A Branch-and-Price Algorithm for the Vehicle Routing Problem with Stochastic Demands and Probabilistic Duration Constraints

Alexandre M. Florio*[1], Richard F. Hartl†[1], Stefan Minner‡[2] and Juan-José Salazar-González§[3]

[1]University of Vienna, Austria
[2]Technical University of Munich, Germany
[3]Universidad de La Laguna, Spain

April 1, 2020

Abstract

In many routing applications, it is necessary to place limits on the duration of the individual routes. When demands are stochastic and restocking during route execution is allowed, the durations of the resulting routes are also stochastic. In this paper, we consider the vehicle routing problem with stochastic demands and (probabilistic) duration constraints (VRPSD-DC). We assume optimal restocking, which means that, during the route execution, replenishment trips to the depot are performed in an optimal way. The resulting optimization problem calls for a set of routes with minimal total expected cost for visiting all customers, such that the duration of each route, with a given probability, does not exceed a prescribed limit. We solve the VRPSD-DC with a novel branch-and-price algorithm. An orienteering-based completion bound is proposed to control the growth of labels in the pricing algorithm. Feasibility of a priori routes is verified by applying Chebyshev's bounds, by Monte Carlo simulation and statistical inference, or by analytically deriving the distribution of the route duration. Consistency checks are incorporated into the branch-and-price framework to detect statistical errors. Computational experiments are performed with demands following binomial, Poisson, or negative binomial probability distributions, and with duration constraints enforced at the levels of 90%, 95% and 98%. Optimal solutions to the VRPSD-DC may contain routes that serve an expected demand that is larger than the capacity of the vehicle. These solutions actively employ optimal restocking to reduce traveling costs and the number of required vehicles. Sensitivity analyses indicate that high demand variability negatively impacts the solution, both in terms of total expected cost and the number of routes employed.

1 Introduction

We consider a variant of the vehicle routing problem (VRP) where the demands of the customers are stochastic. At route planning time, the demands are only known in probability distribution. During route execution, the demands are not disclosed before the vehicle arrives at the customers' locations. This problem is known as the VRP with stochastic demands (VRPSD), and it is the most studied stochastic variant of the VRP (Gendreau, Jabali, and Rei 2016). Under the a priori paradigm, the VRPSD calls for a set of a priori routes of minimum total expected cost, such that every customer is visited by exactly one a priori

*alexandre.de.macedo.florio@univie.ac.at
†richard.hartl@univie.ac.at
‡stefan.minner@tum.de
§jjsalaza@ull.es

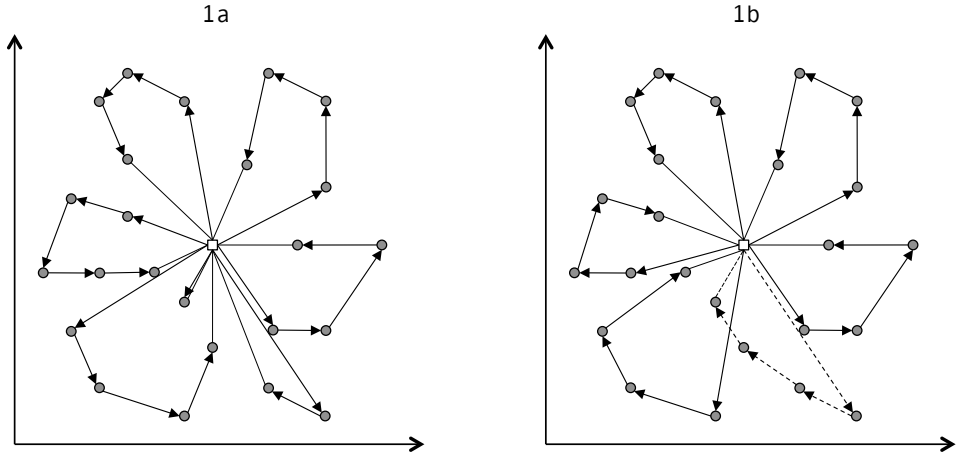

Figure 1: Optimal solutions of instance R101.25 *(Solomon 1987) with capacity constraint (solution 1a) and with probabilistic duration constraint (solution 1b). The route indicated with the dashed arrows serves a total expected demand of* 77*, above the capacity of the vehicle of* 60*.*

route. Since the total demand along a route may exceed the capacity of the vehicle, the vehicles are allowed to perform replenishment trips to the depot in most of the VRPSD settings. Consequently, the expected cost of a route consists of a deterministic part (the a priori route cost) and a stochastic part (the expected restocking cost). In the event of a restocking, the remaining unserved customers must still be visited according to the sequence prescribed by the a priori route.

Until recently, all exact algorithms for the VRPSD assumed that the vehicle could only replenish when arriving at a customer with insufficient load for serving that customer's demand, an event known as *failure* (e.g. Gauvin, Desaulniers, and Gendreau 2014, Jabali et al. 2014). This reactive policy is known in the literature as the *detour-to-depot* or *classical* policy, and it was first described in Dror, Laporte, and Trudeau (1989). Considering that failures could occur at customers located far from the depot, it became common practice in these algorithms to enforce that the total expected demand in a route does not exceed the capacity of the vehicle. This additional capacity constraint would not only control the number of failures, but also make them more likely to only occur at the end of the routes, i.e. when customers located relatively close to the depot are served. Hence, capacity constraints were reasonable when solving the VRPSD under the detour-to-depot policy.

It is well-known that, given an a priori route, an optimal restocking policy can be computed by dynamic programming (Yee and Golden 1980). Depending on customer specific threshold levels, an optimal policy may prescribe preventive replenishment trips. Exact algorithms for the VRPSD under optimal restocking have been proposed in Louveaux and Salazar-González (2018), Salavati-Khoshghalb et al. (2019), Florio, Hartl, and Minner (2020). As in the earlier detour-to-depot methods, capacity constraints were also enforced in these approaches. In this work, we focus on probabilistic duration constraints. The individual routes are allowed to serve a total expected demand that exceeds the capacity of the vehicle. The capacity constraint is then managed only by the restocking policy. We now provide a simple example in order to further motivate the study of the VRPSD with duration constraints, and to illustrate why capacity constraints are not beneficial when solving this problem variant.

In Figure 1, two solutions for the literature instance R101.25 (the capacity of the vehicle is set to 60, and time windows are ignored) are introduced. The demands of the customers follow Poisson distributions, with parameters as given in the original instance. Solution 1a corresponds to the optimal solution of the VRPSD under optimal restocking, when the capacity constraint is enforced (i.e., the total expected demand along each a priori route cannot exceed 60). Solution 1b is an alternative solution for the same problem, but with a relaxed capacity constraint. It turns out that solution 1b is superior to solution 1a with regard to the total expected distance (505.3 vs 508.2), and the number of vehicles needed

(6 vs 7). We now add a *duration* dimension, and assume that the duration of a route is proportional to the distance traveled. In many applications there are limits on the number of working hours of the driver (e.g., due to contractual terms, legal regulations or a company's overtime policy), so the durations of the routes are important. Suppose that a transportation company requires that each route must finish before a maximum time limit with a probability of 95%. For solution 1a to comply with such a requirement, this limit cannot be smaller than 124. Any integer value less than 124 will render solution 1a infeasible. We can verify that all routes of solution 1b, with a probability of 95%, also finish within a maximum time frame of 124, which would satisfy the real-world constraint imposed by the company. We conclude that, from a practical standpoint and considering minimization of transportation costs (fixed vehicle costs and traveling costs), there is no reason to adopt solution 1a in place of solution 1b.

The previous example motivates solving the VRPSD by enforcing only the duration constraints. In this variant, routes that serve a total expected demand larger than the capacity of the vehicle are also allowed, provided that their durations are within the maximum established limit. The vehicle capacity constraint is then managed entirely by the optimal restocking policy. In Figure 1, solution 1b is the optimal solution of the VRPSD with a probabilistic constraint (or chance constraint) at level 95% ensuring that each route finishes before 124, and was obtained with the algorithm we describe in this paper.

The VRPSD with duration constraints (VRPSD-DC) was first studied in Erera, Morales, and Savelsbergh (2010). Before that, duration limits had only been included in models for the VRPSD as soft constraints (e.g., Tan, Cheong, and Goh 2007), or taken into account in an expected value sense (e.g., Yang, Mathur, and Ballou 2000). In Erera, Morales, and Savelsbergh (2010), duration limits were considered deterministic constraints, and a tabu search heuristic was developed to find solutions to the problem, where every route is duration feasible regardless of the customer demand realizations. In Mendoza, Rousseau, and Villegas (2016), duration limits were incorporated as probabilistic constraints into the VRPSD-DC, which was also solved by means of a heuristic method. In both cases the detour-to-depot restocking policy was assumed. To the best of our knowledge, despite its practical relevance, these are the only contributions to the VRPSD-DC under the a priori paradigm. Compared to these approaches, our branch-and-price algorithm **(i)** solves the VRPSD-DC with (probabilistic) duration limits exacly; and **(ii)** solves the problem under optimal restocking and without capacity-based constraints, thus allowing superior solutions to be found, as can, for instance, be seen in solution 1b from the previous example. Finally, under the reoptimization paradigm, where changes in the sequence of customers to be visited are permitted, the VRPSD with duration limits has been solved by approximate dynamic programming methods in Goodson, Ohlmann, and Thomas (2013) and in Goodson, Thomas, and Ohlmann (2016).

