Fast Shapley Value approximation in routing problems through machine learning models

Johannes Gückel, Pirmin Fontaine
Catholic University Eichstätt-Ingolstadt, Ingolstadt School of Management & Mathematical Institute for Machine Learning and Data Science, Ingolstadt, Germany
e-mail: pirmin.fontaine@ku.de

Extended abstract submitted for presentation at the 12th Triennial Symposium on Transportation Analysis conference (TRISTAN XII)
June 22-27, 2025, Okinawa, Japan

March 4, 2025

Keywords: costs allocation, machine learning, vehicle routing problem, traveling salesman

1 INTRODUCTION

Routing problems are widely studied in the literature and many methods exist to minimize the total travel costs. But besides cost minimization, the question of how to allocate these costs to the different involved customers arises. Recently, this became even more relevant in the context of CO_2-emissions since companies want and sometimes even need to report emissions in their sustainability reports.

In this context, one of the most used methods is the Shapley Value since it fulfills the major fairness criteria. However, the major disadvantage of the Shapley Value is its computational complexity since it requires the calculation of the costs of all sub coalitions of customers. Thus, it can only be calculated for very small problem sizes in reasonable time, especially if the considered optimization problem is NP-hard, as it is for the vehicle routing problem.

Therefore, the literature suggests also several approximation methods, both general computational expensive approximation schemes (Touati *et al.* , 2021) and problem specific methods (Levinger *et al.* , 2021). However, the quality of these approaches differs and the solution times might still be too slow to be used in practice if real-time or fast calculations are necessary.

To overcome these drawbacks, we introduce a general machine learning-based approximation algorithm that can be trained offline to efficiently calculate Shapley Values afterwards. We test its performance against state-of-the-art approximation methods for the traveling salesman problem (TSP) and the capacitated vehicle routing problem (CVRP). Further, the generalizability of our methodology is demonstrated on a bin packing problem. Finally, we show how our approach can scale efficiently to even large instances by using biased (easier computable) labels.

2 PROBLEM SETTING

We consider a TSP with a set of customers $\mathcal{N}$ and an optimal tour with total costs $C(\mathcal{N})$. Then, the goal is to find the cost allocation ϕ_n for each customer $n \in \mathcal{N}$. Following the seminal work of Shapley (1953), such a cost allocation is defined as

$$\phi_n^{SV} = \sum_{\mathcal{S} \subset \mathcal{N} : n \in \mathcal{S}} \frac{(|\mathcal{S}| - 1)!(|\mathcal{N}| - |\mathcal{S}|)!}{|\mathcal{N}|!} [C(\mathcal{S}) - C(\mathcal{S} - \{n\})] \tag{1}$$

where $C(\mathcal{S})$ defines the costs of each subcoalition $\mathcal{S} \subseteq \mathcal{N}$. Due to the high number of optimal costs for each subcoalition that need to be determined, the calculation of the Shapley Values becomes fast intractable if the number of customers increases. Note that we can define the Shapley Value in a similar way for the capacitated vehicle routing problem. But in this abstract, we focus on the TSP and only show some results for the CVRP.

3 METHODOLOGY

We introduce a Machine Learning-based Shapley Value Approximator (MLSVA) that uses a supervised machine learning approach to predict Shapley Values for unseen instances and customers after learning from true Shapley Values and influencial features.

We consider a set of instances (e.g., TSP or CVRP) $\mathcal{T}^{all}$ and the set of customers $\mathcal{N}(t)$ in instance t with total routing cost $C(\mathcal{N}(t))$. For each customer n, $\hat{\phi}_n^{ML}$ defines the cost allocation prediction of our machine learning model and $\hat{\phi}_n$ the final approximation of the Shapley Value. Using, this notation, we can define our general methodology in Algorithm 1.

First, we generate a set of instances $\mathcal{T}^{all}$. Then, we determine for each instance t the characteristic function $v(\mathcal{N}(t))$ in order to label each customer observation n with the real Shapley Value ϕ_n^{SV} and prepare the features $f(n)$ that influence the Shapley Value (lines 2 to 8). Here, we use instance- and customer-specific features that characterize the routing specific structure. Such features include distance to the depot, distance to the gravity center, x-y-coordinates, number of clusters, size of clusters, marginal cost of a customer, for example. For the CVRP, further capacity related features are considered.

