Estimation of Marginal Cost to Serve Individual Customers

Akang Wang[1,2], Jeffrey E. Arbogast[3], Gildas Bonnier[4], Zachary Wilson[4], and Chrysanthos E. Gounaris*[1,2]

[1]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[2]Center for Advanced Process Decision-making, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[3]Air Liquide, Shanghai Innovation Campus, China

[4]Air Liquide, Newark, DE 19702, USA

January 16, 2020

Abstract

This paper proposes a scenario-sampling framework to estimate the expected incremental routing cost required so as to incorporate a target customer into the stochastic supply chain network. The cost estimate is shown to converge to its true value with statistical guarantee as the number of samples increases, while the Hoeffding's inequality can be utilized to determine a sufficient sample size for a desired estimation accuracy. Inspired from a real-life setting arising in distribution of industrial gases, we sample instances of multi-depot vehicle routing problems with inter-depot routes, and we use these as scenarios towards a demonstration of the marginal cost estimation framework and towards a detailed study to elucidate the quality of our estimates. In order to solve such rich routing problems exactly, we also develop a tailored branch-price-and-cut algorithm, which is shown to be able to solve to optimality instances of up to 70 customers within reasonable time, significantly outperforming the previous state-of-the-art exact method.

Keywords: marginal cost estimation, multi-depot vehicle routing problems with inter-depot routes, branch-price-and-cut, scenario-sampling, Hoeffding's inequality

*Corresponding author: gounaris@cmu.edu

1 Introduction

Supply chain logistics contribute a significant portion of total cost for many businesses, and thus receive a lot of effort for optimization. In the literature, lots of attention has been paid to optimize supply chain systems in strategic, tactical and operational levels, usually with the objective of obtaining designs that are the most economical in the long run. The most classic example is the one where, given a distribution center, a fleet of vehicles and a group of customers to be served, we aim to identify the minimum-cost routes for vehicles to traverse, such that customer demands are satisfied and relevant system constraints, e.g., vehicle capacities, time windows, and route duration limits, among others, are respected. A wide variety of vehicle routing problems have been addressed in the literature by both exact and heuristic approaches [38]. However, little effort has to-date been devoted to understanding the marginal cost of serving an individual customer, i.e., the incremental cost of delivering to an extra customer on top of the current supply chain design. This problem is of great significance in the following business situations: (i) the distributor wants to estimate the customer lifetime value, in order to distinguish profitable customers; (ii) the distributor wants to understand the expected marginal cost of serving a new customer for pricing purposes. We highlight that our focus in this paper is on *cost estimation*, which can be informative in the context of selecting customer portfolios and/or designing updates to the distribution network; we do not consider *cost allocation* of past operations, which is often sought for purposes of fair accounting.

Estimating the incremental distribution cost of serving a specific customer is challenging, mainly because of the following two reasons. First, the extra cost emanates from the change in routing due to the additional service that must be performed, and thus, identifying the marginal cost in a deterministic setting usually entails solving two $\mathcal{NP}$-hard vehicle routing problems (one accounting for the customer in question, and one not involving that customer). Second, the supply chain network of interest is intrinsically stochastic. For example, not all potential customers request a visit on the same day. Furthermore, the amount of goods to be delivered to customers usually varies on a daily basis. Hence, on different days the distributor might face completely different delivery challenges, as it aims to implement least-cost routing plans that often possess little geographical similarity. In this setting, the addition of a new customer may on certain days be "almost free" because the new customer "fits well" with otherwise optimal routes, but may be completely cost-prohibitive on other days when, for example, the new customer is far away from the existing

customers or when the demand of the new customer cannot be served as "balance load" of existing routes. This leads to the situation where cost allocation to a specific customer in a long term becomes much more perplexing, having to account for such intrinsic stochasticity.

Cost estimation in a stochastic supply chain system is an interesting problem and was approached in the literature mainly from a machine learning perspective. The work of [14] studied approximations to the average travel distance of vehicle routing problems with varying numbers of customers, demands and locations. The authors proposed six approximations with the total number of customers, the customer dispersion area, the average distance between customers and the depot, and the number of vehicles being input variables, and then utilized linear regression to estimate parameters. In [39], the authors focused on analyzing the effect of the geographical dispersion of potential customers on the expected routing cost of the distribution system and derived four estimates using continuous distance approximation [10]. The work of [22] considered the prediction of the cost to serve new customers in the industrial gas business and addressed this problem by first applying a supervised learning technique to group customers based on their features and then utilizing a linear regression model to forecast the distribution cost. A similar approach was adopted in the work of [36]. The authors first identified seven important customer features including neighborhood density, latitude and longitude, among others, and then utilized those to build a predictive model. The work of [26] considered the inventory routing cost allocation problem in a deterministic setting from a cooperative game theory perspective and proposed four cost allocation mechanisms to determine a cost-to-serve for customers.

In this work, we propose a novel scenario-sampling approach that aims to rigorously account for the intrinsic stochasticity in the supply chain system. The basic idea is that the expected value of a given function can be approximated by a sample average estimate derived from random samples. This approach has been widely used for histogram construction [7], function estimation [16], and stochastic optimization [21], among many other settings. In particular, the work of [21] studied a stochastic optimization problem with its objective being an expected function. The authors proposed to approximate the expected objective by its corresponding sample average function and then showed that, with probability approaching one exponentially fast with increase of the sample size, an optimal solution of the sample average approximation problem provides an exact optimal solution of the "true" problem. In our context, the objective is to identify the expected marginal cost

of serving an extra customer in a stochastic supply chain network. Our scenario-sampling framework first generates representative daily delivery scenarios that can simulate the stochasticity in customer presence and demands, and it then quantifies the incremental cost of delivering to the target customer in each such deterministic scenario. The sample average is then returned as an estimate of the expected marginal cost. We estimate the number of required scenarios that need to be sampled by utilizing the Hoeffding's inequality [17], which statistically guarantees that we identify a good estimate of the true marginal routing cost. We highlight that, in contrast with previous works on distribution cost estimation which are mainly based on machine learning principles and lack statistical guarantees [14,22,36,39], our approach combines classic probability theory with exact vehicle routing techniques and thus provides statistical insurances for the estimation accuracy.

In order to quantify the extra cost of serving a specific customer in each sampled scenario, we need to solve routing problems. The routing application we face is a distribution problem arising in the industrial gas business where trucks can be replenished at intermediate production plants and each truck usually performs several trips during a day shift. This setting can be mapped to the *multi-depot vehicle routing problem with inter-depot routes* (MDVRPI) [9]. In our context, we define a *trip* to be a sequence of nodes with "depots" only being its starting and ending nodes, and we define a *route* to be a feasible combination of trips assigned to a vehicle. The MDVRPI is a variant of the *multi-depot vehicle routing problem* in which vehicles are allowed to be replenished at intermediate depots along their routes, while the route duration limit is respected. This problem was formally defined in the work of [9], in which the authors encountered a real-life application of the MDVRPI in a grocery distribution problem. Other important applications arise in municipal service, especially in waste collection problems [1,5,20], where vehicles have to renew their capacities by uploading the waste at one of the treatment plants and return to the depot only when the work shift is over. Interested readers are referred to the recent work of [35] for a comprehensive review on routing problems with intermediate stops and their applications.

In the literature, the scientific research on solving the MDVRPI is dominated by heuristic methods [5,9,37]. In the work of [9], the authors proposed a three-phase methodology: (i) adaptive memory and tabu search are applied to generate a set of routes; (ii) an integer program based on the set-partitioning formulation is executed to determine a minimum-cost routing design; (iii) a post-optimization phase is performed in an attempt to improve the solution. The authors of [37]

renamed the MDVRPI to be the *vehicle routing problem with intermediate replenishment facilities* to emphasize the use of a single central station for the fleet, and they proposed a three-step meta-heuristic algorithm in which tabu search and variable neighborhood search are combined to improve the solution quality after the construction heuristic and a guided local search, combined with a customer removal and reinsertion procedure, is applied to produce the final solution. Their approach was tested on the set of 12 benchmark instances used in [9] and generated 6 new best known solutions. Finally, in [5], the authors considered a waste collection vehicle routing problem with time windows and proposed an adaptive large neighborhood search method.

To the best of our knowledge, the only exact approach for solving the MDVRPI is a branch-and-price algorithm proposed in [25]. The authors modeled the MDVRPI as a *set covering* formulation in which variables are routes corresponding to feasible combinations of trips, and they proposed two pricing subproblems to generate routes. The first one generates routes directly by solving an *elementary shortest path problem with resource constraints* (ESPPRC), while the second one first enumerates all non-dominated trips by solving an ESPPRC and then combines trips into routes by exploiting the relationship between the sets of trips and routes. Their computational studies showed that the second pricing mechanism performs better and their branch-and-price algorithm could solve MDVRPI benchmark instances involving up to 50 customers to optimality.

Whereas satisfactory for instances of the demonstrated sizes, the enumeration step becomes too prohibitive to execute when the number of customers grows larger, and hence, in this work we adopt the first pricing mechanism, which we evolve with several modifications. Furthermore, we develop a *branch-price-and-cut* (BPC) algorithm for exactly solving the MDVRPI. Our BPC algorithm has the following distinctive features from the work of [25]: (i) we enforce only partial elementarity and solve a *shortest path problem with resource constraints* (SPPRC) to generate routes with negative reduced costs, given the fact that the ESPPRC is strongly $\mathcal{NP}$-hard, while the relaxed one is still $\mathcal{NP}$-hard but in a weak sense [12, 32]; (ii) we enhance it with several state-of-the-art pricing techniques, including *ng*-routes, variable fixing and routing enumeration; and (iii) we incorporate rounded capacity inequalities and limited-memory subset row cuts as strengthening constraints.

