A Vehicle Routing Problem with Multiple Service Agreements

European Journal of Operational Research 313 (2024) 129–145

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

A vehicle routing problem with multiple service agreements

Vincent C. G. Karels[a,*], Walter Rei[b], Lucas P. Veelenturf[c], Tom Van Woensel[a]

[a] School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, the Netherlands
[b] Département de management et technologie, Université du Québec à Montréal, Montréal, Canada
[c] Rotterdam School of Management, Erasmus University, Rotterdam, the Netherlands

ARTICLE INFO

Article history:
Received 5 May 2022
Accepted 21 July 2023
Available online 24 July 2023

Keywords:
Transportation
Routing
Stochastic
Service agreements
Branch-and-bound

ABSTRACT

We consider a logistics service provider which arranges transportation services to customers with different service agreements. The most prominent feature of this service agreement is the time period in which these customers send their orders and want to retrieve delivery information. After customers place their orders, they require information about the driver and an early indication of the arrival times. At the moment, this information needs to be provided. The order information of other customers with a different service agreement that needs to be serviced in the same period might still be unknown. Ultimately all customers have to be planned, constrained by the information provided to the customers in the earlier stage. In this paper, we investigate how the logistic service provider plans its routes and communicates the driver and arrival time information in the phase where not all customers are known (stage 1). Once all customer orders are known (stage 2), the final routes can be determined, which adhere to the already communicated driver and arrival time information from stage 1, minimizing total routing cost. For this problem, an exact algorithm is presented. This problem is solved using a novel tractable branch-and-bound method and re-optimization in stage 2. Detailed results are presented, showing the improvements of using re-optimization. We show that integrating the planning of the customers with the different service agreements leads to significant cost savings compared to treating the customers separately (as is currently done by most logistics service providers).

1. Introduction

In the modern world, competition, public expectations, and regulating authorities put a lot of pressure on logistics service providers. The challenge of efficiently organizing the physical distribution process became harder and harder to solve. Over the past years, substantial efforts have been dedicated to finding solutions to this challenge, especially focusing on one of the core elements of physical distribution, routing, and scheduling, leading to significant savings for logistics service providers.

Goods transportation is a central activity within modern economies. This activity poses important challenges for logistics service providers, who are tasked with efficiently planning and organizing the transportation operations required to fulfill customer requests (e.g., commercial shippers, citizens, governmental organizations, etc.). On the one hand, logistics service providers must plan and organize operations while minimizing the transportation costs incurred. On the other hand, they must also meet their customers' desired service quality levels (e.g., shippers expecting deliveries to be made within certain time frames). At the heart of many transportation planning processes, one thus finds routing and scheduling problems that need to be solved. First, how goods are distributed to the customers via routes (i.e., sequences of visits) performed by a fleet of vehicles directly impacts the costs incurred. Secondly, the schedules that are fixed to perform such routes define the timing at which each visit occurs, simultaneously establishing whether or not service quality levels are met. This being said, Vehicle Routing Problems (VRPs) are notoriously hard to solve, even in their basic form, as shown by Toth & Vigo (2002) and Gendreau et al. (2014).

Most of the literature is based on the assumption that all information is known in advance. As a result, the routing problem is deterministic in its input. However, during execution, logistics service providers face uncertainty in unexpected events (additional demand) and expected variations (stochastic travel times). When routes and schedules are planned, a part of the parameters defines the stochastic problem. This necessitates that additional (sometimes costly) decisions be made when these stochastic parameters

* Corresponding author.
E-mail address: v.c.g.karels@tue.nl (V.C.G. Karels).

https://doi.org/10.1016/j.ejor.2023.07.029
0377-2217/© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

become eventually known to ensure that the routing and scheduling plans can still be feasibly executed.

Optimizing a route plan is usually a computationally difficult and time-consuming task. Therefore, as more information becomes available, the logistics service provider usually does not have the time to re-optimize. Instead, routes should be designed to absorb unexpected events. As a result, the question becomes how much of the routing operations should be planned versus how much should be decided as a reactive measure to the new information revealed (i.e., stochastic parameters becoming known).

This research investigates the effect of different service agreements between the customers and the logistics service provider. Especially, service agreements stipulate when the customer provides the order information and when the logistics service provider indicates when he visits the customer. To be most efficient, logistics service providers prefer early order information and a late announcement of the visiting time. For the customer, the reverse holds: it prefers to announce the orders as late as possible before the delivery takes place and wants to know the visiting time as early as possible such that the workforce can be organized. If specific agreements are considered, some decisions need to be made a priori, e.g., the needed number of required vehicles/drivers and the quoted information to each customer (the specific driver that will perform the visit and the timing of the visit), while still, some uncertainties exist.

The problem investigated in this paper is inspired by real problems faced by logistics service providers in the Business to Business (B2B) retail sector in a multichannel environment. Here contracts differ between customers, where larger customers such as supermarkets or larger retailers order more in advance and need more detailed information on the approximate arrival time for workforce planning. Smaller retail shops with small inventory capacity reveal orders as late as possible (to be as accurate as possible with their replenishment and to have the option to include online pick-up-at-store orders).

In our problem, we consider such a logistics service provider with customers with one out of two different service agreements. Under the first agreement, customers order two days in advance and return and receive information about their allocated driver and time window that same day. Under the second agreement, the same guarantee is offered, though one day in advance. An intermediate plan is formulated to provide customers under the first agreement with the required information. During the formulation of this intermediate plan, customer orders under the second agreement are uncertain in both their presence and demand. However, some information about this uncertainty is known by, for example, historical information. When customers' orders under the second agreement are known, routes need to be constructed to serve all customer demands appearing on a given day, including customers from both service agreements, while minimizing costs. Each operating day of the logistics service provider, the problem is resolved again. This happens continuously for the duration of operations. In our paper, we assume that the demand between operating days is independent, and therefore we investigate the problem from the viewpoint of a single operational day.

The *vehicle routing problem with stochastic demands and customers* (VRPSCD), which shares similarities with our problem, has already received some attention in the literature. The VRPSCD is a special combination of two uncertainties: stochastic demands and stochastic customers. Due to the special characteristics of our problem, we could not directly use the methodologies available for the VRPSCD. Therefore, a completely new solution method is introduced.

Specifically, we introduce an exact algorithm for this problem. Exact algorithms are generally less tractable than heuristics but provide the *proven* optimal solution. This can help to build a solid foundation for further research into heuristics with more practical applications. Similar to existing literature, we divide the problem, the Vehicle Routing Problem with Multiple Service Agreements, into two stages. During the first stage, an intermediate plan is formulated, which could subsequently be considered a constraint for the plan formulated during stage 2. The algorithm is a variation of the branch-and-bound algorithm, where we branch on which request is allocated to which route (the allocated driver) and in which sequence they appear on the route (approximation of the time window). However, we allow for *re-optimization* during the second stage, whereas most of the literature considers a *recourse* policy. Finally, vehicles requisitioned during the second stage (which is applied the evening in advance) are more expensive than those ordered in the first stage (applied two days in advance).

Our contributions to the body of stochastic vehicle routing research are as follows:

- We introduce the Vehicle Routing Problem with Multiple Service Agreements. This model cannot be solved with methods currently available in the literature.
- For this problem, we present a novel exact algorithm based on branch-and-bound techniques. This methodology can solve instances of up to 15 customers, of which six are stochastic.
- A comparison is made between our algorithm and several other solution methods for which numerical results and insights are presented.

The structure of this paper is as follows: The literature related to our problem is investigated in Section 2, and the differences between our problem and similarities are explained. The problem is detailed in Section 3, including discussions of why existing solution methods are insufficient to solve the problem. The solution method is formulated in Section 4, with each step presented in detail in the following sections until Section 7. We present the performance of our solution method on the problem in Section 8.

2. Literature review

Our problem relates to vehicle routing problems with stochastic customers and demands. In this section, the current relevant body of literature is presented. The first three sections present the general stochastic optimization methodologies developed for variants of the VRP that are directly related to our own. In contrast, the last part presents the studies conducted on how service consistency plays a role when planning distribution operations.

2.1. Vehicle routing problem with stochastic demands

The Vehicle Routing Problem with Stochastic Demand (VRPSD) is the most researched stochastic variant of routing problems. The first research on this topic was performed by Tillman (1969). Dror et al. (1989) subsequently studied the properties of this model and provided valuable insights into the structure of the optimal solution to the VRPSD. Bertsimas (1988) introduced *a priori* optimization, which divides the problem in two stages. In the first stage, a planned solution is designed, while in the second stage, uncertainties are revealed, and the solution is repaired based on a predetermined *recourse* policy. The *classical* policy is to return to the depot when capacity is exceeded or when the vehicle is empty, depending on whether it is a pickup (e.g., waste collection) and/or delivery problem. Several different recourse policies have been considered in the literature. The optimal policy for one vehicle was determined by Florio et al. (2020). The problem was subsequently solved under the optimal restocking policy by Salavati-Khoshghalb et al. (2019a). An important milestone was reached with the publication of Laporte et al. (2002), where the integer-L-shaped algorithm was applied to the problem. For an extensive overview of the literature on the VRPSD we refer to Vigo (2015) and Oyola et al. (2018). A problem with additional stochastic service times and time windows was investigated by Goel et al. (2019). One important difference between the previously investigated problems and our problem is associated with the fact that we know the complete demand at the beginning of the second stage, such that we can apply a more advanced partial re-optimization recourse strategy, where in most works, the demands will be revealed dynamically (e.g., when visiting a customer).

Usually, a simple recourse policy is chosen to keep the model tractable. There has been prior research into allowing for re-optimization, which was introduced by the work of Secomandi (2001). This author formulated the problem as a stochastic shortest-path problem, a finite-state Markovian Decision Problem. It should be noted that considering the number of stochastic parameters and their associated random distributions, the number of states can become very large. This results in the problem formulation rapidly becoming intractable for many customers. Nevertheless, this was one of the first methodologies that allowed for re-optimization. The author continued his work on this method, publishing Secomandi & Margot (2009) and Bertazzi & Secomandi (2018a). Comparisons were made with restocking in Bertazzi & Secomandi (2018b).

To define the types of recourse actions that have been proposed for the VRPSD we follow the general classification that was proposed in Salavati-Khoshghalb et al. (2019b). Two general types of recourse actions can be implemented: 1) reactive (e.g., classical) and 2) proactive (e.g., restocking). Our proposed partial re-optimization strategy falls in the latter category (i.e., proactive actions). Once all the information regarding the stochastic parameters becomes known, our recourse strategy seeks to utilize the available capacity buffers included in the first-stage routes to service as many of the newly revealed customer requests to minimize the additional vehicles required.

2.2. Vehicle routing problem with stochastic customers

The Traveling Salesman Problem with Stochastic Customers (TSPSC) was introduced by Jaillet (1985), who studied some of its properties and proposed several solution methods. Each customer was given a probability p of being present in the problem. The problem was formulating a tour for all customers, which minimized costs in which customers who were not present were skipped. This work was expanded upon by Jaillet & Odoni (1988) and Bertsimas & Howell (1993), who proposed a series of heuristic algorithms. Laporte et al. (1994) developed an exact branch-and-cut algorithm for the problem. Jezequel (1985) and Jaillet (1985) investigated a version of the TSPSC where the demands were of unit size. It was further noted by Jaillet (1985) that large vehicle capacities may yield higher solution costs and that the costs of the solutions depend on the travel orientations chosen for the routes, even for problems that involve symmetric distances. Waters (1989) considers three strategies for the TSPSC, which are the following: 1) apply the original plan, 2) skip absent customers and finally, 3) re-optimize the route. It was noted that, as the number of uncertain customers rises, rescheduling becomes preferable. They use stochastic programming to solve the problem. In general, the literature concerning stochastic customers specifically remains relatively limited.

