Computers & Operations Research 50 (2014) 141–153

Contents lists available at ScienceDirect

Computers & Operations Research

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

A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands

Charles Gauvin [a,b,c,*], Guy Desaulniers [a,b], Michel Gendreau [a,c]

[a] École polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7
[b] GERAD, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[c] CIRRELT, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

ARTICLE INFO

Available online 12 April 2014

Keywords:
Vehicle Routing
Stochastic programming
Logistics
Branch-cut-and-price algorithm

ABSTRACT

This paper proposes a state-of-the-art branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands (VRPSD). We adapt the model of Christiansen and Lysgaard [6] and formulate the VRPSD as a set partitioning model with additional constraints. Feasible routes are generated using a dynamic programming algorithm executed over a state-space graph. Our method combines 2-cycle elimination with *ng*-routes. In addition, our pricing problem is significantly accelerated by the introduction of a new aggregate dominance rule. To speed up the generation of negative reduced cost columns, we use a tabu search heuristic and a bidirectional labeling algorithm. We also add capacity and subset-row inequalities dynamically in order to strengthen the linear relaxation of the master problem. As extensive computational tests illustrate, our algorithm is very competitive with the one of [6]. We solve 20 additional instances from the 40-instance set considered by these authors and we considerably improve the computing times for instances already closed. We also solve 17 new instances from the literature.

1. Introduction

The well-known deterministic vehicle routing problem (VRP) has been widely studied. Given a set of customers with known demands to serve with a fleet of identical vehicles of finite capacity, it consists of determining a set of routes starting and ending at the depot that minimizes the total travel cost.

Although interesting from a theoretical perspective, the deterministic VRP does not correspond to various real-life situations where the demands of customers are not known precisely in advance. This uncertainty gives rise to the vehicle routing problem with stochastic demands (VRPSD) where the demand of each customer can be modeled as a random variable of known probability distribution. In contrast to the VRP, the stochastic nature of this optimization problem makes its definition ambiguous. Consequently, solving the VRPSD requires the use of several mathematical and operational hypotheses (see [15] for more details on stochastic vehicle routing formulations and numerical solution methods).

As proposed in Christiansen and Lysgaard [6], Golden and Stewart [17], Gendreau et al. [14], Hjorring and Holt [18], and Laporte et al. [23], we choose to model the VRPSD as a two-stage stochastic program with simple recourse. With this framework, we assume that it is the *total expected travel cost* that is subject to minimization. We also cast this problem as an *a priori* optimization problem and adopt the first recourse policy formulated by Bertsimas [4].

In this context, the VRPSD consists of the preliminary design of a set of routes covering every customer exactly once. The planned routes are then followed by the vehicles and the demands of the customers are only revealed once they are visited. Since the total demand on a route is also a random variable, it is possible that the total *actual* demand on a given route exceeds the capacity of a vehicle. In this case, we say that a failure occurs and the vehicle returns to the depot to replenish before continuing the planned route. This is the only situation in which split deliveries can occur. We suppose that more than one failure can occur at a given customer on a given route.

If the actual demand of a given customer is exactly equal to the residual capacity of a vehicle, it could be possible to continue directly to the next customer on the planned route (or stay at the depot if the vehicle is at the last customer). However, the increased complexity of the computation of the expected failure cost is not likely to justify the cost savings generated. As in [6], we therefore assume that when the actual cumulative demand, including the actual demand at the given customer, is exactly equal to a multiple of the capacity we simply return to the customer where the failure occurred. In addition, we rule out the possibility of preventive returns or any other form of reoptimization during the routes.

* Corresponding author.

http://dx.doi.org/10.1016/j.cor.2014.03.028
0305-0548/

The VRP is an $\mathcal{NP}$-hard problem and the introduction of randomness increases the intractability of the problem. It is therefore not surprising that the first numerical methods for the VRPSD consisted of heuristics (see [27,17]). These relatively simple techniques based on savings were later complemented by more efficient algorithms such as the tabu search of Gendreau et al. [16]. For further information, see Gendreau et al. [15].

Exact algorithms have also been developed for the problem. Since certain types of VRPSD can be formulated as integer arc flow problems with simple recourse, they are well-suited for the Integer L-shaped method of Laporte and Louveaux [22]. This branch-and-cut algorithm iteratively determines a first-stage solution by using lower bounds on the optimal total expected cost. In order to obtain an integer solution, subtour elimination constraints are dynamically added and branching on arc flow variables is performed. For every non-dominated integer first stage solution, optimality cuts are added to gradually define the true expected cost function. This method was improved by Gendreau et al. [14], Hjorring and Holt [18], and Laporte et al. [23] through the use of stronger optimality cuts and the use of lower bounding functionals based on partial routes at any integer or fractional solution.

Nonetheless, the L-shaped algorithm suffers from important limitations. When many vehicles must be used, many good feasible integer solutions often exist and numerous optimality cuts must be added to solve the problem exactly. Hence, the most competitive algorithms only succeeded to solve instances requiring 4 vehicles or less and having loose capacity constraints.

In comparison, branch-and-price (BP) algorithms have proven to be efficient for the VRP and VRPTW (Vehicle Routing Problem with Time Windows) precisely when the capacity and time window constraints are restrictive (see [2,8]). Effectively, these tighter constraints reduce the solution space of the column generation subproblem and potentially limit the number of feasible routes that need to be considered.

This is the approach used in [6]. These authors consider Poisson distributed demands and use a BP algorithm where the master problem is a set covering problem imposing that each customer be serviced at least once. Routes are dynamically generated by a dynamic programming algorithm executed over a discretized acyclic state-space digraph that associates with each customer a total expected cumulative demand. Their dynamic programming algorithm specifically identifies routes without 2-cycles, i.e., cycles of the form (i,j,i) (see [19]).

Christiansen and Lysgaard [6] also statically add a single capacity cut to strengthen the linear relaxation of the master problem. In order to obtain integer solutions, they branch on the expected cumulative demand at a given customer or a single arc flow variable. Their algorithm manages to solve 18 out of the 40 instances considered with up to 60 customers and 15 vehicles within the allotted time of 20 min.

This work is important because it represents the first application of a BP algorithm to the VRPSD, as far as we know. The computational results also prove that this technique can be competitive and is a good complement to the L-shaped method when many vehicles are considered and capacity constraints are tight.

Nevertheless, it is possible to improve the performance of the algorithm of [6] significantly. In this paper, we propose to do so by using a branch-cut-and-price (BCP) algorithm based on state-of-the-art techniques in column generation. Our method uses the concept of *ng*-routes introduced by Baldacci et al. [3] in the pricing procedure. We combine this technique with 2-cycle elimination and modify the dominance criterion accordingly.

We also use a bidirectional label-setting algorithm based on the work of Righini and Salani [26] and introduce an aggregate dominance rule that significantly increases the number of eliminated partial routes during the labeling algorithm. As in Desaulniers et al. [8], we use a simple tabu search heuristic that rapidly identifies elementary routes to avoid solving the exact *ng*-route subproblem too often.

In order to derive integer solutions more rapidly and strengthen the linear relaxation of the master problem, we add two types of valid inequalities: capacity cuts (CC) (see namely [25]) and subset-row inequalities (SRI) introduced by Jepsen et al. [21]. When the algorithm fails to identify violated cutting planes, we use an efficient branching strategy.

It is clear from our computational results that our algorithm is very competitive with the BP algorithm of [6]. Indeed, we manage to solve 38 out of the 40 instances proposed by [6] within the 20-min time limit and significantly reduce the computing time required to solve the 18 instances already closed by these authors.

The rest of this paper is structured as follows. In Section 2 we present the mathematical model and notation. Section 3 discusses the different procedures of our BCP algorithm and Section 4 reports our computational results. Concluding remarks are given in Section 5.

2. Mathematical model

Before formally describing our model, we state some important hypotheses. We assume that goods are divisible and are all collected (or delivered) along routes. As done by various authors, such as Christiansen and Lysgaard [6] and Laporte et al. [23], we also suppose that routes designed *a priori* must be feasible on average, i.e., the cumulative expected demand must not exceed the vehicle capacity. This assumption is reasonable, because it would be counter-intuitive to deliberately plan routes that fail on average.

This paper also assumes that demands are independent, follow an additive probability distribution and have a positive expected value less or equal to the vehicle capacity. The first and second assumptions are acceptable if the demands are independent and follow well-known distributions such as the Normal or Poisson distributions. The third is intuitively clear as it would not make sense to accept a customer that cannot successfully be serviced on average.

We now formalize the VRPSD by considering $\mathcal{G}=(\mathcal{N}'=\mathcal{N}\cup\{o,o'\},\mathcal{A})$, a digraph where the set $\mathcal{N}$ represents the n customers, while o and o' represent the central depot and its copy, respectively. $\mathcal{A}=\{(i,j): i,j\in\mathcal{N}; i\neq j\}\cup\{(o,j): j\in\mathcal{N}\}\cup\{(j,o'): j\in\mathcal{N}\}$ is the arc set. c_{ij} denotes the travel cost from $i\in\mathcal{N}'$ to $j\in\mathcal{N}'$. $\mathcal{V}$ represents the set of identical vehicles of capacity Q. We also let ξ_i be the random variable of expected value $\mathrm{E}[\xi_i]$ and variance $\mathrm{Var}[\xi_i]$ representing the demand of customer i. For simplicity, we may consider that the depot has zero demand.

A route is defined as a path of the form $p=(i_1,\ldots,i_{|p|})$ where $i_1=o$ and $i_{|p|}=o'$ with $i_h\in\mathcal{N}$ for $h\in\{2,\ldots,|p|-1\}$. Given such a route, we let $\Psi(\mu_{i_h},\sigma^2_{i_h})=\sum_{l=1}^{h}\xi_{i_l}$ denote the random variable indicating the total *actual* cumulative demand at customer i_h for $h\in\{2,\ldots,|p|-1\}$ with expected value μ_{i_h} and variance $\sigma^2_{i_h}$. Because of the hypothesis on the distributions, we have $\mu_{i_h}=\sum_{l=1}^{h}\mathrm{E}[\xi_{i_l}]$ and $\sigma^2_{i_h}=\sum_{l=1}^{h}\mathrm{Var}[\xi_{i_l}]$. A route cycles if $i_m=i_l$ for $m,l\in\{2,\ldots,|p|-1\}, m\neq l$ and a route without cycles is elementary. Finally, $\mathcal{N}_p$ denotes the set of all customers visited by the route p.