Finally, we remark that the MDVRPI generalizes another well-studied problem, namely the *multi-trip vehicle routing problem* (MTVRP).[1] The MTVRP extends the classical vehicle routing

[1]In the literature, the MTVRP is also named as the *vehicle routing problem with multiple use of vehicles* and it has been well studied by both heuristic and exact approaches. Interested readers are referred to [6] for a comprehensive

problem inasmuch as each vehicle is allowed to perform several trips, subject to route duration constraints. Clearly, the MDVRPI reduces to the MTVRP when the number of depots is equal to one, and consequently, our proposed BPC algorithm can also be applied to solve the MTVRP as well as a number of its variants.

In summary, the distinct contributions of our work are as follows.

- We propose a scenario-sampling framework to estimate the expected marginal cost of serving individual customers. Specifically, we obtain independent scenarios by sampling customer demands from their distribution and consider the sample average as an estimate of the cost. We prove that our proposed framework provides statistical guarantee of the estimation accuracy, provided that a sufficiently large–informed by Hoeffding's inequality–sample size has been obtained.

- We model the MDVRPI as a set partitioning formulation and propose a branch-price-and-cut algorithm that incorporates several state-of-the-art techniques, including *ng*-routes, variable fixing, route enumeration, and limited-memory subset row cuts, among others.

- We conduct computational studies to show that our branch-price-and-cut algorithm significantly outperforms the state-of-the-art exact approach for the MDVRPI. In particular, our algorithm proves optimality for all previously open MDVRPI benchmark instances involving up to 40 customers. We further push the envelope by extending the literature benchmarks with 70-customer instances and solving to proven optimality the vast majority of those as well.

- We demonstrate the scenario-sampling framework and the quality of its cost estimates, utilizing thousands of MDVRPI instance samples in each case, and we elucidate the effect on the marginal routing cost of factors such as customer locations and demand levels.

The remainder of the paper is organized as follows. In Section 2, we provide a formal problem definition. In Section 3, we propose our scenario-sampling framework and prove its validity through the lens of probability theory. In Section 4, we discuss the implementation details of our proposed BPC algorithm. Section 5 presents computational results on the BPC algorithm's performance as

review on the MTVRP and its variants.

well as a detailed analysis of the marginal cost estimation framework. Finally, we conclude our work with some remarks in Section 6.

2 Problem Definition

Consider a supply chain network where n customers with stochastic demands might be served on a daily basis. Let $\xi \in \mathbb{R}^n_{\geq 0}$ denote the corresponding demand vector and we assume that ξ is drawn from some given probability distribution, e.g., $\xi \sim \Xi$. We emphasize that a customer's demand can be zero on a given day, which indicates that this customer does not request a visit. Now, let us assume that the distributor wants to serve an "additional" customer[2] whose demand, denoted by $\vartheta \in \mathbb{R}_{>0}$, is also stochastic and is drawn from some given probability distribution, e.g., $\vartheta \sim \Theta$. We further assume that ξ and ϑ are independent and that, for any $(\xi, \vartheta) \sim \Xi \times \Theta$, the resulting routing problem is always feasible; that is, there always exists a feasible routing plan that can be implemented, such that customer demands are satisfied and applicable system constraints are respected. The distributor aims to figure out how much marginal cost will be incurred on average when serving the additional customer with demand ϑ. In other words, the objective is to quantify the expected incremental cost of incorporating the additional customer into the current supply chain network, while all other customer demands are stochastic. We do not describe the specific routing setting here, since our proposed framework is generic and remains valid irrespective of this aspect. A detailed definition of the routing problem of interest to us is deferred to Section 4.

3 Proposed Framework

In this section, we first properly define the marginal cost and then present our scenario-sampling framework to estimate it. We also prove the validity of this framework using probabilistic arguments, as well as we present a detailed procedure for its deployment.

3.1 Marginal Cost Estimation

Consider a scenario $(\xi, \vartheta) \sim \Xi \times \Theta$. The marginal cost under this specific scenario is defined as

$$\texttt{MC}\,(\xi, \vartheta) := \frac{\texttt{VRP}\,(\xi, \vartheta) - \texttt{VRP}(\xi, 0)}{\vartheta}. \tag{1}$$

[2]This customer may be either an existing or a prospective one.

Here, for a given supply chain network with customer demands fixed to be ξ, $\texttt{VRP}(\xi, 0)$ and $\texttt{VRP}(\xi, \vartheta)$ denote the optimal routing costs before and after serving an extra customer with demand ϑ, respectively. Note that the marginal cost in our definition is normalized by ϑ, representing the incurred incremental cost per unit of extra demand served.[3] Also note that, with the above definition, we make no assumption regarding the sign of the marginal cost. In fact, although in the vast majority of settings the marginal cost is non-negative, i.e., one cannot save costs by performing more service, it is conceivable that, due to some special contract activation or some other synergy between the extra customer with the rest of customers to be served, the marginal cost attains a negative value.

In any case, it is important to emphasize that the optimal routing costs in the numerator of equation (1) have to be computed from exact approaches to appropriately formulated vehicle routing problems. Despite being much cheaper to compute via efficient (meta-)heuristic algorithms, heuristic solutions cannot be used in the definition of the marginal cost, since their lack of optimality guarantee might result in a comparison between optimal and suboptimal costs, causing $\texttt{MC}(\xi, \vartheta)$ to attain a misleading value.

The goal of this work is to estimate $\mathbb{E}_{\xi,\vartheta}[\texttt{MC}(\xi, \vartheta)]$ numerically, which can be very challenging to compute exactly. First, there is no closed-form formula for $\texttt{MC}(\xi, \vartheta)$ for a given (ξ, ϑ), since it entails solving two $\mathcal{NP}$-hard routing problems, in general. Second, (ξ, ϑ) is a high-dimensional vector, and thus, computing $\mathbb{E}_{\xi,\vartheta}[\texttt{MC}(\xi, \vartheta)]$ exactly would entail high-dimensional integration, which is impractical.

Let (ξ^1, ϑ^1), (ξ^2, ϑ^2), $\ldots$, (ξ^N, ϑ^N) be an independent and identically distributed random sample of N realizations of the demand vector $(\xi, \vartheta) \sim \Xi \times \Theta$. For each sample $\left(\xi^i, \vartheta^i\right)$, we first solve two $\mathcal{NP}$-hard vehicle routing problems exactly to obtain $\texttt{VRP}\left(\xi^i, 0\right)$ and $\texttt{VRP}\left(\xi^i, \vartheta^i\right)$ and then compute the specific sample's marginal cost, $\texttt{MC}\left(\xi^i, \vartheta^i\right)$, using equation (1). We consider the average of $\texttt{MC}\left(\xi^i, \vartheta^i\right)$ over N samples as our estimate of the expected marginal cost. The sample average estimate, $\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right)$, coincides with the true value, $\mathbb{E}_{\xi,\vartheta}[\texttt{MC}(\xi, \vartheta)]$, with very high probability, when the sample size N is large enough. See Proposition 1.

Proposition 1. *Given any* $\delta > 0$,

$$\lim_{N\to+\infty} \text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right) - \mathbb{E}_{\xi,\vartheta}[\texttt{MC}(\xi, \vartheta)]\right| > \delta\right) = 0. \tag{2}$$

[3] A sampled ϑ value is always strictly positive, since a zero demand would be meaningless in this context.

Proof. Since (ξ^1, ϑ^1), (ξ^2, ϑ^2), $\ldots$, (ξ^N, ϑ^N) are independent and identically distributed, and since $\texttt{MC}(\xi, \vartheta)$ is a function of (ξ, ϑ), then $\texttt{MC}\left(\xi^1, \vartheta^1\right), \texttt{MC}\left(\xi^2, \vartheta^2\right), \ldots, \texttt{MC}\left(\xi^N, \vartheta^N\right)$ are also independent and identically distributed. By the *weak law of large numbers*, the average of independent and identically distributed random variables converges in probability to the expectation; that is,

$$\frac{1}{N}\sum_{i=1}^{N} \texttt{MC}\left(\xi^i, \vartheta^i\right) \xrightarrow{p} \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right], \quad \text{when } N \to +\infty.$$

The above implies that (2) holds for any specified margin $\delta > 0$, no matter how small. □

We remark that, since all N samples are randomly chosen, the sample average converges to the true value in a probabilistic sense, but not in a deterministic one. Whereas the convergence within a permissible deviation, δ, is only guaranteed to occur when the sample size N approaches infinity, in practice one has to settle with using finitely many samples. In the following section, we seek to determine a finite sample size that would be deemed sufficient to yield an estimate of high quality.

3.2 Bounding the Sample Size

The immediate question after Proposition 1 is how fast is the convergence, namely how fast does the probability defined in equation (2) decrease with the increase in sample size N. We address this question in Proposition 2.

Before we present it, we introduce two quantities, $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$, to denote valid lower and upper bounds for $\texttt{MC}(\xi, \vartheta)$, respectively; that is, $\texttt{MC}\left(\xi, \vartheta\right) \in \left[\underline{\texttt{MC}}, \overline{\texttt{MC}}\right]$, for any $(\xi, \vartheta) \sim \Xi \times \Theta$. The boundedness of $\texttt{MC}(\xi, \vartheta)$ is ensured by the following two observations: (i) $\texttt{VRP}\left(\xi, \vartheta\right)$ and $\texttt{VRP}\left(\xi, 0\right)$ are both finite, due to the feasibility assumption from Section 2; and (ii) in practice, ϑ is bounded from below by some positive value, since the target customer has to order some minimum amount of product each time it requires service.

We remark that the tightest bounds for $\texttt{MC}\left(\xi, \vartheta\right)$ correspond to the solution of the following optimization problems:

$$\min/\max_{(\xi,\vartheta)\sim\Xi\times\Theta} \texttt{MC}\left(\xi, \vartheta\right) \tag{3}$$

However, since $\texttt{MC}\left(\xi, \vartheta\right)$ does not assume an a-priori known form, solving (3) is generally intractable. Fortunately, one can usually deduce practically-relevant, valid bounds for $\texttt{MC}\left(\xi, \vartheta\right)$. As we pointed out earlier, in the vast majority of applications, the marginal cost is always non-negative. In that

case, one can choose $\underline{\texttt{MC}} = 0$ as a safe lower bound. On the other hand, a safe upper bound $\overline{\texttt{MC}}$ can be the normalized cost of performing as many dedicated round-trips as necessary to deliver the least allowable amount of product to the target customer. In this calculation, penalties could also be incorporated for the use of more-than-available vehicles, to reflect overtime and/or outsourcing costs usually incurred in such situations.