2.3. Vehicle routing problem with stochastic demands and customers

The first variant of the Vehicle Routing Problem with Stochastic Demand and Customers (VRPSDC) was formalized by Bertsimas (1992). Gendreau et al. (1995) designed an exact algorithm for this problem. The problem is solved using the *a priori* optimization strategy, as introduced by Bertsimas (1988).

In the considered problem variant, an *a priori* routing plan (i.e., a set of vehicle routes that visit all customers) is sought that minimizes overall *a posteriori* routing costs. Once the routes are performed, and the stochastic parameters are observed, the applied recourse actions are: 1) skip absent customers and 2) apply classical recourse actions whenever a vehicle's residual capacity is insufficient to service a customer's observed demand. The problem is formulated as a stochastic integer program and is solved via the integer L-shaped algorithm. Instances with up to 42 customers were solved using this method.

Gendreau et al. (1996) proposed a new meta-heuristic for the VRPSDC based on TABU search (the algorithm was coined TABUSTOCH). The use of this heuristic was observed to be computationally very expensive. Instances with up to 46 customers and two vehicles were solved. The computational challenges related to TABUSTOCH resulted in evaluating the quality associated with the current solution moves considered in the applied neighborhood at each iteration performed by the algorithm. This was the motivation behind the development of the empirical estimation approach. Finally, additional meta-heuristics were introduced by Balaprakash et al. (2015). They use an empirical estimation approach and show that it is more effective than the approach used in Gendreau et al. (1996). They only investigated single-vehicle routing problems. More recently, Sörensen & Sevaux (2009), Erera et al. (2009), and Beraldi et al. (2010) have all investigated variants of this problem and developed algorithms based on tabu search, insertion heuristic search, and neighborhood search, respectively. In all cases, these methods were used to test the flexibility of solutions to deal with irregular customers.

2.4. Consistent vehicle routing

This paper specifically addresses the challenge of solving a VRP in which two distinct types of customers must be serviced. The problem setting includes explicit service agreements that structure the planning of the distribution operations. Although service agreements can offer different provisions for the customers, we consider the case where a certain level of consistency is imposed in the vehicle routes. We thus briefly review here the literature dedicated to consistent vehicle routing problems.

The consistent vehicle routing problem was first formulated in Groër et al. (2009). In this variant, customer satisfaction was achieved by 1) fixing the assignment of customers to drivers and 2) imposing time consistency. Time consistency is when companies want their drivers to develop relationships with customers on a route and have the same drivers visit the same customers roughly the same time each day they need service. The problem is formulated as a mixed integer program and solved optimally for small instances. A local search algorithm is also designed to solve larger instances. The generalized consistent vehicle routing problem was formulated in Kovacs et al. (2015). Here, multiple drivers can be assigned to customers, relaxing the driver consistency criteria and allowing for greater satisfaction. More recently, Biesinger et al. (2018) investigated a genetic algorithm with a solution archive for the problem. This algorithm proved to be effective when compared to other metaheuristics.

In Jabali et al. (2015) the vehicle routing problem with self-imposed time-windows is investigated. Here, a tabu search heuristic assigns customers to vehicles and establishes the order of visits of the customers per vehicle. The LP model subsequently generates detailed timing decisions, whose output also guides the local search in a feedback loop. Later, Spliet & Gabor (2015), contributed to the research on self-imposing time windows. Their work can be considered a variant of the consistent vehicle routing problem where time windows have to be assigned before demand is known, referred to as the time window assignment vehicle routing problem. A similar problem was investigated in Neves-Moreira et al. (2018), where the stochastic demands are product dependent. For more details on the different variants of emerging vehicle routing problems, we refer the reader to Vidal et al. (2020).

Given the current body of scientific literature, in this paper, a new problem is presented. A novel feature of the problem is the introduction of different service agreements, which generally have requirements on the timings orders have to be placed by customers and information on the delivery has to be revealed. This information on the delivery generally includes a time indication and which driver will service the request. If these timings are different for different customers, it causes uncertainty for the planning. A novel solution method is thus additionally required.

3. Problem definition

A logistics service provider performs distribution services for a set of customers. These customers are divided into two categories based on their respective service agreements. The agreement specifies when they inform the logistics service provider of their requests, establishing the service quality provisions offered when performing the distribution operations. Specifically, customers under the first agreement inform the logistics service provider about their requests two days in advance. In return, on the same day, these customers receive information about their estimated arrival time and which driver is assigned to their request. Their estimated arrival time is approximated by order of the visits on the route to customers under this service agreement. The same information is transferred to customers under the second agreement, with the exception that these customers inform the logistics service provider one day in advance. The second agreement offers the same service quality provisions to the customers. Ultimately, all customers need to be serviced.

Let the set V_1 represent the requests under the first service agreement and V_2 the set of requests under the second service agreement. To provide customers from V_1 with the required information, one has to plan in two separate time periods, known as stages. In the first stage (i.e., two days in advance), an intermediate plan for the requests in V_1 is formulated. When formulating the intermediate plan in stage 1, the request locations in category V_2 are stochastic. Neither their presence nor their demand is known exactly when formulating this plan. However, some information is known that can be used to formulate probability distributions for either of these attributes. In the second stage (i.e., one day in advance), all information regarding the requests in V_2 becomes known (i.e., both the presence and the demands of these requests are observed). The routing plan is then adjusted to service all materialized requests. However, the intermediate plan's basic *structure* is preserved when the final vehicle routes are established to serve the request locations in both V_1 and V_2. According to the service agreement with the customers from V_1, the following decisions from the intermediate plan need to be preserved:

- Requests in V_1 remain on their allocated route of the intermediate plan. (*Driver guarantee*)
  The driver guarantee offers value to the customer and the logistics service provider. A customer that receives its service through a driver familiar with their logistics situation will benefit from an experienced delivery service. This should result in a fast service according to the customers preferences. A second benefit is that the logistic service provider can start preparatory work sooner. They can start the order picking processes and filling the trucks of the customers of V_1 even before the customers of V_2 announce their orders. This in turn decreases the pressure on the order pickers.
- The order of the visits in the routes of the intermediate plan is preserved. (*Time-order guarantee*)
  The logistics service provider would, as a service to the customer, report a time estimate for when they expect to arrive with their delivery. The logistics service provider provides time-windows, for which they select both the start and end times. These windows are chosen such that the driving cost for the route is minimised. Modelling these time-windows is a complex problem in itself. We have elected to model this via preserving the sequencing of the requests. As such, large differences between the self reported time-windows and the actual delivery times are avoided.

If not all customer requests in the set V_2 can be accommodated on the routes established in the first stage, then additional vehicles are requisitioned and charged a premium.

Drivers need to be informed of their shifts on time. In general, there is a monetary penalty, usually in the form of additional wages, when a driver is informed late that work is available. Also, from the vehicle utilization perspective, timely knowledge about the usage is important. For example, vehicles require maintenance, which needs to be effectively planned. Inefficiencies may occur when a vehicle scheduled for maintenance is suddenly required to perform operational duties. As a result, we need to charge a premium for using additional vehicles in the second stage.

Summarized, the objective is to establish an *intermediate plan* to serve the requests in V_1 that minimizes the expected cost of the *final plan* that serves all the requests in both V_1 and V_2. A simple example visually represents the overall planning process in Fig. 1. In this example, set V_1 includes the requests represented by the black dots $\{1, \ldots, 6\}$. In the first stage, three vehicle routes (represented by the red-dotted lines) are established to service the requests in V_1. In the second stage, four additional requests appear from set V_2, represented by the black dots $\{7, 9, 10, 11\}$. The final set of vehicle routes is established (represented by the blue lines). Thus, requests $\{9, 10, 11\}$ are accommodated via the routes of the intermediate plan, while an additional vehicle is requisitioned to service the request of request $\{7\}$. In the end, three vehicles were planned in the first stage (i.e., the black vehicles in Fig. 1), while an additional vehicle was used in the second stage (i.e., the blue vehicle in Fig. 1).

3.1. First stage model

A logistics service provider supplies a set of customer locations with a homogeneous fleet of vehicles, each vehicle having a capacity Q and an associated cost γ_1 if the decision to use it is made in stage 1 and γ_2 if the decision to use it is made in stage 2, where $\gamma_2 > \gamma_1$. In a graph G, these request locations are represented by vertices $V = \{1, 2, 3, 4, \ldots, n\}$, where it is assumed that each vertex $i \in V$ resides at a different location. Let $X = [x_{ij}]_{n\times n}$ be the decision matrix to move from vertex i to j. Vertex 0 denotes the depot. The request locations are connected through a set of edges $E = \{\langle i, j\rangle : i, j \in V, i \neq j\}$ where each edge has an associated driving cost defined in set $C = \{c_{ij} : \langle i, j\rangle \in E\}$. It is assumed that the driving costs are symmetric ($c_{ij} = c_{ji}$). In Stage 1, planning the vehicle routes considers the requests emanating from the two subsets of requests, $V_1 \cup V_2 = V$. The requests in set V_1 are considered to be deterministic. Thus their demands $d_i : i \in V_1$ are known discrete values.

The requests originating from requests in V_2 are stochastic. Thus, each request has a probability of appearing on a given day. When a request is present, there is a random distribution to formulate the demand of the request. A combination of specific re-

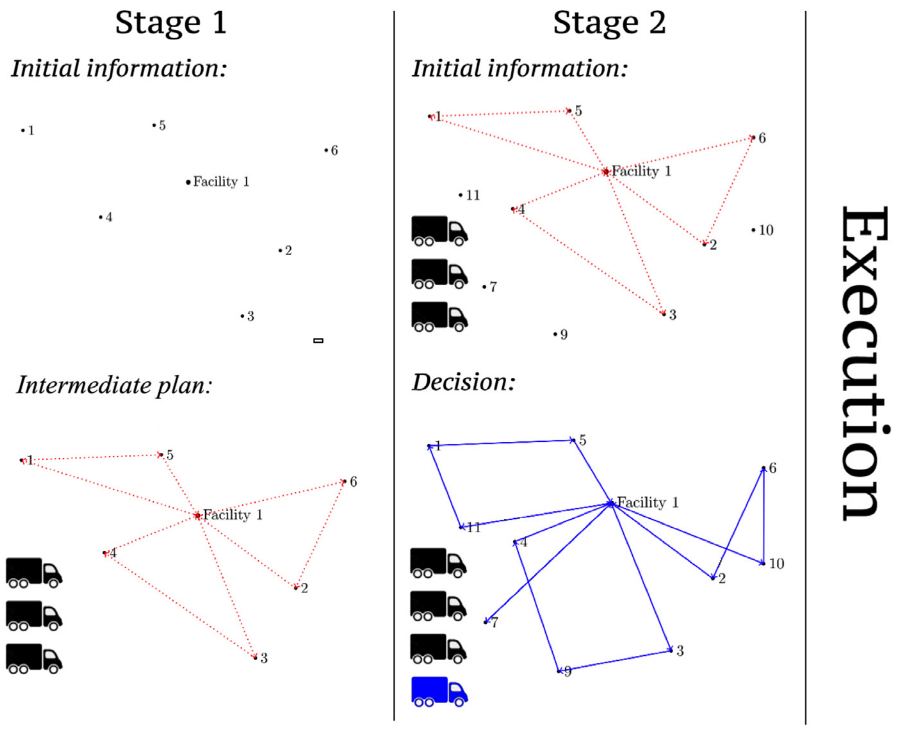


Fig. 1. Example of our single item limited reassignment auction.

quest presences and associated demands is a scenario denoted by s. From now on, we denote V_2^s to indicate the second-stage requests present in scenario s. Thus, let $P = \{p_i : i \in V_2, 0 \leq p_i \leq 1\}$ define the set of probabilities of requests being present. The probability of observing a demand value of d_i^k given that request i is present in the second stage is given by $w^{d_i^k} \in W$. Here each k is an index for which demand realization. Any demand d is always smaller than the vehicle capacity Q. Let m_1 denote the number of vehicles selected in the first stage at a cost γ_1.