The development of the branch-and-price algorithm for the VRPSD-DC followed an entirely different path than what was proposed in Florio, Hartl, and Minner (2020) for the VRPSD with capacity constraints. Even though in both cases the pricing problem was solved with a backward labeling algorithm, most of the machinery previously proposed cannot be applied in the present case. Due to the absence of capacity constraints, rounded capacity cuts to strengthen the set-partitioning formulation and all other types of capacity-based bounds (e.g., the forward RCSP completion bound, employed to control the combinatorial growth of labels) are unavailable. The VRPSD-DC is considerably more difficult. For example, a simple feasibility check of an a priori route is a tricky endeavor, as it involves an understanding of the cumulative distribution of the actual route duration. For the same reason, proposing any type of label dominance is also intricate.

In summary, this paper makes the following contributions:

(1) An algorithm for the VRPSD with route duration constraints, a version of the problem likely to be found in many practical scenarios;

(2) A set of results that show the superiority of solutions obtained when optimal restocking is considered and capacity-based constraints are not enforced;

(3) A sensitivity analysis of important problem parameters such as the level at which the probabilistic constraint is enforced and the variability of the customers' demands, along with new results concerning the negative impact of high demand variabilities in

the solution.

In addition, the branch-and-price algorithm incorporates some new methodological features that could be applied for solving related problems:

(1) An updating procedure to keep track of the variance of the cost of an optimal policy. This is particularly useful whenever some minimal risk assessment is needed alongside an optimal (in terms of expected values) operational policy;

(2) A Monte Carlo procedure to verify feasibility of a priori routes, along with mechanisms for controlling statistical inference errors within the branch-and-price framework;

(3) An exact procedure to derive the distribution of the route duration when optimal restocking is adopted.

The remainder of this paper is organized as follows: in §2, we present general definitions and a VRPSD-DC formulation suitable for the application of branch-and-price. In §3, the branch-and-price algorithm for the VRPSD-DC is described. In §4, extensive computational results are presented and discussed. Finally, in §5 we make some concluding remarks.

2 VRPSD-DC Formulation

2.1 General Definitions

The VRPSD-DC is defined on a complete graph with $N + 1$ nodes. Node v_0 denotes the depot, and nodes $v_1, \ldots, v_N$ represent the customers who need to be visited. The travel time from node v_i to node v_j, $i, j \in \{0, \ldots, N\}$, is known and given by t_{ij}. The triangle inequality holds. We assume that the traveling cost between two nodes is proportional to the travel time between these nodes. Therefore, the traveling cost from a node v_i to another node v_j is also given by t_{ij}. A homogenous fleet of vehicles, each with a capacity of Q, is available for serving the N customers. We assume that an unlimited number of vehicles is available, but the formulation and the algorithm can be easily adapted to also deal with the case of a limited fleet. Both cases have been tested in the computational experiments.

At route planning time, the demand of the customers is only known in probability distribution. We represent the demand of customer v_i by the discrete random variable D_i, and denote the probability mass function of D_i by $p_{D_i}(\cdot)$. Thus, $p_{D_i}(k)$ gives the probability of customer v_i having a demand of k. The demands of the customers are assumed to be independently distributed. For a demand value $k > Q$, it is possible for a customer v_j that $p_{D_j}(k) > 0$. Even though such a case may be seldom observed in practice, this allows us to experiment with unbounded probability distributions with a (significant) part of their tails extending beyond Q.

2.2 A Priori Routes

An a priori route is a sequence of nodes that starts at the depot, visits a number of customers, and finally returns to the depot. We formally define a priori routes as follows:

Definition 1. *An a priori route is a sequence* $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$, *such that* $f > 0$, $s_0 = s_{f+1} = 0$, *and* $s_i \neq s_j$ *for all* $i, j \in \{1, \ldots, f\}$, $i \neq j$.

In the a priori paradigm, the customers are visited according to the sequence prescribed by the a priori route. The demand of each customer is only disclosed upon arrival of the vehicle at the customer's location. If the vehicle arrives at a customer with an insufficient remaining load (or residual capacity) to serve that customer's demand (i.e., if a failure occurs), then the demand is partially served, and a return trip to the depot is performed to replenish (or empty) the vehicle and serve the pending demand. In addition, to decrease the chances of failures at customers located far from the depot, the vehicle may perform a preventive replenishment trip after one customer has been served and before visiting the next customer. Therefore, the duration of a route depends on the realizations of the demands and on the restocking policy adopted, which means it is a random variable. We denote the duration of an a priori route r by the random variable T_r.

The durations of the routes are constrained in a probabilistic way. A feasible route must finish within a maximum time interval of $\overline{T}$ with a positive probability of $\mathcal{P}$, where $\overline{T}$ and $\mathcal{P}$ are parameters of the problem. We formalize feasible a priori route as follows:

Definition 2. *A feasible a priori route is an a priori route r such that* $\Pr[T_r \leq \overline{T}] \geq \mathcal{P}$.

Under optimal restocking, the expected duration (or cost) of an a priori route can be computed with the dynamic programming algorithm from Yee and Golden (1980). The version of the algorithm we provide below also allows for demands that exceed the capacity of the vehicle.

Let $\nu_r(i,q)$ be the expected remaining duration of an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$ just after the i-th customer has been fully served, and the remaining load in the vehicle is q. Let $j = i+1$. Under optimal restocking and if $j \leq f$, as soon as the i-th customer has been served an optimal decision is made about whether to visit the j-th customer directly or after a replenishment trip to the depot. Therefore, $\nu_r(i,q)$ can be calculated with the following recursion:

$$\nu_r(i,q) = \min\Bigg\{ t_{s_i s_j} + \sum_{k=0}^{\infty} [\Gamma_{k,q}(t_{s_j 0} + t_{0 s_j}) + \nu_r(j, q + Q\Gamma_{k,q} - k)] p_{D_{s_j}}(k), \\ t_{s_i 0} + t_{0 s_j} + \sum_{k=0}^{\infty} [\Gamma_{k,Q}(t_{s_j 0} + t_{0 s_j}) + \nu_r(j, Q + Q\Gamma_{k,Q} - k)] p_{D_{s_j}}(k) \Bigg\}, \quad (1)$$

where $\Gamma_{k,q} = \max\{0, \left\lceil \frac{k-q}{Q} \right\rceil\}$ gives the number of round-trips to the depot that are needed when the realized demand is k and the remaining load in the vehicle is q.

The recursion can be solved by dynamic programming starting with the base case $\nu_r(f,q) = t_{s_f 0}$, $q \in \{0, \ldots, Q\}$. Then, the expected duration of r is given by:

$$\mathbb{E}[T_r] = \nu_r(0, Q)\,.$$

With the above definitions, the VRPSD-DC can be formulated as the following set-partitioning problem:

$$\begin{aligned} &\text{minimize} && \sum_{r \in \mathcal{R}} \mathbb{E}[T_r]\theta_r\,, && \\ &\text{subject to} && \sum_{r \in \mathcal{R}} a_{ir}\theta_r = 1 && \forall i \in \{1, \ldots, N\}\,, \\ & && \theta_r \in \{0,1\}\,. && \end{aligned} \quad (2)$$

where $\mathcal{R}$ is the set of all feasible a priori routes and a_{ir} is a binary value that indicates whether customer v_i is visited by the a priori route r.

Some VRPSD formulations also consider a fixed cost for each route in the solution, fixed costs for route failures, and/or fixed costs for replenishment trips to the depot. These costs could also be included in our model by adapting the set-partitioning formulation or the dynamic programming recursion accordingly. The branch-and-price algorithm can be specialized for all these cases. We decided to proceed by focusing only on minimizing traveling costs, because those fixed costs, if applicable, have specific values that are certainly application dependent.

3 Branch-and-Price Algorithm for the VRPSD-DC

Set-partitioning formulations are known to provide very good continuous relaxation bounds for a variety of routing problems. Due to the very large number of feasible routes (and hence of decision variables), the linear relaxation of these formulations is usually solved by column generation (Desaulniers, Desrosiers, and Solomon 2005). When incorporated into a branch-and-bound framework, the resulting technique is called branch-and-price (Barnhart et al. 1998).