It can easily be seen that failures are separable by vehicle and that all vehicles are identical. Given a route $p=(i_1,\ldots,i_{|p|})$, we let $\mathrm{EFC}_{i_h}(\mu_{i_h},\sigma^2_{i_h})$ denote the expected failure cost at customer i_h given a cumulative expected demand and variance of μ_{i_h} and $\sigma^2_{i_h}$, respectively. As in [6] and [23], we can then write

$$\mathrm{EFC}_{i_h}(\mu_{i_h},\sigma^2_{i_h})=2c_{oi_h}\sum_{u=1}^{\infty}\left(\mathrm{P}\{\Psi(\mu_{i_{h-1}},\sigma^2_{i_{h-1}})\leq uQ\}-\mathrm{P}\{\Psi(\mu_{i_h},\sigma^2_{i_h})\leq uQ\}\right) \quad (1)$$

$\mathrm{P}\{E\}$ denotes the probability that event E occurs. $\mathrm{P}\{\Psi(\mu_{i_{h-1}},\sigma^2_{i_{h-1}})\leq uQ\}-\mathrm{P}\{\Psi(\mu_{i_h},\sigma^2_{i_h})\leq uQ\}$ can therefore be interpreted as

the probability of having the uth failure at i_h given that it has not occurred on any previously visited customer along the route. Note that the probability of having a failure at the depot is equal to 0. In practice, the convergent infinite series is approximated by summing until the incremental change is smaller than 2^{-53}, the standard double precision on a 64-bit machine.

We can now calculate $\hat{c}_p$, the total expected cost of any route as the sum of the deterministic travel cost and the expected failure cost at every visited customer:

$$\hat{c}_p = \sum_{h=1}^{|p|-1} (c_{i_h i_{h+1}} + \mathrm{EFC}_{i_{h+1}}(\mu_{i_{h+1}}, \sigma^2_{i_{h+1}})) \tag{2}$$

It then becomes easy to formulate the VRPSD as a route-based integer set partitioning master problem:

$$\min_{\lambda} \quad \sum_{p \in \mathcal{P}} \hat{c}_p \lambda_p \tag{3}$$

$$(IMP) \quad \text{s.t.} \quad \sum_{p \in \mathcal{P}} \alpha_{ip} \lambda_p = 1, \quad \forall i \in \mathcal{N} \tag{4}$$

$$\sum_{p \in \mathcal{P}} \lambda_p \geq \mathcal{V}(\mathcal{N}) \tag{5}$$

$$\lambda_p \in \{0,1\}, \quad \forall p \in \mathcal{P} \tag{6}$$

In this model, α_{ip} is a binary parameter equal to 1 if route p visits customer i and 0 otherwise. λ_p is a binary decision variable indicating whether or not route p is chosen. $\mathcal{P}$ is the set of all routes starting and ending at the depot that are feasible on average. $\mathcal{V}(\mathcal{N}) = \lceil Q^{-1} \sum_{i \in \mathcal{N}} \mathrm{E}[\xi_i] \rceil$ is an easily computable lower bound on the total number of vehicles required to service all customers.

The objective function (3) minimizes the total expected cost of selected routes. Constraints (4) ensure that every customer is visited by exactly one route while (5) imposes that the solution uses at least $\mathcal{V}(\mathcal{N})$ vehicles. The latter constraint is not necessary, but strengthens the linear relaxation of *IMP*. Finally, (6) enforces binary requirement on the route variables.

3. BCP algorithm

3.1. Main scheme

We solve the VRPSD by means of a BCP algorithm. This algorithm is a branch-and-bound procedure where lower bounds on the optimal value are computed by column generation and cutting planes are dynamically added to strengthen *MP* or master problem, the linear relaxation of *IMP*, and accelerate the derivation of integer solutions (see [8]). The column generation procedure iteratively solves a restricted master problem *RMP* that only considers a subset of all feasible routes. *RMP* is initialized by taking $|\mathcal{N}|$ single customer routes. New routes are then dynamically generated by solving a subproblem whose optimization is guided by the value of the dual variables associated with the optimal solution of *RMP*. This procedure is repeated iteratively until no more negative reduced cost columns (routes) exist and the solution is optimal. If this solution is integer, then it is also optimal for the original *MP*. Otherwise, we try to identify violated valid inequalities. If we fail to identify such inequalities or the violation is too small with respect to a pre-specified threshold, then we resort to branching.

BCP algorithms have successfully been applied to a wide range of problems, namely in the field of transportation (see [7] for applications of column generation). As the work of [6] suggests, they can also be efficient to solve the VRPSD.

3.2. Column generation subproblem

The extremely large cardinality of the set $\mathcal{P}$ precludes its exhaustive enumeration. However, in order to guarantee optimality, it is only required to consider routes of negative reduced cost with respect to the optimal solution of *RMP* at the current column generation iteration. These routes can be dynamically generated by solving the following shortest-path problem with resource constraints (SPPRC):

$$(SP) \quad \min_{p \in \mathcal{P}} \bar{c}_p \tag{7}$$

where $\bar{c}_p = \sum_{(i,j) \in p} (\hat{c}_{ij} - \pi_j) - \varphi$ is the (expected) reduced cost of route p. π_j and φ are the dual variables associated with the jth constraint (4) and constraint (5), respectively, at the current iteration. We set $\pi_{o'} = 0$. As we can see, the total expected reduced cost of a route can be decomposed according to the arcs followed.

In order to generate routes with negative reduced cost, one must be able to compute $\hat{c}_p$, the total *expected* cost of a route. This requires knowledge of the total expected cumulative demand and variance at every customer i_h as well as the order in which customers are visited. This additional information is not directly available in the formulation of our problem. As in [6], we therefore introduce $\mathcal{GS} = (\mathcal{NS}' = \mathcal{NS} \cup \{os, os'\}, \mathcal{AS})$, a state-space digraph associating an expected cumulative demand and a cumulative variance with each customer. The nodes os and os' represent the depot and its copy and play the same role as o and o' in the initial graph. The subset $\mathcal{NS}$ is given by: $\mathcal{NS} = \{\mathcal{V}(\mu_i, \sigma_i^2, i) : i \in \mathcal{N}; \mu_i \in [\mathrm{E}[\xi_i], Q]; \sigma_i^2 \in [\mathrm{Var}[\xi_i], V_{max}]\}$.

V_{max} represents an upper bound on the total variance of any elementary feasible route. As in [6], this bound can be obtained by solving a maximum knapsack problem where the capacity of the knapsack is given by Q, the weights of the objects correspond to the expected value of the demands and the profit is their variance.

Furthermore, the arc set $\mathcal{AS} = \bigcup_{i=1}^{3} \mathcal{AS}_i$ can be expressed as the following partition:

- $\mathcal{AS}_1 = \{(\mathcal{V}(\mu_i, \sigma_i^2, i), \mathcal{V}(\mu_i + \mathrm{E}[\xi_j], \sigma_i^2 + \mathrm{Var}[\xi_j], j)) : \mathcal{V}(\mu_i, \sigma_i^2, i), \mathcal{V}(\mu_i + \mathrm{E}[\xi_j], \sigma_i^2 + \mathrm{Var}[\xi_j], j) \in \mathcal{NS}\}$
- $\mathcal{AS}_2 = \{(os, \mathcal{V}(\mathrm{E}[\xi_i], \mathrm{Var}[\xi_i], i)) : \mathcal{V}(\mathrm{E}[\xi_i], \mathrm{Var}[\xi_i], i) \in \mathcal{NS}\}$
- $\mathcal{AS}_3 = \{(\mathcal{V}(\mu_i, \sigma_i^2, i), os') : \mathcal{V}(\mu_i, \sigma_i^2, i) \in \mathcal{NS}\}$.

Fig. 1 illustrates the case where expected demands are assumed to be integers.

With the use of $\mathcal{GS}$, it is possible to statically embed the total expected failure cost at a customer on the arcs of the graph. This is a huge advantage compared to a dynamic calculation of expected failure cost as is done in [23,14,18]. However, the main drawback is the necessity to discretize and assume integrality of the expected cumulative demand and variance at any customer.

In order to solve the column generation subproblem exactly and identify routes of negative reduced cost if some exist, we execute a dynamic programming labeling algorithm on the graph $\mathcal{GS}$ (see [20] for a review of labeling algorithms for column generation). This algorithm is similar to the Bellman–Ford algorithm for shortest paths in acyclic graphs with a modified dominance procedure and where the cost of the arcs is given by their (expected) reduced cost.

In labeling algorithms, partial paths from the source os to a node $\overline{v} \in \mathcal{NS} \backslash \{os\}$ are successively extended using the arcs leaving node $\overline{v}$. Each path is represented by a label $L = (\overline{c}, \overline{v}, \mathbf{r})$ where $\overline{c}(L)$ is the reduced cost of the partial path, $\overline{v}(L)$ is its last node and $\mathbf{r}(L) = (r^1(L), \ldots, r^{|\mathcal{R}|}(L))$ is the vector indicating the consumption of the $|\mathcal{R}|$ resources. In our particular problem, $\mathcal{R} = \{\mathrm{dem}, \mathrm{var}, \mathrm{visit}_1, \ldots, \mathrm{visit}_n\}$. $r^{dem}(L)$ and $r^{var}(L)$ keep track of the total expected cumulative demand and variance, respectively, while each component

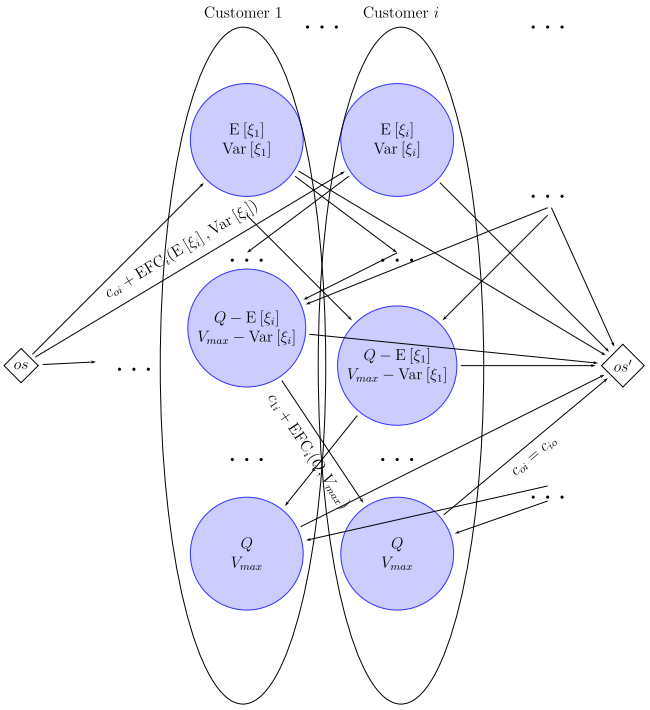

Fig. 1. Extract of the state-space graph $\mathcal{GS}$.

$r^{\mathrm{visit}_i}(L)$ is equal to 1 if a customer i cannot be visited in an extension of label L and 0 otherwise (see [12]).

Each node in the state-space graph gives information on the total expected cumulative demand and cumulative variance of any partial route. However, as will be seen in later sections, it is still useful to consider explicitly $r^{dem}(L)$ and $r^{var}(L)$.