The total amount of scenarios is linked to both the number of requests in V_2 and the size of the set of probabilities W. Each scenario is unique and has a probability

$$\pi_s = \prod_{i \in V_2^s} w^{d_i^{ks}} \cdot \prod_{i \in V_2^s} p_i \cdot \prod_{i \notin V_2^s} (1 - p_i) \tag{1}$$

of occurring. To provide a detailed example, let us assume the case where there is one fixed request and two stochastic ones, each of which has a probability of $\frac{1}{2}$ of being present. Furthermore, let us suppose that each stochastic request also has a stochastic demand. Specifically, each stochastic demand follows a discrete random distribution with two possible realizations. In both cases, demand realization 1 occurs with probability $\frac{1}{3}$ and demand realization 2 occurs with probability $\frac{2}{3}$. Then the scenario where both stochastic requests are present, each of which with demand realization 1 is equal to $(\prod_{i \in V_2^s} w^{d_i^1} = \frac{1}{3} \cdot \frac{1}{3}) \cdot (\prod_{i \in V_2^s} p_i = \frac{1}{2} \cdot \frac{1}{2}) = \frac{1}{36}$.

This probability is equivalent to the product of the probabilities of presence p_i, not presence $(1 - p_i)$ and the probabilities for the demand levels $w^{d_i^k}$ for all requests in V_2^s. S denotes the full set of scenarios. $F(m_1, X, s)$ is the resulting cost from a intermediate plan solution X re-optimized in stage 2 for scenario s. $\sum_{s \in S} \pi_s \cdot F(m_1, X, s)$ is effectively the expected cost of intermediate plan X over all scenarios. The mathematical model Φ is formulated as follows:

$$\min \gamma_1 \cdot m_1 + \sum_{s \in S} \pi_s \cdot F(m_1, X, s) \tag{2}$$

Subject to:

$$\sum_{<0,j> \in E} x_{0j} \leq m_1 \qquad \forall j \in V_1 \tag{3}$$

$$\sum_{<i,j> \in E : i \neq j} x_{ij} = 1 \qquad \forall j \in V_1 \tag{4}$$

$$\sum_{<i,j> \in E : i \neq j} x_{ij} \leq 1 \qquad \forall i \in V_1 \tag{5}$$

$$y_i + d_j x_{ij} - Q(1 - x_{ij}) \leq y_j \qquad \forall \{i, j\} \in V_1, i \neq j \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V_1, i \neq j \tag{7}$$

$$d_i \leq y_i \leq Q \qquad \forall i \in V_1 \tag{8}$$

$$m_1 \in Z^+ \tag{9}$$

The minimization function Φ consists of the cost of a vehicle (γ_1) multiplied by the number of vehicles requisitioned (m_1) plus the expected second stage costs, which is derived from the first-stage route plan X and scenario index s. Constraint (3) ensures that the amount of vehicles leaving the depot is smaller than the amount requisitioned. Constraint (4) ensures each request is visited once. Constraint (5) assures that only one edge leaves each node. Constraint (6) makes sure that the vehicle is always under capacity (Q). Variable y_i is introduced to indicate the vehicle's cumulative volume up to request i. Constraint (6) implicitly captures the order of requests on a route via variable y_i. The probability of its combination of demands multiplied with its probability of the combination of requests multiplied by the second stage costs. $F(m_1, X, s)$ is the expected cost for plan X for a specific scenario s, which can be achieved by a specific vehicle routing problem, for which the mathematical model is presented in Section 3.2.

For convenience, the used notation can be found in Table 1.

Table 1
Legend of the used notation.

Φ	objective minimization function.
i, j	indices for a request in V.
e_{ij}	edge in E
V	set of all requests
V_1	first-stage requests
V_2	second stage requests.
n	size of set V.
$c_{i,j} \in C$	cost for driving an edge e_{ij}
k	index of demand realization.
$d_i^{k^s} \in D$	realization of demand for request i in scenario s.
$w^{d_i^{k^s}} \in W$	probability of demand value $d_i^{k^s}$ occurring for request i
$p_i \in P$	probability of a request i being present.
x_{ij}	decision variable to travel from i to j in the first-stage.
X	the agglomeration of all decision variables x_{ij}, known as the first stage plan.
π_s	probability of a scenario s occurring.
$F(m_1, X, s)$	expected cost for first-stage plan X for scenario s
z_{ij}	decision variable to travel from i to j in the second stage.
Z	the agglomeration of all decision variables z_{ij}, known as the second stage plan.
m_1	stage 1 vehicles requisitioned.
m_2	stage 2 vehicles requisitioned.
Q	vehicle capacity.
y_i	vehicle load after request i
$\Omega(Z)$	collection of routes resulting from a transformation of Z.
R	route: set of decision variables z_{ij} which imply a sequence of requests i and j.

3.2. Second stage model

As previously stated, the intermediate plan produces a set of vehicle routes that visit all requests in set V_1. To compute the expected cost of this plan, as defined in (10) for each scenario, a second stage model should be solved, which minimizes the final routing cost given the constraints set by the intermediate plan, which is considered as input. In this section, we present the mathematical model which, when solved, produces the optimal final plan *constrained by* the provided intermediate plan X for a given scenario s. In essence, the mathematical model enables the usage of any spare capacity present in the routes of the intermediate plan to accommodate the second-stage requests.

When a specific scenario s occurs, the remaining problem to solve is a deterministic Capacitated Vehicle Routing Problem with additional constraints to enforce the service agreements guaranteed to the requests in V_1 by the intermediate plan. The variable z_{ij} is a binary variable that indicates whether edge e_{ij} is used in the final routes of scenario s. Z is the collection of decision variables z_{ij}, which dictates the second stage plan. This is equivalent to the final plan for scenario s. Let m_2, a decision variable, be the number of additional vehicles requisitioned in the second stage.

The mathematical model to compute $F(m_1, X, s)$ can be described as:

$$\min \gamma_2 \cdot m_2 + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} c_{ij} z_{ij} \tag{10}$$

Subject to:

$$\sum_{<0,j>\in E} z_{0j} \leq m_1 + m_2 \qquad j \in V_1 \cup V_2^s \tag{11}$$

$$\sum_{<i,j>\in E: i\neq j} z_{ij} = 1 \qquad j \in V_1 \cup V_2^s \tag{12}$$

$$\sum_{<i,j>\in E: i\neq j} z_{ij} \leq 1 \qquad i \in V_1 \cup V_2^s \tag{13}$$

$$y_i + d_j^{k^s} z_{ij} - Q(1 - z_{ij}) \leq y_j \qquad \forall \{i, j\} \in V_1 \cup V_2^s, i \neq j \tag{14}$$

$$d_i^{k^s} \leq y_i \leq Q \qquad \forall i \in V_1 \cup V_2^s \tag{15}$$

$$z_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in V_1 \cup V_2^s, i \neq j \tag{16}$$

$$m_2 \in Z^+ \tag{17}$$

$F(m1, X, s)$ returns the objective value to the minimization problem. The second stage model is constrained by the intermediate plan, i.e., the solution to the first-stage model, by the service agreements to the requests in V_1. Thus, we include a set of additional constraints that will ensure that time-order guarantees are enforced on the final routes of scenario s. y_i represents the load, which implicitly has an order already known for each request $i \in V_1$ due to these constraints. k_s indicates the index k of the demand realization inherent to scenario s. In more practical terms, if second-stage requests are to be included in routes from the first stage, the ordering of the loads should still be the over both the first and second-stage models. In essence, for this model $x \Rightarrow y$. From this, we obtain constraint (18).

$$x_{ij} \Rightarrow y_i \leq y_j \qquad \forall i, j \in V_1 \tag{18}$$

Similarly, routes of X also have to be preserved along the constraints imposed by the service agreements. For this constraint, (18) is insufficient as it preserves only the ordering of the load, equivalent to the time-order guarantees. Let us first define the route, R, which is a set of decisions z_{ij} between any sequential i and j. Let us define $\Omega(Z)$ as the collection of routes, which is derived by a transformation over Z. Distinct routes R_1 and R_2 are observed as $z_{ij} = 0$ between any i in the sequence of R_1 and j in the sequence of R_2. This means the decision to travel (z_{ij}) is not made for any request i in route R_1 and any request j on route R_2, indicating the routes are completely separate. Our solution method allows the dynamic addition of constraints to the model. Let us consider requests a in the sequence of $R_1 \in \Omega(Z)$ and b in the sequence of $R_2 \in \Omega(Z)$, on distinct routes in the constraints implied from our first-stage model. Suppose our solution for the second stage model contains a route R with a, b in the sequence of R. In that case, we can eliminate this invalid route by adding additional constraint (19) and resolving the model.

$$\sum_{i,j\in R, (i\neq j)} z_{ij} = |R| - 1 \tag{19}$$

Constraint (19) is similar to the dynamic sub-tour elimination constraint found in the routing literature. As an example consider route R $= \{z_{01} = 1, z_{12} = 1, z_{23} = 1, z_{34} = 1\} = \{\ 0 \to 2 \to 3 \to 4$ $(\to 0)\ \}$, and suppose $a = 3$, $b = 4$ and that $a \nrightarrow b$ or $3 \nrightarrow 4$ which implies that 3 and 4 are on distinct routes in the first stage model. We eliminate this route by adding the constraint $z_{01} + z_{12} + z_{23} + z_{34} \leq 3$ to the model. This is equivalent to the currently four occupied edges $|R| = 4$ minus 1, which results in the 3 in the constraint. Observe that as a result of constraint (19), invalid route R cannot return in any future solution. Note that because of the inclusion of constraints (18) and (19), the costs of a solution constrained by an intermediate plan will always be greater or equal than the costs of a solution obtained by solving the scenario as a single stage optimization problem with no restrictions on the requests of set v_1.

A solution of this second stage model is equivalent to the blue solution in Fig. 1 given the intermediate plan (the dotted line), as Fig. 1 implies a scenario.

4. Solution approach

This section presents the algorithm we propose to solve the considered stochastic model. The solution is subdivided into two parts. The first part constructs the sets of requests that must be performed by the same vehicle. This is performed through a branch-and-bound algorithm. The second part then determines the ordering of the requests in a set to form routes.