3.1 Pricing Problem

In order to solve the relaxed set-partitioning formulation by column generation, we start by solving a restricted formulation with a reduced set of variables (for example, the N variables corresponding to all single-customer routes). Column generation is an iterative procedure, where we solve the so-called pricing problem at each iteration to identify and add to the restricted formulation variables with negative reduced cost, or to confirm that no such variable exists. In the latter case, the solution to the restricted formulation is also the solution to the complete formulation. Let $\lambda_1, \ldots, \lambda_N$ be the dual values associated with constraints (2) at an iteration. Then, the pricing problem lies in finding a set $\mathcal{Q} \subset \mathcal{R}$ of routes such that:

$$\mathbb{E}[T_r] - \sum_{i=1}^{N} a_{ir}\lambda_i < 0\,, \qquad \forall r \in \mathcal{Q}\,. \tag{3}$$

Similarly to the branch-price-and-cut (BP&C) algorithm by Florio, Hartl, and Minner (2020) for the VRPSD with capacity constraints, we solve the pricing problem with a backward labeling algorithm. As opposed to the BP&C approach, however, we do not apply dominance rules, and neither can we employ any capacity-based completion bound. In fact, the only component that is shared between the two pricing algorithms is the label updating procedure that keeps track of the expected remaining cost-to-go (or duration-to-go) of partial routes. In the VRPSD-DC, we extend these procedures to also keep track of the *variance* of the remaining duration-to-go.

The general strategy for solving the pricing problem consists in: **(i)** pricing only elementary routes. This was motivated by the difficulty of deriving label dominance rules, and also because when you only price routes without customer repetition, stronger completion bounds can be proposed; **(ii)** verifying route feasibility with Chebyshev bounds, Monte Carlo simulation and statistical inference, and by an exact procedure that derives the distribution of the route duration; and **(iii)** employing completion bounds to control the combinatorial growth of labels.

3.2 Backward Labeling

The reason why we adopt backward labeling is that it allows each label to store the intermediate results of the dynamic programming recursion from equation (1). By doing so, only one stage of the dynamic program needs to be computed whenever a label is (backwardly) extended, since the duration-to-go has already been computed and is stored in the original label from which the extension was created. These updating procedures are described in Florio, Hartl, and Minner (2020).

After the generation of every single extension, the labeling algorithm checks whether the corresponding route has a negative reduced cost. If so, a feasibility check is performed. If the route is feasible, then it is saved for insertion in the restricted formulation. If the route does not have a negative reduced cost, then an orienteering-based completion bound is computed, and the label is discarded if that bound is non-negative. These bounds are the only provisions for containing the number of labels, since label dominance rules are not available. In the remainder of this section we focus on the innovative components of the algorithm, namely, the different ways of determining route feasibility, the completion bounds, and how to deal with statistical errors within branch-and-price.

3.3 Checking Feasibility of Routes

We adopt three distinct strategies for verifying the feasibility of a priori routes. Strategy I consists in applying Chebyshev's bounds on the duration of the routes, and requires procedures to keep track of the variance of the durations. As we will show, these procedures can be incorporated into the pricing algorithm. Strategy I is a *positive* strategy, in the sense that it can confirm that an a priori route is feasible, but it cannot verify infeasibility.

On the other hand, strategies II and III can determine both feasibility and infeasibility (hence, they are both *positive* and *negative* strategies). Strategy II employs Monte Carlo sampling and statistical inference to try to determine, with a very high confidence level, the probability that the duration of a route exceeds a given limit. Strategy III consists

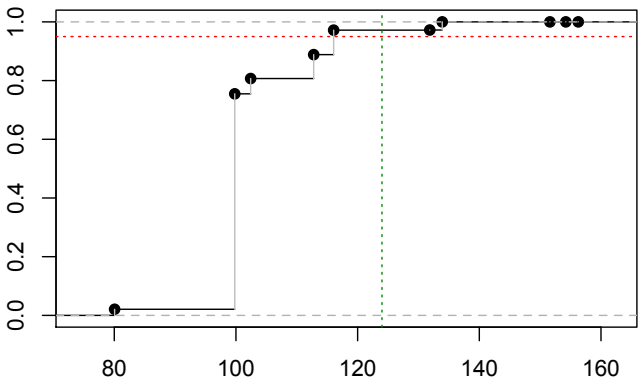

Figure 2: Empirical CDF of the duration of the dashed route from solution 1b *(see Figure 1).*

in deriving the distribution of the duration of an a priori route, from which feasibility or infeasibility can be confirmed. This strategy is more costly, but provides a correct answer in the cases where strategy II is inconclusive, i.e., when the inference process cannot determine feasibility nor infeasibility. This typically happens in routes that are almost feasible (or almost infeasible), meaning that the probability of the route finishing within the time limit is slightly below (or above) the chance-constraint level.

Note that the feasibility check performed by strategy II is the result of a statistical inference process that is subject to statistical errors. These errors are extremely unlikely, considering the adopted confidence level. In addition, such errors, if they happen, can (in most cases) be detected within the branch-and-bound search for the optimal solution. These aspects are discussed thoroughly in §3.6.

Before further describing the strategies, we present a simple example that shows what the cumulative distribution function (CDF) of the duration of a typical a priori route looks like. In Figure 2, the empirical CDF of one of the routes from solution 1b (Figure 1) is depicted. This distribution has been obtained by simulating the a priori route 10^6 times. As we see, the shape of this CDF is a highly irregular step function, with no resemblance to any known probability distribution. Indeed, trying to fit such data to known distributions has not produced any meaningful results. Therefore, the methods we used for deriving bounds on the CDF of a priori routes could not exploit properties of individual probability distributions. This was our motivation for creating such bounds by using Chebyshev's inequality (a relatively weak inequality, but applicable to any distribution with finite expectation and variance) and Monte Carlo sampling.

3.3.1 Strategy I: Chebyshev's bounds.

Consider an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$, and that the i-th customer in r, $i \in \{1, \ldots, f-1\}$, has just been fully served, and that the remaining load in the vehicle is q, $q \in \{0, \ldots, Q\}$. The duration-to-go of r is a random variable, which we denote by $T_r^{i,q}$. Note that, using this notation, $T_r = T_r^{0,Q}$. The expected value of $T_r^{i,q}$ corresponds to the expected duration-to-go when optimal restocking is applied, so $\mathbb{E}[T_r^{i,q}] = \nu_r(i,q)$. Let $j = i+1$. Using the law of total variance, by conditioning on the demand of customer v_{s_j}, the variance of $T_r^{i,q}$ can be decomposed as follows:

$$\begin{aligned}\mathrm{Var}(T_r^{i,q}) = \sum_{k=0}^{\infty} \mathrm{Var}(T_r^{i,q} | D_{s_j} = k) p_{D_{s_j}}(k) + \sum_{k=0}^{\infty} \mathbb{E}[T_r^{i,q} | D_{s_j} = k]^2 (1 - p_{D_{s_j}}(k)) p_{D_{s_j}}(k) \\ - 2 \sum_{k=1}^{\infty} \sum_{l=0}^{k-1} \mathbb{E}[T_r^{i,q} | D_{s_j} = k] p_{D_{s_j}}(k) \mathbb{E}[T_r^{i,q} | D_{s_j} = l] p_{D_{s_j}}(l)\,. \quad (4)\end{aligned}$$

An optimal restocking policy is characterized by threshold values τ_i, such that whenever the remaining load in the vehicle after fully serving the i-th customer is τ_i or less, it is optimal to replenish at the depot before visiting the j-th customer (Yee and Golden 1980).

Consider a demand realization k for the j-th customer. Then, the remaining load in the vehicle after fully serving the j-th customer, which we denote by q', is given by:

$$q' = \begin{cases} q + Q\Gamma_{k,q} - k, & \text{if } q > \tau_i\,, \\ Q + Q\Gamma_{k,Q} - k, & \text{if } q \leq \tau_i\,. \end{cases}$$

Therefore, for any $i \in \{1, \ldots, f-1\}$, the conditional variance of $T_r^{i,q}$ given a demand realization k for the j-th customer can be expressed as follows:

$$\text{Var}(T_r^{i,q} | D_{s_j} = k) = \text{Var}(T_r^{j,q'})\,. \tag{5}$$

In addition, if $i = f$, then $\text{Var}(T_r^{i,q}) = 0$, since the remaining duration-to-go is deterministic. Combining this base case with (4) and (5), we have a mechanism for keeping track of the variance of the route duration when optimal restocking is applied. In the backward labeling algorithm, we simply add to the labels the attributes $\sigma^2(q)$, $q \in \{0, \ldots, Q\}$, which are initialized according to the base case, and updated according to (4) and (5) whenever a label is extended backwardly.

Consider a route r with a negative reduced cost, identified by the labeling procedure. From the one-sided Chebyshev inequality (Feller 2008), we have:

$$\Pr[T_r > \overline{T}] \leq \frac{\text{Var}(T_r)}{\text{Var}(T_r) + (\overline{T} - \mathbb{E}[T_r])^2}\,. \tag{6}$$

Then, the feasibility of r can be confirmed if the right-hand side of (6) is less than or equal to $1 - \mathcal{P}$, where $\mathcal{P}$ is the level at which the maximum duration constraint should be enforced. If this is not the case, then the feasibility (or infeasibility) of r must be verified with the process we will describe next.

