RESEARCH ARTICLE OPEN ACCESS

Fast Shapley Value Approximation Through Machine Learning With Application in Routing Problems

Johannes Gückel | Pirmin Fontaine

Ingolstadt School of Management & Mathematical Institute of Machine Learning and Data Science, KU Eichstätt-Ingolstadt, Ingolstadt, Germany

Correspondence: Johannes Gückel (jgueckel@ku.de)

Received: 28 November 2024 | **Revised:** 9 July 2025 | **Accepted:** 15 July 2025

Funding: This research was supported by the German Research Foundation (ROCOCO FO1468/1-1).

Keywords: approximation | capacitated vehicle routing problem | cost allocation | machine learning | supervised learning | traveling salesman problem

ABSTRACT

For many routing applications, it is not only necessary to minimize total costs but also to allocate them to individual customers. In this context, the allocation according to the Shapley value is a well-known method highly regarded for its fulfillment of major fairness criteria. However, its computational complexity restricts its applicability, especially for NP-hard underlying optimization problems like routing problems, since the exact computability of the Shapley values is no longer possible within a reasonable time. In this study, we propose a general Machine Learning-based Shapley Value Approximator (MLSVA) and apply it to routing problems based on the exploitation of routing-specific problem structures as features, enabling real-time approximations. Within an extensive numerical study, we show that our MLSVA outperforms current state-of-the-art approximation results for the Traveling Salesman Problem (TSP) and, at the same time, achieves very good results for the Capacitated Vehicle Routing Problem (CVRP), for which it is the first efficient method that quickly delivers high-quality approximations. On average, we approximate Shapley values with a mean absolute percentage error of 2.4% for the TSP and 3.5% for the CVRP. Further, the MLSVA achieves strong approximation results even when trained on biased labels. This shows the scalability of the MLSVA and allows high-quality approximations even for large routing problems. Finally, we demonstrate the generalizability of our approach by successfully approximating Shapley values for items in a variant of the bin packing problem.

1 | Introduction

While the literature offers many exact and heuristic methods for solving diverse route planning problems to determine optimal routes and total cost (e.g., [1, 2]), there has been relatively scarce research on allocating the resulting cost to individual customers. Allocation methods are not only used for monetary cost allocation but also for the allocation of environmental emissions, which is likely to become increasingly relevant in the future [3].

The problem of cost allocation in vehicle routing was introduced several decades ago in the context of cooperative game theory [4]. Engevall et al. [5] explored the application of game-theoretic allocation methods, including the Shapley value, in routing problems with heterogeneous vehicle fleets, although computational limits restricted exact calculations to small instances. Similarly, Sun et al. [6] considered the Shapley value for cost allocation on fixed routes. Özener et al. [7] studied an inventory routing application using an approximation approach based on random subset sampling to circumvent computational challenges. Further real-life applications include the work by Engevall et al. [8], who applied the Shapley value to allocate costs among customers of a gas and oil company, highlighting both practical relevance and computational limitations. These studies collectively underline the necessity for efficient approximation methods capable of delivering accurate Shapley value estimates for larger problem instances. Although some cooperative game settings allow efficient computation of Shapley values, this is not the case for routing problems. In such settings, the main computational challenge lies not in the Shapley formula itself, but in evaluating the cost of each subcoalition, which requires solving a complex optimization problem. Due to this combinatorial nature, exact Shapley value computation becomes intractable even for moderately sized instances.

Inspired by concepts from cooperative game theory, the Shapley value has emerged as a method that ensures fair cost allocation to individual customers. Despite its alignment with various fairness criteria, the practical application of the Shapley value in routing problems remains limited to instances with only a small number of customers. This limitation stems from the need to solve the routing problem for every subcoalition of customers to compute the Shapley value exactly, resulting in significant computational expenses. Therefore, there is a need for approximation methods that can provide fast and accurate estimates of the Shapley value while remaining computationally efficient.

Previous approximation methods for the Shapley value can be categorized into two types. The first type includes statistical sampling methods (e.g., [9]), that are computationally intensive and do not exploit the underlying problem structure of routing problems. The second type involves approximations explicitly designed for the Traveling Salesman Problem (TSP) (e.g., [10]) and is not adaptable to other routing problems.

In this study, we propose a general machine learning-based approach to approximate Shapley values and demonstrate its application to the TSP and the Capacitated Vehicle Routing Problem (CVRP). Through this approach, we exploit the particular problem structure of the underlying routing problem and create problem-specific features on which the models can learn and then approximate Shapley values for unseen problem instances rapidly. Although Shapley value approximations can generally only be tested on small instances, as exact Shapley values can only be computed for instances with a limited number of nodes, we propose methods to effectively train our approach for larger instances as well. These methods include training on quickly generated but biased Shapley values, as well as training on smaller instances than those used for later application. Furthermore, we can provide managerial insights about the influential factors determining each customer's cost share. As a result, we contribute to the literature in five ways:

1. We propose a new machine learning-based approach for approximating Shapley values, which utilizes both routing problem-specific features and the efficiency property of the Shapley value.
2. We show options that can be used to create quickly generated but biased labels. This is an important feature, especially for larger instances, as exact labels may no longer be created there.
3. Through an extensive numerical study, we demonstrate the performance of our method compared to simple proxies and current state-of-the-art approximation methods for the TSP and CVRP. Further, we show that this machine learning-based approach also leads to good approximations even with biased labels. As a result, we are the first efficient method that delivers high-quality solutions not only for the TSP but also for the CVRP.
4. Unlike other methods, our approach provides managerial insights into the influential factors on the Shapley value, which is highly relevant for logistics service providers.
5. We show that the methodology that we propose can be generalized by applying it to a bin packing problem.

The structure of this study is organized as follows: Section 2 details the problem description by presenting the problem of cost allocation, the Shapley value and its properties as well as the necessity of an approximation method for Shapley values. Section 3 provides a comprehensive literature review on applications and approximations of Shapley values as well as on machine learning applications in routing problems. Section 4 introduces our machine learning-based approximation approach and discusses the features, the machine learning models, and the generation of biased labels. Section 5 presents an extensive numerical study on the performance of our approach and analyzes the impact of biased labels in the training process, which is critical for larger instances where exact labeling is computationally prohibitive. Section 6 provides managerial insights by analyzing the feature's importance. Section 7 demonstrates the generalizability of our approach by applying it to the allocation of an item's cost in a bin packing variant. Section 8 concludes our study and provides further research directions.

2 | Problem Description

In this section, we first present the general problem description of allocating cost to customers in the context of routing problems (Section 2.1). Then, we discuss the Shapley value and its properties as well as its computational limitations and highlight the importance of approximated Shapley values (Section 2.2).

2.1 | Allocating Cost to Customers in Routing Problems

Cost allocation in routing problems presents a significant challenge. The general problem is explained using the example of a TSP instance shown in Figure 1. Nevertheless, the resulting implications also hold for other routing problems.

Let $\mathcal{N}$ be the set of all customers in a TSP instance. $C(\mathcal{N})$ denotes the total cost of the optimal TSP solution. The goal is to determine a cost allocation ϕ where each customer $n \in \mathcal{N}$ is assigned a cost ϕ_n. We also include a depot node 0, which incurs no cost. Therefore, the total cost $C(\mathcal{N})$ must be fully allocated to the customers, i.e., $C(\mathcal{N}) = \sum_{n\in\mathcal{N}} \phi_n$, which in the example in Figure 1 results in $\phi_1 + \phi_2 + \phi_3 = 18$. Even in this simple example instance, it is not straightforward to determine which customer is responsible for which share of the total cost. For instance, if the cost were distributed depending on the distance to the depot, customer 3 would bear significantly less than customers 1 and 2.

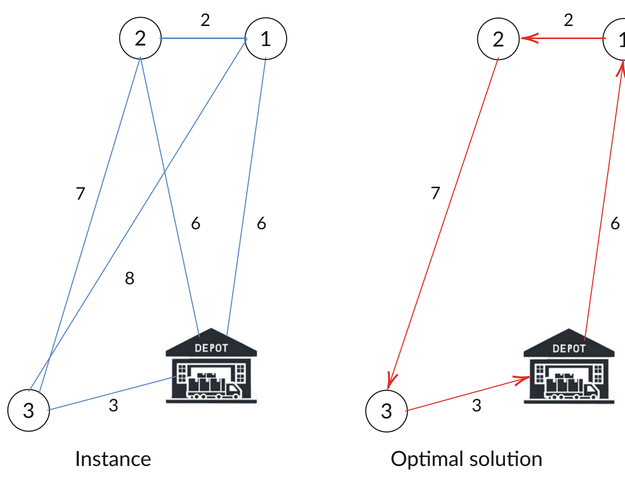

FIGURE 1 | TSP instance solved.

However, this might be perceived as unfair because customers 1 and 2 only incur low marginal cost due to their geographical proximity. As an alternative, the cost could be allocated in proportion to the marginal cost $C(\mathcal{N}) - C(\mathcal{N} \setminus \{n\})$ of each customer n. This might be unfair to customer 3 because he is much closer to the depot.

To address fairness concerns, cooperative game theory has introduced a variety of more sophisticated cost allocation methods that are particularly applicable to such allocation scenarios. These include core allocation [11], the Equal Profit Method [12], and the nucleolus [13]. For a comprehensive review of cost allocation methods in transportation problems, we refer to [14]. Each of these approaches has distinct characteristics and emphasizes different aspects of fairness. Among these, the Shapley value, which will be discussed in detail in the following section, stands out as the most widely used allocation method due to its unique and well-established properties.

2.2 | Shapley Value Cost Allocation

The Shapley value was first introduced by Shapley [15]. Since then, it has been applied in many different contexts. Formally, the Shapley value is defined as follows:

$$\phi_n^{SV} = \sum_{\mathcal{S} \subsetneq \mathcal{N} \setminus \{n\}} \frac{|\mathcal{S}|!(|\mathcal{N}| - |\mathcal{S}| - 1)!}{|\mathcal{N}|!} [C(\mathcal{S} \cup \{n\}) - C(\mathcal{S})] \quad (1)$$

In this Equation (1), the sum is over all subcoalitions $\mathcal{S}$ within the set of all customers $\mathcal{N}$ that do not include customer n. The term $[C(\mathcal{S} \cup \{n\}) - C(\mathcal{S})]$ represents the marginal cost of the customer to the subcoalition $\mathcal{S}$. The fraction $\frac{|\mathcal{S}|!(|\mathcal{N}|-|\mathcal{S}|-1)!}{|\mathcal{N}|!}$ represents the weighting of subcoalitions within all possible permutations. Additionally, especially important for the understanding of sampling approximation methods, the formula can be expressed as an average over all possible permutations, as shown in Equation (2) [16]:

$$\phi_n^{SV} = \frac{1}{|\mathcal{N}|!} \sum_{p \in \mathcal{P}} \left(C(p_{n-1} \cup \{n\}) - C(p_{n-1}) \right) \quad (2)$$

Thereby, $\mathcal{P}$ represents the set of all permutations, and p_{n-1} represents the set of customers ranked lower than customer n in the order of permutation p. Consequently, the Shapley value allocates cost according to each customer's average marginal cost to each possible permutation. The cost allocation of the Shapley value is unique due to the fulfillment of the following properties:

- *Efficiency*: $\sum_{n \in \mathcal{N}} \phi_n^{SV} = C(\mathcal{N})$. The sum of the Shapley values is equivalent to the total cost.
- *Symmetries*: if $C(\mathcal{S} \cup \{n\}) = C(\mathcal{S} \cup \{\tilde{n}\}) \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{n, \tilde{n}\}$, then $\phi_n^{SV} = \phi_{\tilde{n}}^{SV}$. Two customers with the same marginal cost have the same Shapley value.
- *Additivity*: for any two games C and C', the Shapley value satisfies $\phi_n^{SV}(C + C') = \phi_n^{SV}(C) + \phi_n^{SV}(C')$. That is, the Shapley value of the sum of two games equals the sum of their individual Shapley values.
- *Dummy*: if $C(\mathcal{S} \cup \{n\}) = C(\mathcal{S}) \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{n\}$, then $\phi_n^{SV} = 0$. A customer that has no marginal impacts has a Shapley value of zero.

In addition to these four most frequently mentioned properties, the Shapley value also fulfills the property of monotonicity, according to which, if a game changes in a way that increases or maintains a player's contribution to all coalitions, their allocated cost should not decrease [17]. Further, the Shapley value also satisfies the balanced contribution property [18], which ensures that for any pair of players, one player's claim against another is always balanced by a corresponding counterclaim from the other player. Consequently, the Shapley value embodies numerous well-established properties of cooperative game theory, each emphasizing different aspects of fairness in cost allocation. As a drawback, it is important to note that, unlike other cost allocation methods such as the nucleolus [13] or the core allocation [11], the Shapley value does not necessarily lie within the core of a cooperative game. This means that there may be cases where a subcoalition can find a more favorable cost allocation, violating the rationality property, which ensures that no subcoalition has an incentive to deviate from the allocation [19]. Computing the Shapley value requires evaluating every possible subcoalition, represented by the characteristic function $C(\cdot) : 2^{\mathcal{N}} \rightarrow \mathbb{R}$. The characteristic function assigns a value to each subset $\mathcal{S} \subseteq \mathcal{N}$, which in this context represents the routing cost of serving the customers in $\mathcal{S}$. For a problem instance with $|\mathcal{N}|$ customers, this requires solving $2^{|\mathcal{N}|}$ routing problems. For 15 customers, this already amounts to $2^{15} = 32,768$ problems. Figure 2 illustrates the computing time for a single instance as a function of the number of nodes. These times include the exact solution of the corresponding routing problem for every subcoalition required to compute the Shapley value. It shows that the computing time exponentially increases with the number of nodes. This computational burden highlights that there is a need for efficient approximation methods for Shapley values within a short computational time.

3 | Literature Review

This section is divided into four parts. First, we review the existing literature on the applications of the Shapley value (Section 3.1).

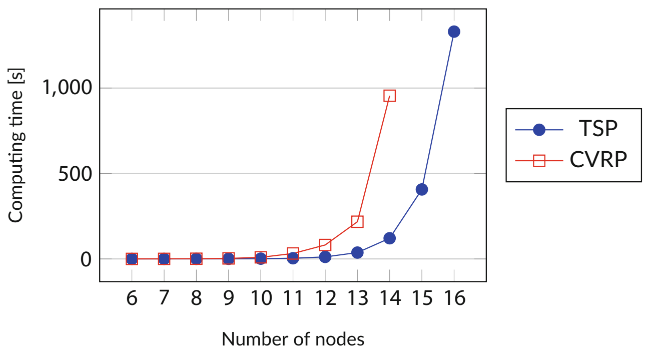

FIGURE 2 | Computing time [s] per number of nodes required to compute a single instance of the Shapley value (averaged over ten randomly generated instances with the numerical setup described in Section 5).

Then, we examine the literature on approximations of the Shapley value (Section 3.2). Further, we review the current state of the literature on machine learning in routing and transportation (Section 3.3). We conclude by outlining research gaps (Section 3.4).

