Vrije Universiteit Amsterdam

Master Thesis

Fair Pricing Mechanisms for Demand Steering

Author: Eda Vedzhhi (2630523)

VU supervisor:	Prof. Dr. Rob van der Mei
CWI supervisor:	M.Sc Joris Slootweg
second reader:	Dr. Caroline Jagtenberg

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

August 13, 2024

"Our greatest glory is not in never falling, but in rising every time we fall."
-Confucius

Abstract

In last-mile deliveries, demand steering is applied by logistics firms to influence customer choice and ensure demand is placed at cost-efficient time-slots and days. For businesses to be able to apply demand steering, costs need to be allocated to customers in a fair manner. The current paper defines fair cost allocation in terms of Shapley values and constructs Travelling Salesman Problem (TSP) games. Due to the computational burden of calculating Shapley values for a TSP problem, novel approaches are introduced by utilising Minimum Spanning Trees (MST) to construct algorithms which can efficiently allocate costs to customers while ensuring accuracy. The novel approaches are evaluated against naive and state-of-the-art methods using benchmark instances in terms of accuracy and speed. The performance of the approaches are also evaluated with a case-study from real-life data, as well as an analysis of robustness. Results show that the novel approaches introduced can efficiently allocate costs while outperforming accuracy of previous methods.

Acknowledgements

Working on this thesis, and finalising my Master's has been one of my biggest challenges yet, but also one of the most rewarding experiences. This experience has been made possible by various people who have supported my master's degree journey. Therefore, I would like to express my deepest gratitude to them.

First and foremost, I would like to express my deepest gratitude to my VU supervisor, Prof. Dr. Rob van der Mei, who has not only provided this opportunity of conducting my research at CWI, but also for his constant support and guidance with his feedback. Moreover, I would like to express my deepest appreciation to my CWI supervisor, M.Sc Joris Slootweg, whose weekly input through meetings and constant guidance has enabled me to complete my thesis. Furthermore, I am grateful for my second reader, Dr. Caroline Jagtenberg, who has also taught me Combinatorial Optimisation during my Bachelor's, and ignited an interest in optimisation problems in me. Without her, I would not be finding myself conducting a research on this topic.

I am deeply appreciative of the financial support and sponsorship of the Open Technology Program of the Dutch Research Council (NWO) for the project number 18938. I would like to additionally express my gratitude to The Driving Force, who was provided their data and made the case study in this thesis possible. Their data has allowed me to observe the credibility of my algorithms and bring the research a new and valuable depth.

Words cannot express my gratitude to Annemieke van Goor-Balk, who has been my academic supervisor during my Bachelor's, as well as my Dual Master and Master thesis coordinator. Since the first year of my Bachelor's, she has guided me and given extensive advice, and without her guidance I would not have come this far.

I would like to extend my sincere thanks to all my professors and teacher's assistants during my studies, who have taught me valuable knowledge and allowed me to see different applications of Business Analytics. Their stimulating academic environment has made me passionate in the field of Data Analytics and Science.

On a personal note, I would like to thank all my colleagues in CWI, who have been supportive not only in an academic aspect, but has also provided a fun and sociable working environment. Their good humour and welcoming stances have made CWI an extremely valuable and enjoyable work environment for me. I am further grateful to my friends in Business Analytics, Computer Science and Artificial Intelligence. Not only have they accompanied me in this thesis journey, but their good support, constant encouragement and lively outlooks have made me believe in myself, and made me enjoy my entire thesis journey to a great extent.

Finally, I would like to acknowledge the emotional support of my family for their unwavering love and support. To my parents, thank you for believing in me, and for always encouraging me to pursue my dreams. I would not be here without your constant support: when I doubted my capabilities you always put your trust into me and the work I did. I am forever indebted to you.

Thank you all for being a part of this amazing journey and for making this thesis possible.

