t \leq Q_t + z_t \qquad \forall j \in N,\ \forall t \in T \tag{4.6}$$

$$\sum_{t \in T} \sum_{j \in N} (g_{ji}^t - g_{ij}^t) = \sum_{t \in T} \sum_{j \in N} d_{ji} x_{ji}^t \qquad \forall i \in N_c \tag{4.7}$$

$$(d_{ij} - d_{j0}) x_{ij}^t \leq g_{ij}^t \leq dl_t x_{ij}^t \qquad \forall (i, j) \in A,\ \forall t \in T \tag{4.8}$$

$$\sum_{j \in N_c} x_{0j}^t = k_t \qquad \forall t \in T \tag{4.9}$$

$$\sum_{j \in N, j \neq i} x_{ij}^t = \sum_{j \in N, j \neq i} x_{ji}^t \qquad \forall i \in N_c,\ \forall t \in T \tag{4.10}$$

$$x_{ij}^t = 0 \qquad \forall (i, j) \in A, \text{ if } t > dd_j \vee t < rd_j \tag{4.11}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall (i, j) \in A,\ t \in T \tag{4.12}$$

$$f_{ij}^t, g_{ij}^t, z_t \in \mathbb{R} \qquad \forall (i, j) \in A,\ \forall t \in T \tag{4.13}$$

The objective in (4.1) is to minimize total cost for the entire planning period of T days. Constraints (4.2) and (4.3), are the in- and outdegree constraints, respectively, which ensure that each node is visited

exactly once during the planning period. Constraints (4.4) and (4.5) are the capacity flow constraints. The former ensures that, for each node, the difference between the incoming and outgoing flow is exactly equal to its demand, such that each node is fully serviced. The latter sets bounds for the flow f_{ij}^t across arc (i, j) and ensures that it is set to zero if arc (i, j) is not used on day t. Constraint (4.6) makes sure that the outgoing flows from the depot do not exceed the vehicle capacity. If an outgoing flow does exceed the vehicle capacity Q_t, than z_t must be at least equal to the excess demand.

The first four constraints constitute the strengthened single commodity flow formulation as proposed by Letchford and Salazar-González (2015). I implement similar flow constraints for the route duration constraints. Constraint (4.7) sets the difference between the g-variables of two consecutive arcs equal to the distance of the first arc. Additionally, constraint (4.8) sets bounds as g_{ij}^t can never exceed the duration limit and it sets g_{ij}^t to zero if arc (i, j) is not used on day t. Furthermore, constraint (4.9) limits the number of outgoing arcs from the depot each day to the number of available vehicles. Constraint (4.10) ensures that each route is performed on a single day and that it is impossible for a route to span several days. The release and due dates are enforced by constraint (4.11) as all x_{ij}^t for which t falls outside of the interval $[rd_j, dd_j]$, are set to zero. Lastly, constraints (4.12) and (4.13) are the variable constraints. Note that by setting $dd_i = rd_i$ for all nodes $i \in N_c$ and setting $\tau = 0$, this problem is equivalent to a single CVRP with route duration constraints, which is an $\mathcal{NP}$-hard problem. Thus, the problem of interest is also $\mathcal{NP}$-hard.

Chapter 5 Data

Two types of data instances are used in this research, different in size and generation process. To test the finalized algorithm, I use a set of benchmark instances of the CVRP from Augerat et al. (1995). Because of the relatively long running times using these instances, during the development and initial testing of the algorithm, I used smaller randomly generated instances. The generation of the small instances is described in Section 5.1 followed by an introduction of the benchmark instances in Section 5.2. In either case, the planning period consists of five days. Furthermore, for the purpose of this research, the route duration limit is always set sufficiently high such that the route duration constraint is never binding. Thus, only the number of vehicles and the vehicle capacity are considered as properties of the fleet of vehicles.

5.1 Random Instance Generation

The smaller randomly generated instances consist of five days with four or five nodes per day. The number of nodes is chosen randomly. The x- and y-coordinates of each node are taken from a uniform distribution in the interval $[0, 20]$. The demand of each node is taken from a discrete uniform distribution in the interval $[0, 10]$. The release date of each node is its initial day and the due date is one day later. For each day, there is one vehicle available of which the capacity is set according to the following process. To set the capacity, a sample of total day demands is taken from the distribution described above. The capacity is then set such that 95% of the time, the capacity is sufficient to serve all nodes on the day. The capacity is set in this way to simulate the decision a manager might have to make. It is too costly to invest in a larger number of vehicles to have enough capacity for any scenario because there will be many days where there is a large remaining capacity. Furthermore, this setting of the capacity will also force the algorithm to deal with situations where there is an excess in demand which provides more insight into the performance of the algorithm.