The first part of our algorithm is based on a branch-and-bound search strategy. Our description of the proposed solution method focuses on its main algorithmic components: 1) the bounds that are computed to guide the search process and 2) the strategy that is applied to perform the exhaustive search of the feasible region of the considered stochastic model. In the second part of our model, we determine a property of the orders on requests that we can subsequently use for an algorithm to determine the best ordering for any route. This section is subdivided into the following subsections:

1. Creating sets (which will form the basis for routes).
2. Preserving the ordering on sets (to form valid routes).

4.1. Creating sets

We introduce several concepts we use to explain our branch-and-bound methodology for the first part, forming sets that are the basis for our routes. In our branch-and-bound model, we branch on nodes. Allow us to introduce the concept of a node in our branch-and-bound tree. Each node K in our branch-and-bound tree always contains the full collection of scenarios S and their **respective solutions** given the constraints passed through the branching process.

Each scenario is solved optimally in the *initial node* K' in our branch-and-bound tree. The costs associated with this initial node are the expected value of perfect information. The value of this node is also the **lower bound** on costs for this node for our branch-and-bound search. We end the search if the sets for all first-stage requests are equivalent.

Recall the service agreement that is provided to the customers from V_1:

1. **Driver guarantee**: Requests on the same route in the intermediate plan remain on the same route in the final plan.
2. **Time-order guarantee**: The order of the requests on the routes in the stage 1 plan is preserved.

When the sets for all first-stage requests are equivalent, we fulfilled the *Driver guarantee* part of the service agreement. When the sets over the first-stage requests are different, our branch-and-bound methodology (Fig. 4) starts to branch on the initial node K'. During the branching, constraints are iteratively added to the branching nodes.

We introduce the applied disjunction to obtain the feasible subregions created each time branching in the search tree is performed. This disjunction consists of constraints, which in turn consist of two first-stage requests (say a and b), which in the current node are in the same set for the solutions to some scenarios while in different sets to others. When we branch on this node, we enforce that a and b are in the same set $a \to b$, versus that they are not $a \nrightarrow b$. As a result, if we branch on a node in the tree, it results in two other nodes, one where $a \to b$ in the solutions to all scenarios, and one where $a \nrightarrow b$. Each constraint added maintains or increases the cost of the node, as some scenarios need to be resolved with the added constraint. The sum of these costs is equivalent to the *lower bound* of the costs of the node. Constraints continue to be added until the sets over the first-stage requests are the same, equivalent to no more a and b available to branch on. When this occurs, we have found a valid first-stage plan. This valid plan provides us with the first *upper bound* in costs. If we explore a node for which the computed costs are higher than this upper bound, we know that this node is not worth exploring further (additional constraints only make subsequent branching nodes more expensive). We continue the algorithm until all nodes have been explored and the entire solution space has been investigated. The amount of branches is finite, as there is a limited amount of permutations possible with the constraints $a \to b$ and $a \nrightarrow b$. In the worst-case scenario, however, the full set of all possible allocations of requests to routes is calculated. Branching on the initial node is presented in Fig. 2.

Over the scenarios, multiple first-stage requests exist in a node that could potentially be selected as the a and b to branch on. As such, we need to select the best a and b from the pool of potential requests. This solution method uses the difference in the proportion of scenarios for which $a \to b$ versus $a \nrightarrow b$ need to be re-calculated. The combination of the first-stage requests (i, j) where the largest proportion is selected for a and b. Thus, for one of the branching nodes, only a small set of scenarios need to be re-calculated. The node with the smallest number of a solution changes over the scenarios is then selected to branch on next. The benefit is that one expects this node to have lower costs since less “change” is implemented in the node (the added constraint impacts fewer scenarios). The algorithm to select the appropriate a and b as branching nodes from first-stage request set $(i, \dots, n)$ is presented in Fig. 3.

As a more practical example, consider a node in which 80% of the plans has $a \to b$, and 20% has $a \nrightarrow b$. In this situation, the optimal intermediate plan likely contains $a \to b$ since that already holds for most scenarios. This mechanic allows us to find the optimal intermediate plan relatively quickly, while proving that this intermediate plan is optimal is more time-consuming.

4.2. Preserving ordering on routes

In the previous section, the notion of branching on a and b being on the same route (i.e., $a \to b$) versus a and b not being on the same route (i.e., $a \nrightarrow b$), was introduced. It enables the solution space to be explored through the enumeration of the possible partitions of the requests in V_1 and assignment to vehicle routes. Once this partitioning is complete, the sequencing in the partitions needs to be investigated.

Consider the following example, where an end-node in the tree contains constraints $a \to b$, $b \to c$, and $c \to d$. As a result, it is known that a, b, c, and d are on the same route in this node. However, the order in which they appear on that route is still undetermined. Ultimately, the routing plan over the first-stage requests in

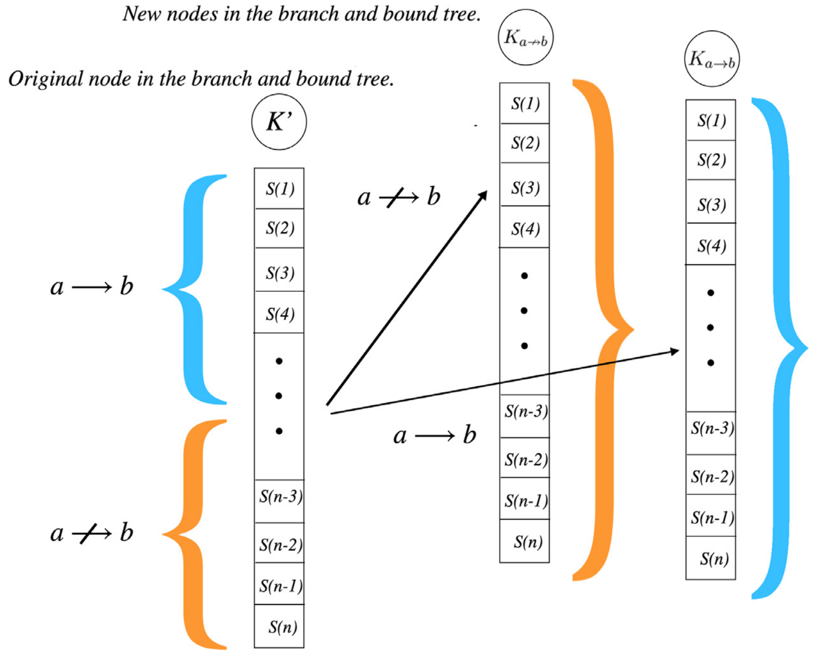

New nodes in the branch and bound tree.
Original node in the branch and bound tree.
K'
$K_{a\nrightarrow b}$
$K_{a\rightarrow b}$
$a \longrightarrow b$
$a \not\longrightarrow b$
S(1)
S(2)
S(3)
S(4)
S(n-3)
S(n-2)
S(n-1)
S(n)

Fig. 2. Example of branching on the initial node K'.

```
1:  $f^{\rightarrow}(i,j) = 0$ and $f^{\nrightarrow}(i,j) = 0$ for all $i$ and $j$.
2:  for all $i \in V_1$ do
3:      for all $j \in V_1$ do
4:          if $i \rightarrow j$ then
5:              $f^{\rightarrow}(i,j) = f^{\rightarrow}(i,j) + 1$
6:          end if
7:          if $i \nrightarrow j$ then
8:              $f^{\nrightarrow}(i,j) = f^{\nrightarrow}(i,j) + 1$
9:          end if
10:     end for
11: end for
12: $a, b = \text{argmax}_{i,j \in V_1} |f^{\rightarrow}(i,j) - f^{\nrightarrow}(i,j)|$   if $f^{\rightarrow}(i,j) \neq 0$ and $f^{\nrightarrow}(i,j) \neq 0$, otherwise empty.
```

Fig. 3. Algorithm for determining a and b.

V_1 needs to be identical for all scenarios, including the assignment to routes and the sequencing on those routes.

The number of sequences over a, b, c, and d is equivalent to the number of possibilities in a TSP over these requests, which is an NP-hard problem. In this section, we show that we can limit the number of sequences that need to be investigated. Initially, the sequences resulting from the preferred plans in the solution node are considered, which are intuitively a good starting point. In addition to the preferred plans, the important question to consider is the following: is there an alternative sequence that could be optimal for all scenarios? To help answer this question in the affirmative, we define a property of this sequence that, should it exist, can be verified with relative ease. It should be noted that when the second stage occurs, the demands from the requests in set V_2 become known and, at this point, no longer affects the sequence of visits established for the requests in set V_1. Here, demand does not affect the sequence since requests to routes have already been allocated. This allocation considers the capacity constraint, which interacts with demand. However, demand has no such interaction in sequencing the requests on the route once the allocation is complete.

Consider K^* a potential end-node in our branch-and-bound tree containing scenarios. For these scenarios, the partitioning of V_1 over routes has already been completed. While the partitions between the scenarios are now the same, the sequence over the first-stage requests on a route in each partition can differ. Collecting all different individual sequences provides us with a list of potential sequences for node K^*, one of which could be optimal.

Let's illustrate this with some examples. Consider a problem with a single stochastic request, referred to as q, and a set of requests V_1. Since q is the single stochastic request, two scenarios exist. When comparing scenarios, the set V_1 could be considered deterministic, as the set remains unchanged in the comparison. A single stochastic request results in each scenario having a preferred plan and thus a preferred sequence σ over V_1. These are referred to as σ^{-q} (optimal when q not present) and σ^{+q} (optimal when q present).

Two cases now exist: σ^{-q} is equivalent to σ^{+q}, or not. When both sequences are equivalent, there exists no other sequence that could be better (since the preferred sequence is the optimal sequence for both scenarios). As a result, this particular case requires no further investigation. When the sequences are unequal, a unique sequence can exist that minimizes the expected cost over the two scenarios. Two visual representations of unequal sequences are presented in Fig. 5. In Fig. 5(a) σ^{-E} is

```
1: UB (upper bound on cost) = $\infty$
2: $\Omega \leftarrow K'$, the initial node is the collection of initial intermediate plans.
3: while $\Omega \neq \emptyset$ do
4:     $k := \text{argmin}_{K\in\Omega}\ \Psi(K)$, select the node with the lowest cost.
5:     Select $a$ and $b$ using Algorithm from Figure 3.
6:     if ($a$ and $b$ do not exist) then                    ▷ We have found an end node.
7:         Calculate optimal sequence on $k$.
8:         Update UB.
9:         Update $Z^*$, the optimal solution.
10:    else                                                   ▷ Here we branch.
11:        Create $K_{a\to b,....}$ from $K_{,...}$, add constraint $a \to b$.
12:        Calculate $\Psi(K_{a\to b,....})$.
13:        if $\Psi(K_{a\to b,....}) < UB$ then
14:            $\Omega \leftarrow K_{a\to b,....}$
15:        end if
16:        Create $K_{a\nrightarrow b,....}$ from $K_{,...}$, add constraint $a \nrightarrow b$.
17:        Calculate $\Psi(K_{a\nrightarrow b,....})$.
18:        if $\Psi(K_{a\nrightarrow b,....}) < UB$ then
19:            $\Omega \leftarrow K_{a\nrightarrow b,....}$
20:        end if
21:    end if
22: end while
```

Fig. 4. Branch-and-bound algorithm.