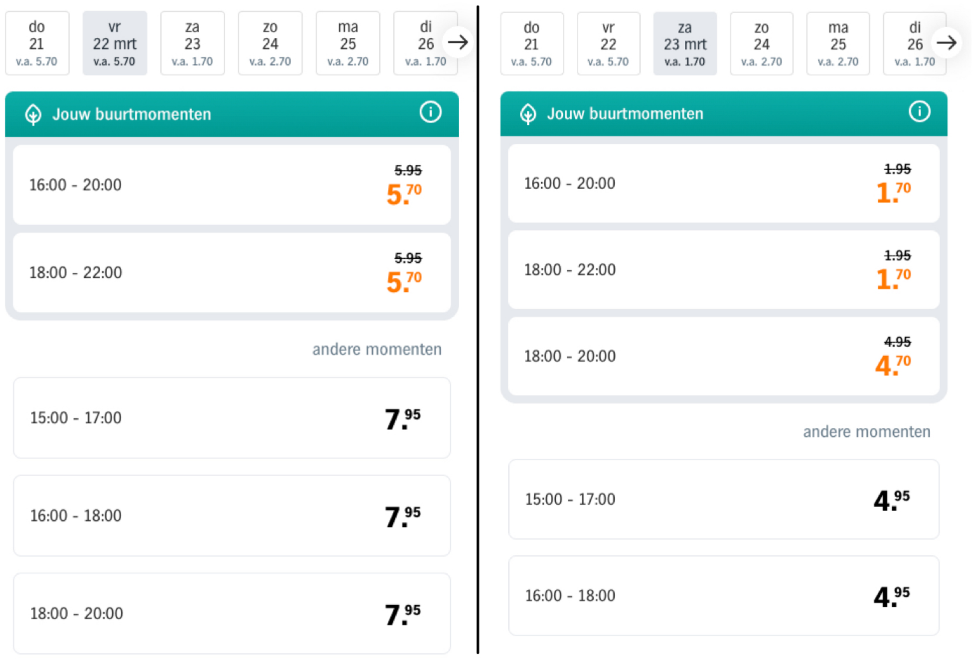
Contents

1

Introduction

Within supply chain management, last-mile deliveries play a significant role in terms of costs and efficiency of the logistics planning. This crucial phase generally determines customer satisfaction, as efficient last-mile delivery requires timely fulfilment of orders and minimising delays while concurrently ensuring sustainable and cost-efficient delivery routes (1, 2). Moreover, with the surge of e-commerce and growing demand for same-day or next-day delivery services, optimising the last-mile logistics has become imperative for businesses to remain competitive and reduce costs. In addition to the commercial benefits of last-mile logistics, efficient and effective delivery route planning can help reduce traffic congestion in urban areas and improve carbon emissions, thereby contributing not only to the business, but additionally to urban planning and sustainability (3). Therefore, last-mile deliveries have a central role in logistics, and is of crucial importance to ensure a systematic approach to delivering goods to customers.

Businesses employ various methods, such as solving travelling salesman problems (TSP) or vehicle routing problems (VRP) in order to reduce costs and plan efficient delivery routes for customers. An alternative method utilised is demand steering, which consists of strategically influencing customer behaviour to balance demand on delivery times across different time-slots. Businesses steer customers by setting incentives in a manner that can make a delivery day or time-slot more attractive in comparison to others (4). Such incentives can consist of discounts, cheaper delivery, faster delivery times or sustainable options. Behavioural nudges are also applied, which are subtle cues to influence customer behaviour, such as highlighting preferable time-slots. For example, a supermarket firm might offer a discount for selecting a delivery slot during non-peak hours to reduce traffic congestion, or to optimise routes by delivering to customers within the same area. Consider placing an order with a supermarket firm, and choosing the delivery date and time based on various options. Such an example can be seen with Figure 1.1. This Figure illustrates the cost of delivery for two dates: the left-hand side is a Friday, whereas the right-hand side is a Saturday. From this figure, methods of demand steering can be observed by discounts, green/sustainable options and behavioural nudges such as placing non-favourable time-slots as 'other moments', in a less highlighted manner.



do 21 v.a. 5.70 | vr 22 mrt v.a. 5.70 | za 23 v.a. 1.70 | zo 24 v.a. 2.70 | ma 25 v.a. 2.70 | di 26 v.a. 1.70

Jouw buurtmomenten

16:00 - 20:00 ~~5.95~~ 5.70

18:00 - 22:00 ~~5.95~~ 5.70

andere momenten

15:00 - 17:00 7.95

16:00 - 18:00 7.95

18:00 - 20:00 7.95

do 21 v.a. 5.70 | vr 22 v.a. 5.70 | za 23 mrt v.a. 1.70 | zo 24 v.a. 2.70 | ma 25 v.a. 2.70 | di 26 v.a. 1.70