5.2 Benchmark Instances

To test the finalized algorithm, I use set A of the benchmark instances for the CVRP from Augerat et al. (1995). This set consists of 27 CVRP instances with number of nodes ranging from 32 to 80 and number of vehicles available from 5 to 10. All vehicles have a capacity of 100. In these benchmark instances, coordinates were generated at random within a square of 100. Demands are taken from a uniform distribution in the interval $[0, 30]$, but $n/10$ of those demands are multiplied by 3, where n is the number of nodes. An instance for the problem discussed in this report is constructed by randomly selecting five benchmark instances and assigning those to the five days of the planning period by setting the release dates accordingly and the due dates to one day later. I created a list of 100 training instances

according to this process and a list of 100 test instances. The training instances are used in calibrating the algorithm, whereas the test instances are used for testing of the algorithm.

Table 5.1 shows the summary statistics of the tigh in each benchmark instance. The tigh is the proportion of the total capacity that is used.

Mean	0.92
Stdev	0.05
Minimum	0.81
Maximum	0.99

Table 5.1: Tigh statistics

Table 5.1 shows that, on average, there is 8% free capacity. Thus, there should be enough space for the algorithm to move nodes around and find a better solution than the same-day solution. However, as will be the case in reality, there might be some days where there is almost no capacity left and days with much more remaining capacity. This is depicted by the maximum tigh of 0.99 and minimum of 0.81, respectively. Table A.1 in Appendix A shows some properties of all benchmark instances.

Chapter 6 Methodology

In this chapter, I discuss the different stages that are involved in the process of solving each variation of the problem. The variations differ by delivery approach, either same-day or same-or-next-day, and by amount of information known. Complete information means that we know exactly for each day in the planning period what orders will come in, whereas in the case of partial information we only know today's orders and are only given the orders of the next day once today is solved. Thus, we use a probability distribution for the demand on each day to represent what the set of orders in the future may look like. I discuss the different variations in increasing complexity.

6.1 Same-Day Approach

By employing the same-day delivery approach, the problem reduces to a sequence of CVRPs with no interdependencies. Thus, we can solve each day separately using a CVRP formulation similar to the formulation presented in Section 4.2 with $\tau = 0$ and due dates equal to release dates. For small instances we can solve the problem using this formulation. However, for larger instances we have to resort to a heuristic to solve the CVRP. I use the open-source implementation of the hybrid genetic search (hygese) algorithm provided in Vidal (2022), as discussed in Section 3. The code used for this research is from Github version $v2.0.0$. Note that for the same-day approach it does not make a difference whether the information is complete or not as each day is solved independently. Thus, with partial information, each day can be solved once the information of that day becomes available.

6.2 Same-or-Next-Day Approach with Complete Information

By relaxing the same-day constraint and allowing for next-day deliveries as well, the sequence of CVRPs becomes interdependent and, thus, they can no longer be solved separately. For small instances we can find the exact solution by solving the MIP formulation as presented in Section 4.2. Solving larger instances becomes more complex as, before solving each day with the hygese algorithm, we have to decide for each day what nodes to move to the next day. For that purpose, I implement several removal heuristics in an Adaptive Large Neighbourhoud Search algorithm, as in Pisinger and Ropke (2007), and a simulated annealing acceptance criterion. After all decisions are made on what nodes to move to a later day, I simply use the hygese algorithm to solve each day separately and determine the final solution.

The initial solution is the same-day solution obtained from the hygese heuristic. Then, in each iteration, a random day is selected. Next, a removal heuristic is selected which is used to select a node on the chosen day. The selected node is then moved and the new solution is computed. If the new solution is accepted, the current solution is updated. This process is repeated until the stop criterion is met, after which the best solution found across all iterations, is return as the final solution. The stop criterion is a prespecified number of iterations. This master level algorithm is also described as Algorithm 1.

Algorithm 1 Master Level Framework

$x \leftarrow$ initial solution
repeat
    $t \leftarrow$ a randomly selected day
    Select heuristic to use
    Select a node on day t
    Move the selected node
    $x' \leftarrow$ new found solution
    if x' is accepted **then**
        $x \leftarrow x'$
    end if
until stop criterion met
Return Best found solution

I implemented a set of five removal heuristics. The *random removal* heuristic (RR) randomly selects a node to be moved from the chosen day. Except for the random removal heuristic, all heuristics follow the same outline as described in Algorithm 2. First, all nodes on the selected day are ordered by some criteria. Next, a node is selected. The randomness parameter p determines the degree of randomness in choosing a node. For larger values, it is increasingly more likely to select the first node in the list. For smaller values of p, nodes that rank lower according to the criteria may be selected instead.