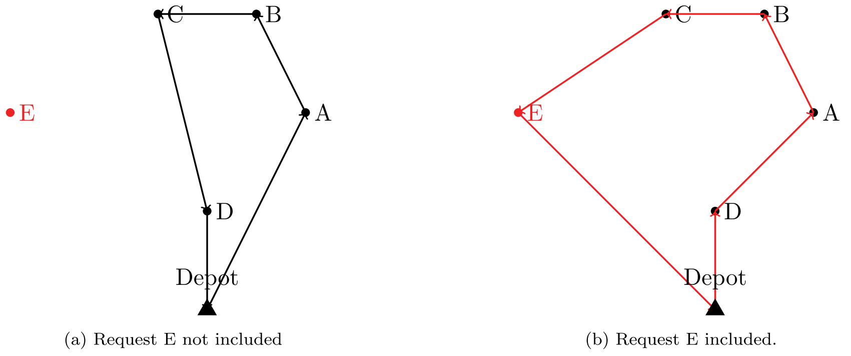


Fig. 5. Two examples of the preferred sequence changing as the result of the addition of a request.

equivalent to $A \to B \to C \to D$, but in Fig. 5(b) σ^{+E} is equivalent to $D \to A \to B \to C$.

In these examples, the solution to the optimal sequence is one of the sequences resulting from the preferred plans. However, outside of these sequences, there could be an additional sequence that is cheaper concerning the weighted average (with the probabilities π_s being the weights) in cost over the scenarios, but not the cheapest in cost for any scenario individually.

4.2.1. A property of the weighted-cheapest sequence.

Considering Fig. 5, aside from σ^{-q} (the optimal sequence when q is not present) and σ^{+q} (optimal sequence when q is present), there could exist sequence $\tilde{\sigma}$ which is optimal in costs with respect to the *weighted average* over the scenarios. An example can be found in the Appendix (Fig. 11). In order to find the optimal sequence, we have to consider σ^{-q}, σ^{+q}, and all potential sequences $\tilde{\sigma}$. Sequence $\tilde{\sigma}$ does not occur in our end-node in our branch and bound model.

We have to obtain optimal sequence σ^*. One way to approach that is referenced in Fig. 6, with i, j requests in route R, with q being a stochastic request, and TSP referencing the Traveling Salesman Problem. $sol(T_1)$ is the solution to the TSP related to T_1, which contains sequence $\sigma_{sol(T_1)}$. Note that by using this algorithm sequences σ^{-q} and σ^{+q} are considered.

For any requests i and j in the route R we answer the following questions: What is the cost of the cheapest route if stochastic request q was inserted between i and j? For this we a Traveling Salesman Problem has to be computed with an additional constraint $\{i \to q \to j\}$. What is the cost if we traveled that same route when q is not present? Is the weighted average over those two scenarios the cheapest? If the answer is yes, update the optimal sequence to the sequence contained within the current considered cheapest route. In the end, by testing all i and j, we have effectively tested all possible sequences for the route when considering stochastic request q.

Suppose 15 requests have been added to the route. To complete this algorithm we have to compute about 105 TSP's, which requires significant computation times. Note that the constraint that we add $\{i \to q \to j\}$ is related to the insertion of q between i and j which also carries its own cost. Let us define $\iota_v(\sigma^{-q})$ as the *cheapest insertion costs* of q on sequence σ^{-q}. We wish to limit the amount of TSP's that have to be computed. To this end, we determine bounds on the insertion costs. In other words, if the insertion costs for inserting q between i and j is outside of the computed bounds, do not compute the TSP as the sequence contained in the solution to that TSP can never be the optimal sequence.

Considering Fig. 5, aside from σ^{-q} and σ^{+q}, there could exist a sequence $\tilde{\sigma}$ which is optimal in costs with respect to the *weighted*

```
for all $i \in R$ do
    for all $j \in R$ do
        $T_1$ = Cost of TSP over $R \cup q$ with the constraint that $i \to q \to j$.
        $T_2$ = Cost over $R$ of the previous solution to the TSP. (without $q$)
        if $p_q * T_1 + (1 - p_q) * T_2 \leq T^*$ then
            $\sigma^* = \sigma_{sol(T_1)}$
        end if
    end for
end for
```

Fig. 6. Algorithm to determine the optimal sequence σ^* for q (brute force).

average over the scenarios. An example of such a sequence can be found in the Appendix (Fig. 11). We next state a property that holds for $\tilde{\sigma}$, if such a sequence does indeed exist. Let us consider a specific stochastic request q, this property defines valid bounds on the insertion cost of q in $\tilde{\sigma}$ when comparing the preferred sequences of two scenarios that only differ with respect to q (i.e., σ^{+q} versus σ^{-q}). In turn, these bounds enable an algorithm that finds the optimal sequence in an end-node K^* to be defined. Subsequently, it is shown that if this property holds for a pair of scenarios with a single stochastic request difference, then it holds for all scenario pairs. Let Υ define the set of overlapping requests between any scenario pair. We further define function $\gamma(\sigma, \Upsilon)$ as the cost of implementing a sequence σ on the set of requests Υ and function $\iota_v(\sigma)$ as the *cheapest insertion costs* of v on the sequence σ. The following result then holds:

Lemma 4.1.

$$\iota_q(\sigma^{+q}) \leq \iota_q(\tilde{\sigma}) \leq \iota_q(\sigma^{-q}) \tag{20}$$

Proof. Let us first consider the following inequality:

$$\gamma(\sigma^{-q}, \Upsilon) \leq \gamma(\sigma^{+q}, \Upsilon). \tag{21}$$

Inequality (21) simply states that, when implemented on a give set of requests Υ, the cost of sequence σ^{-q} is at most as high as the cost of sequence σ^{+q} (as directly implied by the definitions of both sequences). Furthermore, the following result also holds:

$$\gamma(\sigma^{-q}, \Upsilon \cup q) \geq \gamma(\sigma^{+q}, \Upsilon \cup q). \tag{22}$$

Inequality (22) simply states that, if one considers the set of requests $\Upsilon \cup q$, then the implementation cost of sequence σ^{-q} is necessarily higher than that of σ^{+q}, considering that request q is already part of the preferred sequence σ^{+q}. It should be noted that the implementation cost of sequence σ on the set $\Upsilon \cup q$ is equivalent to the cost of sequence σ on Υ plus the cheapest insertion costs of q. We thus obtain the following equations:

$$\iota_q(\sigma^{-q}) = \gamma(\sigma^{-q}, \Upsilon \cup q) - \gamma(\sigma^{-q}, \Upsilon), \tag{23}$$

$$\iota_v(\sigma^{+q}) = \gamma(\sigma^{+q}, \Upsilon \cup q) - \gamma(\sigma^{+q}, \Upsilon). \tag{24}$$

Considering inequalities (21) and (22), when combined with (23) and (24), results in the following inequality being observed:

$$\iota_q(\sigma^{-q}) \geq \iota_q(\sigma^{+q}), \tag{25}$$

which simply states that the insertion cost of request q is more expensive when considering the preferred sequence that did not account for it, i.e., σ^{-q}, when compared to the preferred sequence that did, i.e., σ^{+q}. Let request q be present with probability p_q. By the original definition of sequence $\tilde{\sigma}$, it is considered optimal in terms of cost when considering the probability p_q and for the given scenarios. We can now bound the value of $\iota_q(\tilde{\sigma})$. If a sequence $\tilde{\sigma}$ exists, then the following inequalities must hold:

$$\gamma(\sigma^{-q}, \Upsilon) \leq \gamma(\tilde{\sigma}, \Upsilon), \tag{26}$$

$$\gamma(\tilde{\sigma}, \Upsilon \cup q) \geq \gamma(\sigma^{+q}, \Upsilon \cup q). \tag{27}$$

Considering that q is a stochastic request (with probability p_q) and by the definition of $\tilde{\sigma}$ as an optimal sequence in terms of the weighted average, then we obtain the following:

$$\begin{aligned}(1 - p_q) \cdot \gamma(\tilde{\sigma}, \Upsilon) + p_q \cdot \gamma(\tilde{\sigma}, \Upsilon \cup q) \leq (1 - p_q) \cdot \gamma(\sigma^{-q}, \Upsilon) \\ + p_q \cdot \gamma(\sigma^{-q}, \Upsilon \cup q),\end{aligned} \tag{28}$$

$$\begin{aligned}(1 - p_q) \cdot \gamma(\tilde{\sigma}, \Upsilon) + p_q \cdot \gamma(\tilde{\sigma}, \Upsilon \cup q) \leq (1 - p_q) \cdot \gamma(\sigma^{+q}, \Upsilon) \\ + p_q \cdot \gamma(\sigma^{+q}, \Upsilon \cup q).\end{aligned} \tag{29}$$

Using the insertion cost function ι, inequalities (28) and (29) can be rewritten as:

$$\gamma(\tilde{\sigma}, \Upsilon) + p_q \cdot \iota_q(\tilde{\sigma}) \leq \gamma(\sigma^{-q}, \Upsilon) + p_q \cdot \iota_q(\sigma^{-q}) \tag{30}$$

$$\gamma(\tilde{\sigma}, \Upsilon \cup q) - (1 - p_q) \cdot \iota_q(\tilde{\sigma}) \geq \gamma(\sigma^{+q}, \Upsilon \cup q) - (1 - p_q) \cdot \iota_q(\sigma^{+q}). \tag{31}$$

Given (30) and (31) and applying the inequalities (26) and (27), we can now show that the cheapest insertion cost of q on $\tilde{\sigma}$ is bounded by:

$$\iota_q(\sigma^{+q}) \leq \iota_q(\tilde{\sigma}) \leq \iota_q(\sigma^{-q}),$$

which gives us the stated result. In words, the cheapest insertion cost of q on $\tilde{\sigma}$, $\iota_q(\tilde{\sigma})$, is cheaper (or equivalent) than $\iota_q(\sigma^{-q})$ but more expensive (or equivalent) when compared to $\iota_q(\sigma^{+q})$. □

For a given request q, let us consider a sequence $\tilde{\sigma}$ for which the insertion cost associated with q is smallest. Then, to insert q in $\tilde{\sigma}$, a specific edge (i, j) from sequence $\tilde{\sigma}$ needs to be removed and two edges (i, q) and (q, j) need to be added. This results in the following additional cost: $\iota_q(\tilde{\sigma}) = c(i, q) + c(q, j) - c(i, j)$. Considering the bounds associated with $\iota_q(\tilde{\sigma})$, which are defined by (20), there are a limited number of potential edges for which these bounds hold.

Algorithm for optimal sequence. In Section 4.2.1, we developed bounds on the insertion costs should a sequence $\tilde{\sigma}$ exist between *pairs* of scenarios. However, the full problem considers pairs instead of a full scenario set. In this section, we show it is sufficient to check all pairs of scenarios where one additional request differs. Consider the following small scenario tree, η:

Consider a set of deterministic requests V_1 and two stochastic requests a and b. For notation, $s(V_1 \cup a)$ indicates the scenario where both V_1 and a are present. Assume that $\tilde{\sigma}$ does not exist between scenarios $s(V_1)$ versus $s(V_1 \cup a)$. Similarly, $\tilde{\sigma}$ does not exist between $s(V_1 \cup a)$ and $s(V_1 \cup a \cup b)$. We can then claim that there is also no sequence $\tilde{\sigma}$ between $s(V_1)$ and $s(V_1 \cup a \cup b)$. Should sequence $\tilde{\sigma}$ exist, it would have been either found in the comparisons $s(V_1)$ versus $s(V_1 \cup a)$ or $s(V_1 \cup a)$ versus $s(V_1 \cup a \cup b)$.