3.1 | Shapley Value Applications

The Shapley value was initially introduced in the literature on cooperative game theory for allocating costs to players of a cooperative game [15]. Since then, it has been used in many different contexts to allocate total costs. Beyond allocating costs to customers in routing problems, which has been discussed in the introduction, the Shapley value has also been applied to many other applications in the transportation sector. For instance, it can be used to allocate cost to carriers in cooperative routing problems. In this context, Krajewska et al. [20] used the Shapley value to distribute profits to cooperating carriers. While allocating costs and distributing profits are conceptually distinct, both processes can be modeled using the Shapley value to fairly allocate the total amount based on each participant's contribution. In another application, Kimms and Kozeletskyi [21] used the Shapley value for cost allocation in the context of a TSP with a rolling horizon and cooperative salesmen. Additionally, Gückel et al. [22] studied cost allocation among cooperative logistics service providers in a city logistics context, where they compared the Shapley value with other cost allocation methods. Further, the Shapley value has been used as a cost allocation method for distributing highway construction and maintenance costs among different user groups [23, 24].

Besides its use in transportation and routing, the Shapley value has been applied in a variety of other domains. For instance, in cooperative queueing games, Bendel and Haviv [25] demonstrated the Shapley value's utility for allocating costs when servers pool resources to optimize service rates. Similarly, Timmer et al. [26] applied the Shapley value in the context of inventory management among cooperating companies, ensuring that the cost allocation incentivizes collaboration. In the realm of peer-to-peer networks, Altan and Özener [27] proposed an approximation of the Shapley value to achieve fair cost allocation between service providers and users workingz together to minimize joint costs. Moreover, Zolezzi and Rudnick [28] utilized the Shapley value to design a cooperative cost allocation method for transmission systems in the energy sector. Additionally, Choudhury et al. [29] introduced the Generalized Egalitarian Shapley value, which provides more flexibility in determining the level of marginality based on coalition size.

Beyond cost allocation, the Shapley value has found widespread applications in other fields. In machine learning, it has become a widely used tool for assessing feature importance in model performance (e.g., [30–32]). Additionally, it has been used in sports analytics to evaluate player importance in basketball [33]. In healthcare, Roshanaei et al. [34] used the Shapley value to distribute cost savings from cooperating hospitals. Gnecco et al. [35] applied the Shapley value to measure the contribution of each node in cooperative transportation network games to assess centrality. Notably, the Shapley value has also been investigated in the context of environmental cost allocation, as seen in studies such as [3, 36].

Despite its broad applicability, the Shapley value is seldom used in applications involving NP-hard problems with a large number of participants due to significant computational limitations.

3.2 | Shapley Value Approximations

To date, the approximation methods found in the literature can be divided into two categories. The first category includes approximations that are independent of the underlying subproblem and that rely solely on the properties of the Shapley value. This category is represented by statistical sampling methods that aim to approximate the Shapley value through selective sampling of permutations or sub-coalitions while reducing computing time. The most straightforward sampling method is the Monte Carlo sampling introduced by Shapley and Mann [37] for electoral college voting games. In this sampling method, random permutations are sampled and evaluated until a predefined number is reached. Then, the average marginal contribution of each participant over all sampled permutations is determined and serves as the approximation. Schopka and Kopfer [38] used a so-called α-sampling procedure to approximate Shapley values in horizontal carrier coalitions. Horizontal carrier coalitions involve collaboration between competing carriers at the same level of the supply chain to reduce costs through joint operations, in contrast to vertical cooperation, which occurs between different supply chain levels (e.g., carriers and suppliers). Unlike our work, where costs are allocated to customers, the cost allocation here occurs at the carrier level. This procedure does not sample random permutations but only subcoalitions with a small number α of carriers, which leads to a much lower computing time. Touati et al. [9] recently showed a new approach by combining Monte Carlo sampling with Bayesian methods. Further, Castro et al. [39] introduced a polynomial calculation approach for the Shapley value on the basis of sampling. However, the limitation here is that the value of each coalition in the subproblem must be computed polynomially. With the goal of attributing predictions to features in a machine learning context, Mitchell et al. [16] introduced a new approximation method of Shapley values on the basis of a specific selection of permutation samples.

The second category includes problem-specific approximation methods that explicitly exploit properties of the underlying routing problem. Aziz et al. [40] investigated several proxies for Shapley values in TSPs. Both simple rules of thumb, such as distance to depot or distribution based on marginal cost, and more advanced methods, such as linear regression or sampling methods, were used as proxies. In their research outlook, it became clear that approximation methods that also perform on more general vehicle routing problems are needed. Popescu and Kilby [10] presented two approximation approaches for the Euclidean TSP, which build on an extension of the one-dimensional TSP. These approximations are on the basis of shared distances, where the cost associated with any two locations is based on the Euclidean distance between their coordinates and the depot. These require only polynomial computing time. They showed that they outperform all other proxies from the literature and are only outperformed by sampling methods with very high iteration numbers and computing time. The current state-of-the-art method is from Levinger et al. [41]. Their approximation method, SHAPO, is based on a fixed order of customers. However, they showed that this approximation method can be generalized by considering the customer sequence of a TSP as a fixed order. In a numerical study, they achieved better results than the previous state-of-the-art method by Popescu and Kilby [10].

3.3 | Machine Learning in Routing and Transportation

Machine learning methods have been used in the fields of routing and transportation for prediction or decision support in various areas. Li et al. [42] used decision tree ensemble methods for the travel time prediction. Gmira et al. [43] predicted the travel speed to get from one location to another in a road network with the use of different machine learning and data mining methods. Basso et al. [44] proposed a probabilistic Bayesian machine learning approach for predicting the expected energy consumption and variance in the electric vehicle routing problem.

In addition to actual predictions, machine learning models are also used to support decision-making or adapt policies. Ghiani et al. [45] used a neural network (NN) to adjust online policy parameters in a pickup and delivery problem learned offline by using simulation experiments. Van der Hagen et al. [46] used supervised machine learning approaches to support time slot decisions for the feasibility check to offer time slots to passengers in an online booking system. In the context of hierarchical vehicle routing problems, Sobhanan et al. [47] integrated a NN for predicting the objective function value of the underlying vehicle routing problem without actually solving it into a genetic algorithm. Another possible application is the reduction of problems. For example, by using a support vector machine in the TSP, Sun et al. [48] predicted which decision variables could be removed from the problem without significantly affecting the quality of the solution. Bertsimas and Kim [49] developed a machine learning approach to accelerate the solution process for the mixed-integer convex optimization problem and applied it to problems from the field of transport optimization. In the context of load planning for rail transportation, Larsen et al. [50] developed a two-stage stochastic programming model, where the expected solution of the second-stage problem is predicted based on the first-stage variables. Further, machine learning models have been successfully applied in heuristics (e.g., [51, 52]), and exact solution methods (e.g., [53]). In recent years, there has been a growing number of contributions showcasing new machine learning-based methodologies applied conceptually to fundamental routing problems. Asín-Achá et al. [54] demonstrated how machine learning models can be used to select a best performing fast algorithm and parameterize it at the same time by exploiting routing-specific features for the CVRP. Morabit et al. [55] proposed a machine learning-based heuristic for the reoptimization of repeatedly solved routing problems after minor changes in the problem data. Chamurally and Rieck [56] integrated a machine learning approach into an adaptive large neighborhood solution method and used historical data to predict customer demands for a vehicle routing problem with stochastic demands. For a literature review about machine learning methods in a vehicle routing context, we refer to Bai et al. [57]. For a more general review of the use of machine learning in combinatorial optimization, we refer to Bengio et al. [58]. Overall, machine learning is already used to predict outcomes and enhance solution approaches by leveraging problem-specific features, making it a suitable choice for the case at hand.

3.4 | Research Gaps

While the Shapley value is well-regarded for cost allocation in literature, its resource-intensive computations usually limit its application to instances with a small number of customers.

The literature lacks a general approximation method that can be applied to different routing problems and utilizes the specific problem components in each case. Machine learning models seem ideally suited for this problem setting as there are a lot of influential factors on the Shapley value that have not been explored in any of the approximation methods before. In addition, none of the current approximation methods provides managerial insights about the main cost drivers for customers in routing problems. We fill this research gap with our machine learning-based approximation approach.

4 | Methodology

This section starts describing the general outline of our Machine Learning-based Shapley Value Approximator (MLSVA) (Section 4.1). Then, we describe our methodology step by step. We start with the machine learning models (Section 4.2). Next, we continue with the feature generation (Section 4.3). After that, we show the scaling of our approximations to ensure the efficiency property (Section 4.4). Finally, we describe how to create fast but biased labels to make our approach feasible for larger instances, since exact labels for them cannot be created in acceptable time (Section 4.5).

4.1 | General Outline

Our MLSVA is based on supervised machine learning, utilizing true Shapley values and influential features to learn and predict Shapley values for unseen instances and customers. An

ALGORITHM 1 | MLSVA.

```
1: Generate set of instances $\mathcal{T}^{all}$
2: for $t \in \mathcal{T}^{all}$ do
3:     $C(\cdot) \leftarrow$ determineCharacteristicFunction($\mathcal{N}(t)$)
4:     for $n \in \mathcal{N}(t)$ do
5:         $\phi_n^{SV} \leftarrow$ computeShapley($n, C(\cdot)$)
6:         $f(n) \leftarrow$ prepareFeatures($n, t$)                ▷ Section 4.3
7:     end for
8: end for
9: Separate $\mathcal{T}^{all}$ into $\mathcal{T}^{train}$ and $\mathcal{T}^{test}$
10: Use $\phi_n^{SV}$ as label $\forall n \in \mathcal{N}(t), t \in \mathcal{T}^{train}$
11: trainMlModels($f(n), \phi_n^{SV} \forall n \in \mathcal{N}(t), t \in \mathcal{T}^{train}$)    ▷ Section 4.2
12: for $t \in \mathcal{T}^{test}$ do
13:     for $n \in \mathcal{N}(t)$ do
14:         $\hat{\phi}_n^{ML} \leftarrow$ predictShapley($f(n), \phi_n^{SV}$)
15:     end for
16: end for
17: for $t \in \mathcal{T}^{test}$ do
18:     for $n \in \mathcal{N}(t)$ do
19:         $\hat{\phi}_n \leftarrow$ scalePrediction($\hat{\phi}_n^{ML}, C(\mathcal{N}(t))$)    ▷ Section 4.4
20:     end for
21: end for
```

instance refers to a specific realization of a routing problem (TSP, CVRP, etc.), characterized by a given number of customers, their locations, and other relevant parameters. The primary distinction between instances lies in their unique combination of problem-specific parameters, such as the number of customers, their positions, and additional factors like customer demands or vehicle capacity. Let $\mathcal{T}^{all}$ denote the set of all generated instances, which serve as the basis for training and testing our models, $\mathcal{N}(t)$ represent the set of customers in instance t, $\hat{\phi}_n^{ML}$ denote the original predictions of our machine learning models, and $\hat{\phi}_n$ denote our final approximation of the Shapley value for customer n. The cost function $C(\cdot)$ defines the total routing cost for any subset of customers within an instance, i.e., it represents the characteristic function of the cost game. For example, $C(S)$ denotes the cost of the optimal routing solution for a coalition $S \subseteq \mathcal{N}(t)$, and $C(\mathcal{N}(t))$ corresponds to the total cost of serving all customers in instance t. Thereby, no predefined customer order is assumed, as costs are based on (optimal) routing solutions. Using this notation, our general methodology is outlined in Algorithm 1.

We begin by generating a set of instances, denoted as $\mathcal{T}^{\text{all}}$. For each instance t, the first step is to compute the characteristic function, i.e., to evaluate the routing cost for all subcoalitions of customers within the instance. This allows us to assign the true Shapley value ϕ_n^{SV} to each customer observation n, and to prepare the corresponding feature vector $f(n)$ that captures the factors influencing the Shapley value (see lines 2–8). These features characterize both the individual customer as well as the instance in which the customer is present, thereby exploiting the specific routing problem structure. It is important to note that, while predictions are made for individual customers, instance-specific data and costs are crucial for generating the features and scaling the predictions.

We then split the instances into training instances $\mathcal{T}^{train}$ and test instances $\mathcal{T}^{test}$ (line 9). For the training instances, we use the Shapley values as labels (line 10). The selected supervised machine learning models are trained exclusively on the training instances (line 11) based on the features and labels. These trained models are then used to generate predicted Shapley values $\hat{\phi}_n^{ML}$ for each observation (lines 12 to 16). Next, these predictions are scaled for each instance so that the sum of the approximated Shapley values corresponds to the total routing costs for that instance, producing the final Shapley value approximations $\hat{\phi}_n$ (lines 17 to 21). This scaling is a critical aspect of our methodology, as the sum of the approximations for all customers in an instance must match the total routing costs that are known in advance (see efficiency property in Section 2).

In the end, the performance of the method is evaluated exclusively on the test data $\mathcal{T}^{test}$ and the best performing machine learning model can be selected. Once the machine learning models are trained, our approximation approach can generate approximations rapidly without high computational effort.

4.2 | Machine Learning Models

In this framework, we use machine learning models that are able to provide an approximation for a numerical value. This is supervised machine learning as we use labeled observations [59].

4.2.1 | NNs

NNs are especially known for recognizing highly nonlinear relationships and can generate numerical, categorical, and binary outputs [59]. A NN essentially consist of a sequence of layers in which data is transferred nonlinearly to the layer behind. Each layer's output represents the next layer's input in the sequence. Each layer consists of several neurons that apply transformations to the input data [60]. These transformations are controlled by learnable parameters and activation functions determining how the input signal is modified. The sequence's last layer determines the final output. In our study, we use a feedforward architecture in which each layer receives input from the previous layer.

4.2.2 | Random Forest

Random forest (RF) is an ensemble learning method in which a large collection of de-correlated trees is formed, the results of which are then averaged. The basic idea behind it is that aggregating multiple different decision trees reduces the variance of the approximation. This technique can be used for both classification and regression of numerical data [59]. As the number of trees in a forest increases, the generalization error converges [61].

4.2.3 | XGBoost

XGBoost is short for eXtreme Gradient Boosting (XGB). In contrast to the RF, new trees are created sequentially, which learn from the approximation errors of the previous trees and thus generate better approximations [59].

4.2.4 | Polynomial Regression

Polynomial regression (PR) extends linear regression by incorporating polynomial terms, capturing complex, nonlinear relationships between variables [62]. A PR model of degree d is given by Equation (3):

$$E(Y|X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_d X^d \tag{3}$$

where $\beta_0, \beta_1, \ldots, \beta_d$ are regression coefficients. The degree d controls model flexibility, with $d = 2$ for quadratic and $d = 3$ for cubic regression. Higher degrees enable more complex curves but risk overfitting.

4.2.5 | Support Vector Regression

Support vector regression (SVR) extends the concepts of support vector machines to regression problems by fitting a hyperplane in a high-dimensional space that best approximates the given training data within a specified margin [63].

4.2.6 | *k*-Nearest-Neighbor

k-nearest-neighbor (KNN) is a straightforward machine learning algorithm that is used for both classification and regression. We include this algorithm to have a simple machine learning method that serves as a benchmark for the more advanced methods introduced before. The basic idea is to approximate the target variable based on the k nearest neighbors. Since our use case is a regression with numerical output, the approximation is determined by the mean value of the k nearest neighbors. Let $\mathcal{K}(n)$ be the set of k nearest neighbors of customer n; the approximation is defined with Equation (4).

$$\hat{\phi}_n^{KNN} = \frac{\sum_{n' \in \mathcal{K}(n)} \phi_{n'}^{SV}}{|\mathcal{K}(n)|} \tag{4}$$

The hyperparameter $k = |\mathcal{K}(n)|$ must be tuned in advance.