Assume now $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$ can be deduced for the routing problem of interest. Proposition 2 applies.

Proposition 2. *Given any $\delta > 0$, then*

$$\text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i,\vartheta^i\right) - \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi,\vartheta\right)\right]\right| > \delta\right) \leq 2\exp\left(-\frac{2N\delta^2}{\left(\overline{\texttt{MC}}-\underline{\texttt{MC}}\right)^2}\right). \tag{4}$$

Proof. Since $\texttt{MC}\left(\xi^i,\vartheta^i\right)$ are independent random variables such that $\texttt{MC}\left(\xi^i,\vartheta^i\right) \in \left[\underline{\texttt{MC}},\overline{\texttt{MC}}\right]$, for all $i = 1, 2, \ldots, n$, applying *Hoeffding's inequality* [17] yields

$$\text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i,\vartheta^i\right) - \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\xi^i,\vartheta^i}\left[\texttt{MC}\left(\xi^i,\vartheta^i\right)\right]\right| > \delta\right) \leq 2\exp\left(-\frac{2N^2\delta^2}{\sum_{i=1}^{N}\left(\overline{\texttt{MC}}-\underline{\texttt{MC}}\right)^2}\right),$$

noting that the right hand side can be simplified by a factor of N. Furthermore, since $\texttt{MC}\left(\xi^i,\vartheta^i\right)$ are also identically distributed, we have that

$$\mathbb{E}_{\xi^i,\vartheta^i}\left[\texttt{MC}\left(\xi^i,\vartheta^i\right)\right] = \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi,\vartheta\right)\right] \text{ for all } i = 1, 2, \ldots, n.$$

Hence, the above inequality results into (4), which completes the proof. □

Proposition 2 shows us that, as the sample size N increases, the probability that our estimate deviates from the true value more than some permissible value, δ, decreases. In fact, the probability that the sample average estimate approaches the "true" value of $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi,\vartheta\right)\right]$ increases exponentially fast with the sample size N.

Proposition 2 also implies a way to derive the sample size for a desired accuracy. Let us choose a probability threshold, $\alpha \in (0,1)$, and use (4) to deduce the sample size N necessary for the probability to be at most α. By requiring that the right-hand side of (4) be less than or equal to α, we obtain that

$$N \geq \frac{\left(\overline{\texttt{MC}}-\underline{\texttt{MC}}\right)^2}{2\delta^2}\ln\left(\frac{2}{\alpha}\right). \tag{5}$$

A key observation from (5) is that the sufficient sample size depends logarithmically on the probability threshold α, causing N to increase only mildly as α decreases. In contrast, we observe that the sample size should grow polynomially as a stricter permissible deviation δ is required.

3.3 A General Framework

Our scenario-sampling framework works as follows. We first choose a desired permissible deviation, $\delta > 0$, and a probability threshold, $\alpha \in (0, 1)$. Then, we deduce valid values for $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$, and using inequality (5), we identify a bound on the sample size, denoted by $\overline{N}$. We then sample $\overline{N}$ scenarios, each having customer demands (ξ, ϑ) from the demand distribution $\Xi \times \Theta$.[4]

For each deterministic scenario, we solve two routing problems exactly to obtain $\texttt{VRP}(\xi, 0)$ and $\texttt{VRP}(\xi, \vartheta)$, and then calculate the marginal cost $\texttt{MC}(\xi, \vartheta)$ using the formula (1). We finally compute the sample average $\frac{1}{\overline{N}} \sum_{i=1}^{\overline{N}} \texttt{MC}\left(\xi^i, \vartheta^i\right)$ as the estimate of the true value $\mathbb{E}_{\xi,\vartheta}[\texttt{MC}(\xi, \vartheta)]$. The overall framework is illustrated in Fig. 1.

Our proposed framework possesses a number of notable features. Firstly, the estimated value will always converge to the true marginal cost with probabilistic guarantee. Secondly, the framework only assumes that the customer demand distributions Ξ and Θ are given, but it does not require that they belong to any particular probability distribution family, such as Guassian or any other distribution commonly used in similar contexts. Finally, the framework is generic in terms of the type of routing problems one faces, and its validity does not depend on the type of algorithm one uses to obtain provably optimal cost evaluations.

4 Solving Routing Problems

We now turn our attention to discussing the exact approach for solving the routing problems of interest. In this work, we focus on the *multi-depot vehicle routing problem with inter-depot routes*, which extends the multi-depot vehicle routing problem inasmuch as the vehicles are allowed to stop at intermediate depots for replenishment. The examined problem was first introduced in the work

[4]We remark that a perfect description of the demand distribution Ξ is seldom available in practice, but distributors often keep historical delivery data for each customer. This data can be used to extract the distribution using many well-known parametric and non-parametric approaches. In this work, we do not explore these techniques but refer interested readers to [41] for details.

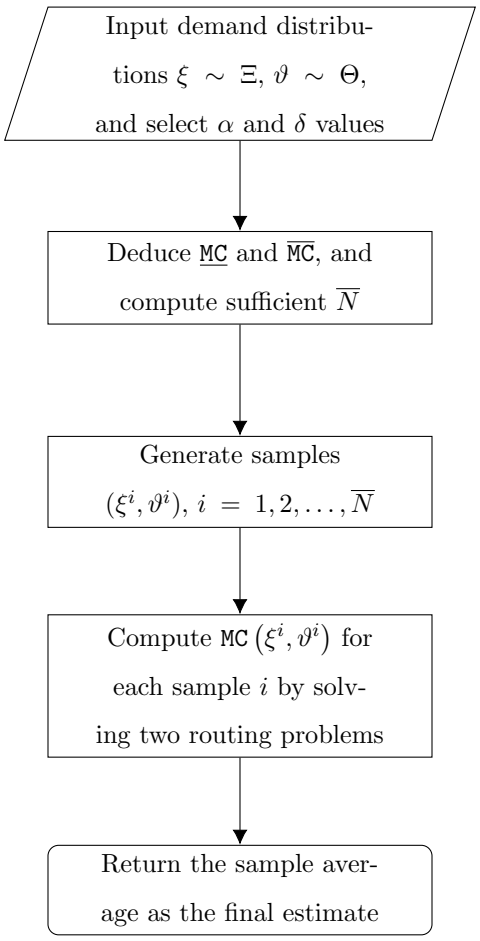

Figure 1. A scenario-sampling framework

of [9]. Note that, in the work of [37], the authors proposed an alternative name, the *vehicle routing problem with intermediate replenishment facilities*, to emphasize both the replenishment role of the intermediate facilities and the use of a single central station for the fleet of vehicles. We follow the work of [9] and use MDVRPI to denote the routing problem of our interest.

Before presenting the exact algorithm, we first formally define the MDVRPI. Let a directed graph $G = (V, A)$, where $V := V_c \cup V_d$ denotes the set of nodes, consisting of customers $V_c := \{1, 2, \ldots, n\}$ and depot facilities $V_d := \{n+1, n+2, \ldots, n+m\}$, and $A := \{(i, j) : i \in V_d, j \in V_c\} \cup \{(i, j) : i \in V_c, j \in V \setminus \{i\}\}$ denotes the set of arcs. Let $c_{ij} \in \mathbb{R}_{\geq 0}$ and $t_{ij} \in \mathbb{R}_{\geq 0}$ represent the cost and time, respectively, for a vehicle to traverse arc $(i, j) \in A$. We assume that the depot facilities have an unlimited supply of goods. A homogeneous fleet of K vehicles of capacity $Q \in \mathbb{R}_{>0}$ is available, with the exact allocation of these vehicles to the depots to be determined by the optimizer. Each customer $i \in V_c$ is associated with a demand $q_i \in \mathbb{R}_{>0}$ and a service duration $s_i \in \mathbb{R}_{\geq 0}$, while a vehicle docking time $\tau_i \in \mathbb{R}_{\geq 0}$ applies at each depot $i \in V_d$.[5] In the MDVRPI, a feasible vehicle route starts from and ends at the same depot, but vehicles are allowed to stop at any depot to replenish and continue with another trip. The vehicle capacity Q has to be respected during each trip, while the total duration of the route cannot exceed a predefined limit $T \in \mathbb{R}_{\geq 0}$. The objective is to identify the minimum-cost set of routes for the fleet of available vehicles to traverse so that each customer is visited exactly once with its demand served, while vehicle capacities and route duration limits are respected.

4.1 Set Partitioning Model

A sequence of vertices $\{i_1, i_2, \ldots, i_p\}$ visited by a vehicle is called a *feasible route*, if $i_1 = i_p \in V_d$, $(i_l, i_{l+1}) \in A$ for $1 \leq l \leq p-1$, and the following conditions are satisfied: (i) each customer vertex is visited at most once (elementary); (ii) the vehicle capacity Q is respected in each trip; (iii) the total route duration does not exceed T. Let R denote the set of all feasible routes and let c_r denote the cost of traversing route $r \in R$ by a vehicle. Let the parameter δ_{ir} denote the number of times customer $i \in V_c$ is covered in route $r \in R$. Let λ_r be a binary variable indicating whether route $r \in R$ is selected in the optimal solution. The MDVRPI can be formulated as the following *set*

[5] Without loss of generality, customer service durations and depot docking times can be suitably incorporated into the travel times along the arcs; moving forward, we always do so for convenience.

partitioning model (6)–(9).

$$\underset{\lambda_r}{\text{minimize}} \quad \sum_{r \in R} c_r \lambda_r \tag{6}$$

$$\text{subject to} \quad \sum_{r \in R} \delta_{ir} \lambda_r = 1 \qquad \forall i \in V_c \tag{7}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{8}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R \tag{9}$$

The objective function (6) is to minimize the total cost for selected routes. The degree constraints (7) guarantee that every customer is served exactly once, while the fleet size constraint (8) enforces that at most K vehicles are used. Finally, constraints (9) enforce binarity of variables.