As the result holds for the base case and the inductive step for any additional stochastic request, we also easily observe that

```
1: for all $s \in S$ do
2:     for all $r \in s$ do
3:         Add sequence $\sigma$ associated with route $r$ to $\zeta(r, \sigma)$.
4:         Check for additional sequence $\tilde{\sigma}$ in $\eta(s)$ by checking all relevant edges for which $\iota_q(\sigma^{+q}) \leq \iota_q(\tilde{\sigma}) \leq \iota_q(\sigma^{-q})$ holds.
5:     end for
6: end for
7: for all $\sigma \in \zeta(r, \sigma)$ do
8:     for all $r \in \zeta(r, \sigma)$ do
9:         Calculate cost of combination on all scenarios $s \in S$.
10:        Update if lowest cost found.
11:    end for
12: end for
13: Apply best $\sigma$ and $r$
```

Fig. 7. Algorithm to determine the optimal sequence.

the property holds for the entire scenario set by induction. This property allows us to formulate an algorithm to determine the optimal sequence. Let $\zeta(r, \sigma)$ be the list that contains all potential sequences σ for each route r. Let $\eta(s)$ be the list of scenarios associated with s with one additional request.

By performing branch-and-bound to find optimal sets for routes and applying Fig. 7 we find the optimal solution for the *intermediate plan*.

5. Computational results and analyses

In this section, our experimental analysis is presented. Computational experiments are performed on an AMD Ryzen 5 3600 with 16 GB of DDR4 RAM. Gurobi 8.0.1 is used to compute the solutions to the sub-problems (of the classical vehicle routing problem class).

The numerical analyses aim to assess the impacts of the stochasticity of demand and the presence of requests on the results (and solutions) obtained. A series of insights are derived from the numerical results obtained. The rest of the section is subdivided as follows:

- We introduce the benchmark approaches.
- We introduce the instances that are used.
- We introduce the results that are gained by running the approaches on the insights.
- Finally, valuable insights are presented.

5.1. Benchmark approaches

As discussed in Section 3, two categories of requests are served under different service quality agreements. The service provided to each request category is thus performed in two distinct stages (i.e., involving different periods). The proposed two-stage optimization model, presented in Sections 3.1 and 3.2, explicitly integrates the planning decisions made within each stage. To properly assess the value of solving such an integrated model, we implement two additional benchmark approaches, i.e., where the planning for both sets of requests is done separately.

In all approaches, all customer requests must be serviced (i.e., requests from V_1 and V_2. The first benchmark approach completely separates the planning of the routes for both request sets. Thus, there are dedicated routes for the V_1 requests and dedicated routes for the V_2 requests. As a result, these requests are planned in distinct periods via two independent capacitated vehicle routing problems, minimizing transportation costs. We call this method the *Seperate* solution method. In the second benchmark approach, the so-called *Using Spare Capacity* method, a capacitated vehicle routing problem is solved for the ν_1 requests, minimizing the transportation cost for visiting all V_1 requests. Then, the spare capacity potentially present in the routes for the V_1 requests might be used to accommodate, as much as possible, the requests observed in the second stage when the requests (and demands) of ν_2 are known. Therefore, the second stage involves determining which requests are added to the first-stage routes and which are planned on additional vehicle routes. The available capacity slack present in the first-stage routes is leveraged by proceeding this way to perform the overall service requests. For both additional solution approaches, the second-stage decisions are not explicitly considered when performing the first-stage planning. Using them as benchmarks for the proposed optimization method, we can assess the added value of integrating the two decision stages involved in the problem.

5.2. Instances

For our experiments, we use adapted versions of the Solomon instances. Of each instance, the first 90 requests are considered and subdivided into batches of 15. Considering that each Solomon instance contains 100 requests, we can obtain six distinct instances. The Solomon instances include coordinates, demand, and the ready, due date, and service times. For the problem investigated in this paper, the ready, due date, and service times are not considered and can be excluded. As a result, the instances are selected based on their uniqueness of coordinates and demand. As a result, four full Solomon instances remain, which are *R*, *C1*, *C2*, and *RC*.

In problem set R, the spacial data is randomly generated. Problem sets C1 and C2 are clustered, and a mix of random and clustered structures are used in the problem set RC. From these instances, we generate four classes. Each class is based on the stochasticity of the demand and the presence of requests. Each instance's last requests are considered stochastic (no random selection). Specifically, if there are either four requests in set V_2, then it is classified as "Low" (L), or six requests, classified as "High" (H). Furthermore, the amount of demand levels per request in V_2 is either 2, classified as "Low" (L), or four, classified as "High" (H). The probability of a request being present in a scenario was always considered to be $\frac{1}{2}$ for each stochastic request. The last requests of each instance were always chosen to be stochastic.

The probability for each demand level is equivalent. Let us clarify this with an example: consider a request of the set V_2 with demand 3.00 in the Solomon instance. In case we add two demand levels, the levels for this request will be the following: demand

Table 2
Computational results on the Solomon instances.

		A		B		C
Inst:	Class:	Result	Add. C.	Result	Add. C.	Result
R	LL	583.16	18.20%	525.48	6.51%	493.37
	LH	726.19	22.66%	636.86	7.57%	592.04
	HL	632.52	42.45%	517.43	16.53%	444.03
	HH	791.30	52.75%	619.98	19.68%	518.03
C1	LL	207.70	19.31%	179.17	2.92%	174.08
	LH	257.87	23.44%	223.48	6.98%	208.90
	HL	203.69	30.01%	172.84	10.32%	156.67
	HH	252.54	38.16%	211.08	15.48%	182.79
C2	LL	234.95	17.67%	209.39	4.87%	199.67
	LH	306.96	28.11%	259.56	8.33%	239.61
	HL	240.34	33.74%	205.46	14.33%	179.70
	HH	290.02	38.33%	242.42	15.63%	209.65
RC	LL	369.07	17.41%	331.19	5.36%	314.34
	LH	469.81	24.55%	416.48	10.41%	377.21
	HL	356.74	26.10%	327.83	15.88%	282.91
	HH	466.83	41.44%	390.82	18.41%	330.06

Avg: 29.65% Avg: 11.20%.

Table 3
Computation times for the different methods.

Legend identifier	Solution type	Computation time in hours
A	Separate	00:02:26
B	Using Spare Capacity	00:04:58
C	Integrated	10:24:13

3.00 with probability $\frac{1}{2}$ and demand 6.00 with probability $\frac{1}{2}$. (For three demand levels, this would be: demand 3.00 (prob: $\frac{1}{3}$, demand 6.00 (prob: $\frac{1}{3}$), demand 9.00 (prob: $\frac{1}{3}$)).

As a result, we obtain classes “LL”, “LH”, “HL” and “HH”, with the first letter signifying the number of requests present and the second letter signifying the number of demand levels present. All values reported are based on the average results obtained over the six instances for each of the three methods we utilize to solve the problem.

5.3. Results

In Table 2, the results, which are the objective function values, obtained when the three solution approaches are applied to solve the different instances are reported. As indicated in Table 3, A, B, and C refer to the three solution approaches that are implemented: the separate planning approach, the separate planning approach with the use of spare capacity in the first-stage routes, and our proposed exact method for the integrated problem, respectively. In addition, Table 3 also reports the average computation times for the different solution approaches when applied to all considered instances. The columns in Table 2 refer to the instance type (i.e., Inst) and the problem class (i.e., Class). In contrast, the column Result represents the average total cost associated with the obtained solutions, and Add. C. is the average relative cost difference between the obtained solutions and the optimal ones of the integrated approach C. The average relative cost difference over all instances is also reported at the bottom of the table (i.e., Avg).

One first observation is the clear advantage of implementing an integrated planning approach when establishing the vehicle routes to service the requests from the two request types in the present case. On average, the separated planning approach produces solutions whose costs are approximate 30% higher when compared to the costs of the optimal solutions of the integrated approach (i.e., obtained by solving the proposed two-stage model). If the spare capacity of the routes for the v_1 requests can be used to accommodate the requests of v_2, then the average relative cost difference is improved. Nonetheless, it remains quite high, i.e., approximately 11%. As expected, these differences are more pronounced when solving instances where the levels of uncertainty are higher (i.e., the highest values for the average relative cost difference are observed for the instances in the problem class HH). Furthermore, the number of stochastic requests seems to have a more significant impact on the obtained results than the random variability of the demands. For example, when considering instance type R and the results obtained using the solution approach A, the value of Add. C. goes from 18.20% to 42.45% for the instances in the problem classes LL and HL, respectively. Using the same example, the Add. C. value difference is even more noticeable when considering the results obtained for the instances in the problem classes LH and HH (22.66% and 52.75%, respectively). The integrated approach significantly outperforms the other two solution approaches in all solved instances.

Table 4
Computation times (hours) for Gurobi TL.

Instance	R			
Class	LL	LH	HL	HH
Comp. Time	1:05	1:20	1:47	2:24
Instance	C1			
Class	LL	LH	HL	HH
Comp. Time	1:07	1:23	1:51	2:41
Instance	C2			
Class	LL	LH	HL	HH
Comp. Time	1:02	1:21	1:46	2:32
Instance	RC			
Class	LL	LH	HL	HH
Comp. Time	1:01	1:27	1:59	2:51

Average: 1:47 .

5.4. Effects of limiting the computation times

As seen from Table 3, directly solving the considered integrated stochastic model requires large computation times. In this section, we thus investigate the effects of limiting the computation times dedicated to different calculations performed by the proposed solution method. Specifically, three limits on the calculations are considered:

1. Limiting the overall computation time to one hour. *(Overall TL)*
2. Limiting the computation time of Gurobi within a node in the search tree to 15 seconds. *(Gurobi TL).*
3. Using both the 15-second limit within a node in the search tree as well as using the overall one hour time-limit. *(Both TL)*

The overall time limit of one hour for variants “Overall TL” and “Both TL” was reached at all times. For variant “Gurobi TL”, we report the computation times in Table 4. In Table 5, we present the optimality gap compared to the optimal solution found if no computation time limits are set. More precise, the optimality gap is computed as:

$$Gap = \frac{Best\ solution\ found - Optimal\ Solution}{Optimal\ Solution} \tag{32}$$

Here we refer to the Best Solution as the cost of the best valid solution found thus far while exploring the branch and bound tree, while the Optimal Solution refers to the cheapest cost found during the full computation.

From Table 5, we observe that a time limit on the Gurobi calculation is not the main contributor to the gap relative to the actual solution. Gurobi generally finds a close-to-optimal solution early, and by limiting its calculation time, time is saved by skipping the proof to optimality. Within generally two hours, a good solution was reached, within 1% of the optimum compared to the exact method. Our algorithm usually finds the optimal solution when the first feasible solutions are obtained (we arrive at an end

Table 5
Reported optimality gaps for the different limits on computation time.

		Overall TL	Gurobi TL	Both TL
Inst:	Class:	Gap	Gap	Gap
R	LL	3.40%	0.71%	0.78%
	LH	3.66%	0.81%	0.87%
	HL	4.45%	0.83%	0.91 %
	HH	5.75%	0.98 %	1.12 %
C1	LL	3.31%	0.42 %	0.47 %
	LH	3.44%	0.68%	0.69%
	HL	4.01%	0.75 %	0.76 %
	HH	4.16%	0.88%	0.89%
C2	LL	3.0%	0.54%	0.54%
	LH	4.64%	0.70%	0.73%
	HL	4.11%	0.73%	0.74%
	HH	5.34%	0.84%	0.85%
RC	LL	3.41%	0.66%	0.78%
	LH	4.15%	0.73%	0.85%
	HL	5.10%	0.85%	0.91%
	HH	8.44%	1.31%	1.43%

Avg: 4.85% Avg: 0.73% Avg: 0.81% .