The selection of machine learning models in this study covers different highly recognized algorithms, each representing distinct paradigms. RF and XGB showcase the strengths of tree-based ensemble techniques through bagging and boosting, respectively. PR captures nonlinear relationships effectively, while an NN manages complex data structures via deep learning. SVR extends support vector machines principles to high-dimensional regression tasks. KNN is included as the simplest machine learning model, serving as a baseline for comparing the performance of more advanced approaches. Together, these models provide a thorough evaluation of diverse approaches, enhancing the robustness and applicability of our results.

4.3 | Feature Generation

To accurately approximate the Shapley value using regression models, we design features that capture both individual customer characteristics and their interactions within the instance. These features are grouped into four categories: (1) **distance-based features**, describing spatial relationships that impact cost distribution; (2) **instance-based features**, summarizing global structural properties of the problem instance; (3) **cost-based features**, incorporating marginal and local cost effects that directly influence allocation; and (4) **cluster-based features**, reflecting structural dependencies among customers that impact cost allocation. Additionally, for the CVRP, we introduce **CVRP-specific features**, which account for demand distributions and vehicle capacity constraints, both of which play a crucial role in shaping cost allocation. The clusters were created using the DBSCAN algorithm from Ester et al. [64]. This algorithm is based on a density-based notion of clusters, which identifies clusters as areas of high density separated by areas of low density. Further, only for the CVRP instances, we use CVRP-specific features. Please note that we often use features in relation to the mean characteristics of the other customers of an instance, as the absolute value of a feature is often not very meaningful, but rather, the ratio to the other feature characteristics of the customers of an instance is relevant. This is indicated with (r). The distance between two nodes $i, j \in \mathcal{N}$ is denoted as d_{ij}, which refers to the distance between the nodes in the original TSP instance, as specified in the distance matrix. This distance remains fixed regardless of the solution to the instance. A detailed overview of the features for a customer n, including their mathematical expressions where applicable, is provided in Table 1.

4.4 | Scaling the Output

The machine learning models provide numerical approximations of the Shapley value for each customer. However, the sum of these approximated Shapley values for all customers in a given instance may not exactly equal the total cost. To preserve the efficiency property (see Section 2.2), we scale the approximations for each instance in accordance with its total cost.

The Shapley value predictions generated by the machine learning models are denoted by $\hat{\phi}_n^{ML}$. To ensure that the total cost for each instance is fully allocated, the condition $\sum_{n \in \mathcal{N}(t)} \hat{\phi}_n = C(\mathcal{N}(t)) \; \forall t \in \mathcal{T}$ must be satisfied. To achieve this, we scale the machine learning predictions in proportion to the total cost of each instance, yielding the final approximation according to Equation (5):

$$\hat{\phi}_n = \frac{\hat{\phi}_n^{ML}}{\sum\limits_{i \in \mathcal{N}(t)} \hat{\phi}_i^{ML}} \cdot C(\mathcal{N}(t)) \qquad \forall n \in \mathcal{N}(t), \; t \in \mathcal{T} \tag{5}$$

4.5 | Generating Fast but Biased Labels

Since the exact determination of Shapley values in this labeling process is only possible for relatively small instances and thus train data with exact Shapley values can only be generated for relatively small instances, we propose three methods for generating faster but biased labels.

The first option is to use approximated Shapley values as labels that are much faster to generate than exact Shapley values. We use Monte Carlo sampling, an unbiased estimator where the errors are identical and independently distributed [37]. In Monte Carlo sampling, the Shapley value is not calculated as an average over all permutations (see Equation (2)) but only over a small proportion of all permutations. Mathematically, it is expressed in Equation (6):

TABLE 1 | Feature description.

Feature name	Description	Expression
Distance-based features		
Distance to Depot (r)	The distance of the customer to the depot node 0	$\frac{d_{0n}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{0\hat{n}}}$
Distance to m closest customer (r)	Distance to the m closest customer. We include this feature for $m = 1, .., 5$	$\frac{d_{nm}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{\hat{n}m}}$
Distance to gravity center (r)	Distance of the customer to the gravity center g of all customers	$\frac{d_{ng}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{\hat{n}g}}$
Mean distance to other customers (r)	Mean distance of the customer to other customers	$\frac{\frac{1}{\vert\mathcal{N}\vert-1}\sum_{i\in\mathcal{N}\setminus\{n\}} d_{ni}}{\frac{1}{\vert\mathcal{N}\vert}\left(\sum_{\hat{n}\in\mathcal{N}}\left(\frac{1}{\vert\mathcal{N}\vert-1}\sum_{j\in\mathcal{N}\setminus\{\hat{n}\}} d_{\hat{n}j}\right)\right)}$
Instance-based features		
Number of customers	Number of customers in the instance	$\vert\mathcal{N}\vert$
Total cost	Total cost of the (optimal) solved routing problem	$C(\mathcal{N})$
Y/X-coordinate	The X and Y coordinates of the customer location	—
Y/X-coordinate depot	The X and Y coordinates of the depot location	—
Descriptive statistics	Descriptive statistics of the coordinates. We consider the min, max, mean, standard deviation, and the skewness of X and Y coordinates	—
Cost-based-features		
Marginal costs (r)	Marginal costs of a customer	$\frac{C(\mathcal{N})-C(\mathcal{N}\setminus\{n\})}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}}(C(\mathcal{N})-C(\mathcal{N}\setminus\{\hat{n}\}))}$
Shortcut cost (r)	Cost savings when skipping a customer in the given optimal tour by directly connecting its predecessor i and successor j. Unlike marginal cost, this only accounts for local cost changes without reoptimizing the solution	$\frac{d_{in}+d_{nj}-d_{ij}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}}(d_{i\hat{n}}+d_{\hat{n}j}-d_{ij})}$
Cluster-based-features		
Number of clusters	The number of different clusters including outliers as own cluster after application of DBSCAN	—
Number of customers in cluster	The number of customers in the same cluster	—
Distance to cluster center (r)	Distance of the customer to the center of the own cluster $z(n)$	$\frac{d_{nz(n)}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{\hat{n}z(\hat{n})}}$
Cluster distance to depot	Distance of the cluster center $z(n)$ to the depot 0	$\frac{d_{0z(n)}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{0z(\hat{n})}}$
Distance to closest other cluster centroid (r)	Distance of the customer to the center of the closest other cluster $\hat{z}(n)$	$\frac{d_{n\hat{z}(n)}}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} d_{\hat{n}\hat{z}(\hat{n})}}$
Cluster area (r)	The area of the cluster customer n is assigned to, defined as the smallest possible square that fully encloses all customers in the cluster. Let A_n denote the area of customer n's cluster	$\frac{A_n}{\frac{1}{\vert\mathcal{N}\vert}\sum_{\hat{n}\in\mathcal{N}} A_{\hat{n}}}$
CVRP-specific features		
Vehicle capacity—demand ratio	The capacity of the vehicles divided by the total demand	$\frac{Q}{\sum_{\hat{n}\in\mathcal{N}} u_{\hat{n}}}$
Demand (r)	Demand volume of customer n	$\frac{u_n}{\sum_{\hat{n}\in\mathcal{N}} u_{\hat{n}}}$
Customer cluster demand	Demand volume of customer n divided by the average demand volume of all customers in the same cluster. The set $\mathcal{A}$ represents the customers in the own cluster	$\frac{u_n}{\frac{1}{\vert\mathcal{A}\vert}\sum_{\hat{n}\in\mathcal{A}} u_{\hat{n}}}$
Cluster demand share	Cumulative demand volume of customers in the same cluster divided by the cumulative demand volume of all customers	$\frac{\sum_{i\in\mathcal{A}} u_i}{\sum_{\hat{n}\in\mathcal{N}} u_{\hat{n}}}$

$$\hat{\phi}_n^{MC} = \frac{1}{s} \sum_{p \in \mathcal{P}^s} C(p_{n-1} \cup \{n\}) - C(p_{n-1}) \tag{6}$$

$\mathcal{P}^s$ represents s random samples from the set $\mathcal{P}$ of all permutations of customers $\mathcal{N}$. The set of customers that appear in the order of permutation before customers n is represented by p_{n-1}. We use the same set of random permutations for each customer within the same instance. The great advantage of using this approximation method is that the bias is not subject to any systematic over- or underestimation of the Shapley value. Still, the approximation error through the bias corresponds to a distribution around the mean Value of 0. The larger s, the better the approximations. In the following, we refer to this modification as MLSVA-MC. The second option to speed up the label generation process is to evaluate the characteristic function not with an exact solver but with a heuristic that is very fast to evaluate and, therefore, speeds up the labeling process (MLSVA-H). This can also be done within the Monte Carlo sampling to have very fast labels generated by Monte Carlo sampling based on evaluating the routing cost of the subcoalitions through a heuristic (MLSVA-H&MC), which further speeds up the labeling process.

The idea behind this approach is that machine learning models, provided they are correctly calibrated and do not overfit the training data, do not learn the noise caused by the approximation of the labels as long as the deviation of the approximated Shapley values from the true Shapley values is not systematic but random.

5 | Numerical Study

First, we describe our numerical setup (Section 5.1). Then, we show the evaluation metrics (Section 5.2). After that, we detail the implementation of our machine learning models (Section 5.3), describe benchmark approaches (Section 5.4), and show our approximation results compared to these approaches for the TSP and CVRP (Section 5.5). In further experiments, we investigate how the approximation performance changes when training our models on biased labels or only on smaller instances (Section 5.6). Additionally, we report the computing times for generating labels and executing predictions (Section 5.7). Finally, we conduct further experiments on instances with different node distributions and on larger instances to further underscore the robustness of our approach (Section 5.8).

5.1 | Numerical Setup and Instance Generation

Since the exact determination of the Shapley values for large instances represents an extreme computational effort (see Figure 2), we measure the performance on new generated instances with up to 15 nodes, as in other works on Shapley value approximations (e.g., [41]).

We create a total of 5000 new instances each for instance sizes from 7 to 15 nodes for the TSP and 7 to 13 nodes for the CVRP. For the CVRP, we assume a random vehicle capacity between 10 and 18 per instance and a demand volume per customer between 1 and 5 to include different demand structures in the instances. Each instance is generated by randomly setting X and Y coordinates in the interval 0 to 100. The depot is also set randomly in this interval. This allows us to achieve a broad mix of instances that does not follow a specific structure. However, if a special structure exists in the real-world environment, the training data should be generated to match this structure. Of these instances, we use 80% as training data and 20% as test data to evaluate and benchmark our machine learning framework against other methods. We use Gurobi 9.7 as the solver to compute the characteristic function by exactly solving the routing problems, limiting it to four threads. We implemented the TSP and the CVRP with classical formulations that are provided in Appendix A. Although advanced exact solution methods could potentially slightly reduce the time to calculate the exact Shapley values, even with state-of-the-art methods, exact Shapley values for larger instances remain computationally infeasible due to the exponential growth in subcoalitions. The data processing, the addressing of the solver, and the implementation of the machine learning models are conducted in Python 3.12. All experiments are carried out on an AMD Ryzen 9 5950X 16-Core Processor, 3.40 GHz with a 128 Gb RAM.

5.2 | Evaluation Metrics

We use the following evaluation metrics to evaluate our approximations for Shapley values.

- *MAPE*: the mean absolute percentage error from the true Shapley value, illustrated in Expression (7).

$$\text{MAPE} = \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \frac{|\hat{\phi}_n - \phi_n^{SV}|}{\phi_n^{SV}} \times 100 \tag{7}$$

- *MMAXPE*: the mean maximum absolute percentage error from the true Shapley value related to an instance, illustrated in Expression (8).

$$\text{MMAXPE} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \max_{n \in \mathcal{N}(t)} \frac{|\hat{\phi}_n - \phi_n^{SV}|}{\phi_n^{SV}} \times 100 \tag{8}$$

where $\mathcal{T}$ is the set of routing problem instances and $\mathcal{N}(t)$ represents the customers in instance $t \in \mathcal{T}$. We introduce this unusual metric because it is particularly important for evaluating the worst deviation of approximations from the real Shapley value for each instance.

5.3 | Implementation and Hyperparameter Tuning

We utilized various implementation environments and hyperparameter tuning methods to minimize the MAPE after scaling for our machine learning models, which are described individually below. Therefore, we conducted hyperparameter tuning separately for the TSP and CVRP. The resulting configurations with the hyperparameters are presented in Appendix B.

- *NN*: We implement the NN with TensorFlow Keras API. For hyperparameter tuning, we use the Bayesian optimization of the Keras-tuner. This probabilistic model-based approach constructs a posterior distribution to approximate and finds the optimal hyperparameters, balancing exploration and exploitation through an acquisition function [65]. We use Adam Optimizer [66] to train the weights.

- *XGB and RF*: We implement the RF algorithm with the Scikit-learn Python library, while XGB is implemented using the XGBoost Python library [67]. For both methods, we use Bayesian search for hyperparameter tuning.
- *PR, SVR, and KNN*: we implement the algorithms with the Scikit-learn Python library [68] and use a grid search for hyperparameter tuning.

To standardize the features, we apply the StandardScaler from the Scikit-learn Python library, which transforms the data to have a mean (μ) of 0 and a standard deviation (σ) of 1.

5.4 | Benchmark Approaches

We use three proportional methods, along with the current state-of-the-art method SHAPO [41], and Monte Carlo sampling, as described in Section 4.5, as benchmark approaches. SHAPO stands for SHapley Approximation based on a fixed Order and is an efficient method for approximating the Shapley value in our context. It simplifies computations by assuming a fixed servicing order for customers, estimating each customer's marginal contribution to the total cost, and allowing for polynomial-time computation. For a comprehensive treatment of the mathematical foundations and implementation of SHAPO, we refer the reader to Levinger et al. [41]. Note that SHAPO can only be used for the TSP as a benchmark, as it is not designed for the CVRP. To implement SHAPO, we reimplemented their algorithm according to the pseudo-code provided in their paper. For Monte Carlo sampling, we ensured fairness in the benchmark by using a high number of sampled permutations (150) to reduce variance and improve accuracy. As simple proportional methods, we use the depot distance and the reroute margin. Mathematically, these are expressed with Equations (9) and (10):

$$\hat{\phi}_n^{Depot} = C(\mathcal{N}) \cdot \frac{d_{0n}}{\sum_{i\in\mathcal{N}} d_{0i}} \tag{9}$$

$$\hat{\phi}_n^{Reroute} = C(\mathcal{N}) \cdot \frac{C(\mathcal{N}) - C(\mathcal{N} \setminus \{n\})}{\sum_{i\in\mathcal{N}}(C(\mathcal{N}) - C(\mathcal{N} \setminus \{i\}))} \tag{10}$$

A more complex proportional benchmark is the Alternative Cost Avoided Method (ACAM). This method, originally introduced by Tijs and Driessen [69], first assigns each customer their separable cost, defined as their marginal contribution to the total cost. The remaining nonseparable cost is then distributed proportionally based on the difference between each customer's standalone cost and their separable cost, reflecting the savings they achieve by joining the grand coalition. Mathematically, this is expressed with Equation (11):

$$\hat{\phi}_n^{ACAM} = m_n + \frac{(C(\{n\}) - m_n)}{\sum_{i\in\mathcal{N}}(C(\{i\}) - m_i)} \cdot g(\mathcal{N}) \tag{11}$$

where $m_n = C(\mathcal{N}) - C(\mathcal{N} \setminus \{n\})$ is the marginal cost of customer n and $g(\mathcal{N}) = C(\mathcal{N}) - \sum_{n\in\mathcal{N}} m_n$ is the total nonseparable cost.