Algorithm 2 General Heuristic Framework

function HEURISTIC(p)
    $L \leftarrow$ List of all nodes i on selected day, sorted by some criteria
    $y \leftarrow$ Random number in $[0, 1)$
    $n \leftarrow L[\lfloor y^p |L| \rfloor]$
    Return n
end function

The *worst removal* heuristic (WR) selects the node which has the worst fit in the current routes on the selected day. All nodes are ordered by descending cost differential. The cost differential of node i is calculated by $costdiff(i) = f(x) - f(x_{-i})$, where $f(x)$ is the cost of solution x, and x_{-i} is the solution excluding node i. The solutions are found using the hygese algorithm. The cost differential of node i is the additional cost incurred by including node i in the routes.

In addition to this version of the worst removal heuristic as proposed by Ropke and Pisinger (2006), I propose a similar heuristic using an approximation of the cost differential, which eliminates the need to solve the day as many times as there are nodes on the selected day. In this *approximate worst removal* heuristic (AWR), the cost differential of node i is calculated as the difference between the cost of traveling arcs $(i-1, i)$ and $(i, i+1)$, and arc $(i-1, i+1)$. Where nodes $i-1$ and $i+1$ represent the nodes than are visited before and after node i, respectively, in the current solution.

The fourth heuristic is the *correlated removal* heuristic (CR) inspired by the concept of time-space correlation proposed by Wen et al. (2010). In this heuristic, the nodes on the selected day are ordered

by increasing correlation. Thus, the correlated removal heuristic selects the node that is least correlated with other nodes on the same day. The correlation between two nodes i and j, $corr(i,j)$ is calculated according to equation (6.1). It is a function of the cost of traveling the arc (i,j), c_{ij}. The parameters ρ, δ, and ε are user-defined. The nodes are eventually ordered by their total correlation with all other nodes on the same day. The closer a node is to other nodes on the same day, the higher its correlation.

$$corr(i,j) = \begin{cases} 0 & \text{if } c_{ij} > \rho \\ \frac{1}{(c_{ij}+\delta)^{\varepsilon}} & \text{if } c_{ij} \leq \rho \end{cases} \tag{6.1}$$

The final heuristic I implement is the *neighbourhood removal* heuristic (NR). Depending on what day it is, this heuristic selects nodes in a certain area to be moved to another day. This could be interpreted as deciding to move all orders that occur within some neighbourhood to the next day. In my implementation, the nodes are ordered by their x-coordinate. On even days, nodes are sorted by increasing x-coordinates and on odd days, nodes are sorted by decreasing x-coordinates.

In each iteration, line 4 of Algorithm 1, one heuristic has to be selected for use. This is done by an adaptive selection procedure as described by Ropke and Pisinger (2006). A weight is assigned to each heuristic, which is initially set to one for all heuristics. In each iteration, the heuristic used is awarded a score based on the new solution found, which is a measure for the effectiveness of the heuristic. There are three scenarios in which a score is given: a new best solution is found (σ_1); a new solution is found which is better than the current solution and was not accepted before (σ_2); a new solution is found which is worse than the current solution and was not accepted before (σ_3). After each segment, which is a specified number of iterations, the weights of all heuristics are updated according to their scores obtained in that segment. The weight of heuristic i, w_i, is updated as shown in equation (6.2).

$$w_i = w_i(1-r) + r\frac{\pi_i}{\theta_i} \tag{6.2}$$

Here, π_i and θ_i are the score and number of attempts of heuristic i in the last segment, respectively. The reaction factor r determines how sensitive the weights are to the performance in the last segment. In each iteration a heuristic is then selected with a probability proportional to their weight.

Lastly, in line 8 of Algorithm 1, the new solution is either accepted or rejected. For this decision, I use a simulated annealing acceptance criterion. Each move is accepted with a probability computed by equation (6.3).

$$P(\text{Accept } x') = \begin{cases} 1 & \text{if } f(x') < f(x) \\ e^{-(f(x')-f(x))/t_k} & \text{otherwise} \end{cases} \tag{6.3}$$

The current solution is denoted by x and the new candidate solution by x'. The objective function is denoted by f and t_k is the temperature parameter where k denotes the current iteration. The temperature parameter decreases with the number of iterations to decrease the probability of accepting moves that worsen the objective value. The temperature is adjusted each iteration according to a cooling schedule defined by the cooling rate c. The temperature in iteration k is calculated as $t_k = Tc^k$, where T is the starting temperature. The starting temperature is determined by the starting temperature control

parameter, φ. The starting temperature is set such that a solution 100φ percent worse than the initial solution is accepted with a probability of 0.5.

6.3 Same-or-Next-Day Approach with Partial Information

With only partial information, we can solve the same-or-next-day delivery problem with a similar algorithm as described in Section 6.2. However, as we have no information on future days, we have to replace those days with instances that approximately represent what can happen on those days. Then each day is solved using approximated data for future days. After a day is solved, the newly available data for the next day is inserted and merged with the nodes that were moved to that day. The new day is solved and the process starts again.