It is well-known that one can relax the feasible space of the above set partitioning model by including non-elementary vehicle routes in R without sacrificing optimality. In our implementation, we replace R with the set of so-called ng-routes, $R^{ng} \supseteq R$, which are not necessarily elementary [4]. Since there exist exponentially many ng-feasible routes, the formulation (6)–(9) is a mixed-integer linear programming model with a very large number of binary variables. As a result, the linear programming (LP) relaxations at each node of the branch-and-bound tree are of large sizes such that they have to be tackled via an efficient *column generation* method. Valid inequalities are dynamically separated and added so as to strengthen the LP relaxations, yielding a *branch-price-and-cut* algorithm to solve the set-partitioning model. Details on this algorithm are presented below.

4.2 Branch-Price-and-Cut Algorithm

In the BPC algorithm, the LP relaxations at every node in the branch-and-bound tree are solved via column generation, while cutting planes are added to strengthen the relaxations. Our implementation incorporates several elements of the algorithm described in the work of [29], including ng-routes [4], variable fixing [18], route enumeration [3], and limited-memory subset row cuts [28], among other features. In what follows, we highlight only the most important of these ingredients; for more details, we refer readers to the works of [27], [29], and [34].

We first replace the binarity constraints (9) in the set partitioning model by non-negativity constraints and obtain its LP relaxation, which is usually referred to as the *master problem*. The

master problem has a very large number of variables, as exponentially (to number of customers) many feasible routes are typically available. Generating all of these routes to explicitly define the master problem is obviously impractical, thus we resort to column generation. Given the premise that most of the variables will be non-basic, and assuming non-zero values in the optimal solution, only a subset of variables need to be considered in practice when solving the problem. Column generation leverages this idea and aims to generate only the variables that have the potential to improve the objective function, i.e., variables with negative reduced costs. In our context, columns correspond to *ng*-feasible routes, and we will use these terms interchangeably. We first consider a *restricted master problem* (RMP) defined by a subset of *ng*-routes $\tilde{R}^{ng} \subseteq R^{ng}$. After optimizing the RMP, columns with negative reduced costs will be appended to $\tilde{R}^{ng}$ and the resulting RMP will be reoptimized. This procedure iterates until no such column exists. In that case, we say that column generation converges and the master problem has achieved optimality. If at that point the solution to the master problem is fractional, we separate and add valid inequalities, or resort to branching. The overall BPC algorithm is illustrated in Fig. 2.

4.2.1 Initialization

To start the algorithm, we utilize single-customer routes to build $\tilde{R}^{ng}$, which are used to define the initial RMP. We highlight that, by constructing $\tilde{R}^{ng}$ in this way, the initial RMP is not necessarily guaranteed to be feasible, e.g., due to the fleet size constraint (8). If that is the case, we can seek for a feasible RMP via the "feasibility-recovery" step [13], which will be discussed in Section 4.2.5.

4.2.2 Solving Pricing Subproblems

After the RMP is solved, we need to check whether it is necessary to enlarge $\tilde{R}^{ng}$ to include some neglected *ng*-feasible routes so as to improve the objective value of the RMP. This entails identifying columns with negative reduced costs, which is achieved by solving *pricing subproblems*. Since vehicle routes in R^{ng} can be categorized by their start depots, we consider to solve a pricing subproblem for each depot $i \in V_d$. Without loss of generality, let vertex $n+1$ denote the depot of interest.

In our context, the pricing subproblem can be modeled as a *shortest path problem with resource constraints* (SPPRC) [32]. The SPPRC can be defined on a directed graph $\overline{G} = (\overline{V}, \bar{A})$, where $\overline{G}$

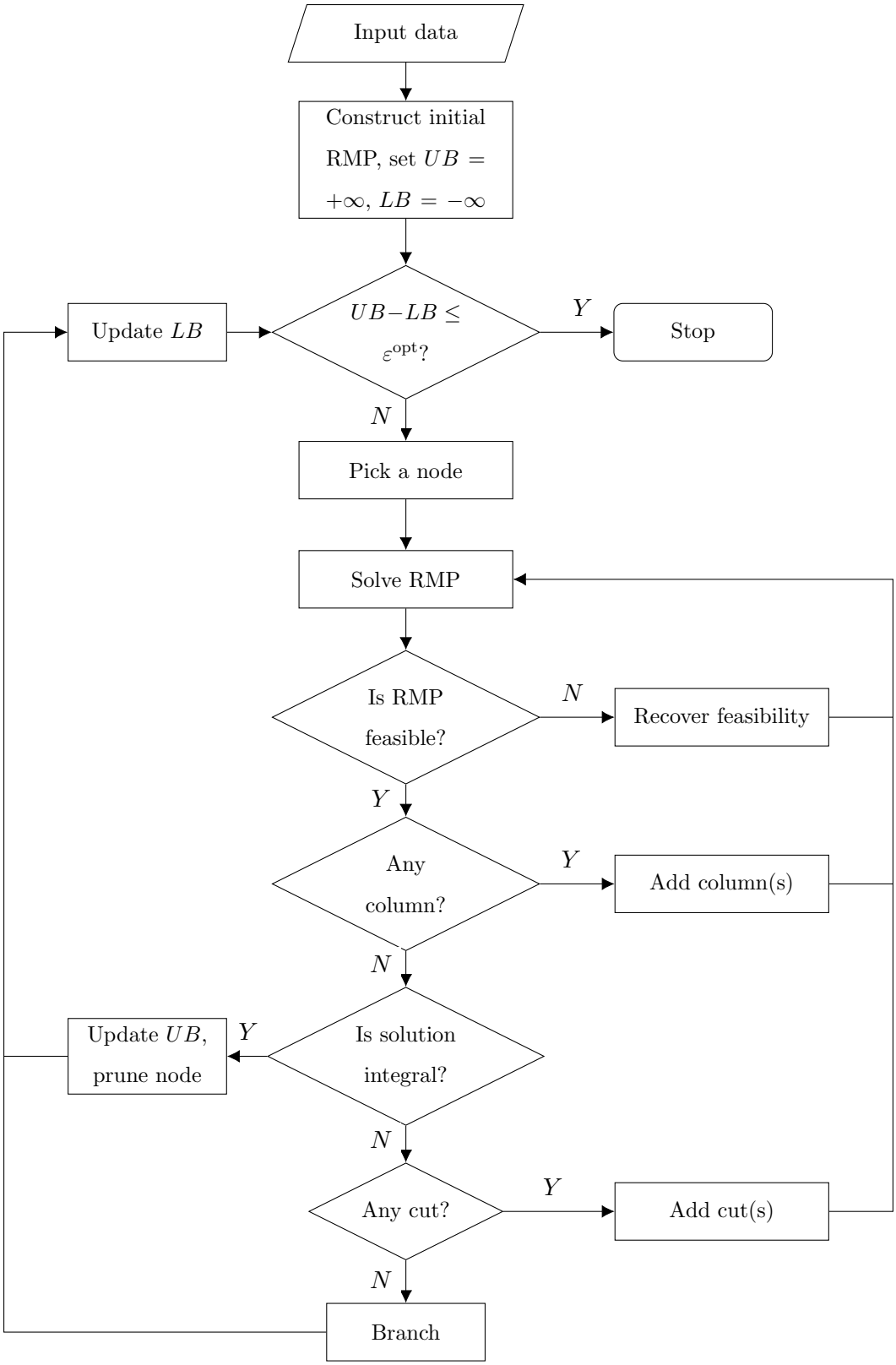

Figure 2. The Branch-Price-and-Cut algorithm.

is obtained by inserting into G two copies of the chosen depot, 0 and $n+m+1$, to represent a virtual start depot and a virtual destination depot, respectively, and by adding edges $(0, n+1)$ and $(i, n+m+1)$ for $i \in V_c$. Clearly, in such a configuration, a feasible route always starts from vertex 0 and ends at $n+m+1$, the chosen depot vertex $n+1$ being the immediate vertex after the starting node, and all depot vertices $i \in V_d$ used as intermediate replenishment facilities. Associated with each arc $(i,j) \in \bar{A}$ are the cost $\bar{c}_{ij}$, travel time t_{ij} and demand q_j. Note that $q_j = 0$, if vertex j corresponds to a depot. The cost $\bar{c}_{ij}$ is obtained by properly modifying c_{ij} in order to account for the contribution from current dual values to constraints (7) and (8). We associate each vertex $i \in \overline{V}$ with the load limit Q and the route duration limit T. While arc $(i,j) \in \bar{A}$ is traversed by a path, denoted by P, time resource t_{ij} and capacity resource q_j are consumed. A resource's accumulated consumption until a vertex visited along a path should not exceed its limit. We highlight that, if vertex j represents some intermediate depot facility, then the load of path P is reset to 0 to effectuate the replenishment process. The goal is to determine the minimum-cost path among all paths that start from the vertex 0 and end at the vertex $n+m+1$, such that all resource constraints are respected.

Exact pricing: It is well-known that the SPPRC is $\mathcal{NP}$-hard [12]. The most successful solution approach is a dynamic programming method called the *labeling algorithm*, which has a pseudo-polynomial time complexity [32]. The labeling algorithm works as follows. We associate an *ng*-feasible partial path P with a label, $L(P) = (\bar{c}(P), v(P), d(P), t(P), \Pi(P), pred(P))$, which stores the reduced cost, end vertex, load, duration, a set of forbidden vertices, and a pointer to its predecessor label. For a given pair (i, t') with $i \in \overline{V}$ and integer t' such that $0 \leq t' \leq T$, we define a bucket $B(i, t')$. A label $L(P)$ is stored in bucket $B(v(P), t(P))$. We remark that, if any entry of the travel time matrix t_{ij} is not integral, we then enforce an extra resource constraint in the SPPRC. In particular, we associate arc $(i,j) \in \bar{A}$ and vertex $i \in \overline{V}$ with the integral travel time $\lfloor t_{ij} \rfloor$ and route duration limit $\lfloor T \rfloor$, respectively. We then use the integral travel time duration as our "reference" resource to store labels.