5.5 | Approximation Performance

We present the approximation performance for the TSP (Section 5.5.1) and the CVRP (Section 5.5.2). Additionally, the 95th Percentile Absolute Percentage Error is reported in Appendix C, along with a detailed illustration of a 7-node TSP and CVRP instance for improved interpretability in Appendix D.

5.5.1 | TSP Results

The results of our benchmark are presented in Tables 2 and 3.

We find that for the TSP, the NN delivers by far the best performance among the machine learning models. With an average MAPE of 2.40%, it also clearly outperforms the current state-of-the-art approximation approach SHAPO, especially for larger instances. For the MMAXPE, the NN outperforms SHAPO even more significantly, which makes it a particularly suitable

TABLE 2 | MAPE [%] for MLSVA differentiated by ML Models and Benchmarks (TSP).

No. of nodes	MLSVA						Benchmarks				
	NN	RF	XGB	PR	SVR	KNN	SHAPO	Depot	Reroute	ACAM	MCS
7	1.61	5.87	3.60	2.94	4.00	22.64	**1.22**	26.40	60.09	10.38	6.35
8	**1.71**	6.05	3.65	2.75	3.04	20.01	1.79	27.65	62.29	12.33	6.83
9	**1.82**	6.60	3.73	2.83	3.42	19.44	2.30	28.58	64.72	14.37	7.26
10	**1.99**	6.59	3.89	3.03	3.41	17.77	2.84	28.60	66.30	15.46	7.63
11	**2.32**	6.72	4.21	3.31	4.02	20.02	3.42	29.21	67.88	16.94	7.78
12	**2.64**	6.84	4.34	3.85	4.42	18.02	4.00	29.30	68.59	17.98	8.02
13	**2.90**	7.09	4.56	4.50	4.72	19.04	4.46	29.34	69.69	18.67	8.29
14	**3.15**	7.38	4.97	4.51	4.80	17.80	5.02	30.10	68.51	19.12	8.38
15	**3.45**	7.65	5.07	4.77	5.03	17.58	5.56	29.98	69.44	19.83	8.70
avg.	**2.40**	6.75	4.22	3.61	4.10	19.15	3.40	28.80	66.39	16.12	7.69

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

TABLE 3 | MMAXPE [%] for MLSVA differentiated by ML Models and Benchmarks (TSP).

No. of nodes	MLSVA						Benchmarks				
	NN	RF	XGB	PR	SVR	KNN	SHAPO	Depot	Reroute	ACAM	MCS
7	4.25	14.35	9.70	9.22	13.72	86.72	**3.08**	52.83	113.93	27.83	14.40
8	**4.79**	16.29	10.18	8.86	9.94	81.00	4.96	56.17	121.26	34.40	16.33
9	**5.13**	19.36	11.34	9.41	12.00	85.38	6.73	58.45	131.19	40.66	17.69
10	**5.87**	20.21	12.23	10.89	12.51	80.05	8.79	59.99	144.67	45.15	19.43
11	**7.14**	21.26	15.19	12.38	17.30	112.07	10.84	61.47	151.87	48.97	20.62
12	**8.51**	22.45	14.77	16.09	20.43	99.70	13.34	62.68	161.00	52.33	21.70
13	**9.77**	24.76	17.19	21.25	22.90	117.46	15.51	64.01	171.32	53.86	22.69
14	**10.21**	26.97	19.74	18.96	21.35	111.81	17.75	65.99	168.88	55.29	23.23
15	**11.93**	28.96	19.42	20.26	23.29	111.95	20.12	66.17	177.30	57.50	24.75
avg.	**7.51**	21.62	14.42	14.15	17.05	98.46	11.24	60.86	149.05	46.22	20.09

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

TABLE 4 | MAPE [%] for MLSVA differentiated by ML Models and Benchmarks (CVRP).

No. of nodes	MLSVA						Benchmarks			
	NN	RF	XGB	PR	SVR	KNN	Depot	Reroute	ACAM	MCS
7	**2.93**	6.43	4.49	4.61	5.31	20.71	21.70	45.54	22.72	4.98
8	**3.21**	6.51	4.47	4.17	4.64	19.76	21.78	46.89	24.26	5.09
9	**3.42**	6.67	4.90	4.30	4.57	18.44	22.53	45.41	22.82	5.19
10	**3.47**	6.62	4.88	4.06	4.41	16.87	23.00	45.08	22.81	5.16
11	**3.69**	6.99	5.08	4.29	4.62	16.53	23.66	45.00	23.91	5.30
12	**3.86**	6.90	5.09	4.40	4.77	17.53	24.04	43.78	24.18	5.34
13	**3.98**	6.97	5.21	4.49	5.24	17.38	24.32	43.52	25.14	5.42
avg.	**3.51**	6.73	4.87	4.33	4.79	18.17	23.00	45.03	23.69	5.21

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

approach when cost allocations are desired where the worst approximation is not far away from the real Shapley value.

We further observe that the advanced machine learning models (all except KNN) significantly outperform the simple proportional proxies, although they do not reach the approximation quality achieved by the NN. Among the proportional methods, ACAM clearly delivers the best performance, yet it remains far behind the advanced machine learning models. Additionally, all advanced machine learning models outperform the MCS benchmark in MAPE. We also observe that as the number of customers in an instance increases, the approximation error tends to rise. However, this effect is even more pronounced for SHAPO and MCS, emphasizing the inherent difficulty of accurately approximating Shapley values. While a clear trend is visible, it remains uncertain whether this pattern would persist with larger instances or eventually stabilize.

5.5.2 | CVRP Results

Our benchmark results are presented in Tables 4 and 5 for the respective evaluation metrics.

Consistent with the TSP results, the NN achieves the best approximations in both MAPE and MMAXPE, although the overall approximation quality is slightly lower compared to the TSP. All advanced machine learning models again clearly outperform the proportional methods in both metrics. The MCS benchmark achieves comparable results to some advanced machine learning models in terms of MAPE and is only surpassed by the NN in MMAXPE; however, it incurs a significantly higher computational cost (see Section 5.7.2). Additionally, we observe the same trade-off as in the TSP: the approximation error increases with instance size, although this effect is less pronounced compared to the TSP.

All reported values represent scaled predictions. Detailed analysis revealed that scaling the approximations to match the total routing cost consistently improves performance across all machine learning models. However, this improvement remains small (e.g., 0.16 percentage points for the NN in the TSP), except for KNN, as the cumulative predictions of the machine learning models already closely match the actual total instance costs. Similarly, improvements in the MMAXPE metric resulting from scaling are also minor (e.g., 0.31 percentage points for the NN in the TSP).

TABLE 5 | MMAXPE [%] for MLSVA differentiated by ML Models and Benchmarks (CVRP).

No. of nodes	MLSVA						Benchmarks			
	NN	RF	XGB	PR	SVR	KNN	Depot	Reroute	ACAM	MCS
7	**7.70**	15.63	11.59	13.48	16.08	76.41	47.97	98.22	56.92	11.25
8	**8.82**	16.19	12.80	12.32	14.36	81.41	49.65	102.16	58.70	12.20
9	**9.57**	17.19	13.70	13.35	14.15	83.73	55.79	103.91	58.10	12.84
10	**9.90**	17.93	13.93	12.34	13.60	78.68	58.28	103.88	59.10	13.23
11	**10.82**	20.54	15.59	14.49	16.10	89.76	62.67	108.50	63.60	14.45
12	**11.76**	21.23	16.20	15.61	17.78	95.43	67.02	109.74	66.80	14.74
13	**12.59**	21.67	16.63	15.64	20.93	94.60	70.77	112.23	71.10	14.82
avg.	**10.17**	18.63	14.35	13.89	16.14	85.72	58.88	105.52	62.05	13.36

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

TABLE 6 | Performance of MC-sampling and MLSVA-MC.

No. of nodes	TSP				CVRP			
	MC		MLSVA-MC		MC		MLSVA-MC	
	MAPE [%]	MMAXPE [%]	Δ MAPE [%]	Δ MMAXPE [%]	MAPE [%]	MMAXPE [%]	Δ MAPE [%]	Δ MMAXPE [%]
7	10.91	24.71	1.84	4.47	8.80	19.91	1.58	3.33
8	11.87	28.17	1.59	4.04	8.84	21.11	1.35	2.95
9	12.45	30.71	1.74	4.86	9.07	22.39	1.11	2.62
10	13.01	33.03	1.73	4.73	9.24	23.72	0.97	2.16
11	13.45	35.04	1.75	4.93	9.27	24.54	0.94	2.74
12	13.97	37.49	1.75	5.24	9.37	25.28	0.85	2.54
13	14.34	39.23	1.82	5.33	9.34	25.97	0.83	2.78
14	14.66	41.14	1.74	4.98	—	—	—	—
15	15.11	43.18	1.75	4.90	—	—	—	—
avg.	13.31	34.74	1.75	4.83	9.13	23.27	1.09	2.73

While our MLSVA approach does not explicitly enforce the axiomatic properties of the Shapley value, the resulting approximations closely align with the true values. Consequently, the fairness properties of the Shapley value are nearly preserved, providing a more game-theoretically justified cost allocation compared to simple proportional methods and other practical cost allocation methods.

5.6 | Training With Approximated Labels

This section shows the effects of fast-to-generate but biased labels (see Section 4.5) in the training process. We show this for Monte Carlo sampled Shapley values (MLSVA-MC) (Section 5.6.1), for the labels generated by evaluating the routing problems with a very fast but simple heuristic (MLSVA-H), as well as for the combination of both biases (MLSVA-H&MC) (Section 5.6.2). We further show the effects of only using smaller instances as training data (Section 5.6.3).

If these experiments demonstrate that the approximation performance decreases only minimally, this shows that we can also apply our methodology to larger instances with multiple customers, as we can generate the necessary biased labels very quickly and still achieve good approximation results. In the following, we conduct all our experiments using only the best-performing model, the NN. Note that we have individually tuned the number of epochs in the training process of the NN for these modifications, as we have found that for biased labels, a lower number of epochs has a better effect on the performance of our approach.

5.6.1 | Monte Carlo Approximated Labels

In this section, we evaluate the effect of labels generated by Monte Carlo sampling. To avoid memory errors, we generate the random permutations using the Fisher-Yates shuffle algorithm, which runs in complexity of $O(n)$. Further, we take advantage of memorization (caching the value of already evaluated subcoalitions) as this significantly reduces computing times for generating the labels. We use a small number of $s = 50$ random samples of permutations for our experiments. We deliberately use this small number to investigate the effect of a strong bias in the training data on the performance of the NN. The approximation quality of the labels (MC performance), as well as the approximation quality of the NN trained with these labels (MLSVA-MC), is shown in Table 6 for both the TSP and the CVRP. The performance of Monte Carlo sampling is reported in terms of MAPE and MMAXPE, while the performance of the MLSVA-MC is evaluated based on the increase in MAPE and MMAXPE compared the results in Sections 5.5.1 and 5.5.2 (expressed in percentage points as Δ MAPE / Δ MMAXPE). For instance, a value of 1.5% indicates that MAPE increases from, e.g., 3.0% to 4.5%.

We observe that the labels generated through Monte Carlo sampling exhibit a strong bias, with MAPE values of 13.31% for the TSP and 9.13% for the CVRP, and MMAXPE values reaching 34.74% and 23.27%, respectively. However, our MLSVA-MC achieves excellent results, performing only slightly worse than MLSVA trained on exact Shapley values. For instance, in the CVRP, the average MAPE increases only marginally from 3.51% to 4.60%. Additionally, while MC produces high MMAXPE values, the impact on MMAXPE when using these labels remains relatively minor.

We could thus show that it is possible to approximate Shapley values with high precision, although using approximated Shapley values with very strong biases as labels. This shows that the performance of our approach only slightly declines when the machine learning models are not trained with exact labels but with biased labels that are much faster to generate. This insight is particularly important for larger instance sizes, for which exact Shapley values can no longer be generated.

5.6.2 | Trained With Heuristically Solved Routing Problems

In this section, we analyze the impact on performance if the routing problems required for labeling are not solved exactly but instead with a very fast but straightforward heuristic. We deliberately use a heuristic that can be solved quickly but no longer finds the optimal solution, even for the smallest instances, to check the effects of routing problems that have not been solved optimally. Therefore, we use a simple *nearest neighbor heuristic* as a starting solution and then improve this solution with a simple *remove and best insert heuristic*. The pseudo-code is shown in Algorithm 2.

We provide two different configurations of the heuristic referred to as H1 and H2 to evaluate the effect of two different performing heuristics. We set the parameters as follows: H1: $\lambda = 2$ and $\tau = 2|\mathcal{N}|$, H2: $\lambda = 4$ and $\tau = 4|\mathcal{N}|$. The parameter λ defines the maximum number of customers that can be randomly selected and removed from the solution in each iteration. Table 7 shows

ALGORITHM 2 | Remove and best insert heuristic.

```
1:  $\mathcal{S} \leftarrow$ nearestNeighborHeuristic()        ▷ Generate initial solution $\mathcal{S}$
2:  $it \leftarrow 0$
3:  while $it < \tau$ do
4:      $\hat{\mathcal{N}} \leftarrow$ randomSelect($\mathcal{N}, 1, \lambda$)     ▷ Select between 1 and $\lambda$ customers
5:      $\mathcal{S} \leftarrow$ remove($\mathcal{S}, \hat{\mathcal{N}}$)       ▷ Remove customers from solution
6:      for $n \in \hat{\mathcal{N}}$ do
7:          $\mathcal{S} \leftarrow$ bestInsertion($\mathcal{S}, n$)    ▷ Insert customer in solution
8:      end for
9:      $it \leftarrow it + 1$
10: end while
```

TABLE 7 | Mean GAP between the optimal solution and heuristic solutions for different instance sizes averaged over 100 instances.

	H1		H2	
No. of nodes	**GAP TSP [%]**	**GAP CVRP [%]**	**GAP TSP [%]**	**GAP CVRP [%]**
7	0.13	2.94	0.00	0.03
8	0.56	3.89	0.00	0.02
9	1.20	6.67	0.00	1.03
10	1.62	5.98	0.17	1.78
11	2.22	6.88	0.22	3.27
12	2.78	8.66	0.40	3.86
13	3.43	8.58	0.64	3.47
14	4.20	—	1.07	—
15	4.42	—	1.63	—
avg.	2.28	6.23	0.46	1.92

the TSP and CVRP performance of the heuristic compared to the optimal routing costs.

As can be seen, the heuristic leads to deviations from the optimal solution even with our small instance sizes. This effect is intentional, as it allows us to evaluate the influence of this source of bias on the approximation performance of our NN. Notably, the heuristic performs slightly worse for the CVRP than for the TSP, and H1 is significantly less effective than H2.