Jouw buurtmomenten

16:00 - 20:00 ~~1.95~~ 1.70

18:00 - 22:00 ~~1.95~~ 1.70

18:00 - 20:00 ~~4.95~~ 4.70

andere momenten

15:00 - 17:00 4.95

16:00 - 18:00 4.95

Figure 1.1: Delivery prices of a supermarket firm per time-slot on a Friday and Saturday

Although the exact mechanisms employed to steer customers are outside the scope of this thesis, it must be noted that demand steering is an essential strategy businesses employ, as customers' choices of delivery days and time-slots subsequently impact the delivery routes and hence the associated costs. (4)

For firms to steer customers to an efficient time-slot, the cost of a customer must be known at the moment a time-slot is picked. Specifically, to be able to apply behavioural nudges to customers, businesses must be able to determine how much a customer is going to cost them if they are served within a specific time-slot. To know how much an incoming customer will contribute to the total cost of the route is imperative for businesses in order to apply demand steering methods and steer customers to a more profitable or cost-efficient time. The total cost of a route is influenced by each new arriving individual, and subsequently it is important to allocate a cost to serve a customer in a fair manner.

The cost of serving a customer can vary significantly depending on several factors, such as time of day, location and other customers who will be served during the route. Even for regular customers with identical orders, the expenses incurred by the firm can vary due to the stochastic nature of customer demands. This can also be observed with Figure 1.1, where the cheapest delivery option for Friday with 5.70€ is more expensive than the most expensive option on Saturday, 4.95€.

A customer may be more costly to serve in one route in comparison with another - this can occur due to the existing customers and their respective locations and order properties. In one route, the vehicle may need to "go out of its way" to serve a customer, thereby leading to higher costs, whereas in another, if existing customers are in close proximity to each other, the costs for the new customer may remain lower. If a new arriving customer is included in a route where previously accepted customers are considered remote for a truck, the cost allocated to the customer and hence the business will be substantially higher. A visualisation of this instance can be observed in Figure 1.2. The left diagram shows a route where the green customer may have lower cost assigned due to being on-route for the delivery truck, whereas the right diagram shows an instance where the green customer may have substantially higher costs due to being off-route and remote with respect to other customers. Although the green customer is in the same location, the locations of other customers significantly influences the route, and hence the cost they will yield.

In other settings, the arrivals of customers may be deterministic - meaning that the customers to serve are known in advance. This deterministic setting is especially relevant in B2B business contexts, when the clients to be served are known in advance, but the costs to be distributed to each customer are yet to be determined. Moreover, this setting is a stepping stone for online algorithms, where customers may arrive sequentially, and a cost needs to be allocated at each arrival.

In both deterministic and dynamic customer arrival settings, the importance of real-time cost estimation is emphasised in order to optimise last-mile delivery operations. Therefore, businesses want to be able to approximate the contribution an incoming customer will have on the route cost, and thereby allocate a certain cost to customers in real time. Allocating costs to customers in real-time can subsequently allow businesses to steer customers to an efficient time-slot and optimise route costs. Moreover, firms are accountable for allocating costs to customers not only in an efficient manner, but also with a fair approach. If the egalitarian method is applied - where every customer pays the same amount - this can be considered unfair, as it may require more time, fuel and hence money to reach certain customers than others. In such a case, it would not be an impartial method, as customers