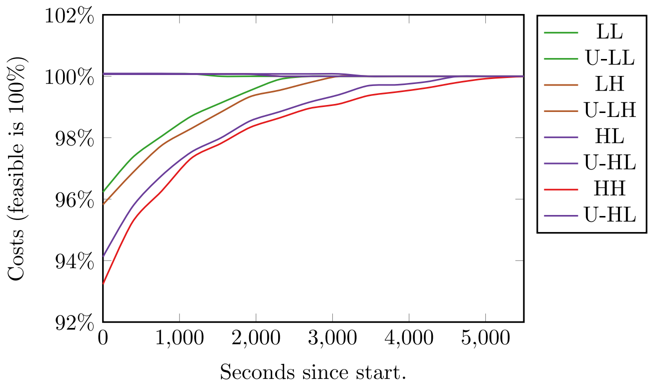


Fig. 8. Average progress of the algorithm in time.

node). This results from the strategy employed, which is discussed in Section 4.1. Here we branch on the combination of requests that minimizes the number of scenarios that need to be recalculated to transition to a new node. This strategy reaches a solution node in the minimum amount of time. However, one needs to compute all end nodes for proof of optimality.

The optimum was found on average in 1h47m, while the average total computation time equals 10h24m. Our algorithm starts from a “preferred” solution, which is optimum for each scenario and converges to a “feasible” solution for our problem. As we explore requests through the search tree, the cost of nodes (the sum of costs for all intermediate plans for all scenarios) increases (the sum of the probabilities of scenarios multiplied by the cost of the associated scenario). This results from the cost of individual scenarios increasing as constraints are added. The increase in cost through time in the algorithm is displayed in Fig. 8. Here, the results of the classes { LL, ..., HH }, which are equivalent to the lower bounds on the cost of the problem, and the upper bounds, { U-LL, ..., U-HH }, are visualized. We also display the upper bound for each category, calculated by applying the most common preferred plan to all scenarios (if all are unique, the preferred plan of scenario where all second-stage requests are present is chosen.).

Often the solution to the problem was the most common “preferred plan” in the original node.

In such a case, the upper bound costs are thus equivalent to the final costs, which explains near horizontal lines for the upper bound.

5.5. Larger instances

Having more requests will lead to larger computation times for solving the Capacitated VRP but also to more scenarios that have to be analyzed. If one stochastic request is added to the HH problem referenced in Section 5.3, the amount of scenarios increases by a factor of 7. To analyze the limits of our method, We tested larger instances. We have generated the following instances of 20 and 25 requests. First, we have subdivided the Solomon instances into five instances of 20 requests. Of these, the last 6 requests are stochastic, being present with probability $\frac{1}{2}$. There are only two demand levels, {3,6}, each with probability $\frac{1}{2}$. Table 6 shows for each instance category the average gap (over the 5 instances) between the lower bound and upper bound after 24 hours of computation. Secondly, we have subdivided each Solomon instance into four instances of 25 requests. Of these, the last 8 requests are stochastic, and present with probability $\frac{1}{2}$. There are again only two demand levels, {3,6}, each with probability $\frac{1}{2}$. Table 7 shows for each instance category the average gap (over the 4 instances) between the optimal solution (found in segment 5.3) and upper bound after 24 hours of computation.

$$Gap = \frac{Upperbound - Lowerbound}{Lowerbound} \tag{33}$$

Here the Upper bound is the best valid solution found thus far while exploring the branch and bound tree. The lower bound is the lowest value for all unexplored nodes. Note that these might be nodes where not all constraints have been added yet (yielding cheaper cost).

From the results, it is implied that both the increase in the number of scenarios, as well as the increase in the problem size of the Capacitated VRP has a significant impact on the computation time of this exact methodology. We were able to solve each the instances of 15 customers within on average 10 hours, computation time rapidly increases by the problem size. None of the instances with 20 and 25 customers was able to be solved within 24 hours, for instances with 20 customers the gap after 24 hours is relatively small, but this gap seems to grow exponentially if the problem size grows. While a potential solution is found within the 24-hour time period, the process of proving that this is the optimum is not nearly complete, as seen from the reported gaps. For larger problems, we propose heuristics are to be used.

5.6. Insights into the balancing act of buffers versus risk

With regard to the stochastic nature of the considered problem, there are some interesting points to highlight. Specifically, finding the optimal first-stage plan over all the scenarios involves a balancing act of introducing buffers in the first-stage routes, and as a result, incurring additional costs, to accommodate the second-stage requests versus the risk of them not appearing and thus having paid these additional costs unnecessarily. In that sense, the scenarios are similarly interesting. The preferred first-stage plan associated with the scenario that includes the maximum number of requests and maximum demands is also the plan with the most buffer. On the other hand, the preferred first-stage plan associated with the scenario that contains only the first-stage requests with their fixed demands is the plan with the least buffer. Moreover, it should be noted that for all instances that we computed the optimal solution was included in the collection of initial preferred plans in the original node in the branch-and-bound tree. In all generality, one can expect that the buffers included in the first-stage routes will depend on a subset of second-stage requests being present while for the other requests the risk of them appearing will simply be accepted.

Table 6
Optimality gap after 24 hours of computation time for instances of 20 customers.

Instance:	Average gap:	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
R	**6.1%**	5.2%	7.6%	6.3%	5.6%	5.7%
C1	**4.6%**	4.7%	4.6%	4.1%	4.4%	4.8%
C2	**4.8%**	4.3%	4.9%	4.7%	4.9%	5.0%
RC	**5.2%**	4.7%	5.1%	4.9%	5.7%	5.6%

Table 7
Optimality after 24 hours of computation time for instances of 25 customers.

Instance:	Average gap:	Instance 1	Instance 2	Instance 3	Instance 4
R	**156.1%**	155,3%	157,1%	156,3%	155,6%
C1	**140.7%**	140,9%	141,6%	140,0%	140,3%
C2	**141.3%**	142,2%	140,6%	141,5%	140,9%
RC	**147.8%**	147,1%	147,4%	148,0%	148,8%

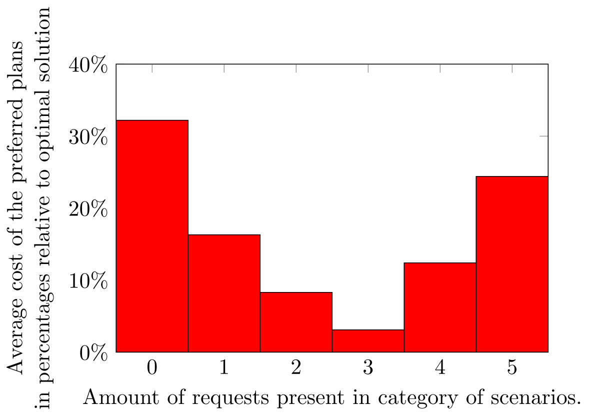


Fig. 9. Average expected costs of preferred plans for the categories of scenarios.

The balancing act of buffers versus risk also holds some interesting analysis. Let us categorize scenarios on the basis of the sum of their requests present. If we have five second-stage requests, there will as a result be six categories (none present to all present). Each scenario is associated with a preferred plan, and each of the preferred plans is a potential solution, for which the total expected cost can be calculated. We can then compute the expected cost of each category (for all scenarios in a category we multiply the cost of that scenario by the probability of that scenario occurring). Fig. 9 shows these average expected costs, defined as relative to the optimum in percentages, over the different categories of preferred plans, which has a convex shape. This trend was observed for all tested instances. Furthermore, the trend similarly held true when the scenarios were categorized based on the total demand level.

5.7. Assessing the impacts of the stochastic parameters

For the results in Section 2, a fixed subset of requests is considered stochastic. In this section, we perform a sensitivity analysis of the demands and requests. Once again the Solomon instances were adapted, similar to those in Section 5.2. For each instance of 15 requests, we select a random set of requests indicated by the class, which we give a probability of being present (in this case, 0.5). The average normalized increase from the base cost is plotted. Here, the base cost is the cost of the scenario with all requests present. We can motivate this as we are assessing the impact of considering an increasing number of stochastic requests and their associated demands. Our point of comparison is obtained by solving the problem based on only considering the worst-case scenario (i.e., all the stochastic requests being present with their associated maximum demand in the second stage). Of course, when there are no stochastic requests present, then this point of comparison provides the optimal solution to the problem and there are no differences between the three solution methods that are proposed (i.e., A, B, and C). Otherwise, one can assess the value of using the proposed solution methods compared to the more conservative approach, where the problem is solved using the worst-case scenario. The increase in the size of this selection is plotted in Fig. 10(a). Note that the cost is decreasing for the optimal solution.

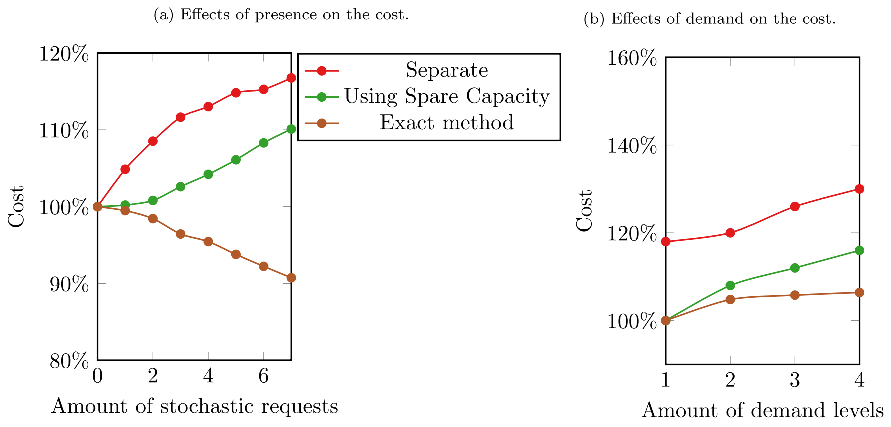


Fig. 10. The effects of dynamism.

In Graph 10(b) a selection of 3 requests was transformed to stochastic and presented with demand levels.

Looking at these graphs, we can observe for the solution method "Separate" that there is an initial greater increase of additional costs because of the increase in uncertainty of the problem. As the amount of uncertainty increases, this decreases, though costs remain monotonically increasing. Using spare capacity has the opposite effect. The spare capacity can initially absorb a few stochastic requests, but as the amount of uncertainty in demand and requests increases, so does the additional cost. Finally, one can observe that the *presence* of requests, which is reflected in Fig. 10(a), has a stronger impact on the expected cost compared to the demand of the request, reflected in 10(b). In the scientific literature, the marginal costs of additional requests are generally higher than the marginal costs of additional demand. This is, for example, reflected in Gendreau et al. (1992) and in Gendreau et al. (2014).

6. Conclusions

This paper introduced a novel stochastic vehicle routing problem in which customers have different service agreements. These service contracts differ when customers make their requests known, creating uncertainty in the problem. These uncertainties are in both the present requests and their demand. For this problem, we introduce a new exact branch-and-bound algorithm. While traditional branch-and-bound algorithms tend to branch on edges within the vehicle routing problem, as observed in Vigo (2015), we introduce a branch-and-bound algorithm that branches on the full set of scenarios that are introduced by the uncertainties generated by the different service contracts.