According to our definition of resource visits, we set $r^{\mathrm{visit}_h}(L) = 1$ if extending the path associated with label L to the *customer* h would result in a new path whose cumulative expected demand exceeds the vehicle capacity. Since the graph $\mathcal{GS}$ only considers feasible paths, this is done by checking the existence of the arc $(\mathscr{V}(\mu_i, \sigma_i^2, i), \mathscr{V}(\mu_i + \mathrm{E}[\xi_h], \sigma_i^2 + \mathrm{Var}[\xi_h], h))$. It is possible to perform this operation, because the expected demand of any customer is strictly larger than 0. Hence, the cumulative expected demand resource respects the triangle inequality.

3.3. ng-route

Constraints (4) indicate that a route can visit a customer *at most once* in any optimal solution to *IMP*. It is therefore possible to strengthen this formulation and consider the set of *elementary* feasible routes. Considering such routes requires an ESPPRC subproblem. Nonetheless, this problem is strongly $\mathcal{NP}$-hard (see [11]) and solving it repetitively can be prohibitive for large instances. As preliminary tests have shown, the stronger bound provided by the ESPPRC is not sufficient to compensate for its prohibitive computation times.

However, it is possible to consider a relaxed version of the ESPPRC that is still stronger than the basic SPPRC. This pricing method informally helps eliminate small cycles and is based on the concept of *ng*-routes introduced by Baldacci et al. [3]. It consists of defining for each customer $i \in \mathcal{N}$ a set $\mathcal{NG}_i$ of cardinality $\Delta(\mathcal{NG}_i)$ such that $i \in \mathcal{NG}_i$. When working with the state-space graph, we simply set $\mathcal{NG}_u = \mathcal{NG}_i, \forall u \in C(i)$ where $C(i)$ is the set of nodes in $\mathcal{GS}$ associated with customer i.

For our particular implementation, experimentations reveal that it is preferable to consider sets composed of the $\Delta(\mathcal{NG}_i) - 1$ closest customers in expected distance (as well as i itself) and fix $\Delta(\mathcal{NG}_i) = 10$. For any partial route $p = (i_1, \ldots, i_k)$ in $\mathcal{G}$ with $o = i_1$ and $i_k \neq o'$ we then forbid the extension of p to any node in the set $\Pi(p) = \{i_r : i_r \in \cap_{s=r+1}^{k} \mathcal{N}_{i_s}, r = 1, \ldots, k-1\} \cup \{i_k\}$ by properly modifying the resource visits associated with given customers.

For example, consider an instance with $\mathcal{N} = \{1, 2, 3, 4, 5\}$, source o and sink o'. We let $\Delta(\mathcal{NG}_i) = 3, \forall i \in \mathcal{N}$ and we consider the following neighborhoods: $\mathcal{NG}_1 = \{1, 2, 5\}$, $\mathcal{NG}_2 = \{1, 2, 5\}$, $\mathcal{NG}_3 = \{1, 3, 4\}$, $\mathcal{NG}_4 = \{3, 4, 5\}$, $\mathcal{NG}_5 = \{1, 2, 5\}$. Then it follows that the partial path $p_2 = (o, 1, 2, 3, 2)$ in $\mathcal{G}$ is a *feasible* *ng*-route while $p_1 = (o, 1, 2, 3, 1)$ is *not* since $\Pi(o, 1, 2, 3) = \{1, 3\}$.

However, we notice that $p_2 = (o, 1, 2, 3, 2)$ is not feasible with respect to 2-cycle elimination. This can be explained by the fact that *ng*-routes cannot guarantee the elimination of cycles of fixed length. We therefore find it desirable to combine 2-cycle elimination with *ng*-routes. Although authors such as [3] do not consider them useful, numerical experiments illustrate that the strengthening of *MP* they provide outweighs their additional computational complexity. 2-cycle elimination is performed by adding the field *pred* to the labels. This new datum represents the predecessor node ($pred(L) \in \mathcal{NS}$) and allows identification of the corresponding customer ($Customer(pred(L)) \in \mathcal{N}$).

In order to obtain an efficient labeling algorithm, it is essential to devise a dominance criterion that will eliminate unnecessary partial paths (labels that cannot lead to an optimal solution) and is compatible with *ng*-routes and 2-cycle elimination. We first consider a standard dominance rule. A label L_3 can be eliminated if there exist labels L_1 and L_2 such that $Customer(pred(L_3)) = h, Customer(pred(L_2)) = i, Customer(pred(L_1)) = j$ and the following conditions hold:

Condition 1. Initial dominance criterion with *ng*-routes, 2-cycle elimination and resource visits

$$\overline{v}(L_1) = \overline{v}(L_2) = \overline{v}(L_3) \tag{8}$$

$$\overline{c}(L_1) \leq \overline{c}(L_3) \quad \text{and} \quad \overline{c}(L_2) \leq \overline{c}(L_3) \tag{9}$$

$$r^s(L_1) \leq r^s(L_3), \quad \forall s \in \mathcal{R} \backslash \{visit_j\} \text{ and}$$
$$r^s(L_2) \leq r^s(L_3), \quad \forall s \in \mathcal{R} \backslash \{visit_i\} \tag{10}$$

The validity of this dominance criterion is straightforward. Indeed, it is a simple extension of the basic 2-cycle elimination for the case when resource visits are considered. Of course, we can also eliminate a label L_3 if there exists another label L_1 associated with the same end node in $\mathcal{NS}$ that has smaller or equal resource consumption for all resources, as well as a smaller or equal total reduced cost.

3.4. Aggregate dominance

It is possible to strengthen the dominance rule stated in Section 3.3 and eliminate more labels by exploiting the structure of the state-space graph. Indeed, condition (8) imposes that labels can be checked for dominance only if they are associated with the same end node in $\mathcal{GS}$. This condition is stringent as it implies that two partial paths $p_1 = (os = i_1, \ldots, i_{|p_1|})$ and $p_2 = (os = j_1, \ldots, j_{|p_2|})$ associated with labels L_1 and L_2 can be compared only if $\overline{v}(L_1) = \mathscr{V}(\mu_{i_{|p_1|}}, \sigma^2_{i_{|p_1|}}, i_{|p_1|}) = \overline{v}(L_2) = \mathscr{V}(\mu_{j_{|p_2|}}, \sigma^2_{j_{|p_2|}}, j_{|p_2|})$, which implies that the two paths have *exactly* the same expected cumulative demands and variances at a given customer.

However, we know that all paths whose last node belongs to a set $C(i)$ are comparable because they correspond to paths with the same end node i in $\mathcal{G}$ having *different* expected cumulative demands and variances. We therefore change condition (8) to

$Customer(\overline{v}(L_1)) = Customer(\overline{v}(L_2)) = Customer(\overline{v}(L_3))$. We refer to this new criterion as the aggregate dominance rule.

Proposition 1. *The aggregate dominance rule is a valid dominance criterion when Poisson distributed demands are considered.*

Proof. Consider L_1, L_2 and L_3, any three labels associated with a common customer i such that $\overline{v}(L_1) = \mathcal{V}(\mu_1, \sigma_1^2, i)$, $\overline{v}(L_2) = \mathcal{V}(\mu_2, \sigma_2^2, i)$; and $\overline{v}(L_3) = \mathcal{V}(\mu_3, \sigma_3^2, i)$ with $\mu_1 \leq \mu_3$ and $\mu_2 \leq \mu_3$. If these labels respect (9) and (10), we know that L_3 has resource consumption greater than or equal to L_1 and L_2 for all resources except perhaps the resource visit of the respective predecessor customer. By definition of the state graph $\mathcal{GS}$ we therefore know that the set of customers reachable from L_3 lies within the union of the set of all customers reachable from L_1 or L_2.

Next, consider two different partial routes that we extend to the same set of customers in exactly the same sequence. The total travel cost of visiting this set of customers can differ only with regards to the total expected failure cost since the deterministic travel cost will be the same for both routes.

Hence, we only need to prove that the total expected failure cost of any route is a non-decreasing function of the total cumulative expected demand and variance associated with a given node $\mathcal{V}(\mu_i, \sigma_i^2, i) \in \mathcal{NS}$. We first observe that by considering Poisson distributed demands, we have $\mu_i = \sigma_i^2$ for every node in $\mathcal{NS}$ and we only need to consider the effect of the cumulative expected demand on the expected failure cost. We prove in Appendix A that the total expected failure cost at any customer $i \in \mathcal{N}$ is a non-decreasing function of μ_i, the expected cumulative demand at that customer. □

It is important to note that this rule only applies when $\mu_1 \leq \mu_3$ and $\mu_2 \leq \mu_3$ hold if and only if $r^{dem}(L_1) \leq r^{dem}(L_3)$ and $r^{dem}(L_2) \leq r^{dem}(L_3)$ (considering Poisson demands). In other words, a label associated with a node with smaller expected demand and variance must also have smaller resource consumption for demand and variance. As we will see in Section 3.5, this is not always the case. In addition, all arcs initially in $\mathcal{AS}$ between any pair of customers must still be present to guarantee exactness. We discuss this in Section 3.8.

3.5. Bidirectional algorithm

To speed up the solution of the subproblem, we use a bidirectional labeling algorithm inspired by the work of Righini and Salani [26]. This algorithm is based on a forward and a backward procedure that gradually extend paths until a "halfway" point is reached. Forward and backward paths are then combined and checked for feasibility and negativity of the reduced cost.

More specifically, our forward phase extends a non-dominated label L from the source to the sink as long as $\mu_i < Q/2$ where $\overline{v}(L) = \mathcal{V}(\mu_i, \sigma_i^2, i)$. The backward algorithm extends a label from the sink to the source until $\mu_i \geq Q/2$. This assures us that all non-dominated paths from os to os' are considered.

It is important to note that for a partial *forward* path $p = (i_0, \ldots, i_h)$ with $os = i_0$ associated with label L where $\overline{v}(L) = \mathcal{V}(\mu_{i_h}, \sigma_{i_h}^2, i_h)$, the equalities $r^{dem}(L) = \mu_{i_h}$ and $r^{var}(L) = \sigma_{i_h}^2$ always hold. Indeed, we have $\mu_{i_h} = \sum_{k=0}^{h} \mathrm{E}[\xi_{i_k}] = r^{dem}(L)$ and $\sigma_{i_h}^2 = \sum_{k=0}^{h} \mathrm{Var}[\xi_{i_k}] = r^{var}(L)$. However, they may not hold for backward labels. For instance, consider the simple backward path $p' = (os', j_1)$ where $j_1 = \mathcal{V}(\mu_{j_1}, \sigma_{j_1}^2, j_1)$ and $\mu_{j_1} > \mathrm{E}[\xi_{j_1}]$. This path represents a feasible route since $r^{dem}(L) = \mathrm{E}[\xi_{j_1}] < Q$ always holds.