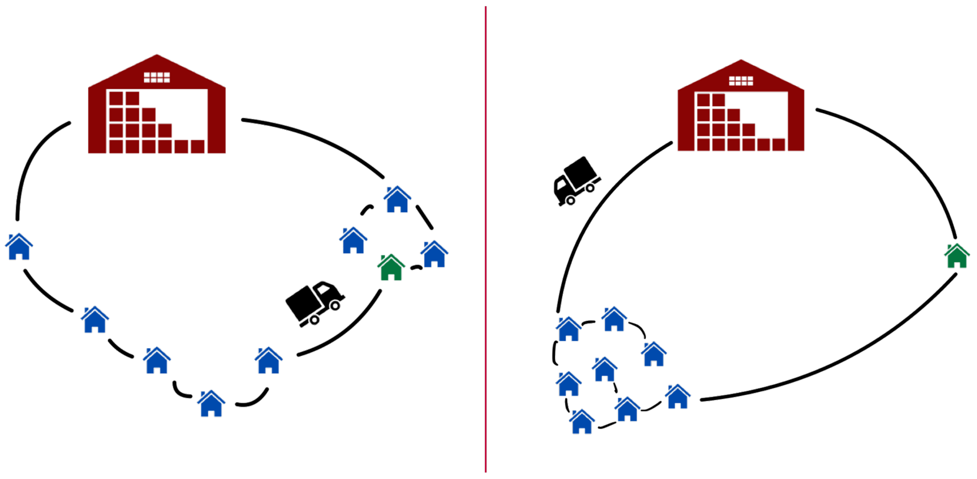


Figure 1.2: Instances with different delivery routes

who may be located close to the depot would bear the burden of paying additional costs that another customer is yielding. Moreover, a negative aspect of the Egalitarian method for cost allocation is that from a company perspective, this method provides no incentive to pick a time that is easy to serve, and hence . Allocating marginal cost of a customer - the additional incurred cost of including another customer to the route - would also be a biased cost allocation method. This is because the marginal cost of including a customer depends the point in time the customer makes an order, as well as the combination of customers - and their respective locations - who are already present.

Shapley values are a promising solution concept in fair cost allocation, as it constitutes useful properties, for example the property of distributing all costs to the agents involved. This solution concept allows for all marginal costs to be considered, and returns their average. This method consequently allows to eliminate the bias in marginal costs which derive from the order of customer arrivals. However, obtaining such values for TSP problems are computationally burdensome, as the computational complexity becomes impractical even for relatively small-scale examples.

Considering the advantages of fair cost allocation - which ensures that customers pay only for their own contribution of the route cost - and the computational burden of fair cost allocation with means of using Shapley values, the objective of this research will be to implement heuristics and algorithms to approximate the Shapley values of customers in a TSP problem in an efficient and accurate manner. The research will observe small TSP benchmark instances to identify the change of Shapley values based on different factors, such as the proximity of a customer to the depot, or to other customers. Based on such observations, naive and advanced proxies will be developed, in addition to combinations of proxies. The results will be evaluated against benchmark instances in terms of speed and accuracy - a crucial trade-off. This thesis focuses on the simpler travelling salesman problems, where a single truck serves the entire route. Moreover, the demands of customers are assumed to be known in advance in order to develop heuristics for Shapley value approximations.

The contributions of this this are as follows: although there has been research for Shapley value approximations within literature, novel approaches for such approximations are introduced by utilising minimum cost spanning trees (MST). During the construction of the tree, costs are assigned to customers in an intuitive form. Variations of advanced proxies are introduced which make use of the MST. Hybrid proxies are constructed by combining advanced proxies with intuitive algorithms and compared against the true Shapley values on the basis of benchmark instances and algorithms from research. The introduced methods in this paper outperform existing best methods in terms of accuracy and speed. Moreover, machine learning models will be employed which will combine naive and advanced proxies to make predictions on Shapley values. Additionally, the proxies will be evaluated using case-study data from The Driving Force in the Netherlands, utilizing road network distance metrics.

The organisation of the thesis is as follows: relevant literature will be discussed in Section 2. In Section 3, a comprehensive introduction for the Shapley values will be provided. Naive and advanced proxies will be introduced and described, in addition to observed patterns from solving simple TSP examples. Blended models will be introduced, along with their purpose within the research. Section 4 will be composed of three segments: an experimental study will be conducted based on benchmark instances. Later, a case study will be applied using data from The Driving Force. The final segment consists of evaluation of the implemented algorithms for robustness. The performance of the algorithms in terms of speed and accuracy will also be assessed in this section. Section 5 evaluates and discusses the strengths of the proposed methods as well as the results obtained, where remarks and recommendations for future research will also be made. Finally, concluding remarks will be provided in Section 6.