The labeling algorithm is initialized by storing a label $(0, 0, 0, 0, \emptyset, null)$ in bucket $B(0, 0)$. From bucket $B(0, 0)$, we use dynamic programming to fill other buckets, starting with lower values of t'. For each integer t' with $0 \leq t' \leq T$, the algorithm goes through $i \in \overline{V}$ and, for each neighbor j such that $(i,j) \in \bar{A}$, evaluates the extension of the walk represented by $B(i, t')$ to j. Given

a label $L(P)$ from bucket $B(v(P), t(P))$, we now attempt to extend it to vertex $j \in \overline{V} \setminus \Pi(P)$ with $(v(P), j) \in \bar{A}$. We first check whether this is a feasible extension, i.e., whether the resource consumption constraints are respected at vertex j. If that is the case, we obtain a new ng-feasible label and store it in bucket $B(j, t(P) + t_{v(P)j})$. As we have discussed above, the load will be reset to 0, if vertex j denotes a depot.

The set of forbidden vertices should be updated in a similar way as the work of [4], but with some modifications due to the existence of intermediate depot vertices. In our implementation, we associate each customer vertex $j \in V_c$ with an ng-set $NG(j)$ composed of 8 nearest customer neighbors. When extending a label $L(P)$ to a vertex $j \in \overline{V} \setminus \Pi(P)$, its successor's forbidden set is updated to be $(\Pi(P) \cap NG(j)) \cup \{j\}$, if vertex j denotes a customer; otherwise, the successor maintains the same forbidden set as the label itself. The idea behind this is that enforcing forbidden sets and thus ng-routes is to seek for partial elementarity of routes generated in the SPPRC. Since the elementarity of a given route is not affected by the existence of depot vertices, those vertices should be excluded from forbidden sets. When no more label extensions can be made, we collect all columns stored in buckets $B(n + m + 1, t')$ with $0 < t' \leq T$ and return all those with negative reduced costs.

To accelerate the labeling algorithm, it is crucial to check for dominance relationships so as to avoid some unnecessary label extensions. More specifically, before storing a new ng-feasible label, we first check whether it dominates or is dominated by existing labels stored in buckets. We say that a label $L(P_1)$ dominates another label $L(P_2)$ if, for any feasible path extended from $L(P_2)$, we can always find a feasible path extension from $L(P_1)$ and this extended path is not more costly. Sufficient conditions for this are given by (10)–(14).

$$\bar{c}(P_1) \leq \bar{c}(P_2), \tag{10}$$

$$v(P_1) = v(P_2), \tag{11}$$

$$d(P_1) \leq d(P_2), \tag{12}$$

$$t(P_1) \leq t(P_2), \tag{13}$$

$$\Pi(P_1) \subseteq \Pi(P_2). \tag{14}$$

A newly generated label $L(P)$ is saved only if it is not dominated, and existing labels that are dominated by $L(P)$ will be removed. In our implementation, the dominance rule is only tested

between a label $L(P)$ and labels stored in buckets $B(v(P), t')$ with $\max\{0, t(P) - 10\} \leq t' \leq t(P)$.

We solve a pricing subproblem for each depot $i \in V_d$ and append into $\tilde{R}^{ng}$ the up to 20 columns with the most negative reduced costs. Let $\bar{c}_{min}$ denote the minimum cost among the returned columns. Clearly, $\bar{c}_{min}$ is the minimum reduced cost of all ng-feasible routes.

Variable fixing: Let z^* denote the current RMP optimal value. It is well-known that one can obtain a valid lower bound, $z^{LB} := z^* + \min\{K, |V_c|\}\bar{c}_{min}$, also known as the *Lagrangian bound* [24]. We emphasize that the fleet size bound K is updated, whenever branching on the fleet size is executed (see later for details on branching rules). Besides pruning branch-and-bound nodes, we also use Lagrangian bounds for variable fixing.

Let us define the primal-dual gap, $z^{gap} := UB - z^{LB}$. One can easily show that a column with a reduced cost larger than $z^{gap} + \bar{c}_{min}$ will not be part of any feasible solution better than the incumbent [34]; thus, such columns can be excluded from consideration. In our context, we employ a technique called arc fixing [18]. Specifically, given arc $(i, j) \in \bar{A}$, if every feasible column traversing this arc has a reduced cost larger than $z^{gap} + \bar{c}_{min}$, then this arc can be omitted from consideration. To reach such a conclusion and omit an arc, we need to apply the labeling algorithm we discussed above in a backward fashion. The backward labeling algorithm works in a similar way as the regular (forward) labeling, but with the following minor modifications: (i) it is initialized by storing the first label $\{0, n + m + 1, Q, T, \emptyset, null\}$ and starts with higher values of t' for label extension; (ii) the inequality conditions (12) and (13) in the dominance check are reversed; (iii) the dominance relationship is tested between a label $L(P)$ and labels stored in buckets $B(v(P), t')$ with $t(P) \leq t' \leq \min\{t(P) + 10, T\}$.

We separately perform a full run of both forward and backward labeling. The minimum reduced cost of a column passing by an arc $(i, j) \in \bar{A}$, denoted by $\bar{C}_{ij}$, can be obtained by concatenating the labels in $B(i, t')$ from the forward run with the labels in $B(j, t' + t_{ij})$ from the backward run for $0 \leq t' \leq T$. If $\bar{C}_{ij}$ is larger than $z^{gap} + \bar{c}_{min}$, then one can eliminate the arc (i, j) from $\bar{A}$. Since arc fixing is typically effective when z^{gap} becomes small, we begin to fix arcs by reduced costs after the initial convergence of column generation and when z^{gap}/UB is below 5%. Thereafter, arc fixing is performed only if z^{gap} decreases by at least 10% since the last call.

Route enumeration: As [3] suggested, one can enumerate all elementary columns with reduced costs less than $z^{gap} + \bar{c}_{min}$, since only these columns may contribute to a solution better than the incumbent. We adopt the idea from the work of [8] and attempt to enumerate all possible routes when the primal-dual gap becomes small. In particular, after each call of variable fixing, if the primal-dual gap is below 1%, we then attempt to enumerate all columns that have the potential to improve the objective function. In order to enumerate elementary routes using the forward labeling algorithm we discussed above, $NG(i)$ is defined to be the customer set V_c. As a result, $\Pi(P)$ is simply the set of visited customer vertices along path P. Furthermore, as suggested by [29], the dominance rule should be updated in the following way. First, the condition (14) is modified to $\Pi(P_1) = \Pi(P_2)$. Second, we define for each label $L(P)$ an extra member $c(P)$ to indicate the original accumulated travel cost along path P, and then, the condition (10) is revised to be $c(P_1) \leq c(P_2)$.

The route enumeration procedure can be accelerated by using the so-called *completion bound* [8], which can be computed for free from the variable fixing step. For the sake of brevity, we refer readers to the work of [27, 29] for details.

The enumeration procedure is interrupted when the number of labels exceeds 1.0×10^6, or when the number of enumerated routes exceeds 1.0×10^5. After a successful enumeration, generated columns are not immediately appended to $\tilde{R}^{ng}$ but saved in a pool. At this point, one can also erase non-elementary columns from $\tilde{R}^{ng}$. Finally, once the total number of columns from $\tilde{R}^{ng}$ and the pool is less than 1.0×10^4, we append integrality constraints (9) to the RMP and solve the resulting integer program. Otherwise, in the coming iterations, we can simply inspect columns saved in the pool and return at most 150 of them with negative reduced costs.

Heuristic pricing: The column generation procedure can be further accelerated by using a fast heuristic pricing methods called "bucket pruning," which was introduced in the work of [15]. Simply put, this method relaxes the dominance check by only considering the condition (10). The trade-off to this efficiency is that non-dominated labels may be dropped, though it is expected that those would be less likely to lead to optimal solutions. In our implementation, we opted to first apply the heuristic labeling algorithm to generate columns with negative reduced costs, and to resort to the exact algorithm only if the heuristic method fails to identify any column. In particular, in each iteration, we return the up to 30 columns with the most negative reduced costs from heuristic pricing.

4.2.3 Cutting Planes

When column generation converges, the master problem achieves its optimality. If the current LP optimal solution is not integral, we try to identify strengthening inequalities to exclude the fractional solution from the relaxed feasible space. We consider two types of valid inequalities, namely *Rounded Capacity Inequalities* (RCI) [23] and *Subset Row Cuts* (SRC) [19].

The separation routine for RCI is synopsized as follows. Let $G^* = (V, A)$ be a support graph corresponding to the current solution to an RMP. We first shrink the set of depot vertices, V_d, to be a single vertex, which results in a support graph for the classic capacitated vehicle routing problem. We then apply a tabu search algorithm inspired from [2] to identify violated RCI. In the context of a branch-price-and-cut algorithm, RCI are so called "robust" cuts [11] because their corresponding dual values can be properly accommodated into the modified arc cost in the SPPRC; thus, adding RCI into the set partitioning model as strengthening inequalities will not complicate the solution of pricing subproblems.