As such, we cannot apply the aggregate dominance rule presented in Section 3.4 to backward paths. Contrary to the forward case, a backward label L_1 that has visited only a subset of the customers visited by L_2 and has smaller expected cumulative demand may *not necessarily* be extended to all the customers to which L_2 can. Moreover, even if L_1 and L_2 share the same set of feasible extensions, it is not guaranteed that extensions of L_1 will have a lower cost, because it is possible that $\mu_1 > \mu_2$ or $\sigma_1^2 > \sigma_2^2$ even if $r^{dem}(L_1) \leq r^{dem}(L_2)$ and $r^{var}(L_1) \leq r^{var}(L_2)$ and the expected failure cost at any customer is increasing in μ and σ^2 (where $\overline{v}(L_1) = \mathcal{V}(\mu_1, \sigma_1^2, i)$ and $\overline{v}(L_2) = \mathcal{V}(\mu_2, \sigma_2^2, i)$).

3.6. Tabu search

Although solving the *ng*-route subproblem is much faster than solving the ESPPRC, it can still be relatively slow and must be carried out numerous times at various nodes in the branch-and-bound tree. To avoid excessive computing times, we therefore introduce a tabu search heuristic that can rapidly generate negative reduced cost columns.

Tabu search is a popular metaheuristic that uses simple operators to explore the solution space. At each iteration it considers a set of solutions that are in the neighborhood of the current solution and chooses the best according to a function to optimize. It attempts to avoid getting trapped in local minima by maintaining a list of tabu moves that are forbidden for a certain number of iterations.

Our algorithm is very similar to the one presented in Desaulniers et al. [8]. At each iteration, it considers an initial route in $\mathcal{GS}$ and determines the least-cost neighbor (route) that can be obtained by the application of non-tabu operators. The algorithm then replaces the current solution with this best solution, even if this leads to a deterioration of the objective function. It then makes the inverse move tabu for 10 iterations. We specifically consider sequential insertion and deletion operators. Deletion consists of eliminating each customer in $\mathcal{N}_p$, the set of customers visited by route p. On the other hand, the insertion procedure tries to insert each customer in $\mathcal{N} \setminus \mathcal{N}_p$ at every possible location in p. The new route is checked for feasibility with respect to expected cumulative demand. Both operators require testing if a route still respects the branching decisions currently imposed (see Section 3.8). Routes are also checked for 2-cycle elimination when applying the deletion operator, since initial routes may visit a customer more than once.

Contrary to the algorithm of [8] that does not deal with expected failure costs, it is necessary to recalculate the cost of the "tail" of the new path each time an operator is applied. Indeed, to obtain the exact total expected cost of a new route, the algorithm must compute the new expected failure costs for all customers from the point of insertion or deletion until the end. This procedure can be relatively time consuming.

However, it is possible to derive lower bounds on the true savings that can occur when removing a customer that do not require these computations. For instance, if we consider the model defined by (3)–(6), then $\overline{c}_p - (c_{i_{k-1},i_k} + c_{i_k,i_{k+1}} - c_{i_{k-1},i_{k+1}} - \pi_{i_k})$ represents an upper bound on the true reduced cost of the route p' that is obtained by removing customer i_k from p. Note that in our final algorithm, we must adjust this bound by taking into account the dual variables associated with the cutting planes (described in Section 3.7). This bound holds for all distributions that ensure that the total expected failure cost at any customer is a nondecreasing function of the cumulative demand of the route at that customer. As described in Appendix A, this is the case for the Poisson distribution.

In order to provide greater diversification, we use a multi-start procedure. More specifically, the algorithm starts by considering an initial path corresponding to a column in the basis of the current optimal solution of *RMP*. After reaching a maximum number of iterations, the algorithm considers a new basic variable. These variables are good candidates, because they have zero reduced cost.

The algorithm is carried out until no more negative reduced cost columns can be identified or until a pre-specified maximum number of columns have been identified. If negative reduced cost columns were found, we execute the tabu search once more. In the other case, we use the exact labeling algorithm. If the exact methods fails to identify negative reduced cost columns, then our algorithm terminates with an exact solution. If this is not the case, then the tabu search algorithm is executed again at the next column generation iteration.

3.7. Cutting planes

Cutting planes are inequalities that are valid for any integer solution, but that may be violated by a fractional solution. In our formulation of the VRPSD, these inequalities can take two forms. If they can be expressed as linear combinations of arc flow variables, they can be introduced directly in *IMP*. This type of inequalities can easily be taken into account by the column generation procedure (see [13]).

Lysgaard et al. [25] have developed a rich group of separation procedures for various inequalities of this type for the VRP that can also be applied to the VRPSD. We performed various tests with these inequalities, but chose to retain only the capacity cuts (CC). Capacity cuts impose a lower bound on the total number of vehicles required to service a given subset of customers based on the total demand of these customers. These inequalities can be expressed in the *IMP* as

$$\sum_{p \in \mathcal{P}} \beta_p^f \lambda_p \geq \mathcal{V}(S^f), \quad \forall f \in \mathcal{H} \tag{11}$$

where $\mathcal{H}$ represents the set of CC that were identified and added to *IMP* at the current iteration of the column generation algorithm while S^f represents the customer set considered by the fth CC. This set induces the following "ingoing" cutset: $\delta^-(S^f) = \{(i,j) \in \mathcal{A} : i \notin S^f, j \in S^f\}$. We can then write $\beta_p^f = \sum_{(i,j) \in \delta^-(S^f)} \beta_{ijp}$ with $\beta_{ijp} = 1$ if route p uses the arc (i,j) in the cutset and 0 otherwise. $\mathcal{V}(S^f) = \lceil Q^{-1} \sum_{i \in S^f} E[\xi_i] \rceil$ is an easily computable lower bound on the total number of vehicles required to service the customers in the set $S^f \subseteq \mathcal{N}$. CC are identified heuristically using the separation package CVRPSEP [24]. All separation routines are executed on the *original* graph $\mathcal{G}$.

On the other hand, it is also possible to use valid inequalities that can not be expressed as linear combinations of arc flow variables. We namely consider the subset-row inequalities (SRI) introduced by Jepsen et al. [21]. These rank 1 Chvátal-Gomory inequalities can be written as

$$\sum_{p \in \mathcal{P}} \left\lfloor \frac{1}{\kappa} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{\kappa} \right\rfloor \tag{12}$$

where $0 < \kappa \leq |S|$ and $S \subseteq \mathcal{N}$. We explicitly consider SRI with $\kappa = 2$ and $|S| = 3$. This corresponds to forbidding the coexistence of routes that cover at least 2 customers out of any triplet of customers and is therefore a special form of clique inequalities. This set of constraints can be written in *IMP* as:

$$\sum_{p \in \mathcal{P}} \gamma_p^g \lambda_p \leq 1, \quad \forall g \in \mathcal{G} \tag{13}$$

$\mathcal{G}$ represents the set of all SRI that are identified and introduced in *IMP* at the current column generation iteration while S_g corresponds to the triplet defining the gth SRI and γ_p^g is equal to 1 if p visits at least 2 customers in S_g and 0 otherwise.

As in Desaulniers et al. [8] and Jepsen et al. [21], we identify violated SRI exactly. That is we enumerate the collection of all triplets of customers and check every route p with $\lambda_p \in (0,1)$ in the current basis of *IMP*. This is done in $O(n^3|B|)$ where $|B|$ is the number of routes in the basis. The use of SRI requires introducing additional resources to keep track of the contribution of the dual variables to the total route reduced cost. This makes it considerably more difficult to eliminate partial routes using the dominance criterion introduced previously. This difficulty increases with the number of active SRI in *IMP*. Nonetheless, the considerable strengthening of *IMP* justifies this added complexity. The reduced cost of a route in $\mathcal{G}$ can now be expressed as:

$$\bar{c}(p) = \sum_{(i,j) \in p} \left(\hat{c}_{ij} - \pi_j - \sum_{f \in \mathcal{H}} \beta_{ijp}^f \vartheta^f \right) - \varphi - \sum_{g \in \mathcal{G}} \gamma_p^g \nu^g \tag{14}$$

where $\vartheta^f \geq 0$ and $\nu^g \leq 0$ are respectively the values of dual variables associated with the fth CC and gth SRI at the current column generation iteration.

3.8. Branching strategy

If we fail to identify violated CC or SRI, we resort to dichotomic branching in order to derive integer solutions. Our branching strategy consists of evaluating a set of decision rules and selecting the best according to a fixed normalized criterion. We specifically consider 2 competing decision rules: branching on a sequence of two customers and branching on the number of arcs adjacent to a subset of customers. The latter branching decision was introduced by Augerat [1] and used by various authors such as Jepsen et al. [21], Fukasawa et al. [13] and Lysgaard et al. [25].

In order to explain branching on a sequence of customers, we introduce the following notation. Let $\bar{J} \subseteq \mathcal{N}$ be the set of customers visited by more than one route in a fractional solution. Branching on inter-customers consists of identifying $j \in \bar{J}$ and its predecessor $i(j) \in \mathcal{N}$, such that

$$0 < \sum_{p \in \mathcal{P}} \lambda_p \alpha_{i(j),j}^p < 1 \tag{15}$$

where $\alpha_{i(j),j}^p$ is a parameter equal to 1 if route p visits j immediately after $i(j)$ in $\mathcal{G}$ and 0 otherwise. In the first branch-and-bound node, we then impose that $i(j)$ be followed by j by eliminating all arcs in $\mathcal{GS}$ from $i(j)$ to every customer except j. We forbid that sequence in the other node by removing all arcs in $\mathcal{GS}$ from $i(j)$ to j. This branching strategy is therefore equivalent to branching on arcs in the original graph $\mathcal{G}$. If all routes that visit j also visit $i(j)$ beforehand, then the sequence $(i(j),j)$ is not a candidate for branching. This branching rule makes the current fractional solution infeasible in both branching nodes. To evaluate the quality of the decision, we use the metric $2 \times \min\{\sum_{p \in \mathcal{P}} \lambda_p \alpha_{i(j),j}^p, 1 - \sum_{p \in \mathcal{P}} \lambda_p \alpha_{i(j),j}^p\}$.

We also consider the type of branching decisions used in [1]. This consists of identifying a set of customers S such that

$$2 < \sum_{p \in \mathcal{P}} \sum_{(i,j) \in \delta(S)} \lambda_p \alpha_{i,j}^p < 4 \tag{16}$$

where $\alpha_{i,j}^p$ indicates if route p visits the arc (i,j) and $\delta(S) = \{(i,j) \in \mathcal{A} : i \in S, j \notin S \text{ or } i \notin S, j \in S\}$ represents the cutset induced by S. Once such a set has been identified, we introduce the constraint $\sum_{p \in \mathcal{P}} \nu_{Sp} \lambda_p \geq 4$ in the *IMP* of the first node and $\sum_{p \in \mathcal{P}} \nu_{Sp} \lambda_p = 2$ in the second (where ν_{Sp} indicates the number of arcs of p incident to S). We evaluate the quality of the decision by computing $1 - |\sum_{(i,j) \in \delta(S)} \sum_{p \in \mathcal{P}} \lambda_p \alpha_{i,j}^p - 3|$.