2

Literature review

The current thesis aims to implement heuristics which can approximate the Shapley values of customers in a routing problem. This approximation is essential for a fair and efficient allocation of costs yielded in a route for last mile deliveries. Cost allocation within logistics and transportation is critical for ensuring fairness and efficiency. Consequently, there have been various research into cost allocation methods. Cost allocation is also crucial to determine how customers should be steered to minimise costs related to last mile deliveries. Due to the significance of demand steering and the time slots in which customers are served, there has also been various research into time-slot management, especially within attended home delivery. Numerous survey and literature review papers have been published, with comprehensive reviews on demand management and attended home delivery. Cost allocation methods in collaborative transportation are also discussed within these surveys. The current section will begin by discussing survey papers, in order to establish a background and general overview of methods available in research. Subsequently, previous research will be evaluated and discussed in terms of implemented methods for time slot management and cost allocation. The relation between time slot management and cost allocation will be established, with a subsequent focus on cost allocation methods.

Guajardo and Ronnqvist (5) provide an extensive overview of cost allocation methods available in literature, where problems consist of planning, vehicle routing, travelling salesman, distribution and inventory. There have been surveys emerging extensively on demand management in attended home deliveries. Fleckenstein et al. (6) provide a methodological review on approaches for integrating demand management with vehicle routing. Another comprehensive review by Wassmuth et al. (7) emphasizes the complex trade-offs between customer preferences and profitable service execution by mapping these trade-offs to different planning levels and synthesize literature according to different demand management decisions. Different demand management levels are classified among strategic, tactical and operational planning levels, where research gaps are identified for each classification. Cordeau et al. (8) similarly provides an extensive review on the state-of-the-art methods for attended home delivery and attended home service problems, while focusing on real-world applications.

Effective cost allocation is crucial in guiding customers toward optimal time slots in the routing process via demand steering. This significance has led to extensive research on time slot management. An efficient time slot management can allow businesses to allocate customers to a time-slot that can help maximise profit and reduce costs. Numerous studies have explored various strategies for time slot optimisation, including dynamic scheduling algorithms (9) and predictive modelling (10).

In a similar research, Hernandez et al. (2017) (11) propose heuristics of addressing the tactical problem of selecting a time-slot schedule in each zone of a geographical area for delivery service. The proposed heuristic evaluates each schedule by corresponding a tactical routing plan of minimum cost, and the schedule with the best tactical routing plan is selected, in which a periodic vehicle routing problem (PVRP) is solved, in addition to solving a PVRP with time windows.

Agatz et al. (12) develop an automated continuous approximation approach to produce time slot schedules, where a computational study is conducted to analyse the viability of the approach. Local search method is performed to improve the time-slot schedule in an iterative process, numerical experiments conducted resulting in significant cost savings. In a subsequent research, Hernandez et al. (11) propose two heuristics of addressing the tactical time slot management problem, which consists of identifying the time slots which should be offered to customers depending on their geographical location. In their first heuristic, they decompose this problem into a periodic vehicle routing problem (PVRP) and a vehicle routing problem with time windows (VRPTW). The second heuristics approaches the time slot management problem directly by involving a unified tabu search. In their studies they compare their heuristics with an LP model of the tactical time slot management problem solved with CPLEX. They result with their second heuristic improving upon their first heuristic for the same computational time, and optimal solutions for small instances.

In a distinct yet relevant approach, Mackert (13) propose a mixed-integer linear program to approximate the opportunity costs of customers to decide whether a time slot should be offered to an incoming customer or not. They evaluate their approach on a computational study and remark that their approach yields higher profits compared to other benchmark methods. In their methodology, they model a customer choice behaviour based on a generalised attraction model introduced earlier by Yang et al. (4). In their research, a dynamic pricing policy is proposed based on the choice model to determine how much incentive - such as discounts - to offer to each time slot at the time a customer is making a booking.

In a recent study, van der Hagen et al. (10) propose a machine learning based framework using classification to classify whether a certain new customer can be feasibly included in a particular delivery time slot, given the already placed orders. In their study, they consider a single-depot VRPTW as a routing context, and define feasibility checks. They further compare their results with insertion heuristics and continuous approximation methods, in which the machine learning model outperforms all traditional methods.