The way I approximate future days is by using subgrids. I divide the grid in which all nodes lie into a specified number of subgrids. Each subgrid is represented by a node placed at the center of the corresponding subgrid. This concept is illustrated in Figure 6.1, in which the total grid is divided into 16 subgrids with nodes placed at the center to represent each subgrid.

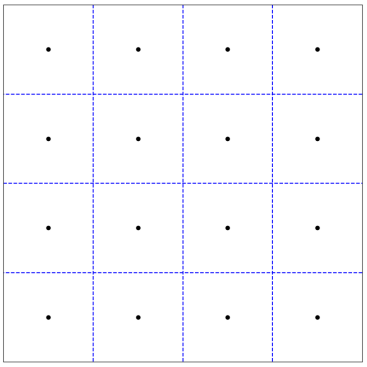


Figure 6.1: Grid divided into subgrids with representitative nodes.

This takes care of the placement of the nodes. The demand of the nodes is determined based on historic data. In principal, the demand of each node is set to the average daily demand that occurs within its corresponding subgrid. As the main difficulty of the uncertain future lies within predicting the remaining capacity, the previously calculated demands are scaled such that the remaining capacity is equal to the average remaining capacity across the historic data. However, according to Toktas et al. (2006), the use of average values may be a simplified approximation scheme that results in solutions far from optimal. They considered approximation schemes other than the sample average and found that the risk-averse trimmed mean performs best. The risk-averse trimmed mean ($RATM$) eliminates outliers, resulting in good performance in cases where the historic data sample is highly optimistic in the sense that it portrays an average remaining capacity much higher than is to be expected. The $RATM$ is the average of a trimmed sample where the largest observations are removed from the sample. Equation (6.4) shows how the $RATM$ is calculated with parameter ξ, which determines what proportion of the sample is kept in the trimmed sample. Let F be the number of observations in the complete sample and let the sample be ordered such that $k^{[1]} \leq k^{[2]} \leq \cdots \leq k^{[F]}$.

$$RATM = \frac{\sum_{f \leq \lfloor \xi F \rfloor} k^{[f]}}{\lfloor \xi F \rfloor} \tag{6.4}$$

The value for the parameter ξ was set to 0.8 by Toktas et al. (2006). The same setting is used in this research. This means that the most optimistic 20% of all observations will be excluded. The use of the risk averse trimmed mean will reduce the risk of moving too many nodes to a later day resulting in insufficient capacity.

In practice, there is not a clear finite planning horizon as the planning of delivery routes is a continuous process. Thus, in reality, it might be possible to move nodes from the last day in the planning period to the day after. Setting a strict planning horizon may lead to accumulation of nodes on the last day, therefore, increasing the possibility of having to deal with excess demand on the last day. This would substantially increase the total cost while it might be possible to move a few nodes to a later day and fix the capacity problem that way. To allow for this, I add an extra day to the planning period which does not consist of realized data but of an approximation using subgrids identical to the approximation discussed above. The costs of this extra day are the additional costs incurred by adding nodes from previous days into the routes. If no nodes are moved to the extra day, the extra day costs are equal to zero. Costs are calculated in this manner, because taking the total cost of the extra day, including visiting the subgrid nodes, would add costs that are not considered in the same-day solution. To be able to fairly compare the solutions obtained from complete and partial information, this extra day is also added in case of complete information such that it is possible to move nodes from the last day in the planning period to the day after.

Chapter 7 Computational Experiments

This chapter discusses the results from the research and its implications. Firstly, I present some computational experiments to test the performance of the individual heuristics and to tune a selection of parameters in Section 7.1 and 7.2, respectively. Secondly, I present the results obtained from the finalized algorithm in Section 7.3. The distances between nodes are Euclidean distances and the cost c_{ij} is set equal to the distance of the corresponding arc d_{ij}.

7.1 Heuristic Performance

The first experiment is that of the heuristic performances individually. This test was done using one hundred randomly generated data instances which are generated according to the process described in Section 5. Each instance includes five days with one vehicle available each day and four or five nodes per day. In solving these small instances, the maximum number of iterations was set to 20. The bar chart in Figure 7.1 shows the relative performances of each heuristic in terms of percentage cost reduction and running time. Both measures are scaled such that the cost reduction and running time of the worst removal heuristic (WR) are both equal to 1.