The technique illustrated above provides a simple way of computing the variance associated with an optimal policy, and could also be applied in other contexts where optimal operational policies are obtained with dynamic programming.

3.3.2 Strategy II: Monte Carlo sampling and statistical inference.

Consider an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$. A random scenario for r is an f-dimensional vector ξ, where the i-th component is a random value generated from the demand probability distribution of the i-th customer in r. Given a scenario ξ, the actual duration of r can be computed by simulating its execution under optimal restocking. This simulation requires $\mathcal{O}(f)$ operations, since the threshold values τ_i, $i \in \{1, \ldots, f\}$, are already available. We denote this simulation procedure by Ξ, and the actual duration of r under scenario ξ by $\Xi(r, \xi)$.

The parameter of interest is $p_r = \Pr[T_r \leq \overline{T}]$. The idea of this Monte Carlo approach for verifying the feasibility of r is to create successive interval estimations for p_r by repeatedly generating a random scenario ξ, then simulating it to check whether $\Xi(r, \xi) \leq \overline{T}$, and applying inferential statistics. The procedure stops as soon as it can be determined, with a high confidence level, that p_r is either greater than or equal to $\mathcal{P}$ (in which case r is deemed feasible), or less than $\mathcal{P}$ (in which case r is deemed infeasible). If feasibility cannot be inferred within a certain number of simulations, then we activate the more costly strategy III. The Monte Carlo feasibility procedure in described in Algorithm 1.

Algorithm 1 Monte Carlo feasibility check of an a priori route

```
function isFeasible(r)                          ▷ attempts to infer whether r is feasible
    in ← 0
    for n = 1 to N_high do
        ξ ← genScenario(r)
        if Ξ(r, ξ) ≤ T̄ then
            in ← in + 1
        (lower, upper) ← AgrestiCoull5(n, in)
        if upper < P then return false
        else if lower ≥ P then return true
    return "inconclusive"
```

The algorithm simulates the execution of the route up to N_{high} times (in the experiments, $N_{high} = 10^3$). Note that each simulation is equivalent to a Bernoulli trial with a success probability of p_r. After each simulation, we define a confidence interval using the method from Agresti and Coull (1998) for estimating binomial proportions, which also performs well for extreme (near 0 or 1) proportion values (Brown, Cai, and DasGupta 2001). The interval is defined considering 5 standard deviations, which translates into a coverage probability of around 99.99994%. This seemingly unnecessary level of precision is actually important, since the number of routes examined during the complete execution of branch-and-price can be quite large. Note also that most routes are clearly feasible or clearly infeasible (i.e., p_r is far from $\mathcal{P}$), and in both cases the procedure returns after a relatively small number of simulated scenarios.

If the algorithm cannot infer feasibility (or infeasibility) within the specified confidence level, then strategy III is activated. Typically, only a few hundred routes, those with p_r values very near to $\mathcal{P}$, need to be examined by strategy III, which is described in the next section.

Strictly speaking, whenever strategy II is employed, the branch-and-price algorithm becomes probabilistic, i.e., at least in theory, there is no guarantee that the optimal solution will always be returned. We will further discuss this issue in §3.6.

3.3.3 Strategy III: Deriving the distribution of the route duration analytically.

Consider again an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$, and that the i-th customer in r, $i \in \{1, \ldots, f\}$, has just been fully served. The remaining load in the vehicle and the route duration so far are random variables, which we denote by L_i and P_i, respectively. Let $p^{r,i}_{L,P}(l, p)$ be the joint probability mass function (PMF) of L_i and P_i (the subscripts in L_i and P_i have been omitted for simplification). After the first customer in r is fully served, $p^{r,1}_{L,P}(l, p)$ can be completely specified as follows:

$$p^{r,1}_{L,P}(Q + Q\Gamma_{k,Q} - k, t_{0s_1} + \Gamma_{k,Q}(t_{s_1 0} + t_{0s_1})) = p_{D_{s_1}}(k)\,. \tag{7}$$

Starting from $p^{r,1}_{L,P}(\cdot,\cdot)$, the joint PMF $p^{r,f}_{L,P}(\cdot,\cdot)$ can be derived by an iterative procedure, which generates each $p^{r,j}_{L,P}(\cdot,\cdot)$, $1 < j \leq f$ from the joint PMF $p^{r,j-1}_{L,P}(\cdot,\cdot)$, the demand distribution of the j-th customer $p_{D_{s_j}}(\cdot)$, and the threshold value related to the $(j-1)$-th customer τ_{j-1}. With the joint PMF of L_f and P_f, $\Pr[T_r > \overline{T}]$ can then be computed exactly. The full procedure is described in Algorithm 2.

Algorithm 2 Exact feasibility check of an a priori route

function isFeasible(r) ▷ returns TRUE iff r is feasible
  $p^{r,1}_{L,P}(\cdot,\cdot) \leftarrow$ initialJointPMF(r) ▷ initialize $p^{r,1}_{L,P}(\cdot,\cdot)$ from equation (7)
  for $j = 2$ **to** f **do**
    $p^{r,j}_{L,P}(\cdot,\cdot) \leftarrow \mathbf{0}$ ▷ initialize with zeros
    for all l, p **such that** $p^{r,j-1}_{L,P}(l,p) > 0$ **do**
      for all k **such that** $p_{D_{s_j}}(k) > 0$ **do**
        $a \leftarrow p^{r,j-1}_{L,P}(l,p)\, p_{D_{s_j}}(k)$
        if $l > \tau_{j-1}$ **then** ▷ visiting s_j directly
          $p^{r,j}_{L,P}(l + Q\Gamma_{k,l} - k, p + t_{s_{j-1}s_j} + \Gamma_{k,l}(t_{s_j 0} + t_{0s_j})) \mathrel{+}= a$
        else ▷ restocking before visiting s_j
          $p^{r,j}_{L,P}(Q + Q\Gamma_{k,Q} - k, p + t_{s_{j-1}0} + t_{0s_j} + \Gamma_{k,Q}(t_{s_j 0} + t_{0s_j})) \mathrel{+}= a$
  return $\sum_{l=0}^{Q} \sum_{p \leq \overline{T} - t_{s_f 0}} p^{r,f}_{L,P}(l,p) \geq \mathcal{P}$

We now briefly discuss the average runtime complexity of Algorithm 2. Our intention is not to formally prove asymptotic properties of the algorithm, but just to estimate the runtime complexity for the cases we are most concerned about. In order to simplify this analysis, we assume that $p_{D_i}(k) = 0$ if $k > Q$ for every customer v_i. Clearly, Algorithm 2

is $\Omega(NQ^2)$, i.e., its runtime complexity cannot be better than that. However, depending on the number of pairs (r, l) such that $p^{r,j}_{L,P}(r, l)$ is positive, the complexity can be considerably higher. Each restocking event leads to a diversion from the planned a priori route, and gives rise to a new possibility for the actual duration of the route. Let $\kappa = \lceil \frac{\sum_{v_i \in r} \mathbb{E}[D_i]}{Q} \rceil$ be a rough estimation of the number of restocking trips needed when executing an a priori route r. The individual demand variabilities also influence the number of restocking trips, but we leave these aspects aside in this analysis. Since a restocking can occur between any two successive customers, and also at any customer in the a priori route (due to failures), it is reasonable to conjecture that the number of possible values for the actual duration of r growths according to the function $(2n)^\kappa$. Based on that, it is also reasonable to estimate that the average-case runtime complexity of Algorithm 2 is $\mathcal{O}(2^\kappa N^{1+\kappa} Q^2)$. However, in most imaginable applications, the value of κ is likely to be small: a couple of restocking trips, at most. In these cases, verifying route feasibility with Algorithm 2 is computationally tractable.

3.3.4 Combining all strategies.

Even though expression (6) can be evaluated in constant time, strategy I requires the execution of additional procedures whenever a label is (backwardly) extended. From a runtime complexity perspective, the critical step is the computation of the last term in (4). Since (4) needs to be calculated for every possible remaining load, extending from a customer v_j requires $\mathcal{O}(K_j^2 Q)$ operations, where K_j is the number of possible demand values of v_j. This is a major increase compared to the $\mathcal{O}(K_j Q)$ operations needed when only the intermediate results of the dynamic programming are updated. For this reason, strategy I must be applied with parsimony whenever K_j is large. In our experiments, where the demands followed Poisson or other unbounded distributions (hence, we have rather large K_j values when only rounding very small probabilities to zero), strategy I was mainly used for confirming the feasibility of the routes generated in the first iterations of column generation. After these initial iterations, the labeling algorithm will no longer update the variances, and the route feasibility checks will then be performed exclusively by strategies II and III.