To evaluate the effects on the approximation performance, we distinguish between two cases. In the first case (MLSVA-H), we use the heuristic to determine the characteristic function used for the Shapley values. In the second case (MLSVA-H&MC), we generate the labels by Monte Carlo sampling with $s = 250$ samples and use the heuristic to evaluate the routing costs of the subcoalitions. As a result, we have two sources of bias in the labels. The first one is by sampling permutations using the Monte Carlo sampling method. The second one is by the heuristic evaluation of the subcoalitions cost. The approximation performance of the NN for these two cases is presented in Table 8 (Δ MAPE [%]) and Table 9 (Δ MMAXPE [%]) as an increase in percentage points.

As demonstrated, even with biased labels, our approach maintains strong approximation performance. Notably, even when both biases are present (MLSVA-H&MC), the decline in MAPE remains minimal compared to the original MLSVA without modifications. Moreover, the results clearly indicate that the stronger the heuristic, the smaller the performance loss. In contrast to MAPE, the impact on MMAXPE is significantly larger, particularly when using the weaker heuristic H1. A detailed analysis reveals that this is primarily driven by customers with very small Shapley values, as the heuristic evaluation of their marginal costs tends to overestimate their contribution to subcoalitions. The comparison between H1 and H2 highlights substantial differences in MMAXPE, reinforcing that a more accurate heuristic reduces this bias effectively.

Nevertheless, we were able to show that our approach achieves good approximation results even with large biases in the labels.

TABLE 8 | Performance of MLSVA-H and MLSVA-H & MC in Δ MAPE [%].

No. of nodes	TSP				CVRP			
	MLSVA-H		MLSVA-H & MC		MLSVA-H		MLSVA-H & MC	
	H1	H2	H1	H2	H1	H2	H1	H2
7	1.13	0.50	1.65	0.84	2.32	0.58	2.73	0.96
8	0.95	0.46	1.47	0.63	2.49	0.46	2.63	0.76
9	0.93	0.52	1.59	0.75	2.56	0.54	3.18	0.90
10	1.06	0.53	1.61	0.71	2.74	0.60	3.20	0.93
11	1.38	0.48	1.83	0.72	2.85	0.86	3.61	1.07
12	1.51	0.56	2.00	0.82	3.23	0.95	4.01	1.20
13	1.69	0.58	2.16	0.89	3.33	1.31	3.87	1.53
14	2.11	0.77	2.31	1.02	—	—	—	—
15	2.14	0.81	2.25	1.03	—	—	—	—
avg.	1.43	0.58	1.87	0.82	2.79	0.76	3.32	1.05

TABLE 9 | Performance of MLSVA-H and MLSVA-H & MC in Δ MMAXPE [%].

No. of nodes	TSP				CVRP			
	MLSVA-H		MLSVA-H & MC		MLSVA-H		MLSVA-H & MC	
	H1	H2	H1	H2	H1	H2	H1	H2
7	3.59	1.20	4.63	2.19	7.81	2.07	8.65	3.31
8	3.48	1.24	4.41	1.50	9.96	1.92	10.27	2.94
9	3.88	1.82	5.92	2.25	11.58	2.37	14.38	3.70
10	4.41	2.02	6.20	2.28	13.84	2.99	15.69	4.33
11	6.90	2.13	8.51	2.84	17.01	5.13	19.66	5.60
12	7.65	2.90	9.42	3.54	21.83	6.44	21.27	6.53
13	8.53	3.05	10.23	4.24	23.08	8.18	22.44	8.38
14	11.57	4.79	12.91	5.27	—	—	—	—
15	11.05	4.77	11.53	4.99	—	—	—	—
avg.	6.78	2.66	8.20	3.23	15.02	4.16	16.05	4.97

This finding is important because it enables us to quickly generate labels for larger instances, allowing our machine learning-based approach to train efficiently without a substantial loss in performance. Overall, Sections 5.6.1 and 5.6.2 demonstrate that our method can effectively learn well-approximated Shapley values from poorly approximated ones, which highlights the applicability of our approach for larger instances. Furthermore, this is an important methodological insight, as it opens the way to improve further approximation algorithms in other contexts through machine learning algorithms.

5.6.3 | Trained on Smaller Instances

In this experiment, we examine the effects of training the NN on smaller instances and then testing it on larger instances. To do this, we use the same training data as before, with the difference that all instances with more than 12 nodes for the TSP and more than 10 nodes for the CVRP are excluded. We then let our NN approximate Shapley values for instances with 13-15 nodes for the TSP and 11-13 nodes for the CVRP. The performance is shown in Table 10.

TABLE 10 | Performance Improvement in Δ MAPE [%] and Δ MMAXPE [%].

TSP			CVRP		
No. of nodes	Δ MAPE [%]	Δ MMAXPE [%]	No. of nodes	Δ MAPE [%]	Δ MMAXPE [%]
13	0.37	1.80	11	0.67	2.40
14	0.66	2.61	12	0.99	3.69
15	1.02	3.15	13	1.40	4.84

As can be seen, the approximation performance only increases marginally if the NN is trained exclusively on smaller instances. For example, in the case of TSP instances with 15 nodes, the MAPE increases only slightly from 3.45% to 4.47%. Consequently, this experiment shows that our methodology also leads to good approximation results when we train on smaller instances than

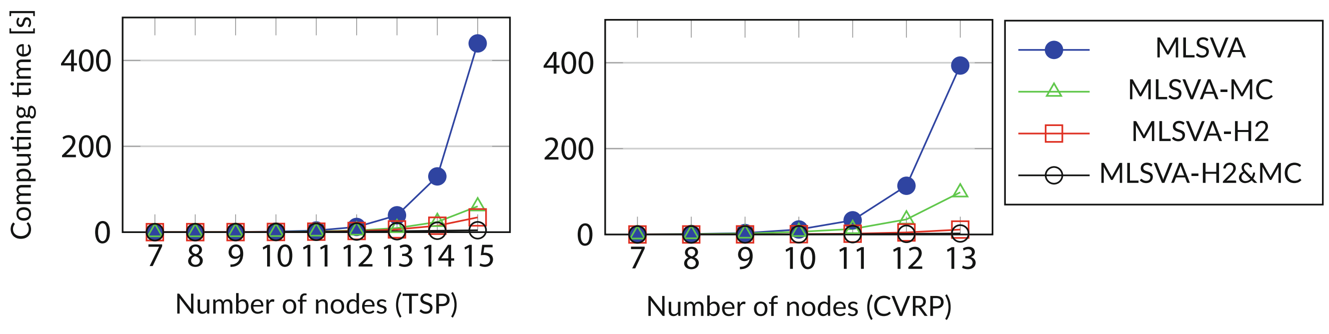

FIGURE 3 | Computing time for labeling a single instance averaged over 100 instances.

we test. Overall, the experiments in Section 5.6 show the scalability of our methodology, as it creates a framework that can be used to train machine learning models efficiently for larger instances.

5.7 | Runtime Analysis

In this section, we analyze the computational burden associated with different stages of our approach. First, in Section 5.7.1, we examine the time required to generate labeled training data, comparing the computational effort of our proposed biased labels to the exact computation of Shapley values. However, it is essential to emphasize that these labeling runtimes do not reflect the actual execution time of our MLSVA approach. Once the machine learning models are trained, only the computational burden for feature generation and the machine learning prediction time is relevant. These execution times are presented in Section 5.7.2.

5.7.1 | Label Generation

Figure 3 presents the runtimes required to generate labels for a single instance of different problem sizes. As already outlined, computing exact Shapley values for labeling (MLSVA) becomes infeasible for larger instances due to the factorial growth in complexity. In contrast, our proposed biased labels—particularly those incorporating heuristics—enable significantly faster labeling. For example, the MLSVA-H2&MC label requires only 4.45 s for a TSP instance with 15 nodes, demonstrating that our method can efficiently generate training data for much larger instances.

These results highlight the practicality of our approach in enabling scalable training datasets for machine learning models.

5.7.2 | Execution Times

Once the machine learning model is trained, execution time consists of two components: feature preparation and prediction. The latter is negligible, with the NN requiring only 0.000028 s per observation in the TSP, and similar times across other models and problems. The primary computational effort arises from feature generation, particularly the evaluation of marginal costs,

TABLE 11 | Runtime in seconds.

	TSP		CVRP	
No. of nodes	Exact MC	Heuristic MC	Exact MC	Heuristic MC
< 11	0.18	0.07	0.6	0.06
12	1.05	0.09	2.50	0.07
13	2.63	0.09	4.16	0.08
14	6.57	0.10	—	—
15	16.49	0.11	—	—

which requires solving the instance while omitting each customer individually.

To address this, we report runtimes for two configurations: one with exact marginal cost calculations (Exact MC) and one where marginal costs are evaluated using the heuristic from Section 5.6.2 (Heuristic MC). Table 11 presents execution times for instances with 11–15 nodes.

While exact marginal cost evaluation remains feasible for the considered instance sizes, it becomes computationally expensive for larger instances. In contrast, the heuristic-based approach ensures consistently low runtimes, making our method scalable and well-suited for generating fast approximations even in large-scale applications. Notably, the heuristic evaluation of marginal costs incurs only a minor performance loss of 0.37% on average for TSP instances and 0.69% for CVRP instances in terms of Δ MAPE—despite relying on a relatively weak heuristic.

For comparison, the MC benchmark in Sections 5.5.1 and 5.5.2 already requires 117.54 s for 15-node TSP instances and 295.33 s for 13-node CVRP instances, whereas SHAPO, being a polynomial-time algorithm, runs almost instantly.

5.8 | Further Experiments

In this section, we conduct two experiments that further demonstrate the robustness of our approach. First, we examine the effects of testing our approach on instances with different characteristics from those it was trained on (Section 5.8.1). Next, we conduct experiments on significantly larger instances (Section 5.8.2).

5.8.1 | Testing on Different Instances

We examine the effects of training the NN on the previously introduced training data and then applying it to newly generated instances with a different node location distribution. Specifically, we created 100 new instances for each instance size ranging from 7 to 15 nodes for the TSP, and from 7 to 13 nodes for the CVRP. These new instances differ from the original ones in that the customer coordinates follow a distinct spatial distribution. In this case, customer locations are drawn from a bimodal triangular distribution: with 50% probability, nodes are sampled from a triangular distribution over the left half of the coordinate space $[0, 50] \times [0, 100]$ with its peak at $(25, 50)$; the other 50% are drawn from the right half $[50, 100] \times [0, 100]$, peaking at $(75, 50)$. This results in two spatially distinct cluster centers. For a visual representation of the distribution, see the probability heatmap in Figure E1 in Appendix E. The depot note is still set randomly. Table 12 presents the increase in MAPE and MMAXPE for these newly generated instances.

As shown, the performance of MLSVA deteriorates only minimally when tested on instances that differ structurally from the training data. This demonstrates the robustness of our approach in handling different spatial distributions. Nevertheless, we recommend that for optimal performance, the training instances should be structurally similar to the instances on which the MLSVA is subsequently applied.

TABLE 12 | Performance on different characterized instances in Δ MAPE [%] and Δ MMAXPE [%].

	TSP		CVRP	
No. of nodes	**Δ MAPE [%]**	**Δ MMAXPE [%]**	**Δ MAPE [%]**	**Δ MMAXPE [%]**
7	0.81	1.53	0.78	2.08
8	0.86	1.54	1.30	2.80
9	0.96	1.86	1.16	2.47
10	0.76	1.13	1.31	2.71
11	0.65	0.83	1.58	3.93
12	0.39	0.83	1.18	2.67
13	0.25	0.58	1.60	4.41
14	0.51	1.06	—	—
15	0.38	0.39	—	—
avg.	0.62	1.08	1.27	3.01

5.8.2 | Stability on Larger Instances

Validating our method on larger instances presents a challenge, as generating true labels becomes infeasible. To demonstrate the stability of our approach, we generated 1500 instances for each individual instance size ranging from 31 to 35 nodes for both the TSP and the CVRP, following the same settings as described in Section 5.1, and using an 80/20 train-test split. The observations were labeled using the biased labels of MLSVA-H2&MC from Section 5.6.2 and the marginal cost were heuristically evaluated.

Since both the biased labels and our approximations are estimates of the Shapley value, traditional evaluation metrics like MAPE and MMAXPE are not directly applicable. Instead, we focus on the relationship between the biased labels and our Shapley value approximations. Figure 4 illustrates this relationship by comparing our approximations with the biased labels for both the TSP and CVRP. The scatter plots use transparency to better visualize the density of data points.

As shown in the figure, there is a clear correlation between the biased labels and the Shapley value approximations, as evidenced by a Pearson correlation coefficient of 0.94 for the TSP and 0.97 for the CVRP. The stronger performance for the CVRP can be attributed to the fact that the Monte-Carlo generated labels are closer to the true Shapley values for the CVRP than for the TSP (see Table 6 in Section 5.6.1). This suggests that our approximation approach remains stable even on larger instances. Moreover, as indicated by the experiments in Section 5.6, which show that we can generate well-approximated Shapley values even with highly biased labels, the deviation from the true Shapley value is expected to be much smaller than the deviation from the biased labels. Further, regarding runtime considerations, we observe that labeling a 35-node instance takes an average of 85.42 s, which

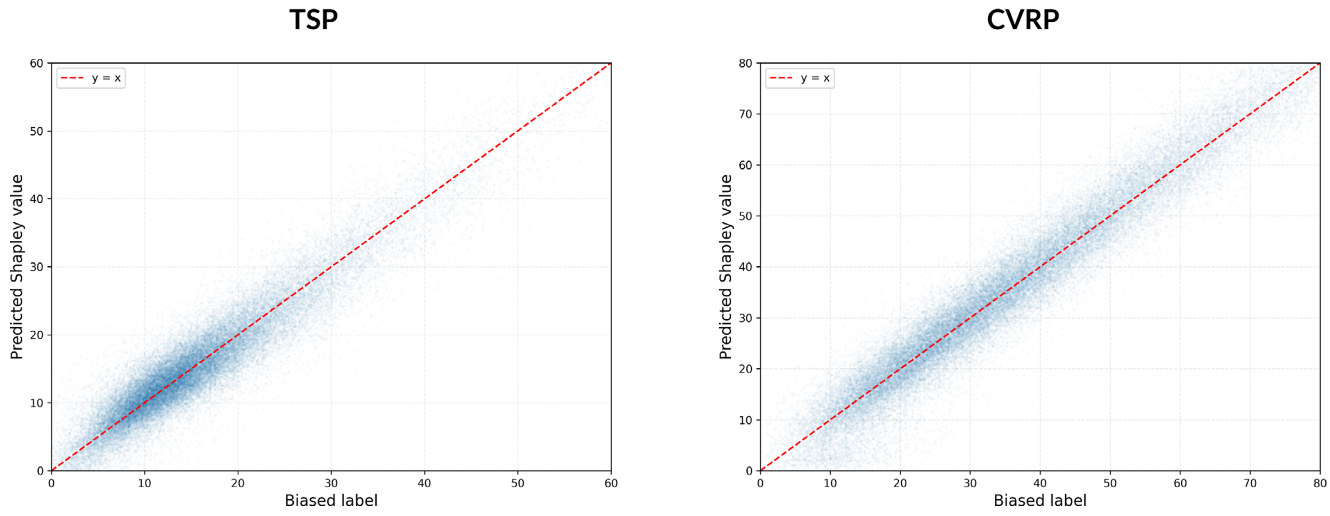

FIGURE 4 | Comparison of Shapley approximations and biased labels.