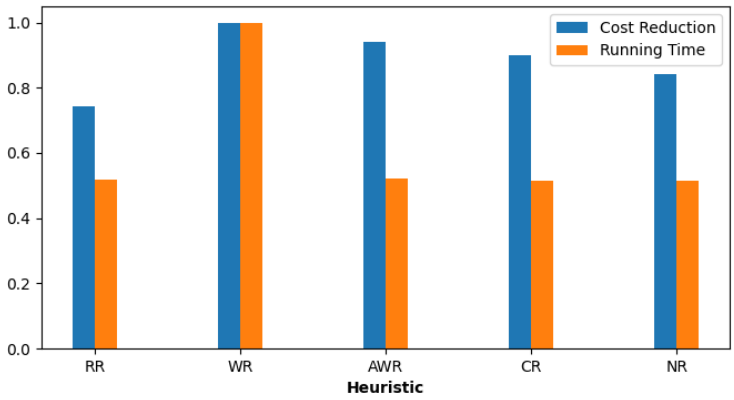


Figure 7.1: Performance of individual heuristics

Firstly, Figure 7.1 shows that the random removal heuristic performs noticeably worse than the other four heuristics. Which is to be expected as the removals in the other heuristics are all based on some measure of similarity of each node with the other nodes, whereas, in the random removal heuristic, every node has equal probability to be removed. Secondly, the running time of the worst removal heuristic is about twice as long as all other heuristics. This can be explained by the fact that in the worst removal heuristic a vehicle routing problem has to be solved for all possible scenarios where one node is removed to decide which node is the worst. Evidently, this process takes a lot of time. Even though the worst removal heuristic takes substantially longer than the approximate worst removal, it does not perform significantly better. For this reason, the worst removal heuristic will be excluded from the algorithm in the remainder of this research.

7.2 Parameter Tuning

From this point onward, the data instances used in all tests and results are constructed from the benchmark instances presented in Augerat et al. (1995) as discussed in Section 5.2 with a planning horizon of five days. For tuning the parameters the set of training instances is used. In solving these instances, the maximum number of iterations was set to 100 with a segment size of 10 iterations.

For parameter tuning, I follow the procedure described in Ropke and Pisinger (2006). Due the amount of time it takes to run the algorithm for several values per parameter to be tuned, the parameters are tuned on ten training instances. The process starts with an initial guess of the parameters, after which one parameter is allowed to take a number of values, keeping all other parameters fixed. For each value, the algorithm is applied to ten training instances and the parameter value giving the highest average cost reduction is chosen. This is repeated for all parameters.

The initial guess for the parameters are the values found during the tuning procedure in Ropke and Pisinger (2006). I chose to retune seven parameters including the randomness parameter, p, the starting temperature control parameter, φ, the cooling rate, c, the reaction factor, r, and the three scores assigned to heuristics in the adaptive heuristic selection algorithm, σ_1, σ_2, σ_3.

The tuning procedure resulted in the following parameter values (p, φ, c, r, σ_1, σ_2, σ_3) = (6, 0.03, 0.9, 0.2, 25, 10, 5). Some notable results are those from tuning the randomness parameter as shown in Figure 7.2.

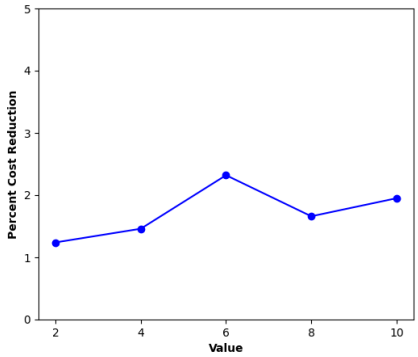
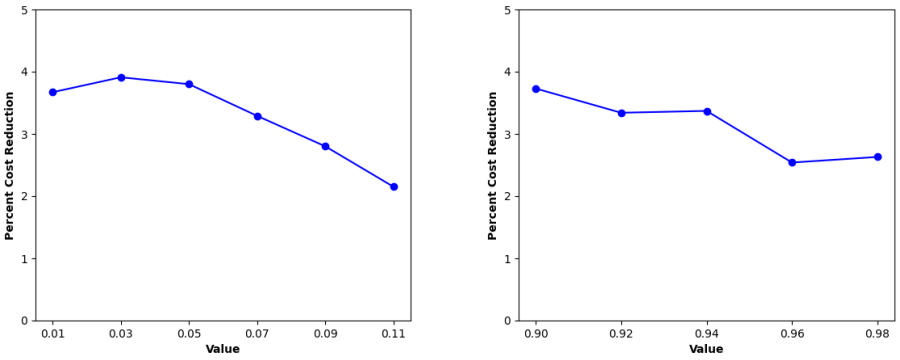


Figure 7.2: Randomness parameter

The initial guess for the randomness parameter was 3, however, the tuning process revealed that a higher value, namely 6, gives better performance. As a higher randomness parameter value reduces the degree of randomness in the node selection, this suggests that the ordering of the nodes before selecting a node is beneficial for the performance of the algorithm.