3.4 Completion Bounds

A completion bound is a lower bound on the reduced cost of all columns that can be obtained from a particular label (Poggi and Uchoa 2014). In our case, since dominance rules are not available, completion bounds are fundamental for efficiently solving the pricing problem.

In the absence of capacity constraints, the only resource available for trying to contain the exponential growth of labels is the route duration. Therefore, instead of capacity-based completion bounds, we propose an orienteering problem (Gunawan, Lau, and Vansteenwegen 2016) based bound, comparable to the one presented in Florio, Feillet, and Hartl (2018) for another routing problem that also involved duration constraints.

In the labeling algorithm, each label corresponds to a partial route, which is defined as follows:

Definition 3. *A partial route is a sequence $p = (v_{u_1}, \ldots, v_{u_g}, v_{u_{g+1}})$, such that $(v_0, v_{u_1}, \ldots, v_{u_g}, v_{u_{g+1}})$ is a feasible a priori route.*

We say that a partial route $p = (v_{u_1}, \ldots, v_{u_g}, v_{u_{g+1}})$ *spawns* an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$ if there exists a k such that $s_{k+j} = u_j$ for all $j \in \{1, \ldots, g\}$. We have a special case when $k = 0$, which leads to the a priori route $p_0 = (v_0, v_{u_1}, \ldots, v_{u_g}, v_{u_{g+1}})$. We also define the a priori cost of p as $c_p = \sum_{i=1}^{g} t_{u_i u_{i+1}}$. With these definitions, a completion bound for partial routes is stated by means of the following proposition:

Proposition 1. *Let $p = (v_{u_1}, \ldots, v_{u_g}, v_{u_{g+1}})$ be a partial route, a_{ip} a binary value indicating whether customer v_i is visited by p, and $\mathcal{K}$ the set of all feasible a priori routes spawned by p. Then:*

$$\min_{r \in \mathcal{K}} \left\{ \mathbb{E}[T_r] - \sum_{i=1}^{N} a_{ir} \lambda_i \right\} \geq \mathbb{E}[T_{p_0}] - \sum_{i=1}^{N} a_{ip} \lambda_i - \Pi \,, \tag{8}$$

where Π *is the value of the optimal solution to an orienteering problem with a time limit of* $\overline{T} - c_p$*, starting at the depot and finishing at customer* v_{u_1}*, and with a set of score nodes* $\{v_i : i \in \{1, \ldots, N\}, a_{ip} = 0\}$*, each associated with a reward of* λ_i*.*

Proof. First, note that p_0 is the route that minimizes $\mathbb{E}[T_r]$, $r \in \mathcal{K}$. Any other route spawned by p visits at least one customer between the depot and v_{u_1}, and, by the triangle inequality, cannot have an expected cost smaller than $\mathbb{E}[T_{p_0}]$.

Secondly, consider an a priori route $r = (v_{s_0}, v_{s_1}, \ldots, v_{s_f}, v_{s_{f+1}})$, $r \in \mathcal{K}$. Let $c_r = \sum_{i=0}^{f} t_{s_i s_{i+1}}$ be the a priori cost of r. Since r is spawned by p, it can be decomposed into two segments, $s = (v_0, \ldots, v_{u_1})$ and p. Let $c_s = c_r - c_p$ be the a priori cost of the segment s. Since r is feasible, it must hold that $c_s \leq \overline{T} - c_p$ (otherwise $\Pr[T_r > \overline{T}] = 1$ since, by the triangle inequality, $T_r \geq c_r$ holds for any scenario). Consequently, s is a feasible solution to the orienteering problem stated in the proposition, and has a value of $\sum_{i=1}^{N} a_{ir}\lambda_i - \sum_{i=1}^{N} a_{ip}\lambda_i \leq \Pi$. □

The left-hand side of (8) corresponds to the minimum reduced cost over all routes spawned by p. Therefore, the label associated with p can be discarded if the right-hand side is nonnegative. Any upper bound to the orienteering problem can be inserted in (8) in place of Π, and a valid (but less tight) completion bound will still be available. Considering that the labeling algorithm is essentially a combinatorial procedure, it is important to compute completion bounds very efficiently. In our implementation, we adopt the linear relaxation of the knapsack bound to the orienteering problem (Laporte and Martello 1990) for computing completion bounds, which leads to bound evaluations in $\mathcal{O}(N \log N)$ time.

3.5 Branch-and-Bound

The branch-and-price algorithm applies two distinct branching rules. Let $\overline{\theta}_r$, $r \in \mathcal{R}$, be an optimal solution to the linear relaxation of the set-partitioning formulation, and $S = \sum_{r \in \mathcal{R}} \overline{\theta}_r$. If S is a fractional value, we branch on the number of routes (first rule), enforcing in one branch that $\sum_{r \in \mathcal{R}} \theta_r \leq \lfloor S \rfloor$, and in the other branch that $\sum_{r \in \mathcal{R}} \theta_r \geq \lceil S \rceil$. This additional constraint does not change the structure of the pricing problem.

If S is integer but some $\overline{\theta}_r$ are fractional, then the second rule is applied: we choose a pair of nodes (v_i, v_j) such that $\sum_{r \in \mathcal{R}} \overline{\theta}_r b_r^{ij}$ is a fractional value, where b_r^{ij} is a binary value indicating whether customer v_j is visited immediately after v_i in the a priori route r. Then, we enforce that v_j is visited immediately after v_i in one branch, and that v_j is not visited immediately after v_i in the other branch. Typically, when applying the second rule, there will be several pairs of nodes to choose from. We select the pair that yields the highest increase in the lower bound, when only the columns generated so far are considered. This can be efficiently assessed by solving several linear programs that share the same feasible region and only differ in the cost vector. The new constraints enforced by the second rule can be readily incorporated into the labeling algorithm.

3.6 A Note on Exactness and on Statistical Errors

The branch-and-price algorithm is exact when only strategies I and III are employed for checking route feasibility, and probabilistic when strategy II is also used. Statistical inference procedures, such as the binomial proportion test performed in §3.3.2, are always subject to statistical errors. Considering the null hypothesis that a route r is feasible, these errors can be of two types:

Type I: Inferring that r is infeasible when it is actually feasible; or

Type II: Inferring that r is feasible when it is actually infeasible.

We provide some arguments to show that type I errors are unlikely to affect the execution of the algorithm, and that type II errors are unlikely to go undetected during the execution of the algorithm. Consider first that a type I error occurred in the column generation procedure, i.e., a feasible a priori route r with a negative reduced cost was deemed infeasible by Algorithm 1. Recall that the pricing problem is solved iteratively until no more columns with a negative reduced cost can be found. If the error occurred at an iteration of the pricing problem that was not the last iteration (in other words, if at least one other negative reduced cost column could be identified in the same iteration where the error occurred),

then the error will not influence the lower bound obtained on that particular node of the branch-and-bound tree, because there are still some iterations left. That is, for a type I error to impact the execution of the algorithm, it would have to occur when the feasibility of the last reduced cost route is examined, *and* that column would have to improve the lower bound being calculated at the branch-and-bound node. The first event alone is very unlikely, which is due to the very high confidence level we use when checking feasibility with strategy II. The second event is also unlikely, which is due to the well-known *tailing-off* effect in column generation (Lübbecke and Desrosiers 2005). Even though theoretically possible, the probability of both events happening simultaneously is extremely small.

Consider now that a type II error occurred, which means that the pool of generated columns contains an infeasible route r. Along the branch-and-bound algorithm, a series of *consistency checks* is implemented to control type II errors. These checks consist of an additional computation of the linear relaxation of a branch-and-bound node, just before a branching decision is made on that node. For this, we use all columns generated so far during execution. Of course, the linear relaxation value has to be the same as computed previously for that node. If it decreases, then this is because a new route is used in the solution. Most likely, it will be an infeasible route, considering again the arguments presented before concerning type I errors. When such an inconsistency happens, the algorithm indicates it and aborts execution. If an infeasible route enters the pool of columns, it is very likely that a branch-and-bound node will use this column to improve its linear relaxation, so type II errors are hard to go unnoticed. These considerations are not exclusively pertaining to the VRPSD-DC. In fact, feasibility checks of probabilistic constraints by statistical inference go very well with the branch-and-price framework because of its iterative nature and the possibilities of easily incorporating inexpensive consistency checks.

In conclusion, no statistical errors could be observed when we enforced a confidence level of 99.99994% (namely, 5 standard deviations) in the binomial proportion test. Note that, if this confidence level cannot be reached within $N_{high} = 10^3$ simulation runs, then route feasibility is verified exactly by strategy III. With these parameters, the Monte Carlo feasibility check provides significant speedups in all the instances we tested, with no observable compromise concerning optimality.