Then, we split the set of instances into the set of training instances $\mathcal{T}^{train}$ and the set of test instances $\mathcal{T}^{test}$ (line 9). In the training, we use the Shapley Values as labels (line 10) to train the selected supervised machine learning model on the training instances (line 11) based on the features and labels. Then, we use the machine learning model to generate predicted Shapley Values $\hat{\phi}_n^{ML}$ for each observation (lines 12 to 16) and scale them for each instance so that the sum of the approximated Shapley Values corresponds to the total routing costs for that instance. These scaled Shapley Values result in the final approximations $\hat{\phi}_n$ (lines 17 to 21). This scaling is necessary, since the sum of the approximations for all customers must match the total routing costs in an instance.

Finally, we evaluate the performance of the method exclusively on the test data $\mathcal{T}^{test}$. Once the machine learning models are trained, our approximation approach can generate approximations rapidly without high computational effort.

Algorithm 1 MLSVA

```
1: Generate set of instances $\mathcal{T}^{all}$
2: for $t \in \mathcal{T}^{all}$ do
3:     $v(\mathcal{N}(t)) \leftarrow$ determineCharacteristicFunction($\mathcal{N}(t)$)
4:     for $n \in \mathcal{N}(t)$ do
5:         $\phi_n^{SV} \leftarrow$ computeShapley($n, v(\mathcal{N}(t))$)
6:         $f(n) \leftarrow$ prepareFeatures($n, t$)
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($f(n), \phi_n^{SV} \forall n \in \mathcal{N}(t), t \in \mathcal{T}^{train}$)
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($\hat{\phi}_n^{ML}, C(\mathcal{N}(t))$)
20:     end for
21: end for
```

4 RESULTS

4.1 Numerical setup

Due to the computational effort and similar to other works (e.g., Levinger *et al.* , 2021), we measure the performance on instances with up to 15 nodes. In total, we create 5000 instances for each size (7 to 15 nodes for the TSP and 7 to 13 nodes for the CVRP). All customer locations and the depot are set randomly on a grid of size 100 x 100. For the CVRP, we generate random demand between 1 and 5 and a random vehicle capacity between 10 and 18 per instance.

We assume a train/test data split of 80/20 to evaluate and benchmark our machine learning framework against other methods. The optimal costs of the TSPs and VRPs are calculated using GUROBI 9.7 and all machine learning models are conducted in Python 3.12. We use a Neural Network, k-nearest neighbor, random forest, and XGBoost as machine learning models and perform for all models a hyperparameter tuning separately for TSP and VRP. All experiments are carried out on an AMD Ryzen 9 5950X 16-Core Processor, 3.40 GHz with a 128 Gb RAM.

4.2 Evaluation metrics and benchmarks

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

- **MAPE:** The mean absolute percentage error from the true Shapley value.

$$\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}\frac{|\hat{\phi}_n-\phi_n^{SV}|}{\phi_n^{SV}} \tag{2}$$

- **MMAXPE:** The mean maximum percentage error from the true Shapley value

$$\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\max_{n\in\mathcal{N}(t)}\frac{|\hat{\phi}_n-\phi_n^{SV}|}{\phi_n^{SV}} \tag{3}$$

  where $\mathcal{T}$ is the set of routing problem instances and $\mathcal{N}(t)$ represents the customers in instance $t\in\mathcal{T}$. The latter is particularly important for evaluating the worst deviation of approximations from the real Shapley Value for each instance.

We use the current state-of-the-art method SHAPO for the TSP (Levinger *et al.* , 2021) and use two simple approximation methods as benchmark approaches. For the latter, we use the depot distance and the reroute margin, which can be calculated as follows:

$$\hat{\phi}_n^{Depot} = C(\mathcal{N})\cdot\frac{d_{0n}}{\sum\limits_{i\in\mathcal{N}} d_{0i}} \tag{4}$$

$$\hat{\phi}_n^{Reroute} = C(\mathcal{N})\cdot\frac{C(\mathcal{N})-C(\mathcal{N}\setminus\{n\})}{\sum\limits_{i\in\mathcal{N}} C(\mathcal{N})-C(\mathcal{N}\setminus\{i\})} \tag{5}$$

4.3 Performance

Table 1 shows the performance for the TSP. We report the results for the neural network (NN), the random forest (RF), the XGBoost (XGB), and the benchmarks. Note that, we do not report the results the k-nearest neighbor because of space limitations in the abstract and the significantly worse performance (which can also be seen in the CVRP results). In a similar way, Table 2 reports the results for the CVRP.