Furthermore, tuning the parameters in the simulated annealing acceptance criterion shows that a lower starting temperature control parameter (0.03 w.r.t. 0.05) and a lower cooling rate (0.9 w.r.t. 0.999) is preferred for this algorithm over the values suggested by Ropke and Pisinger (2006). The results from tuning these parameters are shown in Figure 7.3b.

A lower starting temperature control parameter and a lower cooling rate correspond to a lower starting temperature and a faster decreasing cooling schedule. This can be explained by the fact that this algorithm performs 100 iterations in contrast to the 25000 iterations Ropke and Pisinger (2006). Less



(a) Starting temperature control parameter

(b) Cooling rate

Figure 7.3: Tuning the simulated annealing parameters

iterations means that the algorithm has to get to the best solution more quickly. This is in line with a smaller cooling rate, or a lower temperature in each iteration, as a lower temperature means that the probability of accepting a worse solution is lower. Similar figures for all tuned parameters can be found in Appendix B.

The other parameters that were not retuned are the correlated removal parameters ρ, δ, and ε which were taken from Wen et al. (2010), and the parameter used in the risk-averse trimmed mean, ξ. These parameters were set to $(\rho, \delta, \varepsilon, \xi) = (20, 2, 1.5, 0.8)$.

7.3 Outcome Analysis

Finally, the algorithm with tuned parameters is tested on the set of test instances as discussed in Section 5.2. This test is performed under both complete information and partial information to establish the impact of uncertainty on the performance of the heuristics. In case of partial information, the history used to set the demands in the subgrid approximations, consists of the 22 remaining benchmark instances excluding those which are included in the planning period. The maximum number of iterations is again set to 100 with a segment size of 10. In the case of partial information, this means that 100 iterations are performed for each day. In the approximation of future days, the total grid is divided into 25 subgrids. The running time of the algorithm with complete information took around 5 minutes on average. With partial information, thus performing five times as many iterations, the average running time is about 25 minutes. Summary statistics of the results of both tests are displayed in Table 7.1.

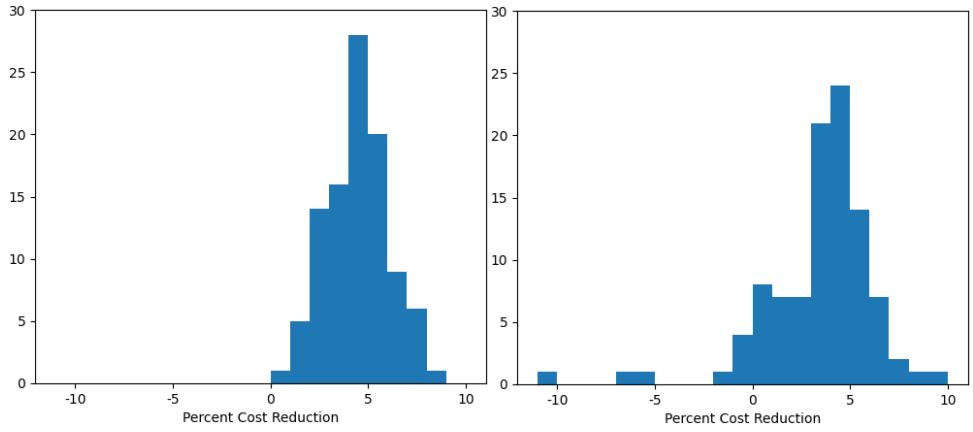
	Complete Information	**Partial Information**
Mean	4.47%	3.40%
Median	4.56%	3.99%
Stdev	1.66%	2.82%
Minimum	0.38%	-10.10%
Maximum	8.51%	9.04%

Table 7.1: Summary statistics of solutions under complete and partial information

The results as shown in Table 7.1 follow expectations. Firstly, on average the cost reduction is lower

in the case of partial information, which can intuitively be explained as having less information makes it more difficult to make the optimal decisions. Secondly, with complete information it is always a possibility to stick to the same-day solution if no better solution can be found. Thus, the cost reduction will never be negative. However, in case of partial information, decisions cannot be reversed after solving a day and, thus, it is possible to end up with a worse solution than the same-day solution. For that reason, we observe a negative minimum under partial information. Also, due to the lack of information, therefore, less accurate decisions, and the possibility of increasing the total costs, the standard deviation of the cost reduction under partial information is higher than under complete information. This is also depicted in Figure 7.4, which shows the histograms of the outcomes under both complete and partial information.