4 Computational Results

We tested the branch-and-price algorithm on several literature instances, assuming different constraint levels and demand variances. In §4.1, we compare our results with those of Mendoza, Rousseau, and Villegas (2016) (MRV-16). In §4.2, we generate and solve new instances obtained by applying duration reduction factors on the MRV-16 instances. In §4.3, we experiment with duration constraints enforced at levels lower (90%) and higher (98%) than the 95% level considered so far. Finally, in §4.4 we solve instances assuming under-dispersed and over-dispersed random demands. All computational experiments were performed in a single thread of an Intel® Core™ i7-4790 (3.60GHz) CPU, with an available memory of 32GB. Linear programs were solved using IBM® CPLEX® version 12.6.1. All probabilities were computed with a precision of 10^{-6}, which means that whenever a probability fell below this value, it was simply truncated to zero.

The implemented algorithm also employs strategy II for route feasibility checks. In any case, we do refer to the obtained solutions as *optimal* solutions, taking into account that the probability of an incorrect result is extremely small (see section §3.6).

4.1 Comparison with MRV-16

In MRV-16, VRPSD-DC instances were created by taking literature instances and defining that the maximum duration is equal to the maximum expected cost among all routes of the optimal VRPSD solution under detour-to-depot restocking. Thus, the maximum durations are rather high, and the resulting instances considerably difficult. We allowed up to 30 hours of running time for solving these instances, which have never been solved to optimality (not even under detour-to-depot restocking and with capacity constraints). The results are presented in Table 1.

Table 1: Comparison with MRV-16 ($\mathcal{P} = 95\%$)

	MRV-16		B&P					B&P-FIX[d]				
Instance	Best	#[a]	Best[b]	#	$\#_{>Q}$	Gap	Time[c]	Best	$\overline{\#}$	$\#_{>Q}$	Gap	Time
A-n32-k5	866.77	5	*866.45	5	0		5.00	*881.81	4	2		14.02
A-n33-k5	735.00	6	*732.82	5	1		6.74	×[e]				
A-n33-k6	793.90	6	*792.54	5	3		10.64	×				
A-n34-k5	839.01	6	*838.95	6	0		0.44	*839.90	5	1		0.82
A-n37-k5	713.99	5	710.78	4	1	1.57%	30	×				
A-n38-k5	777.59	6	*777.55	6	0		29.58	*787.33	5	2		16.52
A-n39-k6	889.40	6	887.04	5	1	0.81%	30	×				
A-n48-k7	1210.79	7	1212.32	7	1	1.60%	30	1251.82	6	2	3.62%	30
P-n19-k2	233.36	3	*227.87	2	1		0.02	×				
P-n20-k2	240.84	3	*240.80	3	0		0.12	*242.12	2	1		0.21
P-n21-k2	234.00	3	*233.98	3	0		0.09	*236.43	2	1		0.33
P-n22-k2	242.19	3	*242.18	3	0		0.43	×				
P-n22-k8	715.81	10	*591.56	7	1		0.17	*592.42	6	3		0.25
P-n23-k8	634.46	10	*620.45	7	3		0.14	*657.23	6	5		0.37
P-n40-k5	488.50	5	*487.61	5	0		0.89	×				
P-n45-k5	539.66	6	539.65	6	0	0.09%	30	545.07	5	1	0.32%	30
P-n50-k8	680.42	9	680.34	9	0	0.24%	30	688.47	8	2	0.97%	30
P-n50-k10	772.25	11	771.49	10	1	0.11%	30	798.59	9	4	1.51%	30
P-n51-k10	833.42	11	832.40	10	1	0.49%	30	842.57	9	3	0.49%	30
P-n55-k10	759.36	11	754.12	10	1	0.55%	30	762.29	9	2	0.75%	30
P-n55-k15	1086.44	17	1075.18	14	4	0.38%	30	1076.89	13	5	0.17%	30
P-n60-k10	811.44	11	809.42	10	2	0.17%	30	849.87	9	4	2.68%	30
P-n60-k15	1110.72	17	1101.41	15	3	0.61%	30	1101.72	14	4	0.50%	30

[a] *Number of routes in the solution.* [b] *Optimal solutions indicated with* *. [c] *Running time in hours.* [d] *Number of routes fixed to* $\overline{\#}$. [e] *Instance is infeasible.*

The branch-and-price (B&P) algorithm was able to compute the exact linear relaxation in 23 out of the 36 instances of the sets `A` and `P`. These are the instances reported in Table 1. The remaining 13 instances have a very large maximum duration attribute, which causes a very fast growth on the number of labels before the completion bounds can act effectively. In these instances, the limiting computational resource was the available memory and not the running time. Stronger completion bounds would be needed for solving these instances. Of the 23 instances where a lower bound could be computed, 12 were solved to optimality, the largest among them instance `P-n40-k5` with 40 nodes. The average optimality gap in the remaining 11 instances was only 0.60%.

Recall that in MRV-16 detour-to-depot restocking is assumed, and capacity constraints are enforced. Therefore, the B&P solutions are not improved solutions to exactly the same problem considered in MRV-16. However, similarly to the example presented in Figure 1, the B&P solutions are superior to the MRV-16 solutions with regard to the total expected cost (22 out of 23 instances) and the number of vehicles utilized (13 out of 23 instances) without violating the maximum duration constraint at a level of 95%, as in MRV-16. Therefore, from an operational costs perspective, solutions obtained by solving the VRPSD-DC under optimal restocking and without capacity constraints are preferable.

More often than not, optimal or near optimal solutions contain one or more routes that serve an average demand that is larger than the capacity of the vehicle (see column $\#_{>Q}$). These solutions exploit optimal restocking along these routes to reduce overall traveling costs. A secondary effect is the reduction on the number of required vehicles (assuming one separate vehicle is required for each route), even when no fixed costs for employing vehicles are charged.

The last columns in Table 1 (B&P-FIX) refer to the limited fleet case. Here, we attempted to solve the same instances, but with one additional constraint enforcing that the number of routes is equal to the number of routes obtained in the best B&P solution minus one. Seven instances became infeasible when this additional constraint was added. In the remaining instances, limiting the fleet size increased traveling costs by 1.7% on average, but in some instances the impact exceeded 5%.

Table 2: MRV-16 instances with duration reduction factors ($\mathcal{P} = 95\%$)

	$\alpha = 0.85$				$\alpha = 0.95$			
Instance	Best	#	Gap	Time	Best	#	Gap	Time
A-n32-k5	*1175.17	7		0.09	*884.37	5		0.54
A-n33-k5	*1027.65	8		0.35	749.92	5	0.74%	5
A-n33-k6	*822.84	7		0.29	*801.39	6		3.57
A-n34-k5	*933.56	7		0.93	863.66	6	0.25%	5
A-n37-k5	724.78	5	0.66%	5	717.73	5	2.12%	5
A-n37-k6	1104.72	6	1.42%	5	$\varnothing^b$			
A-n38-k5	*818.21	6		1.52	794.21	6	0.39%	5
A-n39-k6	951.69	6	0.54%	5	894.43	5	1.71%	5
A-n45-k6	1124.89	8	1.67%	5	1052.54	7	3.04%	5
A-n48-k7	$\times^a$				1231.61	8	3.61%	5
P-n19-k2	*248.15	3		0.01	*240.17	3		0.03
P-n20-k2	*251.15	3		0.01	*240.80	3		0.06
P-n21-k2	*251.00	3		0.01	*234.11	3		0.03
P-n22-k2	*256.03	3		0.01	*242.18	3		0.16
P-n22-k8	*612.68	7		0.33	*591.56	7		0.16
P-n23-k8	*637.14	8		0.01	*626.56	8		0.10
P-n40-k5	*512.74	6		0.01	*495.42	6		0.63
P-n45-k5	558.66	6	0.16%	5	541.74	6	0.20%	5
P-n50-k8	694.27	9	0.52%	5	682.21	9	0.30%	5
P-n50-k10	*811.80	11		0.14	*773.94	11		1.71
P-n51-k10	*919.95	12		0.44	838.92	10	1.05%	5
P-n55-k7	593.35	7	0.80%	5	$\varnothing$			
P-n55-k10	788.62	11	0.69%	5	764.12	11	1.67%	5
P-n55-k15	*1094.15	16		4.78	1079.86	15	0.41%	5
P-n60-k10	$\times$				813.04	11	0.36%	5
P-n60-k15	1130.55	16	0.34%	5	1105.83	15	0.50%	5

[a] *Instance infeasible when $\alpha = 0.85$.* [b] *Exact linear relaxation could not be computed.*

4.2 Results on New Benchmark Instances

In this second round of experiments, we tested the branch-and-price algorithm on new instances generated by applying duration reduction factors on the MRV-16 instances. This effectively changes the maximum duration parameter to a new value $\alpha\overline{T}$ (rounded to the nearest integer), where $\alpha \in \{0.85, 0.95\}$ is the reduction factor, and $\overline{T}$ is the original maximum duration value from the MRV-16 instances. The purpose of these tests was also to provide some initial benchmark solutions for a larger set of instances. A maximum running time of 5 hours was enforced in these runs. The results are presented in Table 2.