The results show that the neural network outperforms all other machine learning models. Moreover, for the TSP, our MLSVA has only an average approximation error of 2.4% compared

$\lvert\mathcal{N}\rvert$	MAPE [%]						MMAXPE [%]					
	MLSVA			Benchmarks			MLSVA			Benchmarks		
	NN	RF	XGB	SHAPO	Depot	Reroute	NN	RF	XGB	SHAPO	Depot	Reroute
7	1.61	5.87	3.60	**1.22**	26.40	60.09	4.09	14.35	9.70	**3.08**	52.83	113.93
8	**1.66**	6.05	3.65	1.79	27.65	62.29	**4.52**	16.29	10.18	4.96	56.17	121.26
9	**1.85**	6.60	3.73	2.30	28.58	64.72	**5.24**	19.36	11.34	6.73	58.45	131.19
10	**2.02**	6.59	3.89	2.84	28.60	66.30	**5.92**	20.21	12.23	8.79	59.99	144.67
11	**2.34**	6.72	4.21	3.42	29.21	67.88	**7.44**	21.26	15.19	10.84	61.47	151.87
12	**2.63**	6.84	4.34	4.00	29.30	68.59	**8.67**	22.45	14.77	13.34	62.68	161.00
13	**2.94**	7.09	4.56	4.46	29.34	69.69	**10.59**	24.76	17.19	15.51	64.01	171.32
14	**3.15**	7.38	4.97	5.02	30.10	68.51	**10.86**	26.97	19.74	17.75	65.99	168.88
15	**3.41**	7.65	5.07	5.56	29.98	69.44	**11.99**	28.96	19.42	20.12	66.17	177.30
avg.	**2.40**	6.75	4.22	3.40	28.80	66.39	**7.70**	21.62	14.41	11.12	60.97	149.49

Table 1 – *MAPE and MMAXPE [%] for MLSVA (TSP)*

$\lvert\mathcal{N}\rvert$	MAPE [%]						MMAXPE [%]					
	MLSVA				Benchmarks		MLSVA				Benchmarks	
	NN	RF	XGBoost	KNN	Depot	Reroute	NN	RF	XGBoost	KNN	Depot	Reroute
7	**2.97**	6.43	4.49	20.71	21.70	45.54	**7.89**	15.63	11.59	76.41	47.97	98.22
8	**3.22**	6.51	4.47	19.76	21.78	46.89	**8.85**	16.19	12.80	81.41	49.65	102.16
9	**3.49**	6.67	4.90	18.44	22.53	45.41	**9.53**	17.19	13.70	83.73	55.79	103.91
10	**3.51**	6.62	4.88	16.87	23.00	45.08	**9.84**	17.93	13.93	78.68	58.28	103.88
11	**3.74**	6.99	5.08	16.53	23.66	45.00	**11.28**	20.54	15.59	89.76	62.67	108.50
12	**3.83**	6.90	5.09	17.53	24.04	43.78	**11.71**	21.23	16.20	95.43	67.02	109.74
13	**4.01**	6.97	5.21	17.38	24.32	43.52	**12.88**	21.67	16.63	94.60	70.77	112.23
avg.	**3.54**	6.73	4.87	18.17	23.00	45.03	**10.28**	18.63	14.35	85.72	58.72	105.52

Table 2 – *MAPE and MMAXPE [%] for MLSVA (CVRP)*

to the state-of the art-method with 3.4%. Additionally, the maximum error is significantly lower. For the CVRP, where SHAPO cannot be applied, the difference is even larger and demonstrates the benefit MLSVA.

In further numerical test, we demonstrate that we can further use heuristic solutions for training our machine learning models without significant loss in performance. Moreover, we can train our model with small instances and still estimate the Shapley Values for large instances. These extensions additionally highlight the advantage of our method since this allows to use MLSVA for large instances without the necessity of training on larger instances, which is computationally too expensive.

5 DISCUSSION

The results have shown that MLSVA outperforms state-of-the-art methods. Additionally, our results allow to calculate Shapley Values in real-time after the model is trained. At the conference, we will present more detailed results and the generalizability of our method which further shows great performance on bin packing problem.

References

Levinger, Chaya, Hazon, Noam, & Azaria, Amos. 2021. Efficient computation and estimation of the Shapley value for traveling salesman games. *IEEE Access*, **9**, 129119–129129.

Shapley, Lloyd S. 1953. A Value for n-Person Games. *Pages 307–317 of:* Kuhn, Harold W., & Tucker, Albert W. (eds), *Contributions to the Theory of Games II*. Princeton: Princeton University Press.

Touati, Sofiane, Radjef, Mohammed Said, & Lakhdar, SAIS. 2021. A Bayesian Monte Carlo method for computing the Shapley value: Application to weighted voting and bin packing games. *Computers & Operations Research*, **125**, 105094.