In addition to time slot management problems, similar problems have been researched in order to determine the cost to serve new customers with various methods. In the research by Kone & Karwan (14), a data classification method is applied, in addition to a regression analysis in order to predict the cost to deliver goods to new customers. The classification method is applied to group similar customers, and the regression model is implemented to determine the cost. In a later study, Klein et al. (9) approximate vehicle routes and subsequently the delivery costs of expected customers using a dynamic seed-based scheme. In their study, they propose an MILP formulation to approximate the opportunity costs for dynamic pricing in attended home delivery, similarly to the subsequent research of Mackert (13). They conclude in their study that the method of dynamic pricing with an MILP approach to approximate the opportunity costs yields the highest expected profit for all delivery capacity levels tested. In 2019, Köhler & Haferkamp (15) present a customer acceptance mechanism to provide guidance to businesses in the estimation of delivery costs, and the number of customers that can be accepted without exceeding capacity. In a recent study by Wang el al. (16), the expected marginal cost of incorporating a customer in a stochastic supply chain network is approximated by proposing a scenario-sampling method. A multi-depot vehicle routing problem with inter-depot routes are considered. To obtain an exact solution, a branch-price-and-cut algorithm is implemented, and daily delivery scenarios are generated to simulate customer demand.

While extensive research has been conducted on time slot management and determining cost-to-serve, the critical role of fair cost allocation cannot be overlooked. Allocating costs to customers in a fair, accurate and efficient manner not only incentivises efficient time slot selection, but also ensures cost savings and subsequent profits of transportation companies. Therefore, the remainder of the literature review will emphasize on cost allocation methods, given their broader impact on operational efficiency.

Cost allocation methods have been emerging within literature, with a focus on logistics and transportation. Faigle et al. (17) make use of TSP games in terms of cooperative game theory, where an LP-based allocation rule is discussed for a moat packing problem. They ensure a fair cost allocation by ensuring that no coalition pays more than α times their own cost, where α denotes the ratio between the optimal TSP tour, and the optimal Held-Karp relaxation. In a further study, Özener et al. (18) similarly design cost allocation methods for an inventory routing problem defined as a cooperative game. The moat-packing problem is similarly introduced in this setting to derive costs for an inventory routing problem, which considers additionally how much to deliver to a customer. Proportional cost allocation methods are proposed which take into account factors such as consumption rate, storage capacity and customer's distance to the depot. They further utilise an approximation method to estimate the Shapley values by generating subset of customers with high synergy.

In transportation, fair cost allocation is a critical concern, particularly in network management. One of the primary methods used to address this concern is the use of Shapley values. This value has been extensively utilised within literature for fair cost allocation. Besozzi et al. (19) introduce a case study, in which they introduce the travelling salesman game (TSG) for a bus service in Castellanza, Italy. In their approach, they study possible allocations of costs via metrics such as standalone costs, and Shapley values. In their study, they compute the Shapley values directly as their case study consisted of 12 locations. In a later study, Kimms & Kozeletskyi (20) observe cooperation among salesmen for a TSP problem with rolling horizon - when demands are stochastic. The researchers aimed to utilise Shapley values in order to determine a cost allocation for salesmen, and provide insight onto benefits of collaboration between salesmen. Their studies differ from this thesis in terms of collaboration - where (20) utilise an optimisation model and observe cooperation between salesmen via Shapley values to see whether cost savings would occur, this study/thesis does not focus on collaboration between companies, and focuses on how the cost of a route can be distributed to customers in a fair manner.

In 2016, Aziz et al. (21) consider sampling based procedures for computing the Shapley values, including the ApproShapley algorithm, which will be introduced in Section 3.1. In their studies, they develop 6 proxies - three being naive, and three advanced proxies, including the moat-packing problem. The naive proxies introduced by Aziz et al. will be heavily inspired upon and will be utilised in this thesis. These proxies will be introduced in Section 3.3. Moreover, a similar approach to (21) will be applied, where two or more proxies will be blended together to result in a hybrid proxy, which will be described in Section

3.6. Their proxies are evaluated on synthetic Euclidean games as well as real-world transportation scenarios. The contribution of this thesis will be to introduce novel proxies based on the Minimum Spanning Tree (MST), and their performances will be evaluated against the synthetics Euclidean games made available in comparison with various literature.