Our solution method shows effective results compared to planning the requests with different services completely separate or using spare capacity. Planning the requests completely separately incurs an estimated 30% additional cost, while using the spare capacity of an earlier formulated plan leads to an additional 11% cost.

The computation times of the full problem are significant. However, by imposing a time limit on the node computation in the branch-and-bound tree and/or a time limit over the full solution time, we reach solution quality within 1% of the optimal solution. Using the novel branch-and-bound algorithm also yielded some interesting results regarding the balancing act of buffers versus risk. Buffers included in the first-stage routes depend on a subset of second-stage requests being present, while for the other requests, the risk, and as a resulting penalty, of them appearing will be accepted. Categorizing the scenarios into different categories and subsequently computing the expected costs of these categories (the cost of each scenario in the category multiplied by the probability of the scenario occurring) yielded some interesting convex relationships.

These convex relationships provide some interesting insights into developing a heuristic to the problem, which is a potential new research direction. In addition, further investigation of how different service agreements with different characteristics (e.g., specific constraints regarding how periodic visits are performed to requests or the introduction of pickup and delivery) may impact the planning of distribution operations is an interesting avenue for further research. When working with real-life cases, it was noted that the bounds towards the optimum are still significant. Future work is needed on scenario-reduction techniques for this problem. Finally, implementing self-imposed time windows is also an interesting future research avenue.

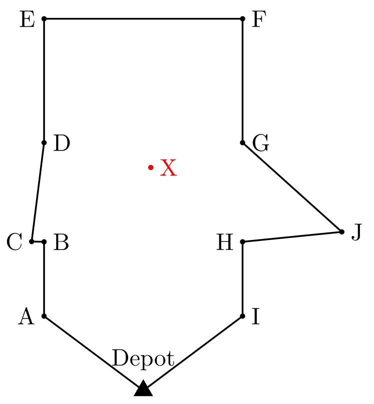
Acknowledgments

The funding for this research was provided by NWO, the Dutch Science Foundation, for the project DATAS for Cities (No. 438-15-507).

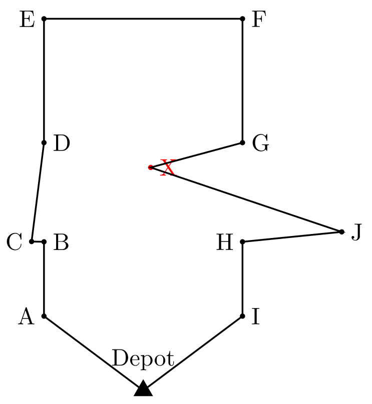
Appendix A

In Fig. 11 we show an example with requests A–I and X. In this example X is the only stochastic request. In Fig. 11(a) we observe the optimal route for the scenario if X is not present and in Fig. 11(d) the optimal route for the scenario if X is present. Fig. 11(b) and (d) show respectively, the adjusted versions of these routes if the other scenario then the one it was optimized for realizes. Finally, in Fig. 11(e) and (f) the route is presented which is the solution being optimum over the weighted scenarios of X being present and X not being present. As can be seen, the weighted cost (with weights 0.5 for each scenario) of routes 11(a) and (b) (equal to 51.21) as well as of routes 11(d) and (c) (equal to 51.09) are more expensive than the weighted cost of routes 11(e) and (f), which equal 50.78. As such, there is a route sequence different than the optimal route sequences of each individual scenario, which has a lower weighted cost.

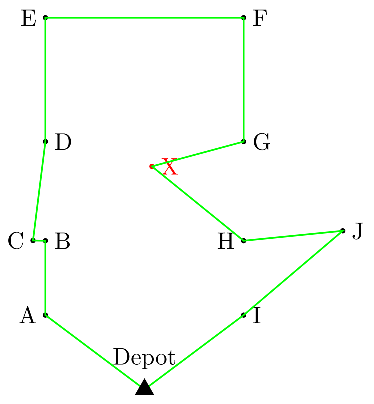
(a) Optimal route determined for the scenario with X not being present (thus the optimal route over request A-I). Total Length: 47.92



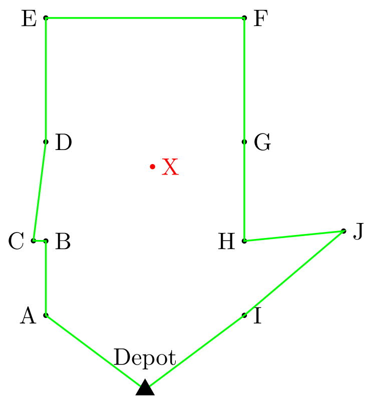
(b) Cheapest insertion of X onto the route of (a), thus representing the route in (a) adjusted when the scenario of X being present realizes. Length: 54.50



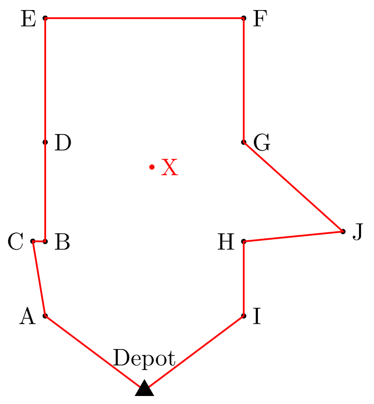
(c) Optimal route for the scenario of X being present (thus the optimal route over request A-I,X). Length: 53.39



(d) The route in (c) adjusted if the scenario of X not being present realizes. Length: 48.79



(e) If the optimal route is determined over the weighted scenarios, this is the route if X is not present. Length: 47.95



32

(f) If the optimal route is determined over the weighted scenarios, this is the route if X is present. Length: 53.61

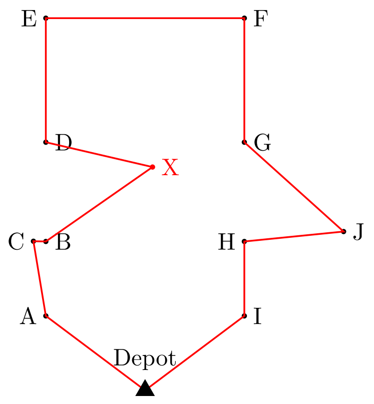


Fig. 11. An example of the existence of sequence $\tilde{\sigma}$.

References

Balaprakash, P., Birattari, M., Stützle, T., & Dorigo, M. (2015). Estimation-based metaheuristics for the single vehicle routing problem with stochastic demands and customers. *Computational Optimization and Applications, 61*(2), 463–487.

Beraldi, P., Ghiani, G., Musmanno, R., & Vocaturo, F. (2010). Efficient neighborhood search for the probabilistic multi-vehicle pickup and delivery problem. *Asia-Pacific Journal of Operational Research, 27*(03), 301–314.

Bertazzi, L., & Secomandi, N. (2018a). Faster rollout search for the vehicle routing problem with stochastic demands and restocking. *European Journal of Operational Research, 270*(2), 487–497.

Bertazzi, L., & Secomandi, N. (2018b). Worst-case benefit of restocking for the vehicle routing problem with stochastic demands. Available at SSRN 3246339.

Bertsimas, D. (1988). *Probabilistic combinatorial optimization problems*. Massachusetts Institute of Technology Ph.D. thesis..

Bertsimas, D., & Howell, L. H. (1993). Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research, 65*(1), 68–95.

Bertsimas, D. J. (1992). A vehicle routing problem with stochastic demand. *Operations Research, 40*(3), 574–585.

Biesinger, B., Hu, B., & Raidl, G. R. (2018). A genetic algorithm in combination with a solution archive for solving the generalized vehicle routing problem with stochastic demands. *Transportation Science, 52*(3), 673–690.

Dror, M., Laporte, G., & Trudeau, P. (1989). Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science, 23*(3), 166–176.

Erera, A. L., Savelsbergh, M., & Uyar, E. (2009). Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks: An International Journal, 54*(4), 270–283.

Florio, A. M., Hartl, R. F., & Minner, S. (2020). Optimal a priori tour and restocking policy for the single-vehicle routing problem with stochastic demands. *European Journal of Operational Research, 285*, 172–182.

Gendreau, M., Jabali, O., & Rei, W. (2014). Chapter 8: Stochastic vehicle routing problems. In *Vehicle routing: Problems, methods, and applications* (pp. 213–239). SIAM.

Gendreau, M., Laporte, G., & Seguin, R. (1992). *The vehicle routing problem with stochastic customers and demands*: CRT. Centre de Recherche sur les Transports Publication.

Gendreau, M., Laporte, G., & Séguin, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science, 29*(2), 143–155.

Gendreau, M., Laporte, G., & Séguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research, 44*(3), 469–477.

Goel, R., Maini, R., & Bansal, S. (2019). Vehicle routing problem with time windows having stochastic customers demands and stochastic service times: Modelling and solution. *Journal of Computational Science, 34*, 1–10.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing and Service Operations Management, 11*(4), 630–643.

Jabali, O., Leus, R., Van Woensel, T., & De Kok, T. (2015). Self-imposed time windows in vehicle routing problems. *Or Spectrum, 37*(2), 331–352.

Jaillet, P. (1985). *Probabilistic traveling salesman problems*. Massachusetts Institute of Technology Ph.D. thesis..

Jaillet, P., & Odoni, A. (1988). The probabilistic vehicle routing problem. In *Vehicle routing: Methods and studies. North Holland, Amsterdam* (pp. 293–318).

Jezequel, A. (1985). *Probabilistic vehicle routing problems*. Massachusetts Institute of Technology Ph.D. thesis..

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2015). The generalized consistent vehicle routing problem. *Transportation Science, 49*(4), 796–816.

Laporte, G., Louveaux, F. V., & Mercure, H. (1994). A priori optimization of the probabilistic traveling salesman problem. *Operations Research, 42*(3), 543–549.

Laporte, G., Louveaux, F. V., & Van Hamme, L. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research, 50*(3), 415–423.

Neves-Moreira, F., Da Silva, D. P., Guimarães, L., Amorim, P., & Almada-Lobo, B. (2018). The time window assignment vehicle routing problem with product dependent deliveries. *Transportation Research Part E: Logistics and Transportation Review, 116*, 163–183.

Oyola, J., Arntzen, H., & 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.

Salavati-Khoshghalb, M., Gendreau, M., Jabali, O., & Rei, W. (2019a). An exact algorithm to solve the vehicle routing problem with stochastic demands under an optimal restocking policy. *European Journal of Operational Research, 273*(1), 175–189.

Salavati-Khoshghalb, M., Gendreau, M., Jabali, O., & Rei, W. (2019b). A rule-based recourse for the vehicle routing problem with stochastic demands. *Transportation Science, 53*(5), 1334–1353.

Secomandi, N. (2001). A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research, 49*(5), 796–802.

Secomandi, N., & Margot, F. (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research, 57*(1), 214–230.

Sörensen, K., & Sevaux, M. (2009). A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling. *Journal of Mathematical Modelling and Algorithms, 8*(4), 387.

Spliet, R., & Gabor, A. F. (2015). The time window assignment vehicle routing problem. *Transportation Science, 49*(4), 721–731.

Tillman, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science, 3*(3), 192–204.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*: vol. 1. SIAM.

Vidal, T., Laporte, G., & Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research, 286*(2), 401–416.

Vigo, D. (2015). *Vehicle routing: Problems, methods and applications*. Society for Industrial and Applied Mathematics.

Waters, C. D. J. (1989). Vehicle-scheduling problems with uncertainty and omitted customers. *Journal of the Operational Research Society, 40*(12), 1099–1108.