We then use the branching decision that has the higher metric. Since all scores are contained in the closed interval $[0,1]$, it is easy to compare the two branching methods. Although this requires an additional amount of overhead compared to a traditional branching scheme based on a single type of decision, computational results have demonstrated the value of this flexibility. Indeed, certain instances become unsolvable within the preestablished time limit when branching on a cutset is not used.

We do not use the first type of branching decision of [6] because it is incompatible with our aggregate dominance rule. Indeed, this rule consists of identifying a customer i visited more than once in a fractional solution and imposing $\mu_i \geq t$ in one node and $\mu_i < t$ in the other for a given $t \in \{E[\xi_i]+1, \ldots, Q\}$. This rule is not valid for our method, because it only eliminates a subset of the arcs between customer i and the other customers in $\mathcal{GS}$. Similarly, we cannot branch on a single arc-flow variable in $\mathcal{GS}$ (though it is possible to branch on arcs in $\mathcal{G}$ since this is equivalent to eliminating all arcs between two customers in $\mathcal{GS}$).

4. Computational results

As in [6], we perform all our tests using a Poisson distribution. This considerably simplifies computations because this distribution has the property that the expected value is equal to its variance. Thus, we can drop the parameter σ_i^2 and every node in the state graph takes the form $\mathcal{V}(\mu_i, i)$. For our initial tests, we use the same instance set considered by [6]: a subset of instances A, E and P obtained from http://branchandcut.org with up to 60 customers. We also complement these with the remaining A, P and E instances as well as the entire set of B, F, G and M instances. We assume that the demand of any customer i is Poisson distributed with parameter $E[\xi_i] \in \mathbb{N}$. We then set $E[\xi_i]$ equal to the demand of customer i in the corresponding deterministic instance. The position of the customers in the 2-dimensional plane is the same as in the deterministic instance.

Like [6], the deterministic travel cost is taken as the Euclidean distance between two nodes *rounded to the nearest integer*. However, we use floating point values to represent the total expected travel cost of an edge. In particular, the expected failure cost is computed as a floating-point value using the *boost* library (http://www.boost.org/).

Observe that the total expected failure cost at any customer is a function of the total expected cumulative demand at that customer. Hence, contrary to the deterministic case, dividing expected demands and capacity by their greatest common divisor will change the nature of the stochastic instance (namely its optimal expected cost). However, this procedure can be seen as an instance construction hypothesis which is analogue to the well-known rounding of distances for the TSP and the VRP. As in [6], we therefore present results based on this assumption.

In order to prevent the explosion of the size of the state-space graph when expected demands are very large and cannot be divided, we use a filtering procedure that eliminates nodes that cannot be reached. More specifically when considering Poisson distributed demands, a node $\mathcal{V}(\mu_i, i) \in \mathcal{NS}$ cannot be reached if the sum of the expected demands for all subsets of customers of $\mathcal{N}\backslash\{i\}$ is never equal to μ_i. We solve this problem by using a procedure based on the subset-sum problem (which can be solved in pseudo-polynomial time with respect to the number of customers and the largest possible expected demand).

All instances were solved using Gencol version 4.5, a professional grade column generation software commercialized by Kronos Inc. and written in C and C++. The *RMP* are solved using ILOG Cplex solver version 12.4.0. All tests were run using a computer with an Intel i7-2600 processor with 3.4 GHz and 16 GB of RAM. In contrast, the algorithm of [6], which dates back to 2007, was run on a Pentium Centrino 1.5 GHz with 480 MB of RAM.

To compare the efficiency of our algorithm with the one of [6] while controlling the effect of increased processing power, we use the benchmarks of Dongarra [10]. We specifically use the "LINPACK Benchmarks" for solving systems represented by a matrix of order 100 and consider 796 MFLOP/s (million of floating point operations per second) corresponding to Intel P4 1700 MHz as a proxy for [6] and 1573 MFOP/s corresponding to Pentium IV with 3.0 GHz for this work. We take the ratio of these two metrics to obtain an estimate of 1.98. The computational times of [6] that we present are divided by this factor. All instances were solved using the following parameters:

- maximal allotted time of 1200 s (20 min),
- only 1 type of cutting plane is added each time such cuts are identified,
- we first try to identify violated CC. If we fail to do so, we then try to identify SRI,
- all SRI that have a violation greater than or equal to 0.1 are added,
- maximum of 15 SRI generated at each B&B node,
- we use a best-first strategy to explore the B&B tree,

Before comparing our algorithm with that of [6], it is important to justify the use of the various features presented in this paper. For this purpose, Table 1 compares different variations of our base algorithm that uses all the 6 algorithmic components (tabu search, bidirectional labeling, aggregate dominance, capacity cuts, subset-row inequalities and branching based on a flow adjacent to a set of customers). We specifically consider each of the 6 variants that do not use one of the components. Only the instances considered by [6] are used in this comparison. The minimum computation time for a given instance is indicated in bold and the average is taken over the 33 instances solved by all variants.

Table 1 shows that the base algorithm has one of the best average computing time for the 33 instances solved by all algorithms. It also solves the largest number of instances within the time limit and proves to be the fastest in many cases. However, in order to establish a more rigorous comparison method, we adopt the methodology of Dolan and More [9] for the evaluation of optimization software. This technique, based on "profile curves", allows a rapid and visual comparison of the efficiency and robustness of various algorithms and parameter sets. It illustrates the worst, average and best performance of an algorithm in relation with the others considered.

To describe this framework, it is necessary to introduce the following notation. Let $\mathcal{AL}$ denote the set of all parameters (algorithms) used and $\mathcal{I}$ represent the set of all 40 instances considered. For every pair $(algo, inst) \in \mathcal{AL} \times \mathcal{I}$, the metric $t_{algo,inst}$ represents the time required to solve the instance $inst$ with the parameter $algo$. $t^*_{inst} = \min_{algo \in \mathcal{AL}}\{t_{algo,inst}\}$ represents the minimum computing time and ρ is a real factor. We then introduce the function $\omega(t, t^*, \rho)$, defined for all $t_{algo,inst} \geq 0, t^*_{inst} \geq 0$ and $\rho \geq 1$:

$$\omega(t_{algo,inst}, t^*_{inst}, \rho) = \begin{cases} 1 & \text{if } t_{algo,inst} \leq \rho t^*_{inst} \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

We also define $\pi_{algo}(\rho)$ as the "performance profile" of the parameter set $algo$ with the factor ρ. This metric is obtained by computing:

$$\pi_{algo}(\rho) = |\mathcal{I}|^{-1} \sum_{inst \in \mathcal{I}} \omega(t_{algo,inst}, t^*_{inst}, \rho) \tag{18}$$

We plot this metric for each algorithm. The results are shown in Fig. 2. $\pi_{algo}(1)$ indicates the fraction of instances for which $algo$ was the fastest while $\pi_{algo}(\rho)$ shows the fraction of instances for which the computing time of $algo$ was less than or equal to ρ times the best time t^*_{inst}. The algorithm with the largest $\pi_{algo}(1)$ is therefore the one that yields the best time for the most instances. In our case, it is the parameter set without the bidirectional algorithm that is the fastest for more than 50% of instances. This can be explained by the fact that the extra overhead and the incompatibility with the aggregate

dominance of the bidirectional labeling algorithm often make it uncompetitive. However, it is essential to solve instances with loose capacity constraints in which the optimal solution can be composed of very long routes.

A curve that stabilizes rapidly indicates that the algorithm has a relatively stable performance compared to the others. For instance the computing time for the base algorithm is always within a factor of 7.2 of the best algorithm for every instance it can solve while there exist instances for which the algorithm without CC can take more than 48 times the computing time of the best algorithm. Moreover, a curve that reaches its maximum value π_{algo}^{max} at small values of ρ and that has a high π_{algo}^{max} indicates that the algorithm has a good worst case performance. This is namely the case for the base algorithm.

According to this framework, we clearly see that the "best" algorithm is the base algorithm that uses all the features described. More specifically, the parameter set that yields the most reliable and rapid performance is the one that uses *ng*-routes, tabu search and bidirectional labeling. Although it is the most rapid for only 18% of the instances, it is the only one that manages to solve 38 instances. Moreover it has the best worst case performance. Indeed, 90% of instances (solvable within 20 min) are solved with a computing time smaller than or equal to 2.2 times the best time and 100% are solved within a ratio of 7.2.

We also observe that the "most important" factors are the addition of SRI and the use of the aggregate dominance. Without the SRI, the algorithm only manages to solve 85% of the instances (4 instances solved by the base algorithm are not closed). We

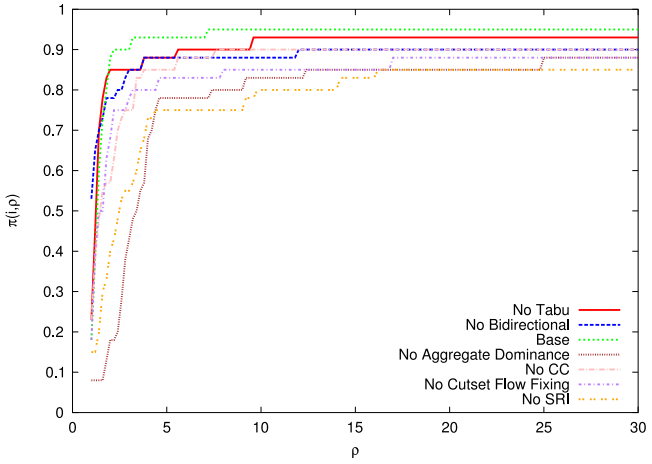

Fig. 2. Performance profile curves.

Table 1
Computational times (in seconds)—comparison with variants that do **not** use one of the 6 features.