An important thing to note is that, on average, the cost reduction under partial information is not very much lower than the cost reduction under complete information (3.40% and 4.47% respectively). This shows that the heuristics still perform well with uncertainty and that the algorithm is still viable. This result is not unexpected as all heuristics base the node selection only on the other nodes on the same day and not on the nodes of future or previous days. Thus, heuristics would select the same nodes to be moved, if not for the added randomness, regardless of the available information on other days. However, with complete information there is the possibility of reversing a move in a later iteration if it turns out that the moved node is also misplaced in some way on the later day. With partial information, this is impossible as the information of the later day is not available. This, together with the uncertainty regarding the remaining capacity, is most likely the cause of the overall decrease in cost reduction under partial information with respect to complete information.



(a) Complete Information

(b) Partial Information

Figure 7.4: Histograms of cost reduction percentages w.r.t. same-day solution

As shown in Figure 7.4, the cost reduction varies significantly across different instances, especially under partial information. In studying the composition of the test instances in terms of the benchmark CVRP instances, it shows that in the instances that resulted in a negative cost reduction, the last day in the planning period was relatively tight in capacity. Consequently, moving nodes to this day lead to excess demand which could not be cleared up by moving nodes to the extra day. On the other hand, in instances that resulted in a high cost reduction, the tigh of the last day was relatively low.

The composition of the instances can be used to explain the general variability in the quality of the solutions. However, part of the variability is due to the inconsistency of the algorithm as depicted in Figure 7.5. I applied the algorithm to the exact same instance 10 times and Figure 7.5 shows the obtained

solutions under complete (CI) and partial (PI) information.

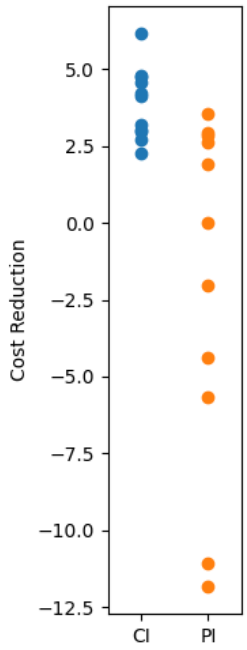


Figure 7.5: Variance of Solutions

As illustrated in Figure 7.5, the solution varies substantially in each run, ranging from a cost reduction of 2.29% to 6.16% with complete information and −11.83% to 3.56% with partial information. This suggests that the algorithm is not fully reliable as the solution could be very different if run again. In this particular example, two days, including the last day, have a relatively high tigh, which makes it a difficult instance to solve with partial information. For other easier instances, the solution variance is less extreme than in this example but still evident. With complete information, one could possibly run the algorithm multiple times and go with the best solution. However, with partial information this is not an option, thus, in a way leaving the solution quality to chance.

Chapter 8 Conclusion

With growing sustainability expectations of the e-commerce industry and the deliveries, it is important for companies within this industry to evaluate different strategies to accomplish their sustainability goals. This research explores a strategy of delaying deliveries to allow for more efficient routing. The same-day approach is compared to the same-or-next-day approach where deliveries can be delayed with a maximum of one day. In solving a planning horizon of five days, each day is represented by a Capacitated Vehicle Routing Problem (CVRP), which can be solved independently for the same-day approach. To solve the sequence of CVRPs with the same-or-next-day approach, an Adaptive Large Neighbourhood Search algorithm was developed that moves nodes across the days using four different heuristics.

Assuming complete information with regard to the deliveries that have to be performed on all days, the algorithm was able to attain an average cost reduction of 4.47%. To solve the problems with partial information, meaning the days in the planning horizon are solved sequentially without knowledge of any deliveries that will appear on later days, future days were approximated. Using partial information, the algorithm attained an average cost reduction of 3.40%. The relatively small difference between the results from complete and partial information shows that the proposed algorithm still performs well under uncertainty. The cost is calculated as the total distance travelled by all vehicles. Thus, the reduction in cost is also directly related to a reduction in the emissions.

Even though this research shows promising results, the proposed algorithm lacks in consistency as solutions may vary significantly across several runs on the same instance. Thus, further research into solution consistency may prove to be very beneficial and result in even higher cost reductions. Furthermore, the addition of route duration limits can be explored as in this research these were set to a very large number such that they are never binding and have no impact on the solution. In reality, route duration is likely to be a very important constraint as the drivers of the vehicles can only work for a limited amount of time consecutively.

In conclusion, the results presented in this report show that increasing the flexibility of when a parcel is delivered can lead to a reduction in total distance travelled by the delivery vehicles. Consequently, not only does this result in a more cost-effective planning of the delivery routes but it also reduces the CO2 emissions from the delivery vehicles. By applying an approach such as the same-or-next-day approach, delivery companies can reduce costs, increase sustainability and, therefore, potentially attract a larger market share in the increasingly sustainability conscious and growing e-commerce industry.

Bibliography

(2022). Retailers: Sustainability is Not a Challenge, It's an Opportunity. Technical report, Descartes System Group, Inc.