Again, we only report results for the instances where the exact linear relaxation (for at least one of the two reduction factors) could be computed. This amounts to 26 out of 36 instances (sets `A` and `P`). The VRPSD-DC becomes less difficult when the maximum duration is shorter, as measured by the number of instances solved to optimality (15 when $\alpha = 0.85$ *vs.* 10 when $\alpha = 0.95$). As usual in branch-and-price, instances with a lower number of customers per route are easier to solve. The largest instance solved to optimality within the running time budget of 5 hours was `P-n55-k15` ($\alpha = 0.85$) with 55 nodes.

4.3 Lower *vs.* Higher Constraint Levels

Depending on the practical scenario, a duration constraint may need to be enforced at a lower (less strict) or higher (more strict) level. The purpose of these tests is to assess the impact of different constraint levels on the optimal solution value, and on the number of routes used in the optimal solution. To this end, we experimented with two additional constraint levels, $\mathcal{P} = 90\%$ and $\mathcal{P} = 98\%$. We selected the MRV-16 instances with a reduction factor $\alpha = 0.85$, since the available reference results for $\mathcal{P} = 95\%$ are better with such a factor (i.e., more confirmed optimal solutions, and smaller optimality gaps in the unsolved instances). The results are presented in Table 3.

On average, enforcing duration constraints at a level of 98% leads to modest increases in

Table 3: MRV-16 instances, $\alpha = 0.85$, constraint levels $\mathcal{P} = 90\%$ and $\mathcal{P} = 98\%$

	$\mathcal{P} = 90\%$				$\mathcal{P} = 95\%$[a]		$\mathcal{P} = 98\%$			
Instance	Best	#	Gap	Time	Best	#	Best	#	Gap	Time
A-n32-k5	*1175.17	7		0.10	*1175.17	7	*1176.37	7		0.04
A-n33-k5	*1027.65	8		0.33	*1027.65	8	*1033.78	8		0.58
A-n33-k6	*822.06	6		1.12	*822.84	7	*823.99	7		0.13
A-n34-k5	*933.56	7		3.18	*933.56	7	951.15	7	0.17%	5
A-n37-k5	723.34	5	0.16%	5	724.78	5	730.57	5	0.19%	5
A-n37-k6	1079.65	6	0.56%	5	1104.72	6	1123.69	6	1.46%	5
A-n38-k5	*812.14	6		2.77	*818.21	6	845.42	6	1.67%	5
A-n39-k6	950.95	6	0.86%	5	951.69	6	960.57	6	0.54%	5
A-n45-k6	1115.34	8	1.52%	5	1124.89	8	1124.89	8	0.22%	5
P-n19-k2	*248.15	3		0.01	*248.15	3	*248.15	3		0.01
P-n20-k2	*251.15	3		0.01	*251.15	3	*251.15	3		0.03
P-n21-k2	*251.00	3		0.01	*251.00	3	*251.00	3		0.01
P-n22-k2	*256.03	3		0.01	*256.03	3	*256.03	3		0.01
P-n22-k8	*612.68	7		0.32	*612.68	7	*616.92	7		0.35
P-n23-k8	*624.95	8		0.01	*637.14	8	*685.03	10		0.02
P-n40-k5	*512.74	6		0.01	*512.74	6	*528.10	6		0.02
P-n45-k5	*558.66	6		4.06	558.66	6	563.44	6	0.31%	5
P-n50-k8	687.08	9	0.43%	5	694.27	9	*705.01	10		2.68
P-n50-k10	*807.61	11		0.42	*811.80	11	*823.11	12		0.06
P-n51-k10	*913.22	12		1.30	*919.95	12	*932.73	12		2.16
P-n55-k7	593.35	7	1.03%	5	593.35	7	601.78	7	2.20%	5
P-n55-k10	766.76	11	0.05%	5	788.62	11	795.16	11	0.25%	5
P-n55-k15	1076.13	15	0.09%	5	*1094.15	16	1130.78	17	0.37%	5
P-n60-k15	*1104.99	15		2.17	1130.55	16	*1140.18	16		1.25

[a] *Results imported from Table 2 for ease of comparison.*

the traveling costs when compared to a level of 90%. Exceptions exist though. For our analysis, we only consider the instances where in both cases ($\mathcal{P} = 90\%$ and $\mathcal{P} = 98\%$) a solution within 1% of the optimal could be found (in total, 20 instances). Among these instances, the average increase in the solution value was only 1.9%. However, in instances `P-n23-k8` and `P-n55-k15`, the solution values increased by 9.6% and 5.1%, respectively. Furthermore, in both cases, additional routes were needed in the best found solutions, which was also the case in 4 other instances. Therefore, assuring with reasonable certainty that routes finish within a limit may require somewhat longer routes on average, and also additional vehicles. This extends the results observed in Erera, Morales, and Savelsbergh (2010) to the case of probabilistic duration constraints and non-identically distributed demands.

We also attempted to solve, with a fixed number of routes, the 6 instances where the number of routes in the best solution for $\mathcal{P} = 98\%$ was larger than the corresponding number of routes for $\mathcal{P} = 90\%$. When the number of routes is fixed (to the value obtained when $\mathcal{P} = 90\%$), instances `P-n23-k8` and `P-n50-k10` become infeasible, and the best solution values in the remaining 4 instances increase only by between 1.1% and 1.5%. This result is in line with what we saw in §4.1 for a fixed number of routes: provided that the instance remains feasible, decreasing the number of routes does not generally considerably impact the traveling costs.

4.4 Over-dispersed *vs.* Under-dispersed Demands

So far, it was assumed in the experiments that the demands follow Poisson distributions, which is a neutrally-dispersed probability distribution (i.e., the variance is equal to the mean). In order to understand the effect of demand variability on the optimal solutions, we performed another round of experiments with under-dispersed and over-dispersed customer demands. In the first case, the demands follow binomial distributions with a variance-to-mean ratio of 0.5, and in the second case, they follow negative binomial distributions with a variance-to-mean ratio of 1.5. In both cases, the mean demand of each customer was kept exactly the same as in the previous (Poisson distributed) cases.

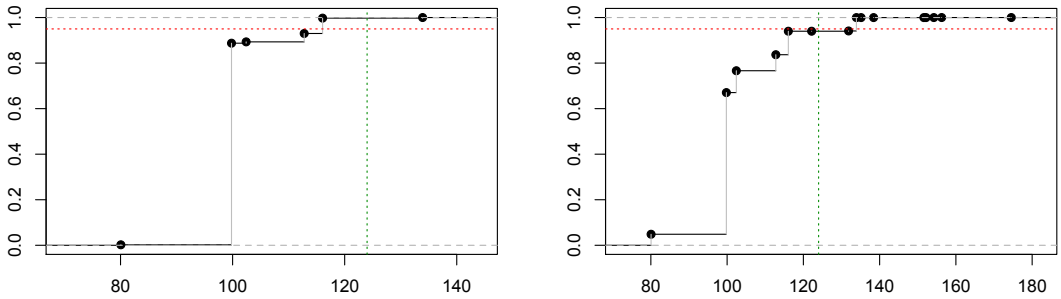

Figure 3: Empirical CDF of the duration of the dashed route from solution 1b *(see Figure 1). On the left, under-dispersed customer demands and, on the right, over-dispersed demands.*

The cumulative distribution of T_r, the duration of an a priori route r, may change considerably, depending on the dispersion of the customers' demands. To illustrate this effect, let r be the dashed route from solution 1b (see Figure 1). Figure 3 outlines the empirical CDF of T_r under both cases: under- and over-dispersed demands (left- and right-hand side plots, respectively), with distributions as indicated above. Note how the tail of the probability distribution of T_r extends much farther when the demands are over-dispersed. Note also that the dashed route would not be feasible (by a small margin, considering $\overline{T} = 124$, and $\mathcal{P} = 95\%$) if the customers' demands were over-dispersed. Interestingly, the mean of T_r does not change much with the dispersion of the demands. Under the three cases (under-, neutral- and over-dispersed demands), the expected value of T_r is, respectively, 101.4, 102.9 and 103.7. On the other hand, the variance of T_r increases from 25.2 (under-dispersed) to 67.4 (neutrally-dispersed) and to 113.2 (over-dispersed).

The results of the experiments with different demand dispersions is presented in Table 4. Again, we restrict the analysis to the instances where the optimality gap was at most 1% (in total, 18 instances). When the demands are over-dispersed, solutions are, on average, 3.4% longer than when demands are under-dispersed. In addition, in 7 instances at least one additional route was employed in the best solution we found for the over-dispersed case. In fact, the effect of increasing the demand dispersion is similar to that of increasing the level of the probabilistic duration constraint, albeit a bit more pronounced. Instances that are sensitive to the variation of the constraint level are also likely to be sensitive to the variation of the demand dispersion. The consequence is the same in both cases: previously feasible a priori routes cease to be feasible, as illustrated in our example from Figure 3. This is a new result in the literature, since the experiments in Erera, Morales, and Savelsbergh (2010) with different demand variabilities (under detour-to-depot restocking and *i.i.d.* demands) could not find a significant impact of the demand variability on the solution cost.