Instance	Base	No tabu	No bidirectional	No aggregate dominance	No CC	No SRI	No flow fixing
A-n32-k5	24.1	20.8	34.0	108.2	26.3	**11.8**	24.8
A-n33-k5	5.3	10.2	4.0	11.4	**2.8**	10.2	5.0
A-n33-k6	4.9	4.2	**3.8**	10.8	4.0	7.2	4.2
A-n34-k5	9.0	8.0	**7.9**	28.1	60.0	11.6	**7.9**
A-n36-k5	46.9	**40.3**	148.0	160.5	129.3	63.0	42.5
A-n37-k5	23.4	28.5	**18.5**	70.7	59.9	50.6	20.9
A-n37-k6	22.8	24.2	**19.0**	50.3	61.7	38.4	20.3
A-n38-k5	**42.8**	55.5	43.5	311.2	59.1	132.6	89.5
A-n39-k5	5.0	**3.9**	5.2	9.9	4.6	4.8	4.6
A-n39-k6	19.3	17.4	17.0	34.9	**15.0**	25.2	17.5
A-n44-k6	**125.0**	125.9	171.3	364.8	283.7	308.2	361.0
A-n45-k6	86.3	**81.3**	191.6	353.0	224.4	180.4	164.5
A-n45-k7	30.5	25.7	**25.5**	62.7	26.4	63.8	26.5
A-n46-k7	18.7	20.5	26.9	47.5	22.9	21.1	**15.8**
A-n48-k7	27.5	27.5	33.3	71.4	**19.6**	69.2	23.8
A-n53-k7	512.4	311.9	**295.1**	#	295.7	1028.1	#
A-n54-k7	310.5	305.6	298.3	509.9	683.8	1159.4	**272.3**
A-n55-k9	33.2	32.4	29.8	78.4	**25.7**	48.3	28.1
A-n60-k9	#	#	#	#	#	#	#
E-n22-k4	**0.1**	**0.1**	**0.1**	0.2	**0.1**	**0.1**	**0.1**
E-n33-k4	32.6	28.2	**21.1**	36.9	33.5	33.5	34.2
E-n51-k5	#	#	#	#	#	#	#
P-n16-k8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
P-n19-k2	9.7	**7.2**	19.3	32.6	27.3	28.2	9.7
P-n20-k2	128.8	174.8	#	459.2	892.8	**18.4**	310.5
P-n21-k2	2.2	2.1	**1.2**	3.1	2.7	2.8	2.3
P-n22-k2	46.7	84.8	180.6	187.8	83.6	**15.3**	47.1
P-n22-k8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**
P-n23-k8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
P-n40-k5	6.9	**5.7**	**5.7**	9.6	9.5	8.3	7.1
P-n45-k5	**1193.8**	#	#	#	#	#	#
P-n50-k10	54.6	42.7	**32.9**	127.0	66.7	296.5	147.4
P-n50-k7	40.4	33.9	**22.4**	73.5	29.2	315.1	40.0
P-n50-k8	35.7	34.3	**26.9**	114.3	55.5	261.9	50.3
P-n51-k10	10.0	7.9	**7.7**	20.7	7.8	25.9	9.9
P-n55-k10	26.4	20.9	**16.0**	56.9	23.5	259.2	26.0
P-n55-k15	19.9	17.8	**13.3**	37.2	17.4	25.6	106.3
P-n55-k7	103.8	111.2	**91.5**	353.2	112.6	#	103.9
P-n60-k10	417.5	505.9	**275.9**	#	305.1	#	#
P-n60-k15	7.0	4.8	4.6	11.9	**4.5**	17.8	7.3
Total number of instances solved	38	37	36	35	37	35	35
Average Time (33 instances)	34.16	34.01	43.32	90.77	62.74	105.94	49.00

conclude that although these cuts increase the complexity of the pricing procedure, they are extremely efficient to derive integer solutions. The aggregate dominance plays a central role because it manages to eliminate significantly more suboptimal labels at every iteration of the labeling procedure.

Although features such as the tabu search algorithm may seem to be less efficient, we keep all of them in our final algorithm because they make it possible to solve strictly more instances than if we omit any one of them.

We compare the final computational results of the base algorithm with the ones of [6] in Table 2. We also present further computational results for A, B, E, M and P instances in Table 3. We only report results for instances for which we were able to construct the state-space graph and obtain a feasible solution. These tables contain the following columns:

- **Instance**: Name of the instance,
- **# Path**: Number of vehicles used by the best computed feasible solution of the VRPSD instance.
- **EEV**: Expected value of the expected value solution (see [5]). Since we assume that the expected demands are equal to their corresponding deterministic values, the expected value solution is the optimal solution of the deterministic VRP counterpart. These solutions are available at http://branchandcut.org. We need to consider the two orientations of each route of this solution since the deterministic VRP cost does not depend on the order in which customers are visited, which is not the case for the stochastic VRP. Note that this column corresponds to the "Best det." column from Table 1 in [6].[1]
- **Cost (UB)**: Best computed feasible solution cost. This cost is divided into two: the total deterministic cost of the routes and the recourse cost. Optimal solutions are marked with a ∗.
- **Time**: Total time required to obtain the optimal solution (# indicates the instance was not solved within the time limit),
- **LB root/UB**: Ratio of the lower bound at the root node over the best feasible (not necessarily optimal) solution found,
- **# B&B nodes**: Total number of branch-and-bound nodes in the B&B tree (excluding the root node),
- **# Rounds of cuts**: Total *number of times* during the algorithm where valid inequalities (CC or SRI) were added to *MP*,
- **# Cuts (CC or SRI)**: Total *number* of cuts (capacity cuts or subset-row inequalities) identified during the course of the entire algorithm,
- **Acceleration factor**: Ratio $\text{Time}_{\text{Lysgaard}}/\text{Time}_{\text{Base}}$. For the computation of this ratio, we use $\frac{1200}{1.98} = 606.06$ s as a lower bound on $\text{Time}_{\text{Lysgaard}}$ for all instances that were not solved by [6]. We only compute this metric for instances that we solve in at least 1 second (which have a finite ratio).

As Table 2 demonstrates, our algorithm is very competitive with the one of [6]. Indeed, only the easily-solvable instances A-n33-k5, A-n39-k5 and P-n40-k5 are solved in less time by [6]. Our algorithm solves 19 additional instances compared with [6] within 606 s (which corresponds roughly to the maximum time of 20 min considered by [6] scaled) and 20 within the unscaled time of 20 min. Moreover, it manages to solve instances up to 86 times faster.

[1] We point out that the value presented in column EEV is not totally coherent with our model. Indeed, the deterministic solutions taken from the literature assume that the number of vehicles is given by an *equality* while the constraint (5) in program *IMP* is an *inequality*. To get the EV (expected value solution), we should solve the deterministic counterpart with an inequality. However this only makes a difference for a few instances. For the sake of comparison with previous works and ease of exposition we use the optimal solution from the literature to compute the EEV.

As expected, the combination of *ng*-routes and 2-cycle elimination provides lower bounds at the root node that are at least as good as those of [6]. These stronger bounds are beneficial. Indeed, instances E-n22-k4, E-n33-k4, A-n39-k5 and P-n23-k8 are solved directly at the root node and instances P-n40-k5, P-n21-k1 and P-n16-k8 only required the identification of a set of cutting planes while the algorithm of [6] requires at least 9 branch-and-bound nodes (with the exception of P-n23-k8).

The introduction of cutting planes also significantly boosts the solution procedure. Indeed, the algorithm of [6] creates significantly more branch-and-bound nodes than our algorithm. In addition, large instances such as P-n50-k7 that have a very large gap between the upper and lower bounds at the root node can still be solved rapidly (under 1 min) without resorting to branching.

Although we generally need to solve the *RMP* more often, this does not notably impact the performance of our algorithm since the *ng*-route subproblem combined with tabu search is solved rather rapidly. Nonetheless, as much as 95% of the total running time of our algorithm is dedicated to the solution of the subproblem.

Our algorithm is not even capable of storing the problem data in memory for most of the F, G and M instances. Indeed, the number of clients or the large difference between the maximum and minimum demands in these instances considerably increases the size of the underlying state-space graph and makes them severely intractable for our method.

Tables 2 and 3 illustrate the importance of considering uncertainty in the problem formulation. We observe that the value of the stochastic solution (VSS) given by the difference between EEV and the optimal stochastic solution can be as high as 10% of the optimal stochastic cost. The optimal number of vehicles used can also vary considerably in the stochastic and the deterministic solutions (in the deterministic case, the number of routes is given by the number following k in the instance name).

5. Conclusion

The new state-of-the-art branch-cut-and-price algorithm we propose significantly enhances the work of [6]. The use of a bidirectional labeling algorithm, tabu search heuristic, *ng*-routes combined with 2-cycle elimination, an aggregate dominance criterion, capacity cuts and subset-row inequalities as well as an efficient branching strategy allows us to solve 20 additional instances from a 40-instance benchmark set and solve the majority of previously closed instances in considerably less time. We also solve 17 new instances from the A, B, E, M and P classes. Regardless of the complicated interdependencies between all these algorithmic features and the fact that adding an element yields diminishing marginal returns, their combination makes it possible to solve difficult instances of up to 101 customers and 15 vehicles in less than 20 min.

Although we only reported the results of our algorithm for Poisson distributed demands with integer expectation, the filtering procedure we use to eliminate unnecessary nodes is readily applicable in the case of demands with rational expectations. Experiments have revealed that we can solve the majority of the instances currently solved within the time limit of 20 min when considering Poisson demands with one decimal place. Higher precision quickly increases the number of nodes and arcs in $\mathcal{GS}$ and leads to intractability.

Future work should attempt to generalize our procedure to the case of random variables that are defined by 2 parameters or more such as normal random variables. This represents an important challenge as this would increase the size of the state-graph and

Table 2
Final results—instances A, E, P and comparison with Christiansen and Lysgaard [6].