In a later study, Popescu & Kilby (22) propose heuristics to approximate the Shapley value for a Euclidean TSG, inspired by the 1-dimensional case known as the Airport problem. In their study, they generalise the airport problem to a multidimensional case, where the shared distances and formal intersections of order 1 and 2 are utilised to obtain approximations of the Shapley values. They present two approximations where the second approximation yields higher accuracy, but with the cost of higher computational complexity. In a recent article, Levinger et al. (23) illustrate an efficient computation of the Shapley value for routing games where there is a fixed order of customers to be serviced. They later extend the problem to a general TSG with no predetermined fixed order. They utilise the Shapley values to introduce new notation and provide approximations that are computational in polynomial time. They compare their algorithm SHAPO with the approximations developed by Popescu & Kilby (22), and show that SHAPO outperforms all other proxies. In this thesis, the SHAPO algorithm will be used as a baseline comparison and evaluation of the methods implemented.

In a recent research paper, Arroyo (24) studies a central horizontal collaboration for a vehicle routing problem with time windows (VRPTW), where an approximation to Shapley values are utilised via structured random sampling to allocate costs to customers. The research also consisted of assessing the extent to which factors such as vehicle capacity, demand, distance and time windows affected costs and savings from carriers in collaboration. In their research, they make use of a structured sampling method, which is a method built upon the ApproShapley algorithm to approximate the Shapley values, which has a moderately higher computational burden than ApproShapley. The current thesis does not focus on the collaboration between companies, but rather focuses on TSP problems, where a single company delivers to a set of customers. Moreover, the ApproShapley algorithm is computationally expensive for large instances, therefore this thesis will focus on methods that can approximate the Shapley values in an efficient manner.

Existing methods for determining the cost to serve customers, such as opportunity costs (13) and marginal costs (16), do not adequately ensure a fair allocation of costs to customers on a delivery route. As introduced in Section 1, fairness is a crucial aspect of cost allocation in delivery routes. It is important that costs are not assigned in a biased manner, leading to an unequal distribution. Therefore, this thesis focuses on fair cost allocation methods, particularly the Shapley values. Although Shapley values have advantageous properties that make them suitable for this analysis, they are computationally demanding. Consequently, this thesis explores literature on efficient approximations of Shapley values and introduces novel approaches. These approaches will be evaluated against benchmark instances.

3

Methodology

The objective of this research is to be able to allocate costs to customers in real time. Moreover, the aim is to provide an indication of the contribution of a customer to the total cost of a route in a fair manner. This can allow businesses to steer customers to an efficient time-slot and thereby reduce costs within last-mile deliveries. To achieve this, a quantitative research is conducted, in the form of experiments with TSP instances using various proxies. These proxies are then evaluated using statistical measures such as the R^2 scores and RMSE. The current Section will introduce a fair cost allocation method, namely the Shapley values. An unbiased estimator for the Shapley values, ApproShapley - which will be utilised to assess the performance of the implemented proxies - will be discussed. Subsequently, the TSP games and variants adopted within this thesis will be introduced. Moreover, naive proxies from literature will be briefly discussed. Novel contributions and cost allocation methods will be introduced, which are more involved methods. Moreover, observations on simple delivery routes will be made to illustrate the significance of combining naive and advanced proxies. Subsequently, blended proxies will be introduced, where weighted average models as well as machine learning methods will be discussed.

3.1 Shapley values

Lloyd Shapley (25) introduces the concept of Shapley values, in which a systematic approach is introduced to fairly distribute the total payoff of a game among players based on their individual contribution to the total value. The cost of each player is based on their marginal cost contributions within a game, and can be outlined as the average of all marginal costs of each player. The Shapley value is defined as follows.

Let $N = \{1, 2, ..., n\}$ be a set of n players in a cooperative game, and c defining the cost function for a coalition of players, where $c : 2^N \rightarrow \mathbb{R}$. For each subset of players, a cost is provided, in which $c(\emptyset) = 0$. The Shapley value for player i is then defined as:

$$\Phi(i) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [c(S \cup \{i\}) - c(S)], \tag{3.1}$$

where S represents a subset of players excluding player i, $|S|$ denotes the number of players in subset S, and $[c(S \cup \{i\}) - c(S)]$ is the marginal contribution of player i in the coalition S. Therefore, Shapley values calculate the average contribution of a player among all possible coalitions.

The Shapley values have favourable properties for the problem of allocating costs to customers in a traveling salesman problem: Symmetry, Efficiency, Linearity and Null Player. For this thesis, the properties of Symmetry and Efficiency are relevant. However, each property will be described briefly as follows:

1. **Symmetry**: if two player make the same contributions to all possible subsets, then they would have identical Shapley values. This property ensures that players with close locations yield the same costs.