5 Conclusions

In practical applications, it is often required that routes finish within a prescribed time limit. This may be the case, for example, due to companies' policies concerning overtime. We proposed an algorithm for solving the VRPSD with (probabilistic) duration constraints. This problem has previously only been solved by heuristic methods, and under detour-to-depot restocking. When restocking decisions are optimal, enforcing capacity constraints prevents you from finding superior solutions to the problem with regard to the total expected cost and the number of vehicles utilized. Therefore, optimal solutions to the VRPSD-DC may contain routes that serve an expected demand larger than the capacity of the vehicle. These routes actively exploit optimal restocking to reduce traveling costs without violating the maximum time limit.

We formulated the VRPSD-DC as a set-partitioning problem, and solved it with a branch-and-price algorithm. Compared to the VRPSD, the VRPSD-DC is significantly harder to solve. First, because capacity constraints are not available to strengthen the set-partitioning formulation, nor to derive capacity-based label completion bounds. Secondly, because simple label dominance rules are not available. Finally, because checking feasibility

Table 4: MRV-16 instances, $\alpha = 0.85$, under-, neutral- and over-dispersed demands ($\mathcal{P} = 95\%$)

	Under-dispersed demands				*Neutral*[a]		*Over-dispersed demands*			
Instance	Best	#	Gap	Time	Best	#	Best	#	Gap	Time
A-n32-k5	*1172.28	7		0.07	*1175.17	7	*1178.85	7		0.06
A-n33-k5	*1022.95	8		0.45	*1027.65	8	*1035.95	8		0.47
A-n33-k6	*804.99	6		0.45	*822.84	7	*832.68	7		0.19
A-n34-k5	*918.81	6		1.34	*933.56	7	*954.30	7		4.45
A-n37-k5	718.16	5	0.20%	5	724.78	5	735.17	5	0.39%	5
A-n37-k6	1063.98	6	0.92%	5	1104.72	6	1132.17	6	1.50%	5
A-n38-k5	802.45	6	0.34%	5	*818.21	6	850.55	6	2.02%	5
A-n39-k6	943.41	6	1.21%	5	951.69	6	964.10	6	0.55%	5
A-n45-k6	1104.77	7	2.51%	5	1124.89	8	*1130.02	8		3.58
P-n19-k2	*248.00	3		0.01	*248.15	3	*248.51	3		0.01
P-n20-k2	*251.00	3		0.01	*251.15	3	*251.51	3		0.03
P-n21-k2	*251.00	3		0.01	*251.00	3	*251.06	3		0.01
P-n22-k2	*256.00	3		0.01	*256.03	3	*256.20	3		0.01
P-n23-k8	*613.10	8		0.02	*637.14	8	*676.57	10		0.05
P-n40-k5	*512.06	6		0.01	*512.74	6	*528.54	6		0.02
P-n45-k5	*557.19	6		1.60	558.66	6	*561.50	6		3.22
P-n50-k8	675.03	9	0.35%	5	694.27	9	*709.20	10		2.25
P-n50-k10	801.08	11	0.48%	5	*811.80	11	*828.31	12		0.12
P-n51-k10	*893.35	12		0.40	*919.95	12	*942.19	12		4.41
P-n55-k7	589.25	7	1.73%	5	593.35	7	610.45	7	2.90%	5
P-n55-k10	758.20	11	0.48%	5	788.62	11	798.38	11	0.41%	5
P-n55-k15	*1044.57	15		1.56	*1094.15	16	1143.26	17	0.63%	5
P-n60-k15	1077.53	14	0.22%	5	1130.55	16	*1151.90	16		0.95

[a] *Results imported from Table 2 for ease of comparison.*

of a priori routes requires an understanding of the cumulative distribution of their actual durations. Three strategies were employed to verify route feasibility. The first one consisted in the application of Chebyshev's bounds, and was enabled by incorporating into the pricing algorithm procedures for keeping track of the variance of the route duration. The second strategy was based on Monte Carlo sampling and statistical inference. The last strategy comprised a procedure for deriving the probability distribution of the route duration. We also showed how statistical errors can be identified within the branch-and-price framework.

Computational experiments were performed on sets of literature instances. Three different scenarios for the variability of the customer demands were tested: under-dispersed (binomial distributed with a variance-to-mean ratio of 0.5), neutral (Poisson distributed), and over-dispersed (negative binomial distributed with a variance-to-mean ratio of 1.5). In addition, three probabilistic constraint levels were examined: 90%, 95% and 98%. New benchmark results were produced for more than 150 instance configurations, considering all parameter combinations experimented. The results indicate that the effect of over-dispersed demands is similar to that of enforcing a stricter duration constraint: an increase in the expected solution cost and, in many instances, also a need for additional routes. Thus, we extend the results obtained in Erera, Morales, and Savelsbergh (2010) to the case of probabilistic constraints, and derive a new result concerning the negative impact of demand variability on the solution cost.

References

Agresti A, Coull BA, 1998 *Approximate is better than "exact" for interval estimation of binomial proportions. The American Statistician* 52(2):119–126.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH, 1998 *Branch-and-price: Column generation for solving huge integer programs. Oper. Res.* 46(3):316–329.

Brown LD, Cai TT, DasGupta A, 2001 *Interval estimation for a binomial proportion. Statist. Sci.* 16(2):101–117.

Desaulniers G, Desrosiers J, Solomon MM, 2005 *Column generation* (Springer, New York).

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

Erera AL, Morales JC, Savelsbergh MWP, 2010 *The vehicle routing problem with stochastic demand and duration constraints. Transportation Sci.* 44(4):474–492.

Feller W, 2008 *An introduction to probability theory and its applications*, volume 2 (John Wiley & Sons).

Florio AM, Feillet D, Hartl RF, 2018 *The delivery problem: Optimizing hit rates in e-commerce deliveries. Transportation Res. Part B: Methodological* 117:455–472.

Florio AM, Hartl RF, Minner S, 2020 *New exact algorithm for the vehicle routing problem with stochastic demands. Transportation Sci.* (forthcoming).

Gauvin C, Desaulniers G, Gendreau M, 2014 *A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Comput. Oper. Res.* 50:141–153.

Gendreau M, Jabali O, Rei W, 2016 *50th anniversary invited article—future research directions in stochastic vehicle routing. Transportation Sci.* 50(4):1163–1173.

Goodson JC, Ohlmann JW, Thomas BW, 2013 *Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. Oper. Res.* 61(1):138–154.

Goodson JC, Thomas BW, Ohlmann JW, 2016 *Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. Transportation Sci.* 50(2):591–607.

Gunawan A, Lau HC, Vansteenwegen P, 2016 *Orienteering problem: A survey of recent variants, solution approaches and applications. Eur. J. Oper. Res.* 255(2):315–332.

Jabali O, Rei W, Gendreau M, Laporte G, 2014 *Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. Discrete Appl. Math.* 177:121–136.

Laporte G, Martello S, 1990 *The selective travelling salesman problem. Discrete Appl. Math.* 26(2-3):193–207.

Louveaux FV, Salazar-González JJ, 2018 *Exact approach for the vehicle routing problem with stochastic demands and preventive returns. Transportation Sci.* 52(6):1463–1478.

Lübbecke ME, Desrosiers J, 2005 *Selected topics in column generation. Oper. Res.* 53(6):1007–1023.

Mendoza JE, Rousseau LM, Villegas JG, 2016 *A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. J. of Heuristics* 22(4):539–566.

Poggi M, Uchoa E, 2014 *New exact algorithms for the capacitated vehicle routing problem.* Toth P, Vigo D, eds., *Vehicle Routing: Problems, Methods, and Applications*, chapter 3, 59–86, MOS-SIAM Series on Optimization (SIAM).

Salavati-Khoshghalb M, Gendreau M, Jabali O, Rei W, 2019 *An exact algorithm to solve the vehicle routing problem with stochastic demands under an optimal restocking policy. Eur. J. Oper. Res.* 273(1):175–189.

Solomon MM, 1987 *Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research* 35(2):254–265.

Tan KC, Cheong CY, Goh CK, 2007 *Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. Eur. J. Oper. Res.* 177(2):813–839.

Yang WH, Mathur K, Ballou RH, 2000 *Stochastic vehicle routing problem with restocking. Transportation Sci.* 34(1):99–112.

Yee JR, Golden BL, 1980 *A note on determining operating strategies for probabilistic vehicle routing. Naval Res. Logist. Quart.* 27(1):159–163.