Instance	# Paths	EEV	Cost (UB)			Base (This paper)						[6]			Acceleration factor
			Total	Routing	Recourse	Time	LBroot/UB	# B & B nodes	# Rounds of cuts	# Cuts CC	# Cuts SRI	Time	$\frac{LBroot}{UB}$	# B & B Nodes	
A-n32-k5	5	890.1	853.6*	839*	14.6*	24.1	0.98	0	7	6	90	142.4	0.96	2467	5.9
A-n33-k5	5	723	704.2*	683*	21.2*	5.3	1.00	0	6	29	15	4.0	0.99	117	0.8
A-n33-k6	6	816.6	793.9*	759*	34.9*	4.9	1.00	0	4	3	20	24.7	0.98	909	5.1
A-n34-k5	6	840	826.9*	789*	37.9*	9.0	0.98	0	6	27	28	#	0.97	16 059	$\geq$ 67.3
A-n36-k5	5	907.6	858.7*	816*	42.7*	46.9	0.98	0	10	30	90	#	0.92	8035	$\geq$ 12.9
A-n37-k5	5	709.8	708.3*	670*	38.3*	23.4	0.98	0	7	12	72	#	0.97	9191	$\geq$ 25.9
A-n37-k6	7	1069.3	1030.7*	1003*	27.7*	22.8	0.99	2	11	56	62	#	0.98	16 195	$\geq$ 26.6
A-n38-k5	6	832	775.1*	764*	11.1*	42.8	0.97	6	11	22	74	#	0.95	14 499	$\geq$ 14.2
A-n39-k5	6	903.3	869.2*	832*	37.2*	5.0	1.00	0	0	0	0	1.5	1.00	9	0.3
A-n39-k6	6	960.8	876.6*	834*	42.6*	19.3	0.99	0	8	28	45	140.9	0.97	2431	7.3
A-n44-k6	7	1047.2	1025.5*	971*	54.5*	125	0.99	12	21	21	189	#	0.98	11 077	$\geq$ 4.9
A-n45-k6	7	1096.2	1026.7*	971*	55.7*	86.3	0.98	2	11	9	120	#	0.90	9313	$\geq$ 7.0
A-n45-k7	7	1302.2	1264.8*	1169*	95.8*	30.5	1.00	0	9	8	70	445.5	0.99	5365	14.6
A-n46-k7	7	1069.7	1002.2*	971*	31.2*	18.7	0.99	0	4	58	30	#	0.98	8149	$\geq$ 32.4
A-n48-k7	7	1248.3	1187.1*	1129*	58.1*	27.5	0.99	0	6	42	30	#	0.93	7729	$\geq$ 22.0
A-n53-k7	8	1180.1	1124.3*	1087*	37.3*	512.4	0.99	12	23	37	240	#	0.93	5385	$\geq$ 1.2
A-n54-k7	8	1342.9	1287.1*	1211*	76.1*	310.5	0.99	6	18	55	126	#	0.94	5925	$\geq$ 2.0
A-n55-k9	10	1264.2	1179.1*	1145*	34.1*	33.2	0.99	0	7	50	45	#	0.91	9979	$\geq$ 18.3
A-n60-k9	32	1608.4	3565.0	3565	0.0	#	0.42	20	35	49	219	#	0.93	6889	#
E-n22-k4	4	411.7	411.6*	375*	36.6*	0.1	1.00	0	0	0	0	0.5	1.00	9	5
E-n33-k4	4	850.3	850.3*	846*	4.3*	32.6	1.00	0	0	0	0	43.4	0.99	73	1.3
E-n51-k5	6	553.3	568.0	563	5.0	#	0.95	4	29	23	212	#	0.97	2771	#
P-n16-k8	8	512.8	512.8*	450*	62.8*	0.0	1.00	0	1	1	0	0	1.00	17	#
P-n19-k2	3	229.7	224.1*	219*	5.1*	9.7	0.94	0	7	4	60	77.3	0.94	1815	8.0
P-n20-k2	2	233.1	233.1*	216*	17.1*	128.8	0.95	6	15	5	130	177.8	0.95	3191	1.4
P-n21-k2	2	219	219*	211*	8*	2.2	1.00	0	1	3	0	2.5	0.99	27	1.1
P-n22-k2	2	231.3	231.3*	216*	15.3*	46.7	0.97	0	8	3	90	110.6	0.97	1335	2.4
P-n22-k8	9	707.8	681.1*	598*	83.1*	0.0	1.00	0	2	1	5	0	1.00	65	#
P-n23-k8	9	662.3	619.5*	549*	70.5*	0.0	1.00	0	0	0	0	0.5	1.00	0	#
P-n40-k5	5	475.5	472.5*	461*	11.5*	6.9	1.00	0	1	4	0	4.5	1.00	35	0.7
P-n45-k5	5	546.1	533.9*	514*	19.9*	1193.8	0.98	12	34	22	272	#	0.95	4561	#
P-n50-k10	11	792.2	758.8*	729*	29.8*	54.6	0.99	16	28	29	119	#	0.95	18 901	$\geq$ 11.1
P-n50-k7	7	606.4	582.4*	562*	20.4*	40.4	0.99	0	11	26	75	#	0.95	5311	$\geq$ 15.0
P-n50-k8	9	724.7	669.2*	651*	18.2*	35.7	0.99	2	11	9	98	#	0.91	10 409	$\geq$ 17.0
P-n51-k10	11	859.2	809.7*	766*	43.7*	10.0	0.99	0	8	21	56	217.2	0.99	5431	21.7
P-n55-k10	10	616.4	742.4*	710*	32.4*	26.4	0.99	0	10	23	60	#	0.92	11 257	$\geq$ 23.0
P-n55-k15	18	797.2	1068.1*	1007*	61*	19.9	1.00	36	32	23	61	400.0	0.99	20 027	20.1
P-n55-k7	7	1191.3	588.6*	581*	7.6*	103.8	0.99	0	14	22	120	#	0.94	2441	$\geq$ 5.8
P-n60-k10	11	831.2	803.6*	786*	17.6*	417.5	0.99	34	52	42	273	#	0.95	4969	$\geq$ 1.5
P-n60-k15	16	1133.3	1085.5*	1013*	72.5*	7.0	1.00	0	7	37	26	#	1.00	19 029	$\geq$ 86.6
Average	7.7	824.9	840.6	805.8	34.8	91.7	0.97	4.25	11.9	21	80.6	99.6	0.96	6284.9	$\geq$ 13.6

The row Average represents the arithmetic average over all instances solved.

Table 3
Final results—additional instances A, E, P and instances B and M.

Instance	# Paths	EEV	Cost (UB)			Base (This paper)					
			Total	Routing	Recourse	Time	LB root/UB	# B & B nodes	# Rounds of cuts	# Cuts CC	# Cuts SRI
E-n23-k3	3	569.7	569.7*	569*	0.7*	32.2	0.99	0	1	5	0
E-n30-k3	4	569.9	504.6*	503*	1.6*	59.5	0.99	0	2	0	30
E-n76-k7	41	704.0	2149.0	2149	0.0	#	0.32	0	20	50	150
E-n76-k8	41	791.4	2149.0	2149	0.0	#	0.35	0	20	103	195
E-n76-k10	11	911.7	885.1*	861*	24.1*	175.8	0.99	0	11	20	105
E-n76-k14	16	1187.2	1118.9	1086	32.9	#	0.99	73	103	77	434
E-n101-k8	52	879.0	2881.0	2881	0.0	#	0.29	0	11	79	75
E-n101-k14	52	1233.8	2881.0	2881	0.0	#	0.40	9	31	65	158
A-n61-k9	30	1215.4	2597.0	2594	3.0	#	0.43	28	50	78	323
A-n62-k8	9	1533.1	1430.8*	1375*	55.8*	58.6	0.99	0	6	23	30
A-n63-k10	11	1581.2	1459.5*	1412*	47.5*	958.3	0.99	34	48	109	258
A-n63-k9	36	1991.7	5232.0	5232	0.0	#	0.35	24	49	42	330
A-n64-k9	36	1699.7	4704.0	4704	0.0	#	0.33	8	31	96	215
A-n65-k9	10	1421.5	1313.3*	1266*	47.3*	134.4	0.98	0	16	47	120
A-n69-k9	10	1339.5	1259.4*	1214*	45.4*	755.2	0.99	12	22	43	188
P-n65-k10	10	861.5	854.1*	802*	52.1*	927.8	0.99	34	61	89	308
P-n70-k10	11	929.8	882.0*	851*	31.0*	140.6	0.99	0	14	52	90
P-n76-k4	6	627.0	664.0	664	0.0	#	0.90	0	8	10	90
P-n76-k5	6	649.1	669.9	665	4.9	#	0.94	0	8	10	90
B-n31-k5	5	776.6	714.7*	694*	20.7*	2.6	1.00	0	1	4	0
B-n34-k5	18	926.5	2298.7	2298	0.73	#	0.37	8	20	7	175
B-n35-k5	5	1148.6	1037.9*	988*	49.9*	769.1	0.92	0	11	20	135
B-n38-k6	6	931.4	842.8*	819*	23.8*	8.1	0.97	0	1	13	0
B-n39-k5	5	634.7	590.7*	564*	26.6*	58.5	0.98	0	5	0	75
B-n41-k6	7	973.6	928.0*	897*	301.0*	879.3	0.98	142	133	30	800
B-n43-k6	10	868.3	1038.1	1037	1.1	#	0.77	17	31	43	197
B-n44-k7	23	1123.9	2563.3	2563	0.3	#	0.39	23	26	17	227
B-n45-k5	6	827.0	788.3	765	23.3	#	0.95	16	21	0	243
B-n45-k6	7	828.9	756.4*	727*	29.4*	175.2	0.98	0	14	19	146
B-n50-k7	7	831.3	791.7*	755*	36.7*	167.8	0.95	0	8	17	90
B-n50-k8	32	1587.8	4030.6	4027	3.6	#	0.35	48	73	21	485
B-n51-k7	32	1294.0	3643.0	3643	0.0	#	0.30	14	23	0	224
B-n52-k7	27	878.3	2652.0	2652	0.0	#	0.29	12	27	0	248
B-n56-k7	32	849.1	2803.0	2803	0.0	#	0.26	0	12	0	180
B-n57-k7	40	1492.0	5139.0	5139	0.0	#	0.25	2	20	34	205
B-n57-k9	27	2013.8	4640.0	4640	0.0	#	0.38	11	24	26	198
B-n63-k10	38	1931.6	5045.0	5045	0.0	#	0.34	0	21	34	200
B-n64-k9	49	1049.5	3528.4	3528	0.4	#	0.26	3	16	0	211
B-n66-k9	34	1598.3	4092.0	4092	0.0	#	0.36	8	22	21	179
B-n67-k10	42	1281.4	3346.0	3346	0.0	#	0.34	6	20	30	238
B-n68-k9	39	1597.2	4788.0	4788	0.0	#	0.29	0	19	21	211
B-n78-k10	46	1534.3	4487.0	4487	0.0	#	0.30	0	10	0	150
M-n101-k10	10	984.4	982.8*	827*	155.8*	614.2	0.99	0	12	19	135
Average	21.9	1131.6	2226.3	2208.9	17.4	367.8	0.66	12.4	25.2	32.0	189.3

The row Average represents the arithmetic average over all instances solved.

would considerably increase the complexity of our labeling algorithm.

Acknowledgments

The authors acknowledge the support of the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT). They would also like to thank François Lessard for his technical support as well as two anonymous referees and the area editor for their thoughtful comments.

Appendix A. Proof that $\mathrm{EFC}_i(\mu)$ is increasing in μ for any customer i when the demands are Poisson distributed

Recall that for any customer i on any route such that the expected cumulative demand at that customer is μ, the expected failure cost is given by

$$\mathrm{EFC}_i(\mu) = 2c_{oi} \sum_{u=1}^{\infty} (\mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) \le uQ\} - \mathrm{P}\{\Psi(\mu) \le uQ\}) \tag{A.1}$$

$$\mathrm{EFC}_i(\mu) = 2c_{oi} \sum_{u=1}^{\infty} \sum_{k=0}^{uQ} (\mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = k\} - \mathrm{P}\{\Psi(\mu) = k\}) \tag{A.2}$$

To prove that the function $\mathrm{EFC}_i(\mu) : (\mathrm{E}[\xi_i], \infty) \to \mathbb{R}$ is increasing in μ, the total expected cumulative demand at any customer $i \in \mathcal{N}$ when the demands are Poisson distributed, we show that the derivative of $\mathrm{EFC}_i(\mu)$ exists and is non-negative for $\mu \in (\mathrm{E}[\xi_i], Q]$ for any customer $i \in \mathcal{N}$ (note that we do not require μ to take discrete values). In order to do so, we compute the derivative of $\sum_{k=0}^{uQ} (\mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = k\} - \mathrm{P}\{\Psi(\mu) = k\})$ and then show that the infinite series $\sum_{u=1}^{\infty} \sum_{k=0}^{uQ} (\mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = k\} - \mathrm{P}\{\Psi(\mu) = k\})$ is uniformly convergent.

Lemma 1. $f_{ui}'(\mu) = \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} - \mathrm{P}\{\Psi(\mu) = uQ\}$ *where* $f_{ui}(\mu) = \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) \le uQ\} - \mathrm{P}\{\Psi(\mu) \le uQ\}$, $\mu \in (\mathrm{E}[\xi_i], \infty)$ *and* $u \in \mathbb{N}$.