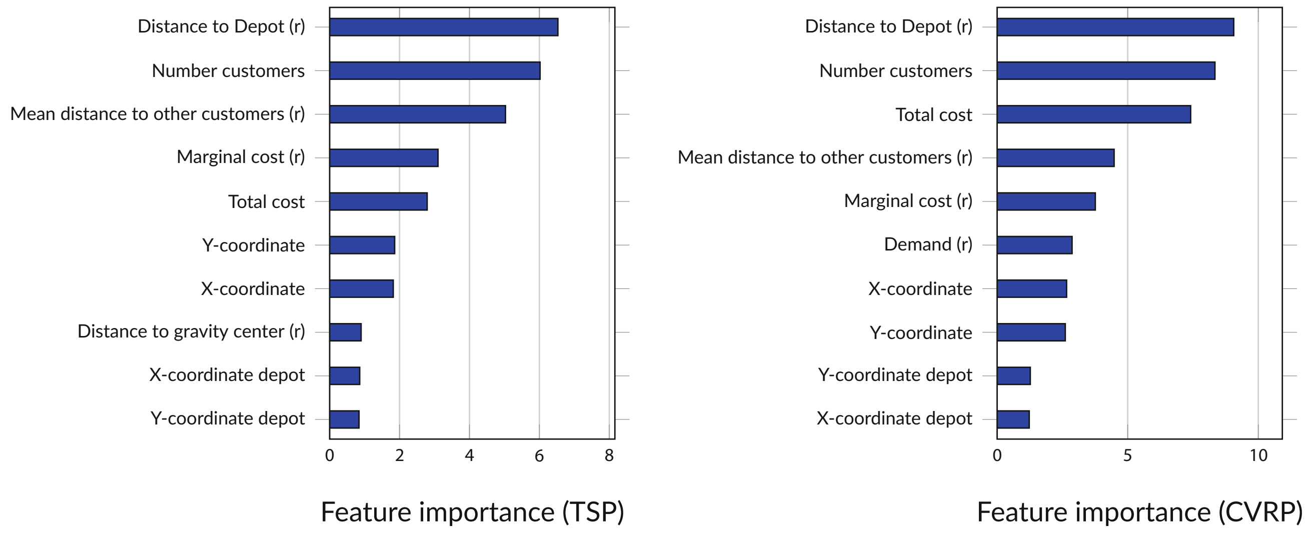

FIGURE 5 | SHAP values of the ten most important features. (a) Feature importance (TSP). (b) Feature importance (CVRP).

makes it feasible to generate large labeled datasets for training the machine learning models. The execution time remains efficient, with an average feature generation time of 0.76 s per instance for the 35-node cases, enabling fast approximations even for larger instances.

6 | Analysis of Feature Importance

In this section, we evaluate the importance of the features and thereby provide managerial insights about the main cost drivers in routing problems. We do this by using the Python package SHAP (SHapley Additive exPlanations) [32]. This package interprets the output of machine learning models by computing the contribution of each feature to the prediction inspired by Shapley values. Figure 5 depicts the SHAP Values of the ten features with the highest value for the TSP and CVRP.

We observe that the ten most important features are nearly identical for both the TSP and CVRP, with the only difference being the addition of the demand ratio in the CVRP. Interestingly, features related to the clustering of individual customers do not rank among the top ten in either case but still contribute to the approximation performance.

Table 13 shows the increase in MAPE of our MLSVA when trained with only these top ten features. For the TSP, the performance remains nearly comparable to the fully trained NN. For the CVRP, there is a slightly more noticeable decline in performance; however, the Shapley values are still approximated with an solid average MAPE. Detailed further analysis revealed the following effects:

- *Correlation among features*: while there is only minor correlation among the top ten features, some features exhibit high pairwise correlation, such as *Marginal Cost (r)* and *Shortcut cost (r)*. Despite this, their joint inclusion still improves the model's performance. Similarly, while distance-based features show some correlation, including all of them leads to the best overall performance.

TABLE 13 | Performance of the MLSVA trained with the ten most important features in Δ MAPE [%] and Δ MMAXPE [%].

	TSP		CVRP	
No. of nodes	**Δ MAPE [%]**	**Δ MMAXPE [%]**	**Δ MAPE [%]**	**Δ MMAXPE [%]**
7	0.71	2.36	1.49	3.88
8	0.89	2.73	1.57	3.61
9	1.12	3.63	1.87	4.53
10	1.11	3.21	1.78	3.95
11	1.18	3.75	1.88	4.12
12	1.18	2.86	1.79	4.02
13	1.27	3.53	1.88	4.04
14	1.34	3.69	—	—
15	1.28	2.96	—	—
avg.	1.12	3.19	1.75	4.02

- *Direction of influence*: for some features—such as *Mean Distance to Other Customers (r)* or *Marginal Cost (r)*—a consistent positive correlation with the model output is observed. In contrast, features like the coordinates exhibit less clear directional effects, as their influence is often shaped by interactions with other spatial attributes, such as the depot location or overall instance-based features.

- *Other machine learning models*: while the results presented here are for the NN, we also computed the SHAP values for RF, XGB, and PR. The obtained SHAP values across the different models are highly correlated, with the lowest Pearson correlation coefficient between any pair of models being 0.86. Note that for these models, a subset of the data was used for SHAP value computation.

7 | Generalizability of the Methodology

In this section, we show the generalizability of our approach by applying it to the approximation of Shapley values for items

TABLE 14 | Features (bin packing).

Feature name	Description	Expression
Weight (r)	Weight of item i	$\frac{w_i}{\frac{1}{\lvert\mathcal{I}\rvert}\sum_{j\in\mathcal{I}} w_j}$
Size (r)	Size of item i	$\frac{s_i}{\frac{1}{\lvert\mathcal{I}\rvert}\sum_{j\in\mathcal{I}} s_j}$
Weight capacity ratio	Weight capacity of bins divided by total weight of all items	$\frac{W}{\sum_{i\in\mathcal{I}} w_i}$
Size capacity ratio	Size capacity of bins divided by total size of all items	$\frac{S}{\sum_{i\in\mathcal{I}} s_i}$
No. of bins	Number of bins in (optimal) solution	$\sum_{b\in\mathcal{B}} y_b$
No. of items	Number of items in the instance	$\lvert\mathcal{I}\rvert$
σ−weight	Standard deviation of item weights	$\sigma(\{w_i\}_{i\in\mathcal{I}})$
σ−size	Standard deviation of item sizes	$\sigma(\{s_i\}_{i\in\mathcal{I}})$
Bin utilization size	Utilization of the bins by size	$\frac{\sum_{i\in\mathcal{I}} s_i}{\sum_{b\in\mathcal{B}} y_b \cdot S}$
Bin utilization weight	Utilization of the bins by weight	$\frac{\sum_{i\in\mathcal{I}} w_i}{\sum_{b\in\mathcal{B}} y_b \cdot W}$
Item-Bin size ratio	Items size divided by bins size capacity	$\frac{s_i}{S}$
Item-Bin weight ratio	Items weight divided by bins weight capacity	$\frac{w_i}{W}$
Weight x-percentile ratio	Weight of item i divided by the x-percentile of weight distribution. We include this feature for $x \in \{0.0, 0.25, 0.5, 0.75, 1\}$	$\frac{w_i}{P_x(w)}$
Size x-percentile ratio	Size of item i divided by the x-percentile of size distribution. We include this feature for $x \in \{0.0, 0.25, 0.5, 0.75, 1\}$	$\frac{s_i}{P_x(s)}$
Weight percentile	Percentile rank of item i's weight	Percentile$(w_i, w_j : j \in \mathcal{I})$
Size percentile	Percentile rank of item i's size	Percentile$(s_i, s_j : j \in \mathcal{I})$
Weight combinations	Combinations of items, including this one fitting in a bin by weight	—
Size combinations	Combinations of items, including this one fitting in a bin by size	—
Total combinations	Combinations of items, including this one fitting in a bin by weight and size	—
Marginal cost (r)	Marginal cost of item i	$\frac{C(\mathcal{I})-C(\mathcal{I}\setminus\{i\})}{\frac{1}{\lvert\mathcal{I}\rvert}\sum_{i\in\mathcal{I}}(C(\mathcal{I})-C(\mathcal{I}\setminus\{i\}))}$

in a variant of the bin packing problem with size and weight constraints.

Let $\mathcal{I}$ be the set of items. Each item $i \in \mathcal{I}$ has a specific weight w_i and a specific size s_i. These items must be packed into bins, whereby the weight capacity W and the size capacity S must not be exceeded for each bin. The objective is to minimize the number of bins used. We assume a uniform cost of one per bin, allowing the number of bins to be interpreted as total cost. The mathematical model formulation and notation is available in Appendix A.

The specific problem characteristic of bin packing problems is that the marginal costs of the items have a binary structure where they are either 0 (no new bin must be opened) or 1 (exactly one new bin must be opened). We apply our MLSVA as described in Section 4 with exact Shapley values as labels. The only difference is the use of bin packing-specific features that are described in Table 14. These features capture the size and weight characteristics of items, bin utilization, and distribution-based statistics, ensuring the model accounts for key structural properties of bin packing instances.

We generated 3000 instances for each number of items, ranging from 7 to 15. The weights and sizes of the items are separately randomly set between 1 and 10. The bin size and weight are separately randomly set between 17 and 25. Once again, we used 80% of the instances for training and 20% for testing. In addition to the performance of the machine learning models, we also show the performance of the most plausible proportional approximation method by weight and size of the items. Formally, WS represents the approximation based on the equally weighted shares of weight and size and is expressed with Equation (12),

$$\hat{\phi}_i^{ws} = C(\mathcal{I}) \cdot \left(0.5 \cdot \frac{w_i}{\sum_{j\in\mathcal{I}} w_j} + 0.5 \cdot \frac{s_i}{\sum_{j\in\mathcal{I}} s_j}\right) \tag{12}$$

Table 15 displays the performance of the MLSVA applied to this bin packing problem. We observe that similar to the routing problems, the NN delivers the best approximation results. In contrast, the performance does not decrease with an increasing number of items in the instance. The results demonstrate that our methodology also leads to excellent approximations for Shapley values in an entirely different problem setting, like bin packing problems.

TABLE 15 | MAPE and MMAXPE [%] for MLSVA differentiated by ML Models (bin packing).

No. of nodes	MAPE							MMAXPE						
	NN	RF	XGB	PR	SVR	KNN	WS	NN	RF	XGB	PR	SVR	KNN	WS
7	**2.81**	3.26	3.27	4.03	3.73	6.88	23.49	**6.85**	8.36	8.38	9.99	9.25	16.97	55.70
8	**3.37**	4.00	3.69	4.16	3.88	6.16	22.62	**8.96**	10.71	9.77	10.81	10.23	15.93	57.94
9	**3.59**	4.36	3.90	4.32	4.04	6.06	22.39	**9.45**	11.65	10.40	11.15	10.67	15.86	57.91
10	**3.48**	4.24	3.67	4.12	3.86	5.39	21.17	**9.62**	11.93	10.26	11.37	10.59	14.84	56.78
11	3.50	4.03	**3.49**	3.86	3.62	5.15	21.68	**10.08**	11.61	10.10	11.19	10.53	14.53	61.29
12	**3.15**	3.90	3.29	3.69	3.43	4.79	22.01	**9.48**	11.49	9.63	10.68	9.87	13.87	63.90
13	**3.42**	4.19	3.51	4.10	3.79	4.94	21.97	**10.11**	12.81	10.81	12.18	11.06	14.74	66.43
14	**3.05**	3.78	3.13	3.70	3.32	4.37	21.83	9.84	11.48	**9.64**	11.29	9.82	12.96	65.84
15	**3.15**	3.90	3.19	3.72	3.33	4.41	21.72	**9.73**	12.21	9.89	11.63	10.02	13.46	68.54
avg.	**3.28**	3.96	3.46	3.97	3.67	5.35	22.10	**9.35**	11.36	9.88	11.14	10.23	14.80	61.59

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

All machine learning models deliver by far a better performance compared to the proportional benchmark.

8 | Conclusion

Our study introduced a new machine learning-based approach for approximating Shapley values. Extensive numerical experiments showed that this approach leads to outstanding approximation results for Shapley values in the TSP and CVRP.

We demonstrated that the approximation through the exploitation of problem-specific knowledge leads to better approximations than simple proxies and current state-of-the-art approximation methods.

In addition, we showed through further experiments that this approach allows us to learn from fast but poorly generated labels and still achieve excellent approximation results. Furthermore, we could show the generalizability of our approximation approach by using it in the context of a bin packing problem to approximate Shapley values for the items. Overall, we present an approach that: (1) effectively utilizes the routing-specific problem structure, (2) enables real-time approximations, (3) can be trained on biased labels with only a moderate loss in performance, (4) identifies key factors influencing Shapley values, and (5) demonstrates versatility by being applicable to entirely different contexts, such as bin packing. Although, questions about the scalability of Shapley value approximations in general remain an exciting field for future research as we still observed a trade of between instance size and approximation quality.

The results in the context of routing are particularly relevant for logistics service providers that need to allocate costs to individual customers. This approach enables them to generate cost allocations close to the real Shapley values immediately. Furthermore, if alternative game-theoretic cost allocation methods are preferred, which also require evaluating all possible subcoalitions, the same methodology can be applied, with the only distinction being the use of the selected cost allocation method as the label. Another promising direction would be to leverage machine learning techniques to directly predict cost allocations that satisfy specific game-theoretic properties, particularly in real-world applications where certain fairness criteria are essential or desired.

In general, we have demonstrated that machine learning models are an efficient tool for leveraging problem-specific knowledge in routing problems. We encourage further research in this direction by exploring similar approaches and assessing their applicability to other vehicle routing problem variants, such as those with time windows or additional constraints, as well as to other combinatorial optimization problems, such as bin packing problems with differently characterized bins (e.g., varying sizes or opening costs). Beyond Shapley value approximation, future research could also focus on broader applications of machine learning in routing, such as predicting feasibility in highly constrained routing problems, estimating marginal costs, forecasting partial routes, or even directly approximating total costs. We hope that our study serves as a foundation for further advancements in integrating machine learning with combinatorial optimization.

Acknowledgments

The paper was written as part of a project funded by the German Research Foundation. We would like to express our gratitude to the German Research Foundation (project number: ROCOCO FO1468/1-1) for the financial support of our research project. We would also like to thank Dennis Götz for his support with feature generation and implementation as part of his final thesis. In addition, we would like to thank the anonymous reviewers and the editor for their valuable recommendations, which have significantly improved our paper. Open Access funding enabled and organized by Projekt DEAL.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. L. Costa, C. Contardo, and G. Desaulniers, “Exact Branch-Price-And-Cut Algorithms for Vehicle Routing,” *Transportation Science* 53, no. 4 (2019): 946–985.

2. R. Elshaer and H. Awad, “A Taxonomic Review of Metaheuristic Algorithms for Solving the Vehicle Routing Problem and Its Variants,” *Computers & Industrial Engineering* 140 (2020): 106242.

3. F. Kellner and M. Schneiderbauer, “Further Insights Into the Allocation of Greenhouse Gas Emissions to Shipments in Road Freight Transportation: The Pollution Routing Game,” *European Journal of Operational Research* 278, no. 1 (2019): 296–313.

4. M. Göthe-Lundgren, K. Jörnsten, and P. Värbrand, “On the Nucleolus of the Basic Vehicle Routing Game,” *Mathematical Programming* 72 (1996): 83–100.