With regards to SRC, it is well-known that they usually help to close the primal-dual gap significantly; however, these are known to be "non-robust" cuts, since their dual values cannot be easily combined into the arc cost in the SPPRC, changing the pricing subproblem structure. In essence, each additional SRC makes pricing harder. To mitigate the negative effect of SRC on solving SPPRC subproblems, we adopt a variant of SRC, called *limited-arc-memory subset row cuts* (lm-SRC), which was proposed in the work of [28]. The lm-SRC are still non-robust cuts but have much less impact on solving pricing problems [29]. We use the procedure proposed in the work of [30] to separate lm-SRC. In particular, for every $i \in V_c$, we define $N(i) \subseteq V_c$ as the set containing the 15 other customers closest to i and then test the violation of all SRC obtained by applying multipliers to all base sets S with cardinalities 3, 4 and 5, and for each i and j in S, $j \in N(i)$. After violated SRC are identified, we add their corresponding limited-arc-memory version. To accommodate the use of lm-SRC, the labeling algorithm is modified in the following aspects: (i) we define a state for each active cut; (ii) the contributions from dual values to lm-SRC are taken into consideration when computing the reduced cost for a new label; (iii) the condition (10) in the dominance check is updated to account for the dual values to lm-SRC. Readers are referred to [30, 34] for implementation details.

We now discuss our overall cut separation protocol. At the root node, we first attempt to

identify violated RCI and resort to lm-SRC only if no RCI is found or if "tailing-off" is detected for RCI. We define tailing-off when the node primal-dual gap z^{gap} has not improved by at least 0.1% during the past 3 iterations. In all other nodes of the branch-and-bound tree, both RCI and lm-SRC are separated in every round. If the tailing-off condition is satisfied, we stop the cut separation and perform branching. Since, as discussed, the lm-SRC have a negative effect on our ability to keep solving SPPRC subproblems, we add into the set partitioning model only the 5 most violated lm-SRCs per round and add at most 300 lm-SRCs in total. We also remove non-active lm-SRC from the model before each cut separation round. After route enumeration at a given node is completed, all lm-SRC are lifted to their SRC counterparts, since we no longer need to solve the SPPRC.

4.2.4 Branching Strategy

When the optimal solution to the master problem is not integral and no violated cuts can be identified, or if we have decided to stop adding cuts due to the tailing-off issue, we then perform branching. We first attempt to branch on the number of used vehicles, since from our experience it often has a more profound impact on the branch-and-bound node lower bound; if this is not applicable, then branching on edges performed by choosing the edge with value closest to 0.5. Readers are referred to [13] and [33] for details. We emphasize that the branching rules we adopt do not change the pricing structure and, thus, do not add complexity when solving the SPPRC.

4.2.5 Other implementation details

Feasibility-recovery: When the RMP becomes infeasible (e.g., due to branching or because the initial RMP itself is infeasible), we resort to the feasibility-recovery step as suggested by [13], which corresponds to the Phase I step in the classic simplex algorithm. In particular, we introduce a dummy variable into the RMP with its coefficient at each constraint being the right-hand side and its objective coefficient being $UB + \epsilon$, where $\epsilon > 0$ (we used $\epsilon = 0.1$). Clearly, the modified linear program becomes feasible. The resulting linear program is then solved and its dual values are used to generate new columns with negative reduced costs so as to recover the RMP feasibility. When column generation converges, i.e., the LP optimality is achieved, and assuming the dummy variable attains a value of zero, the primal feasibility of the RMP is recovered; otherwise, if the dummy

variable attains a non-zero value, the LP optimal value must be $UB + \epsilon$ (due to complementary slackness), resulting in the node being pruned by bound in the branch-and-bound tree.

Primal heuristics: Whenever the RMP is solved, we apply a simple rounding heuristic in an attempt to find better feasible solutions than the incumbent. In our implementation, elementary columns with their current corresponding LP optimal values above 0.5 are collected and we check whether the routing plan composed of these columns is feasible. If this leads to a feasible solution with an objective value better than the incumbent, we then update the upper bound UB.

5 Computational Studies

Our algorithm was implemented in C++ and all subordinate linear and mixed-integer linear programs were solved using the IBM ILOG CPLEX Optimizer 12.9.0 through the C application programming interface, with all settings being default except that the relative optimality gap tolerance was set to the value of 0. In the branch-price-and-cut algorithm, the absolute optimality gap tolerance was set to $\varepsilon^{\text{opt}} = 1.0 \times 10^{-4}$, and the best-bound node selection strategy was chosen. All computations were performed on an Intel Xeon CPU E5-2689 v4 server running at 3.10 GHz. A total of 128GB of available RAM was shared among 10 copies of the algorithm running in parallel on the server. In Section 5.1, each instance was solved by one copy of the algorithm using a single thread. The results presented in Section 5.2 were obtained with OpenMP by using up to 10 threads in parallel.

5.1 Evaluation of BPC Implementation Performance

We first evaluate the performance of our BPC algorithm on solving MDVRPI benchmark instances and compare our algorithm with the state-of-the-art branch-and-price algorithm proposed in the work of [25]. The differences between these two algorithms can be synopsized as follows:

1. our BPC algorithm models each pricing subproblem as an SPPRC and generates routes with negative reduced costs in a single stage; in contrast, the branch-and-price algorithm considers a two-phase approach where trips between each pair of depots are first constructed and then routes with negatived reduced costs are obtained by combining these trips;

2. our BPC approach incorporates certain state-of-the-art pricing techniques, such as ng-routes, variable fixing, route enumeration, among others;

3. we add valid inequalities to strengthen the LP relaxations at each node of the branch-and-bound tree.

We consider the instances from the work of [9] and we adapt them in the same way as done in [25]. There are 12 `random` (a1-l1) and 10 `CGL` (a2-j2) instances. In these instances, which are available at `http://chairelogistique.hec.ca/data/mdvrpi/`, the number of depots ranges from 3 to 7. For each instance, we consider the first 25, 40 and 70 customers with the total number of vehicles limited to 4, 6 and 10, respectively. Note that datasets a1, d1, and a2 from [9] were not adapted to 70-customer instances because they only feature 48 customers.

In each instance, the vehicle capacity Q and the route duration limit T is set to be 50 and 450, respectively, as done in [25]. Also following that work, the travel time is calculated from the coordinates and we do not round these values. Thus, each entry of the travel time matrix is a double-precision non-negative number. As discussed earlier, in order to use our labeling algorithm in such setting, we need to round down all entries of the matrix and to define an extra resource as the "reference" to store labels in an SPPRC.

Finally, as it is common practice in the VRP literature using BPC algorithms (see, e.g., [15, 28, 29, 31, 34, 40]), tight initial upper bounds are provided so that the various pricing techniques like variable fixing and route enumeration are activated in the early stage of the algorithm. Since we did not have at hand sophisticated heuristics for solving the MDVPRI, we chose to use as the initial upper bound the optimal (or best known) value of each instance modified upwards by an offset of 1.0×10^{-2}. Note that, since we do not round the travel costs, and since 1.0×10^{-2} is larger than the branch-and-bound absolute optimality gap tolerance ε^{opt}, our BPC algorithm always has to locate by itself a feasible solution with a value better than the initial upper bound provided.

Our computational results are presented in Tables 1 and 2. In the instance name field, "X" is a placeholder for the number of customers. For our three different values of "X", the columns "Opt [UB]" then report the corresponding optimal objective values, while the columns "t (sec) [LB]" provide the time to solve the instance to optimality; if an instance could not be solved within the allotted time limit of 2 hours, these columns report (in brackets) the best upper and lower bounds (rounded to 3 decimal places) found within this time limit.

Focusing first on the 25- and 40-customer instances, the tables show that our BPC algorithm solved all 44 instances to optimality, with the solution time ranging from 4 to 279 seconds. This performance significantly improves upon the current state-of-the-art exact approach from [25], which could only solve 23 instances to proven optimality within a time limit of 10 hours, noting however that, according to `www.cpubenchmark.net/singleThread.html`, our computer's computational speed was approximately double that of the machine used in [25]. Focusing on the larger, 70-customer instances, our BPC algorithm could solve to optimality 17 out of a total of 19, with an average solution time of $2,201$ seconds, while the remaining 2 instances yielded an average residual gap of 0.90% at the time limit of 2 hours.

Overall, our computational results demonstrate that our BPC algorithm is able to routinely solve to provable optimality MDVRPI instances with up to 40 customers within a mere couple of minutes, while it can solve most MDVRPI instances involving up to 70 customers within 2 hours. We report our optimal and best known solutions to all these benchmark instances in the following link: `http://gounaris.cheme.cmu.edu/bks/mdvrpi/`.

Table 1. Computational results for our BPC algorithm on 25-, 40-, and 70-customer `random` instances

Instances	X = 25		X = 40		X = 70	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
a1-X-50-450	693.810	4	998.431	22	–	–
b1-X-50-450	731.414*	5	1,059.370	27	1,559.669*	4,968
c1-X-50-450	855.831*	23	1,148.669*	23	1,692.074*	1,951
d1-X-50-450	763.303*	12	1,056.868*	74	–	–
e1-X-50-450	803.715	5	1,236.620	32	1,806.888*	937
f1-X-50-450	551.828	13	854.108*	85	1,365.147*	4,760
g1-X-50-450	654.016	15	1,034.936*	191	1,405.974*	4,979
h1-X-50-450	558.646	21	875.552	76	[1,380.122]	[1,364.404]
i1-X-50-450	823.775*	14	1,222.845	54	1,804.683*	609
j1-X-50-450	777.701*	79	904.281*	109	1,330.820*	3,177
k1-X-50-450	868.604*	10	1,255.505*	164	1,803.937*	2,042
l1-X-50-450	818.280	9	1,085.320	62	1,643.854*	1,129

*Instances solved to optimality for the first time are indicated with an asterisk.

Table 2. Computational results for our BPC algorithm on 25-, 40-, and 70-customer `CGL` instances

Instances	X = 25		X = 40		X = 70	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
a2-X-50-450	725.157*	13	1,010.607	87	–	–
b2-X-50-450	912.429	21	1,238.944	42	1,636.112*	403
c2-X-50-450	683.188	15	1,159.053*	38	1,859.551*	3,981
d2-X-50-450	876.113	17	1,199.848*	36	[1,864.038]	[1,851.572]
e2-X-50-450	699.363*	5	1,087.712*	54	1,801.178*	1,959
f2-X-50-450	781.176	8	1,046.454*	43	1,617.750*	983
g2-X-50-450	793.633*	19	1,035.008	44	1,521.121*	702
h2-X-50-450	716.220	15	940.924	114	1,528.441*	793
i2-X-50-450	910.505	10	1,171.579*	74	1,696.617*	2,381
j2-X-50-450	609.378	20	887.270*	279	1,488.898*	1,656

*Instances solved to optimality for the first time are indicated with an asterisk.