Proof.

$$f_{ui}'(\mu) = \frac{d}{d\mu} \sum_{k=0}^{uQ} \left(\frac{(\mu - \mathrm{E}[\xi_i])^k}{e^{\mu - \mathrm{E}[\xi_i]} k!} - \frac{\mu^k}{e^{\mu} k!} \right) \tag{A.3}$$

$$f_{ui}'(\mu) = \sum_{k=0}^{uQ} \left(\frac{k(\mu - \mathrm{E}[\xi_i])^{k-1}}{e^{\mu - \mathrm{E}[\xi_i]} k!} - \frac{k\mu^{k-1}}{e^{\mu} k!} \right) - \sum_{k=0}^{uQ} \left(\frac{(\mu - \mathrm{E}[\xi_i])^k}{e^{\mu - \mathrm{E}[\xi_i]} k!} - \frac{\mu^k}{e^{\mu} k!} \right) \tag{A.4}$$

$$f_{ui}'(\mu) = \sum_{j=0}^{uQ-1} \left(\frac{(\mu - \mathrm{E}[\xi_i])^j}{e^{\mu - \mathrm{E}[\xi_i]} j!} - \frac{\mu^j}{e^{\mu} j!} \right) - \sum_{k=0}^{uQ-1} \left(\frac{(\mu - \mathrm{E}[\xi_i])^k}{e^{\mu - \mathrm{E}[\xi_i]} k!} - \frac{\mu^k}{e^{\mu} k!} \right) + \frac{\mu^{uQ}}{e^{\mu}(uQ)!} - \frac{(\mu - \mathrm{E}[\xi_i])^{uQ}}{e^{\mu - \mathrm{E}[\xi_i]}(uQ)!} \tag{A.5}$$

$$f_{ui}'(\mu) = \left(\frac{\mu^{uQ}}{e^{\mu}(uQ)!} - \frac{(\mu - \mathrm{E}[\xi_i])^{uQ}}{e^{\mu - \mathrm{E}[\xi_i]}(uQ)!} \right) \tag{A.6}$$

$$f_{ui}'(\mu) = (\mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}) \quad \square \tag{A.7}$$

Lemma 2. *The series* $\sum_{u=1}^{\infty} f_{ui}'(\mu)$ *is uniformly convergent for all* $i \in \mathcal{N}$ *on* $(\mathrm{E}[\xi_i], \infty)$ *where* $f_{ui}'(\mu) = \mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}$.

Proof. We show there exists $M_u \in \mathbb{R}$ such that $|f_{ui}'(\mu)| \le M_u$, $\forall u \in \mathbb{N}, \mu \in (\mathrm{E}[\xi_i], \infty)$ and that $\sum_{u=1}^{\infty} M_u$ is convergent.

$$|f_{ui}'(\mu)| = |\mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}| \tag{A.8}$$

$$|f_{ui}'(\mu)| \le \mathrm{P}\{\Psi(\mu) = uQ\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} \tag{A.9}$$

We therefore let $M_u := \mathrm{P}\{\Psi(\mu) = uQ\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}$. We know that for all $k \in \mathbb{N}$:

$$s_k = \sum_{u=1}^{k} (\mathrm{P}\{\Psi(\mu) = uQ\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}) \tag{A.10}$$

$$s_k \le \sum_{a=0}^{kQ} (\mathrm{P}\{\Psi(\mu) = a\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = a\}) \tag{A.11}$$

$$s_k \le \sum_{a=0}^{\infty} (\mathrm{P}\{\Psi(\mu) = a\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = a\}) \tag{A.12}$$

$$s_k \le 2 \tag{A.13}$$

Since $\mathrm{P}\{\Psi(\mu) = uQ\} + \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} \ge 0, \forall u \in \mathbb{N}$, it follows that (s_k) is a non-decreasing bounded sequence and is therefore convergent. $\square$

Lemma 3. *The derivative of* $\mathrm{EFC}_i(\mu)$ *exists and is equal to* $2c_{0i} \sum_{u=1}^{\infty} (\mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\})$ *for* $\mu \in (\mathrm{E}[\xi_i], \infty)$.

Proof. By Lemmas 1 and 2, the series $\sum_{u=1}^{\infty} f_{ui}(\mu)$ converges uniformly on $(\mathrm{E}[\xi_i], \infty)$ to $\mathrm{EFC}_i(\mu)$. We therefore have

$$\frac{d\mathrm{EFC}_i(\mu)}{d\mu} = 2c_{0i} \frac{d}{d\mu} \sum_{u=1}^{\infty} f_{ui}(\mu) \tag{A.14}$$

$$\frac{d\mathrm{EFC}_i(\mu)}{d\mu} = 2c_{0i} \sum_{u=1}^{\infty} f_{ui}'(\mu) \tag{A.15}$$

$$\frac{d\mathrm{EFC}_i(\mu)}{d\mu} = 2c_{0i} \sum_{u=1}^{\infty} (\mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\}). \quad \square \tag{A.16}$$

Lemma 4. $\mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} \ge 0, \forall \mu \in (\mathrm{E}[\xi_i], Q]$, $i \in \mathcal{N}, u \in \mathbb{N}$

Proof. For any $\mu \in (\mathrm{E}[\xi_i], Q], i \in \mathcal{N}, u \in \mathcal{N}$, we have

$$\mathrm{P}\{\Psi(\mu) = uQ\} \ge \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} \Leftrightarrow \frac{\mu^{uQ}}{e^{\mu}} \ge \frac{(\mu - \mathrm{E}[\xi_i])^{uQ}}{e^{(\mu - \mathrm{E}[\xi_i])}} \tag{A.17}$$

The inequality (A.17) is verified if $x \le y \Rightarrow x^{uQ}/e^{x} \le y^{uQ}/e^{y}$ for all $x, y \in (\mathrm{E}[\xi_i], Q]$, which holds if

$$\frac{d}{dx} \frac{x^{uQ}}{e^{x}} \ge 0, \quad \forall x \in (\mathrm{E}[\xi_i], Q] \tag{A.18}$$

$$\Leftrightarrow uQ \ge x, \quad \forall x \in (0, Q] \tag{A.19}$$

(A.19) always holds for $x = \mu - \mathrm{E}[\xi_i]$ and $u \in \mathbb{N}$, since $0 \le \mu - \mathrm{E}[\xi_i] < Q, \forall \mu \in (\mathrm{E}[\xi_i], Q], i \in \mathcal{N}$. $\square$

Proposition. *The function* $\mathrm{EFC}_i(\mu)$ *is increasing in* μ, *the total expected cumulative demand at any customer* $i \in \mathcal{N}$ *when the demands are Poisson distributed and* $\mu \in (\mathrm{E}[\xi_i], Q]$.

Proof. By Lemmas 1–4, we have

$$\frac{d}{d\mu}\mathrm{EFC}_i(\mu) = 2c_{0i} \sum_{u=1}^{\infty} \mathrm{P}\{\Psi(\mu) = uQ\} - \mathrm{P}\{\Psi(\mu - \mathrm{E}[\xi_i]) = uQ\} \tag{A.20}$$

$$\frac{d}{d\mu}\mathrm{EFC}_i(\mu) \ge 0, \quad \forall i \in \mathcal{N}, \ \mu \in (\mathrm{E}[\xi_i], Q]. \quad \square \tag{A.21}$$

References

[1] Augerat P. Approche polyédrale du Problème de Tournées de Véhicules (Ph.D. thesis), Institut National Polytechnique de Grenoble, 1995.
[2] Baldacci R, Bartolini E, Mingozzi A, Roberti R. An exact framework for a broad class of vehicle routing problems. Comput Manag Sci 2010;7:229–68.
[3] Baldacci R, Mingozzi A, Roberti R. New route relaxation and pricing strategies for the vehicle routing problem. Oper Res 2011;59:1269–83.
[4] Bertsimas D. A vehicle routing problem with stochastic demand. Oper Res 1992;40:574–85.
[5] Birge JR, Louveaux F. Introduction to Stochastic Programming. New York: Springer; 1997.
[6] Christiansen HC, Lysgaard J. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. Oper Res Lett 2007;37:773–81.
[7] Desaulniers G, Desrosiers J, Solomon MM, editors. Column Generation. New York: Springer; 2005.
[8] Desaulniers G, Lessard F, Hadjar A. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transp Sci 2008;45:387–404.
[9] Dolan E, More J. Benchmarking optimization software with performance profiles. Math Progr Ser A 2002;91:201–13.
[10] Dongarra J.J. Performance of Various Computers Using Standard Linear equation software, (linpack benchmark report). Technical Report, University of Tennessee Computer Science, 2011.
[11] Dror M. Note on the complexity of the shortest path models for column generation in VRPTW. Oper Res 1994;42(5):977–8.
[12] Feillet D, Dejax P, Gendreau M, Gueguen C. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks 2004;44:216–29.
[13] Fukasawa R, Longo H, Lysgaard J, Poggi de Aragao M, Reis M, Uchoa E, et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Math Progr Ser A 2006;106:491–511.
[14] Gendreau M, Laporte G, Séguin R. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp Sci 1995;29:143–55.
[15] Gendreau M, Laporte G, Séguin R. Stochastic vehicle routing. Eur J Oper Res 1996;88:3–12.
[16] Gendreau M, Laporte G, Séguin R. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Oper Res 1996;44:469–77.
[17] Golden BL, Stewart WJ. Stochastic vehicle routing: a comprehensive approach. Eur J Oper Res 1983;14:371–85.
[18] Hjorring C, Holt J. New optimality cuts for a single-vehicle stochastic routing problem. Ann Oper Res 1999;86:569–84.
[19] Houck DJ, Picard JC, Queyranne M, Vemuganti RR. The travelling salesman problem as a constrained shortest path problem: theory and computational experience. Opsearch 1980;17:93–109.

[20] Irnich S, Desaulniers G. Shortest path problems with resource constraints. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column Generation. New York: Springer; 2005. p. 33–65.
[21] Jepsen M, Petersen B, Spoorendonk S, Pisinger D. Subset-row inequalities applied to the vehicle-routing problem with time windows. Oper Res 2008;56:497–511.
[22] Laporte G, Louveaux F. The integer L-shaped method for stochastic integer programs with complete recourse. Oper Res Lett 1993;13:133–42.
[23] Laporte G, Louveaux F, Van Hamme L. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. Oper Res 2002;50:415–23.
[24] Lysgaard J. CVRPSEP: A Package of Separation Routines for the Capacitated vehicle Routing Problem. Technical Report, Aarhus School of Business, University of Aarhus, Department of Business Studies, 2004.
[25] Lysgaard J, Letchford A, Eglese R. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math Progr 2004;100:423–45.
[26] Righini G, Salani M. Symmetry helps: bounded bi-directional dynamic programming. Discret Optim 2006;3:255–73.
[27] Tillman F. The multiple terminal delivery problem with probabilitic demands. Transp Sci 1969;3:192–204.