5. S. Engevall, M. Göthe-Lundgren, and P. Värbrand, “The Heterogeneous Vehicle-Routing Game,” *Transportation Science* 38, no. 1 (2004): 71–85.

6. L. Sun, A. Rangarajan, M. H. Karwan, and J. M. Pinto, “Transportation Cost Allocation on a Fixed Route,” *Computers & Industrial Engineering* 83 (2015): 61–73.

7. O. Ö. Özener, Ö. Ergun, and M. Savelsbergh, “Allocating Cost of Service to Customers in Inventory Routing,” *Operations Research* 61, no. 1 (2013): 112–125.

8. S. Engevall, M. Göthe-Lundgren, and P. Värbrand, “The Traveling Salesman Game: An Application of Cost Allocation in a Gas and Oil Company,” *Annals of Operations Research* 82 (1998): 203–218.

9. S. Touati, M. S. Radjef, and S. Lakhdar, “A Bayesian Monte Carlo Method for Computing the Shapley Value: Application to Weighted Voting and Bin Packing Games,” *Computers & Operations Research* 125 (2021): 105094.

10. D. C. Popescu and P. Kilby, “Approximation of the Shapley Value for the Euclidean Traveling Salesman Game,” *Annals of Operations Research* 289, no. 2 (2020): 341–362.

11. D. B. Gillies, “Solutions to General Non-Zero-Sum Games,” *Contributions to the Theory of Games* 4, no. 40 (1959): 47–85, Princeton University Press.

12. M. Frisk, M. Göthe-Lundgren, K. Jörnsten, and M. Rönnqvist, “Cost Allocation in Collaborative Forest Transportation,” *European Journal of Operational Research* 205, no. 2 (2010): 448–458.

13. D. Schmeidler, “The Nucleolus of a Characteristic Function Game,” *SIAM Journal on Applied Mathematics* 17, no. 6 (1969): 1163–1170.

14. M. Guajardo and M. Rönnqvist, “A Review on Cost Allocation Methods in Collaborative Transportation,” *International Transactions in Operational Research* 23, no. 3 (2016): 371–392.

15. L. S. Shapley, “A Value for *n*-Person Games,” in *Contributions to the Theory of Games II*, ed. H. W. Kuhn and A. W. Tucker (Princeton University Press, 1953), 307–317.

16. R. Mitchell, J. Cooper, E. Frank, and G. Holmes, “Sampling Permutations for Shapley Value Estimation,” *Journal of Machine Learning Research* 23, no. 1 (2022): 2082–2127.

17. H. P. Young, “Monotonic Solutions of Cooperative Games,” *International Journal of Game Theory* 14, no. 2 (1985): 65–72.

18. R. B. Myerson, “Conference Structures and Fair Allocation Rules,” *International Journal of Game Theory* 9 (1980): 169–182.

19. L. S. Shapley, “Cores of Convex Games,” *International Journal of Game Theory* 1 (1971): 11–26.

20. M. A. Krajewska, H. Kopfer, G. Laporte, S. Ropke, and G. Zaccour, “Horizontal Cooperation Among Freight Carriers: Request Allocation and Profit Sharing,” *Journal of the Operational Research Society* 59 (2008): 1483–1491.

21. A. Kimms and I. Kozeletskyi, “Shapley Value-Based Cost Allocation in the Cooperative Traveling Salesman Problem Under Rolling Horizon Planning,” *EURO Journal on Transportation and Logistics* 5, no. 4 (2016): 371–392.

22. J. Gückel, T. G. Crainic, and P. Fontaine, *Resource Planning and Cost Allocation for Tactical Planning in Cooperative Two-Tier City Logistics Systems* (Publication Centre interuniversitaire de recherche sur les réseaux d’entreprise, la logistique et le transport, 2024).

23. M. Gómez-Rodríguez, L. Davila-Pena, and B. Casas-Méndez, “Cost Allocation Problems on Highways With Grouped Users,” *European Journal of Operational Research* 316, no. 2 (2024): 667–679.

24. J. Kuipers, M. A. Mosquera, and J. M. Zarzuelo, “Sharing Costs in Highways: A Game Theoretic Approach,” *European Journal of Operational Research* 228, no. 1 (2013): 158–168.

25. D. Bendel and M. Haviv, “Cooperation and Sharing Costs in a Tandem Queueing Network,” *European Journal of Operational Research* 271, no. 3 (2018): 926–933.

26. J. Timmer, M. Chessa, and R. J. Boucherie, “Cooperation and Game-Theoretic Cost Allocation in Stochastic Inventory Models With Continuous Review,” *European Journal of Operational Research* 231, no. 3 (2013): 567–576.

27. B. Altan and O. Ö. Özener, “Cost Allocation Mechanisms in a Peer-To-Peer Network,” *Networks* 73, no. 1 (2019): 104–118.

28. J. M. Zolezzi and H. Rudnick, “Transmission Cost Allocation by Cooperative Games and Coalition Formation,” *IEEE Transactions on Power Systems* 17, no. 4 (2002): 1008–1015.

29. D. Choudhury, S. Borkotokey, R. Kumar, and S. Sarangi, “The Egalitarian Shapley Value: A Generalization Based on Coalition Sizes,” *Annals of Operations Research* 301, no. 1 (2021): 55–63.

30. E. Borgonovo, E. Plischke, and G. Rabitti, “The Many Shapley Values for Explainable Artificial Intelligence: A Sensitivity Analysis Perspective,” *European Journal of Operational Research* 318, no. 3 (2024): 911–926.

31. S. Cohen, G. Dror, and E. Ruppin, “Feature Selection via Coalitional Game Theory,” *Neural Computation* 19, no. 7 (2007): 1939–1961.

32. S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems*, vol. 30, ed. I. Guyon, U. V. Luxburg, S. Bengio, et al. (Curran Associates, Inc., 2017).

33. R. Metulini and G. Gnecco, “Measuring Players’ Importance in Basketball Using the Generalized Shapley Value,” *Annals of Operations Research* 325, no. 1 (2023): 441–465.

34. V. Roshanaei, C. Luong, D. M. Aleman, and D. R. Urbach, “Collaborative Operating Room Planning and Scheduling,” *INFORMS Journal on Computing* 29, no. 3 (2017): 558–580.

35. G. Gnecco, Y. Hadas, and M. Sanguineti, “Some Properties of Transportation Network Cooperative Games,” *Networks* 74, no. 2 (2019): 161–173.

36. F. Ciardiello, A. Genovese, and A. Simpson, “A Unified Cooperative Model for Environmental Costs in Supply Chains: The Shapley Value for the Linear Case,” *Annals of Operations Research* 290 (2020): 421–437.

37. L. S. Shapley and I. Mann, *Values of Large Games, IV: Evaluating the Electoral College by Montecarlo Techniques* (RAND Corporation, 1960).

38. K. Schopka and H. Kopfer, “Cost Allocation for Horizontal Carrier Coalitions Based on Approximated Shapley Values,” in *Operations Research Proceedings 2015: Selected Papers of the International Conference of the German, Austrian and Swiss Operations Research Societies (GOR, ÖGOR, SVOR/ASRO), University of Vienna, Austria, September 1–4, 2015* (Springer, 2017), 133–140.

39. J. Castro, D. Gómez, and J. Tejada, “Polynomial Calculation of the Shapley Value Based on Sampling,” *Computers & Operations Research* 36, no. 5 (2009): 1726–1730.

40. H. Aziz, C. Cahan, C. Gretton, P. Kilby, N. Mattei, and T. Walsh, “A Study of Proxies for Shapley Allocations of Transport Costs,” *Journal of Artificial Intelligence Research* 56 (2016): 573–611.

41. C. Levinger, N. Hazon, and A. Azaria, “Efficient Computation and Estimation of the Shapley Value for Traveling Salesman Games,” *IEEE Access* 9 (2021): 129119–129129.

42. X. Li, R. Bai, P.-O. Siebers, and C. Wagner, “Travel Time Prediction in Transport and Logistics: Towards More Efficient Vehicle GPS Data Management Using Tree Ensemble Methods,” *Vine Journal of Information and Knowledge Management Systems* 49, no. 3 (2019): 277–306.

43. M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, “Travel Speed Prediction Based on Learning Methods for Home Delivery,” *EURO Journal on Transportation and Logistics* 9, no. 4 (2020): 100006.

44. R. Basso, B. Kulcsár, and I. Sanchez-Diaz, “Electric Vehicle Routing Problem With Machine Learning for Energy Prediction,” *Transportation Research Part B: Methodological* 145 (2021): 24–55.

45. G. Ghiani, A. Manni, and E. Manni, “A Scalable Anticipatory Policy for the Dynamic Pickup and Delivery Problem,” *Computers & Operations Research* 147 (2022): 105943.

46. L. Van der Hagen, N. Agatz, R. Spliet, T. R. Visser, and L. Kok, “Machine Learning–Based Feasibility Checks for Dynamic Time Slot Management,” *Transportation Science* 58, no. 1 (2024): 94–109.

47. A. Sobhanan, J. Park, J. Park, and C. Kwon, “Genetic Algorithms With Neural Cost Predictor for Solving Hierarchical Vehicle Routing Problems,” *Transportation Science* 59, no. 2 (2024): 322–339.

48. Y. Sun, A. Ernst, X. Li, and J. Weiner, “Generalization of Machine Learning for Problem Reduction: A Case Study on Travelling Salesman Problems,” *OR Spectrum* 43, no. 3 (2021): 607–633.

49. D. Bertsimas and C. W. Kim, “A Prescriptive Machine Learning Approach to Mixed-Integer Convex Optimization,” *INFORMS Journal on Computing* 35, no. 6 (2023): 1225–1241.

50. E. Larsen, S. Lachapelle, Y. Bengio, E. Frejinger, S. Lacoste-Julien, and A. Lodi, “Predicting Tactical Solutions to Operational Planning Problems Under Imperfect Information,” *INFORMS Journal on Computing* 34, no. 1 (2022): 227–242.

51. M. E. Bruni, E. Fadda, S. Fedorov, and G. Perboli, “A Machine Learning Optimization Approach for Last-Mile Delivery and Third-Party Logistics,” *Computers & Operations Research* 157 (2023): 106262.

52. U. Ermağan, B. Yildiz, and F. S. Salman, “A Learning Based Algorithm for Drone Routing,” *Computers & Operations Research* 137 (2022): 105524.

53. X. Zhang, L. Chen, M. Gendreau, and A. Langevin, “Learning-Based Branch-And-Price Algorithms for the Vehicle Routing Problem With Time Windows and Two-Dimensional Loading Constraints,” *INFORMS Journal on Computing* 34, no. 3 (2022): 1419–1436.

54. R. Asín-Achá, A. Espinoza, O. Goldschmidt, D. S. Hochbaum, and I. I. Huerta, “Selecting Fast Algorithms for the Capacitated Vehicle Routing Problem With Machine Learning Techniques,” *Networks* 84, no. 2 (2024): 200–219.

55. M. Morabit, G. Desaulniers, and A. Lodi, “Learning to Repeatedly Solve Routing Problems,” *Networks* 83, no. 3 (2024): 503–526.

56. S. Chamurally and J. Rieck, “A Practical and Robust Approach for Solving the Multi-Compartment Vehicle Routing Problem Under Demand Uncertainty Using Machine Learning,” *Networks* 84, no. 3 (2024): 300–325.

57. R. Bai, X. Chen, Z.-L. Chen, et al., “Analytics and Machine Learning in Vehicle Routing Research,” *International Journal of Production Research* 61, no. 1 (2023): 4–30.

58. Y. Bengio, A. Lodi, and A. Prouvost, “Machine Learning for Combinatorial Optimization: A Methodological Tour D’horizon,” *European Journal of Operational Research* 290, no. 2 (2021): 405–421.

59. T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2 (Springer, 2009).

60. Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature* 521, no. 7553 (2015): 436–444.

61. L. Breiman, “Random Forests,” *Machine Learning* 45 (2001): 5–32.

62. S. Weisberg, *Applied Linear Regression*, vol. 528 (John Wiley & Sons, 2005).

63. H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support Vector Regression Machines,” *Advances in Neural Information Processing Systems* 9 (1996): 155–161.

64. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 96 (AAAI Press, 1996), 226–231.

65. T. O’Malley, E. Bursztein, J. Long, et al., “Kerastuner,” (2019), https://github.com/keras-team/keras-tuner.

66. D. Kinga and J. B. Adam, “A Method for Stochastic Optimization,” in *International Conference on Learning Representations (ICLR)*, vol. 5 (San Diego, California, 2015), 6.

67. T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, 2016), 785–794.

68. F. Pedregosa, G. Varoquaux, A. Gramfort, et al., “Scikit-Learn: Machine Learning in Python,” *Journal of Machine Learning Research* 12 (2011): 2825–2830.

69. S. H. Tijs and T. S. Driessen, “Game Theory and Cost Allocation Problems,” *Management Science* 32, no. 8 (1986): 1015–1028.

Appendix A

Mathematical Model Formulations

TSP Formulation

Given a complete graph $G = (\mathcal{N}, \mathcal{A})$ with the set of nodes $\mathcal{N}$ and the set of arcs $\mathcal{A}$, where each arc $(i, j) \in \mathcal{A}$ has an associated non-negative cost c_{ij}, and where x_{ij} is a binary decision variable that takes the value 1 if arc (i, j) is used in the tour and 0 otherwise, the TSP can be formulated as follows:

$$\begin{aligned}
\min \sum_{i,j\in\mathcal{A}} c_{ij} \cdot x_{ij} & \\
\sum_{i\in\mathcal{N}:i\neq j} x_{ij} = 1 & \qquad \forall j \in \mathcal{N} \\
\sum_{j\in\mathcal{N}:j\neq i} x_{ij} = 1 & \qquad \forall i \in \mathcal{N} \\
\sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}:j\neq i} x_{ij} \leq |\mathcal{S}| - 1 & \qquad \forall \mathcal{S} \subset \mathcal{N}, |\mathcal{S}| \geq 2 \\
x_{ij} \in \{0, 1\} & \qquad \forall i, j \in \mathcal{A}
\end{aligned}$$

CVRP Formulation

Given a complete graph $G = (\mathcal{N}, \mathcal{A})$ with the set of nodes $\mathcal{N}$ and the set of arcs $\mathcal{A}$, where each arc $(i, j) \in \mathcal{A}$ has an associated non-negative cost c_{ij}, and each node a certain demand volume q_i, and where x_{ij} is a binary decision variable that takes the value 1 if arc (i, j) is used in the tour and 0 otherwise, and u_i represents a decision variable for the cumulative demand up to node i, and Q is the vehicle capacity, the CVRP can be formulated as follows:

$$
\begin{aligned}
& \min \sum_{i,j \in \mathcal{A}} c_{ij} \cdot x_{ij} \\
& \sum_{j \in N : j \neq i} x_{ij} = 1 && \forall i \in \mathcal{N} \\
& \sum_{i \in N : i \neq j} x_{ij} = 1 && \forall j \in \mathcal{N} \\
& x_{ij} = 1 \Rightarrow u_i + q_j = u_j && \forall i,j \in \mathcal{A} : j \neq 0, i \neq 0 \\
& q_i \leq u_i \leq Q && \forall i \in \mathcal{N} \\
& x_{ij} \in \{0,1\} && \forall i,j \in \mathcal{A}
\end{aligned}
$$

Bin Packing With Size and Weight Formulation

Given a set of bins $\mathcal{B}$ and a set of items $\mathcal{I}$. Let w_i and s_i be the weight and size of item $i \in \mathcal{I}$ and W and S be the uniform weight and size capacity for each bin $b \in \mathcal{B}$. With the binary decision variables y_b indicating whether a bin $b \in \mathcal{B}$ is selected and x_{ib} indicating whether item $i \in \mathcal{I}$ is packed into bin $b \in \mathcal{B}$, where M is a sufficiently large constant, the problem can be modeled as follows:

$$
\begin{aligned}
& \min \sum_{b \in \mathcal{B}} y_b \\
& \sum_{b \in \mathcal{B}} x_{ib} = 1 && \forall i \in \mathcal{I} \\
& \sum_{i \in \mathcal{I}} x_{ib} \leq M \cdot y_b && \forall b \in \mathcal{B} \\
& \sum_{i \in \mathcal{I}} x_{ib} \cdot w_i \leq W && \forall b \in \mathcal{B} \\
& \sum_{i \in \mathcal{I}} x_{ib} \cdot s_i \leq S && \forall b \in \mathcal{B} \\
& y_b \in \{0,1\} && \forall b \in \mathcal{B} \\
& x_{ib} \in \{0,1\} && \forall i \in \mathcal{I}, b \in \mathcal{B}
\end{aligned}
$$

Appendix B

Hyperparameters

See Tables B1–B6.