5.2 Marginal Cost Analysis

The previous section demonstrated the effectiveness of our BPC implementation in terms of its ability to solve MDVRPI instances. Being equipped with this state-of-the-art code, we now turn our focus to utilizing the latter as the VRP solver in our framework to estimate the marginal cost to serve individual customers. Since there is no available benchmark instance in the literature for this setting, we generated those as follows. We consider the first five `random` instances we used in Section 5.1, namely instances a1-25-50-450 through e1-25-50-450, each having 25 customers, and in every instance we inserted a 26^{th} customer as our target customer to perform the marginal cost analysis. The target customer's coordinates, fixed service time and nominal demand are chosen to be the rounded average (rounded to the nearest integer) of the 25 original customers, respectively.

The stochastic customer demands are designed in the following way. Firstly, since not all 25 original customers generally request a visit on the same day, we assign each of them a probability to indicate how likely the customer is to request an order on any given day. In particular, we assign 75%, 50% and 25% to the first 6, intermediate 12 and last 7 customers, respectively. The target customer always requests an order on any given day, and is thus given a probability of 100%.

Secondly, for any customer i who requests an order, we consider the demand from the instance data as its nominal value, denoted by $\bar{q}_i$, and assume that the stochastic demand q_i is an integer value that always falls into the set $\mathcal{J}_i := \left[\left\lceil 0.7\bar{q}_i \right\rceil, \left\lfloor 1.3\bar{q}_i \right\rfloor\right] \bigcap \mathbb{N}$. Let p_{ij} denote the probability that q_i takes a specific value j from this set. We set p_{ij} to be proportional to the probability density function of a normal distribution with mean $\bar{q}_i$ and standard deviation $0.1\bar{q}_i$, evaluated at q_i, and normalized such that $\sum_{j \in \mathcal{J}_i} p_{ij} = 1$. We highlight that, although the demand distribution $\Xi \times \Theta$ was designed in the way described above so as to allow us to easily implement the sampling process in our computational studies, our proposed scenario-sampling framework is generally valid for any given demand distribution. Furthermore, in order to ensure that every possibly sampled routing scenario remained feasible, the fleet size was set to be unlimited.

We now discuss how to conduct scenario-sampling from the given demand distribution and how to perform the marginal cost calculation. When generating a scenario, we first draw from a Bernoulli distribution with the order-requesting frequency as the parameter, to decide whether a given customer requests an order or not. If it does, we then determine the customer's demand in this scenario by sampling from its distribution. After this process is repeated for each customer, a routing scenario with customers having fixed demands has been defined. In order to computing the marginal cost of serving the target customer in this scenario, we utilize our BPC algorithm to solve to optimality two MDVRPI instances, one with and another one without the target customer, and then apply the formula (1). The average marginal cost across all sampled scenarios is returned as the estimate. We emphasize that, since each scenario can can be solved independently, we utilize OpenMP to parallelize our code using up to 10 threads in the following experiments and report the CPU wall time.

5.2.1 Marginal Cost Estimation

In our context, $\underline{\texttt{MC}} = 0$ is a valid lower bound to the marginal cost, since serving an additional customer never reduces the routing cost. Let C denote the cost for the direct round-trip route from the depot that is closest to the target customer. The value $\overline{\texttt{MC}} = C / \left\{ \min_{\vartheta \in \Theta} \vartheta \right\}$ is a safe upper bound. The probability threshold is chosen to be $\alpha = 5\%$, while the permissible deviation is chosen as $\delta = 2\% \times \left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)$. Consequently, from equation (5), the minimum number of samples that is required for the specified accuracy is $\overline{N} = 4,612$. Fig. 3 shows the progress of the marginal cost

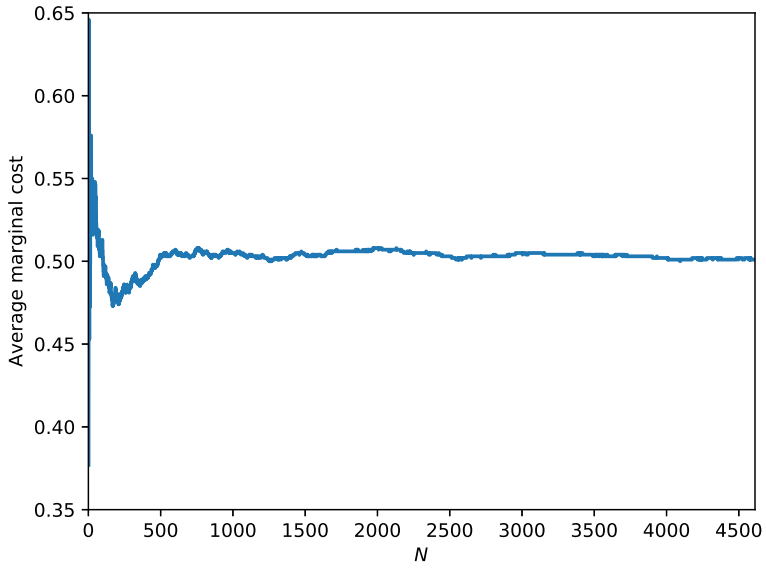

Figure 3. Example progress of the average marginal cost, as the sample size N increases

average, as the sample size increases, for the benchmark instance d1-25-50-450, which is selected here as a representative of all instances.

There are several noteworthy observations. Firstly, the marginal cost average initially fluctuates strongly, but then gradually stabilizes around the value of 0.501, which empirically confirms the conclusion made in Proposition 1, i.e., that the sample average $\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i,\vartheta^i\right)$ converges to a constant value (specifically, the value $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi,\vartheta\right)\right]$) when the sample size N is large enough. The claim that the estimate is of high quality is further strengthened by the fact that the chosen value of the permissible deviation ($\delta = 0.020$ in this example) ended up being nearly negligible when compared to this estimate. Secondly, the stabilized estimate was produced relatively early in the execution of the algorithm, but many additional scenarios had to be sampled until the probability of a sufficiently accurate estimate was lowered below the required threshold to warrant stopping the process (pre-computed sample size $\overline{N}$). We remark that, from a practical point of view, one can keep track of the progress of the marginal cost average and terminate the algorithm when the sample average satisfactorily stabilizes, which might occur significantly before $\overline{N}$ samples have been collected and processed.

5.2.2 Customer Locations

We now investigate the effect of a customer's location on the marginal cost of serving this customer. Generally speaking, when a customer is located at a place far away from the depot facilities, or far away from the other customers, a vehicle has to drive a longer distance for deliveries to this customer, which causes more routing costs. In order to quantify this effect in the context of a stochastic supply chain network, we consider two additional locations for the target customer in each of the five proposed instances: (i) the first location, dubbed "Central," having coordinates at the rounded (to the nearest integer) average of the depots' coordinates, and (ii) the second location, dubbed "Remote," having coordinates $(\max_i x_i, \max_i y_i)$, where (x_i, y_i) represent the i-th customer's coordinates in this instance. Along with the original placement of the target customer, a total of three customer locations are considered. We shall again choose $\alpha = 5\%$ and $\delta = 2\% \times \left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)$ for our estimation parameters, while we shall again generate $4,612$ representative scenarios and compute the average marginal cost.

Table 3 shows the computational results for our five instances under the three customer location cases. The column "δ" denotes the applicable value of the permissible deviation parameter, as determined for the each specific instance, while the column "MC" reports the marginal cost estimate. The relative estimation error is reported in the column "$\delta/\texttt{MC}$ (%)". Finally, we also report the total CPU wall time for running each instance in the column "t (sec)".

The entries in this table reveal that, when the target customer is located further away from the depot facilities, the marginal cost of serving this customer increases, as also plotted in Fig. 4 for all five benchmark datasets. Notably, we can claim in each case that the probability that the expected marginal cost deviates from the returned estimate by *at most* the corresponding relative estimation error is above $1 - \alpha = 95\%$. In particular, this demonstrates a satisfactory estimation accuracy, since the average (among all instances) deviation turns out to be merely 4.5% of the respective estimates. Finally, we remark that the computational time in these experiments ranges from approximately $1,200$ to $2,500$ seconds, which is quite short considering that a total of $9,224$ MDVRPI instances have to be solved to proven optimality in each case.

Table 3. Marginal cost estimation under three customer location cases

Instances	Central				Original				Remote			
	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)
a1-25-50-450	0.001	0.044	2.3	1,566	0.030	0.777	3.9	1,883	0.179	2.451	7.3	1,707
b1-25-50-450	4×10^{-4}	0.011	3.8	1,431	0.037	0.953	3.9	1,221	0.347	6.924	5.0	1,425
c1-25-50-450	5×10^{-4}	0.018	2.6	1,811	0.027	0.764	3.5	1,417	0.276	4.418	6.2	1,463
d1-25-50-450	0.001	0.042	2.4	2,099	0.020	0.501	4.0	2,509	0.237	4.435	5.3	2,277
e1-25-50-450	0.002	0.029	6.9	1,266	0.026	0.569	4.6	1,668	0.423	6.510	6.5	1,324

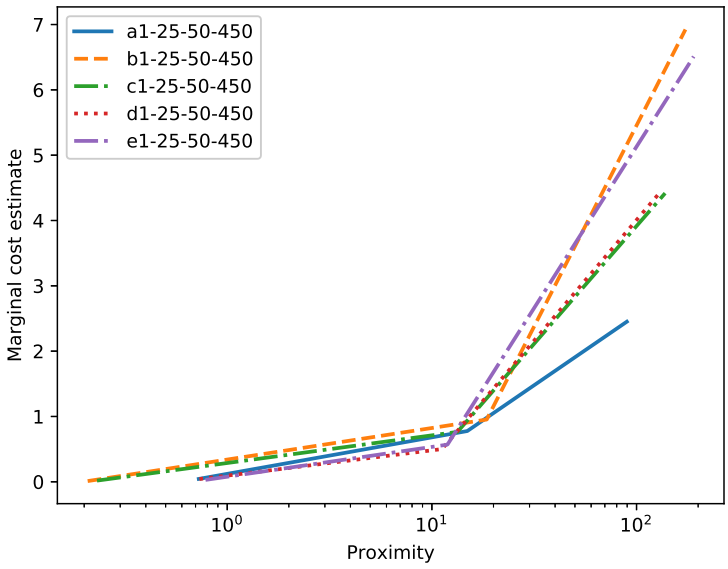

Figure 4. Effect of a customer's proximity to the depot facilities on the marginal routing cost. The proximity is defined as the length of the direct round-trip route from the depot that is closest to the target customer.