Augerat, P., Naddef, D., Belenguer, J., Benavent, E., Corberan, A., and Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, 63(1):2–15.

Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.

Letchford, A. N. and Salazar-González, J.-J. (2015). Stronger multi-commodity flow formulations of the capacitated vehicle routing problem. *European Journal of Operational Research*, 244(3):730–738.

Mor, A. and Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, pages 1–21.

Oyola, J., Arntzen, H., and Woodruff, D. L. (2018). The stochastic vehicle routing problem, a literature review, part I: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435.

Psaraftis, H. N., Wen, M., and Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Shelbourne, B. C., Battarra, M., and Potts, C. N. (2017). The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, 29(4):705–723.

Toktas, B., Yen, J. W., and Zabinsky, Z. B. (2006). Addressing capacity uncertainty in resource-constrained assignment problems. *Computers & operations research*, 33(3):724–745.

Vidal, T. (2022). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

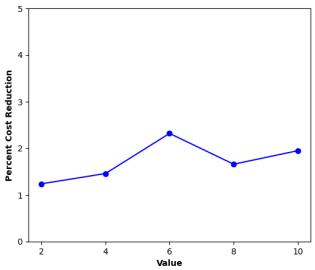
Wen, M., Cordeau, J.-F., Laporte, G., and Larsen, J. (2010). The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37(9):1615–1623.

Appendix A Benchmark Instances

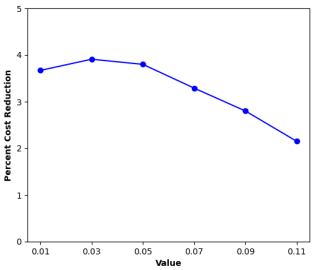
Instance	n	k	Tigh	Instance	n	k	Tigh
A-n32-k5	31	5	0.82	A-n48-k7	47	7	0.89
A-n33-k5	32	5	0.89	A-n53-k7	52	7	0.95
A-n33-k6	32	6	0.90	A-n54-k7	53	7	0.96
A-n34-k5	33	5	0.92	A-n55-k9	54	9	0.93
A-n36-k5	35	5	0.88	A-n60-k9	59	9	0.92
A-n37-k5	36	5	0.81	A-n61-k9	60	9	0.98
A-n37-k6	36	6	0.95	A-n62-k8	61	8	0.92
A-n38-k5	37	5	0.96	A-n63-k9	62	9	0.97
A-n39-k6	38	6	0.88	A-n63-k10	62	10	0.93
A-n39-k5	38	5	0.95	A-n64-k9	63	9	0.94
A-n44-k7	43	7	0.81	A-n65-k9	64	9	0.97
A-n45-k7	44	7	0.91	A-n69-k9	68	9	0.94
A-n45-k6	44	6	0.99	A-n80-k10	79	10	0.94
A-n46-k7	45	7	0.86				

Table A.1: Some properties of the benchmark instances

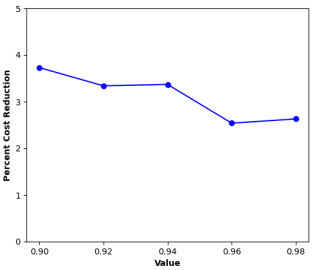
Appendix B Tuning Graphs



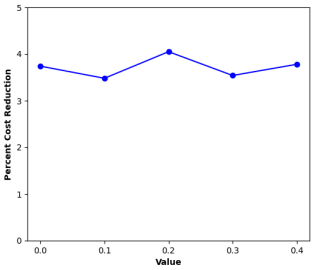
(a) Randomness parameter



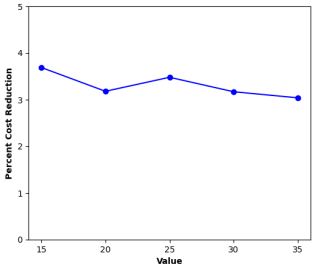
(b) Starting temperature control parameter



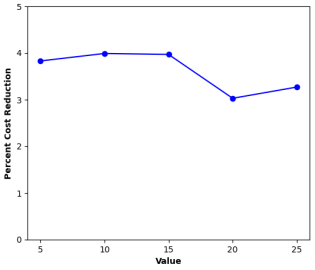
(c) Cooling rate



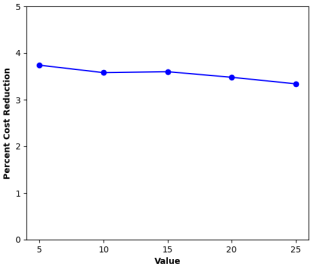
(d) Reaction factor



(e) New best solution score



(f) Better solution score



(g) Worse solution score

Figure B.1: Tuning Parameters