TABLE B1 | Configuration of neural network.

Hyperparameter	TSP	CVRP	Bin packing
Number of hidden layers	4	3	4
Units per Layer	[208, 504, 32, 40]	[408, 184, 160]	[208, 136, 296, 8]
Activation function	[tanh, relu, relu, relu]	[relu, relu, relu]	[relu, tanh, relu, relu]
Learning rate	0.0017	0.0032	0.0016
Epochs[a]	200	200	200
Optimizer	Adam	Adam	Adam

[a]Tuned individually for the different MLSVA modifications.

TABLE B2 | Configuration of random forest.

Hyperparamter	TSP	CVRP	Bin packing
nEstimators	350	350	400
maxFeatures	None	None	None
maxDepth	25	28	25
minSamplesLeaf	2	2	2
bootstrap	True	True	True

TABLE B3 | Configuration of xgboost.

Hyperparameter	TSP	CVRP	Bin packing
nEstimators	250	300	273
maxDepth	11	10	16
minChildWeight	10	12	10
subsample	0.58	1.0	1.0
learningRate	0.06	0.07	0.05
lambda	1.7	5.0	4.99
alpha	0.0	0.0	0.25
booster	dart	gbtree	gbtree
colsampleBytree	1	1	0.77

TABLE B4 | Configuration of polynomial regression.

Hyperparameter	TSP	CVRP	Bin packing
Polynomial Features Degree	3	3	3
Ridge Alpha	1	0.1	0.1

TABLE B5 | Configuration of support vector regression.

Hyperparameter	TSP	CVRP	Bin packing
C	100	10	0.1
Epsilon	0.1	0.01	0.01
maxiter	200 000	200 000	200 000

TABLE B6 | Configuration of KNN.

Hyperparamter	TSP	CVRP	Bin packing
K	10	14	20

Appendix C

Robustness of Approximations

In this section, we report the 95th Percentile Absolute Percentage Error (PAPE95) for different approximation methods. This metric provides insight into the robustness of each method by indicating the absolute percentage error that is not exceeded in 95% of the cases.

The PAPE95 is formally defined with Expression (C1):

$$\text{PAPE95} = \text{Percentile}_{95}\left(\left|\frac{\hat{\phi}_n - \phi_n^{SV}}{\phi_n^{SV}}\right| \times 100,\ \text{for } n \in \mathcal{N}\right) \tag{C1}$$

Tables C1–C3 show the results for the TSP, CVRP and Bin Packing.

TABLE C1 | PAPE95 [%] for MLSVA differentiated by ML Models and Benchmarks (TSP).

No. of nodes	MLSVA						Benchmarks				
	NN	RF	XGB	PR	SVR	KNN	SHAPO	Depot	Reroute	ACAM	MCS
7	**4.56**	18.16	11.18	8.48	11.58	76.02	5.59	63.94	114.48	37.52	17.94
8	**4.67**	19.27	11.32	8.10	9.62	69.45	7.96	66.78	120.39	42.18	19.44
9	**5.34**	20.90	11.16	8.52	11.31	63.86	10.03	66.99	132.49	46.50	20.25
10	**5.78**	20.69	11.45	8.88	10.57	55.48	11.82	67.32	143.72	48.52	21.39
11	**6.57**	20.99	11.88	10.03	11.10	52.68	13.72	67.83	148.94	51.19	21.92
12	**7.63**	21.04	12.59	11.45	12.24	48.39	15.36	67.07	153.54	53.78	22.16
13	**8.27**	21.47	12.71	12.92	13.00	49.77	17.27	67.56	161.40	54.18	22.92
14	**9.14**	22.24	13.56	13.17	14.04	45.82	18.95	68.86	155.97	53.35	22.72
15	**9.90**	23.46	14.72	14.24	14.77	44.90	21.01	68.02	163.06	55.36	23.97
avg.	**6.87**	20.92	12.29	10.64	12.03	56.26	13.52	67.15	143.78	49.18	21.41

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

TABLE C2 | PAPE95 [%] for MLSVA differentiated by ML Models and Benchmarks (CVRP).

No. of nodes	MLSVA						Benchmarks			
	NN	RF	XGB	PR	SVR	KNN	Depot	Reroute	ACAM	MCS
7	**9.48**	18.45	13.03	13.84	16.16	76.87	59.08	102.94	82.10	14.40
8	**9.42**	17.88	13.54	12.39	13.34	67.58	57.32	99.99	82.02	14.78
9	**10.01**	18.48	13.42	12.50	13.03	59.48	61.35	99.93	72.61	14.35
10	**9.58**	18.11	13.70	11.63	12.79	53.87	61.36	99.84	71.70	14.36
11	**10.39**	19.34	13.88	12.19	12.89	57.16	63.56	99.85	74.61	14.49
12	**10.85**	19.01	13.88	12.03	13.01	52.31	66.80	99.79	74.50	15.12
13	**11.06**	18.91	14.06	12.72	13.91	54.18	68.20	99.87	76.72	14.78
avg.	**10.11**	18.60	13.64	12.47	13.59	60.21	62.52	100.32	76.32	14.61

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

TABLE C3 | PAPE95 [%] for MLSVA differentiated by ML Models (Bin Packing).

No. of nodes	NN	RF	XGB	PR	SVR	KNN	WS
7	**9.64**	11.93	11.19	13.37	12.62	20.32	61.28
8	**11.21**	13.34	11.88	13.31	12.74	17.90	58.49
9	**11.54**	13.48	11.98	12.94	12.53	17.68	57.76
10	**10.97**	13.11	11.30	12.61	12.09	15.94	54.55
11	11.26	12.14	**10.53**	12.08	11.41	15.14	56.91
12	**10.06**	11.93	10.32	11.42	10.86	14.10	59.96
13	11.21	12.74	**11.13**	12.60	12.05	14.87	59.83
14	10.18	11.22	**9.86**	11.45	10.20	12.73	58.69
15	10.37	11.87	**9.96**	11.43	10.45	13.40	60.69
avg.	**10.72**	12.42	10.91	12.36	11.66	15.79	58.68

Note: Signicance of bold values indicate the lowest value for each evaluation metric.

Appendix D

Detailed Illustration

TSP Illustration

In the following, we provide a detailed breakdown of a 7-node TSP instance, consisting of one depot and six customers. The optimally solved instance is visualized in Figure D1.

The corresponding distance matrix is presented in Table D1, while a subset of the characteristic function, showing optimal cost values for selected coalitions, is displayed in Table D2.

Finally, Table D3 presents the true Shapley values alongside their approximations obtained from different methods.

CVRP Illustration

In the following, we provide a detailed breakdown of a 7-node CVRP instance, consisting of one depot and six customers. The vehicle capacity in this instance is set to 13. The optimally solved instance is visualized in Figure D2.

The corresponding distance matrix and the demands *q* of the customers are presented in Table D4, while a subset of the characteristic function, showing optimal cost values for selected coalitions, is displayed in Table D5.

Finally, Table D6 presents the true Shapley values alongside their approximations obtained from different methods.

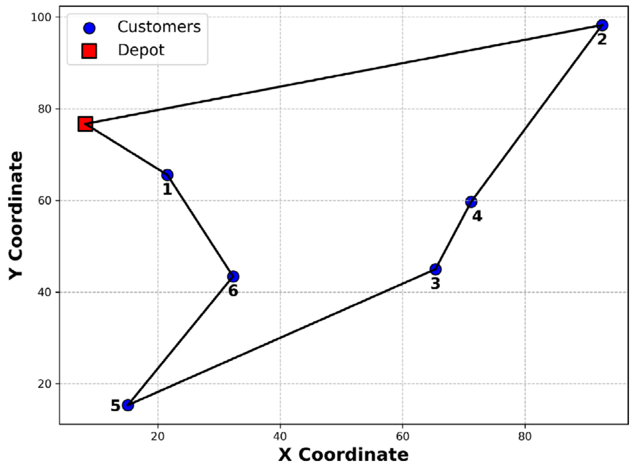

FIGURE D1 | Optimally solved 7-node TSP instance (one depot and six customers).

TABLE D1 | Distance matrix for the 7-node TSP instance.

ID	0	1	2	3	4	5	6
0	0.00	17.39	87.14	65.40	65.29	61.76	41.10
1	17.39	0.00	78.21	48.42	50.01	50.68	24.63
2	87.14	78.21	0.00	59.82	44.10	113.51	81.52
3	65.40	48.42	59.82	0.00	15.82	58.37	33.11
4	65.29	50.01	44.10	15.82	0.00	71.54	42.18
5	61.76	50.68	113.51	58.37	71.54	0.00	32.94
6	41.10	24.63	81.52	33.11	42.18	32.94	0.00

TABLE D2 | Characteristic function: Selected coalition costs.

Coalition (subset)	Total cost
{1}	34.77
{2}	174.29
{3}	130.79
{1, 2}	182.74
{1, 3}	131.20
{2, 3}	212.36
{1, 2, 3}	212.78
{1, 2, 3, 4}	212.87
{1, 2, 3, 4, 5}	273.50
{1, 2, 3, 4, 5, 6}	280.40

TABLE D3 | True Shapley values and approximations by different methods for TSP.

Method	1	2	3	4	5	6
Shapley value	8.72	99.41	39.25	38.48	71.36	23.18
MLSVA (NN)	8.77	99.94	39.68	38.62	70.59	22.81
SHAPO	7.73	98.51	38.31	38.59	72.98	24.28
MCS	8.99	99.09	51.54	27.74	69.26	23.78
Depot	14.42	72.28	54.24	54.15	51.23	34.09
Reroute	10.60	133.17	11.47	0.19	111.74	13.25
ACAM	12.93	95.96	37.61	33.16	74.76	25.98

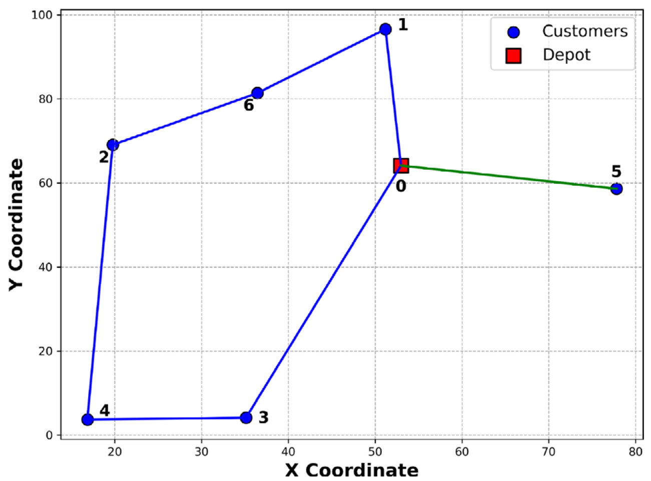

FIGURE D2 | Optimally solved 7-node CVRP instance (one depot and six customers).

TABLE D4 | Distance matrix for the 7-node CVRP instance with demands.

ID	Demand	0	1	2	3	4	5	6
0	0	0.00	32.52	33.59	62.58	70.42	25.43	23.92
1	4	32.52	0.00	41.80	93.84	99.06	46.36	21.20
2	1	33.59	41.80	0.00	66.69	65.43	58.98	20.74
3	1	62.58	93.84	66.69	0.00	18.29	69.22	77.25
4	3	70.42	99.06	65.43	18.29	0.00	82.08	80.14
5	3	25.43	46.36	58.98	69.22	82.08	0.00	47.22
6	3	23.92	21.20	20.74	77.25	80.14	47.22	0.00

TABLE D5 | Characteristic function: Selected coalition costs for CVRP.

Coalition (subset)	Total cost
$\{1\}$	65.04
$\{2\}$	67.17
$\{3\}$	125.16
$\{1, 2\}$	107.91
$\{1, 3\}$	188.94
$\{2, 3\}$	162.86
$\{1, 2, 3\}$	203.59
$\{1, 2, 3, 4\}$	220.62
$\{1, 2, 3, 4, 5\}$	252.69
$\{1, 2, 3, 4, 5, 6\}$	271.61

TABLE D6 | True Shapley values and approximations by different methods for CVRP.

Method	1	2	3	4	5	6
Shapley value	45.68	28.27	59.39	74.94	41.69	21.64
MLSVA (NN)	46.81	28.14	60.02	73.86	41.02	21.76
MCS	44.50	33.00	60.28	71.94	41.15	20.75
Depot	35.55	36.72	68.41	76.99	27.80	26.15
Reroute	75.30	16.46	16.19	55.50	78.82	29.34
ACAM	53.52	27.56	44.81	67.28	50.86	27.59

Appendix E

Visualisation of Modified Instances

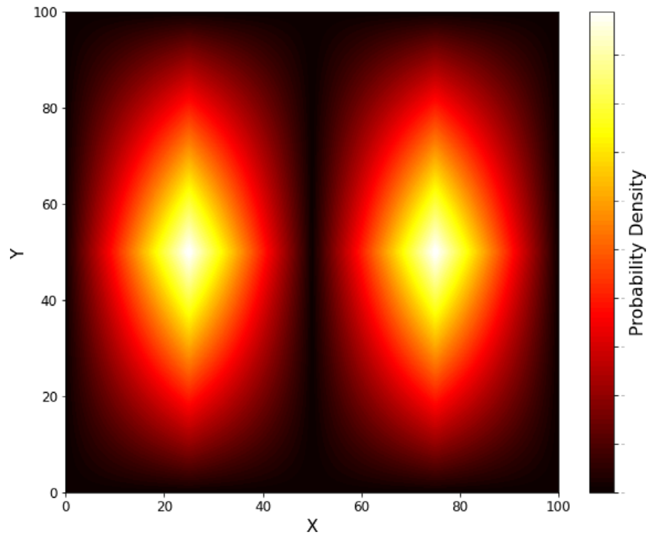

FIGURE E1 | Heatmap of probability distribution of modified instances.