5.2.3 Demand Levels

In this section, we consider the effect of a customer's demand level on the marginal routing cost. Let $\bar{q}$ denote the nominal demand of the target customer in each of the five proposed instances, and let us now also consider two additional demand levels, namely $\bar{q} \pm 5$. The three demands are classified, based on their values, as "Low," "Medium," and "High." The customer demand distribution is created in the same way as before. Again, $\alpha = 5\%$ and $\delta = 2\% \times \left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)$. Table 4 shows the relevant computational results. As expected, when the customer demand increases, the marginal routing cost decreases, as also plotted in Fig. 5. The actual deviation is on average within 4.1% of the marginal cost estimate, which is considered acceptable from a practical point of view. Again, the computational times are deemed quite affordable.

Table 4. Marginal cost estimation under three customer demand levels

Instances	Low				Medium				High			
	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)
a1-25-50-450	0.050	1.215	4.1	1,623	0.030	0.777	3.9	1,883	0.023	0.589	3.9	1,354
b1-25-50-450	0.053	1.192	4.4	1,224	0.037	0.953	3.9	1,221	0.026	0.755	3.4	1,482
c1-25-50-450	0.044	1.191	3.7	1,810	0.027	0.764	3.5	1,417	0.021	0.598	3.5	1,209
d1-25-50-450	0.032	0.722	4.4	2,437	0.020	0.501	4.0	2,509	0.016	0.391	4.1	1,948
e1-25-50-450	0.048	0.894	5.4	1,016	0.026	0.569	4.6	1,668	0.020	0.421	4.8	1,298

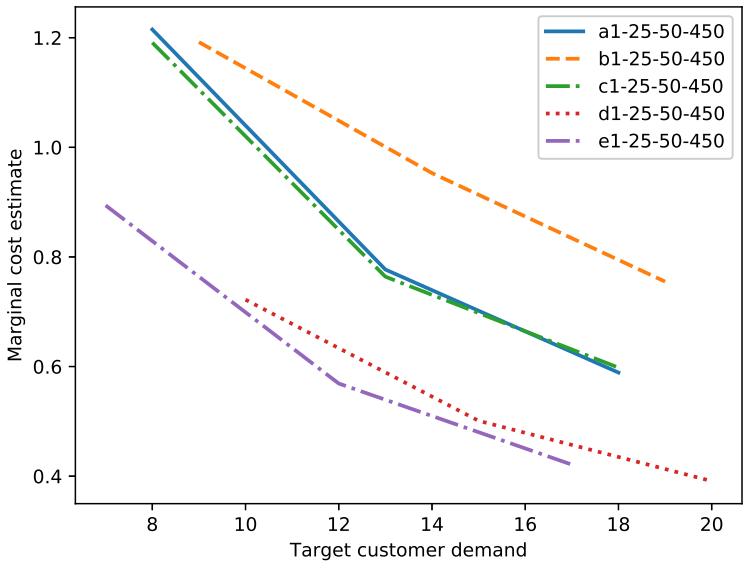

Figure 5. Effect of a customer's demand level on the marginal routing cost

6 Conclusions

We considered the problem of estimating the marginal routing cost of serving an extra customer on top of a given distribution network. In this context, the main challenge stems from the intrinsic stochasticity in customer demands, and to counteract this, we proposed a systematic scenario-sampling framework that can rigorously quantify the marginal routing cost. In particular, we used probability theory to demonstrate that the estimate returned by our framework will converge to the true value with sufficient increase of the sample size. For purposes of practicality, we proposed to utilize the well-known Hoeffding's inequality to bound the sample size for a desired accuracy.

In our computational studies, considering each scenario entailed solving two $\mathcal{NP}$-hard multi-depot vehicle routing problems with inter-depot routes. To this end, we develop a tailored branch-price-and-cut algorithm that incorporates several state-of-the-art techniques, including ng-routes, route enumeration, and limited-memory subset row cuts, among others. Our implementation was shown to perform significantly better than the previous state-of-the-art exact approach on solving MDVRPI benchmark instances. Specifically, within a time limit of 2 hours, our algorithm was able to close all 21 MDVRPI benchmark instances involving up to 40 customers that remained opened in the literature, while it was additionally able to solve 17 out of a total of 19 newly-defined

benchmark instances with 70 customers.

Utilizing our branch-price-and-cut code, we conducted extensive computational studies to demonstrate the deployment of our proposed framework for estimating the incremental routing cost, elucidating the accuracy of our estimates as well as quantifying the effect of two important factors, namely customer location and demand, on the marginal cost estimate.

Acknowledgments

We thank Dr. Ibrahim Muter and Dr. Jean-François Cordeau for providing us with useful clarifications about the datasets that were utilized in their previous work. We acknowledge financial support from Air Liquide through the Center for Advanced Process Decision-making at Carnegie Mellon University. Akang Wang also greatly acknowledges support from the James C. Meade Graduate Fellowship and the H. William and Ruth Hamilton Prengle Graduate Fellowship at Carnegie Mellon University.

References

[1] Enrico Angelelli and Maria Grazia Speranza. The periodic vehicle routing problem with intermediate facilities. *European journal of Operational research*, 137(2):233–247, 2002.

[2] Philippe Augerat, José-Manuel Belenguer, Enrique Benavent, Angel Corbéran, and Denis Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2-3):546–557, 1998.

[3] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[4] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283, 2011.

[5] Katja Buhrkal, Allan Larsen, and Stefan Ropke. The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia-Social and Behavioral Sciences*, 39:241–254, 2012.

[6] Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *4OR*, 14(3):223–259, 2016.

[7] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. Random sampling for histogram construction: How much is enough? *ACM SIGMOD Record*, 27(2):436–447, 1998.

[8] Claudio Contardo and Rafael Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146, 2014.

[9] Benoit Crevier, Jean-François Cordeau, and Gilbert Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

[10] Carlos F Daganzo. The distance traveled to visit n points with a maximum of c stops per vehicle: An analytic model and an application. *Transportation science*, 18(4):331–350, 1984.

[11] M Poggi de Aragao and Eduardo Uchoa. Integer program reformulation for robust branch-and-cut-and-price algorithms. In *Mathematical Program in Rio: a Conference in Honour of Nelson Maculan*, pages 56–61. Citeseer, 2003.

[12] Moshe Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978, 1994.

[13] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4or*, 8(4):407–424, 2010.

[14] Miguel Andres Figliozzi. Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. *Transportation Research Record*, 2089(1):1–8, 2008.

[15] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[16] Phillip B Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 281–291. ACM, 2001.

[17] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[18] Stefan Irnich, Guy Desaulniers, Jacques Desrosiers, and Ahmed Hadjar. Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313, 2010.

[19] Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

[20] Byung-In Kim, Seongbae Kim, and Surya Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

[21] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[22] Estelle RS Kone and Mark H Karwan. Combining a new data classification technique and regression analysis to predict the cost-to-serve new customers. *Computers & Industrial Engineering*, 61(1):184–197, 2011.

[23] Gilbert Laporte and Yves Nobert. A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum*, 5(2):77–85, 1983.

[24] Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

[25] Ibrahim Muter, Jean-François Cordeau, and Gilbert Laporte. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, 48(3):425–441, 2014.

[26] Okan Örsan Özener, Özlem Ergun, and Martin Savelsbergh. Allocating cost of service to customers in inventory routing. *Operations Research*, 61(1):112–125, 2013.

[27] D Pecin. Exact algorithms for the capacitated vehicle routing problem. *Unpublished doctoral dissertation, Pontifícia Universidade Católica do Rio de Janiero, Rio de Janiero, Brazil*, 2014.

[28] Diego Pecin, Claudio Contardo, Guy Desaulniers, and Eduardo Uchoa. New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 29(3):489–502, 2017.

[29] Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017.

[30] Diego Pecin, Artur Pessoa, Marcus Poggi, Eduardo Uchoa, and Haroldo Santos. Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206–209, 2017.

[31] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 354–369. Springer, 2019.

[32] Luigi Di Puglia Pugliese and Francesca Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013.

[33] Stefan Røpke. Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. *Presentation in Column Generation*, 2012.

[34] Ruslan Sadykov, Eduardo Uchoa, and Artur Pessoa. A bucket graph based labeling algorithm with application to vehicle routing. *Cadernos do LOGIS*, 7, 2017.

[35] Maximilian Schiffer, Michael Schneider, Grit Walther, and Gilbert Laporte. Vehicle routing and location routing with intermediate stops: A review. *Transportation Science*, 2019.

[36] Lei Sun, Mark H Karwan, Banu Gemici-Ozkan, and Jose M Pinto. Estimating the long-term cost to serve new customers in joint distribution. *Computers & Industrial Engineering*, 80:1–11, 2015.

[37] Christos D Tarantilis, Emmanouil E Zachariadis, and Chris T Kiranoudis. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing*, 20(1):154–168, 2008.

[38] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[39] Marcel Turkensteen and Andreas Klose. Demand dispersion and logistics costs in one-to-many distribution systems. *European Journal of Operational Research*, 223(2):499–507, 2012.

[40] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

[41] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.