2. **Efficiency**: the total route costs distributed among all players equals to the total value generated by the coalition of all players. This property ensures that contributions of customers are fully allocated.

$$\sum_{i \in N} \delta_i(c) = c(N), \tag{3.2}$$

   where $\delta_i(c)$ denotes the Shapley values for a customer i given a cost c, and $c(N)$ denotes the cost of the entire route.

3. **Linearity**: the Shapley values are linear, meaning if a game is composed of multiple sub-games, the Shapley value for the whole game can be obtained by adding up the Shapley values of the individual parts. If two coalition games described by cost functions c and d are combined, the distributed gains should correspond to the gains derived from c and gains derived from d:

$$\delta_i(c + d) = \delta_i(c) + \delta_i(d) \tag{3.3}$$

4. **Null Player**: if a player does not contribute to any coalition, they should have a Shapley value of zero. Formally, if $c(S \cup \{i\}) = c(S) + c(i)$ for all subsets S that do not contain i, then player i is null.

Shapley values have desirable properties which allow for a fair allocation of costs between customers. Instead of assigning the marginal cost of customers based on the order of their arrivals, Shapley values consider all possible combinations of customer arrivals, and computes the marginal cost of all customers based on each permutation. Subsequently, the marginal costs are averaged. Essentially, Shapley values can be regarded as the expected marginal costs of customers. Considering all subsets and orders of customers to a delivery route makes Shapley values an appropriate method for fair cost allocation.

To illustrate the significance of fair cost allocation, consider the following scenario in which a route consists of three customers. In this scenario, the marginal costs for customer 1 will be analysed. The marginal cost allocated to customer 1 is then dependent on the customers who are already present when customer 1 makes an order. Further consider the following coalitions and costs: $\{1, 2, 3\} = 70$, $\{1, 2\} = 55$, $\{2, 3\} = 65$, $\{1\} = 40$, $\{2\} = 50$, $\{\emptyset\} = 0$. The grand coalition consisting of all three customers constitutes a cost of 70.

Consequently, the marginal cost of customer 1 is dependent on the point in time the customer makes an order, as well as the customers already present in the route.

$$\begin{aligned}\textbf{Cost of customer 1 at \{1\}} &= \{1\} - \{\emptyset\} \\ &= 40 - 0 \\ &= 40\end{aligned} \tag{3.4}$$

$$\begin{aligned}\textbf{Cost of customer 1 at \{1,2\}} &= \{1, 2\} - \{2\} \\ &= 55 - 50 \\ &= 5\end{aligned} \tag{3.5}$$

$$\begin{aligned}\textbf{Cost of customer 1 at \{1,2,3\}} &= \{1, 2, 3\} - \{2, 3\} \\ &= 70 - 65 \\ &= 5\end{aligned} \tag{3.6}$$

Equations 3.4, 3.5 and 3.6 illustrate the marginal cost of customer 1 if they are the first, second or last customer to be included in the route, respectively Figure 3.5 demonstrates the marginal cost if customer 1 arrives after customer 2. The full network can be visualised in Figure 3.1. The routes formed based on the order of arrival of customer 1 can be visualised in Figures 3.2, 3.3 and 3.4. If the customer is the first to be included in the route, they are likely to have a higher marginal cost than when they are included in the route at a later stage in time. This observation can be made in Figure 3.2 - when there are no customers, there are no costs. However when customer 1 arrives, the cost of the route becomes 40, thereby making the marginal cost contribution of this customer as 1. If customer 2 is already present in the route when customer 1 joins, the resulting route can be visualised in Figure 3.3. The total cost of this route is 55, whereas before customer 1 arrives the total route cost of going from the depot to customer 2 and back to the depot would cost 50. Therefore, the marginal cost of customer 1 in this setting would be 5. Finally assume customer 1 is the last customer to arrive in the route. Customers 2 and 3 are already present. The cheapest route cost of considering all three customers is 70, as can be seen from Figure 3.4. Before the inclusion of customer 1, the cost of serving customers 2 and 3 would be 65. Consequently, the marginal cost of customer 1 would be 5 if they were to join at this point in time.

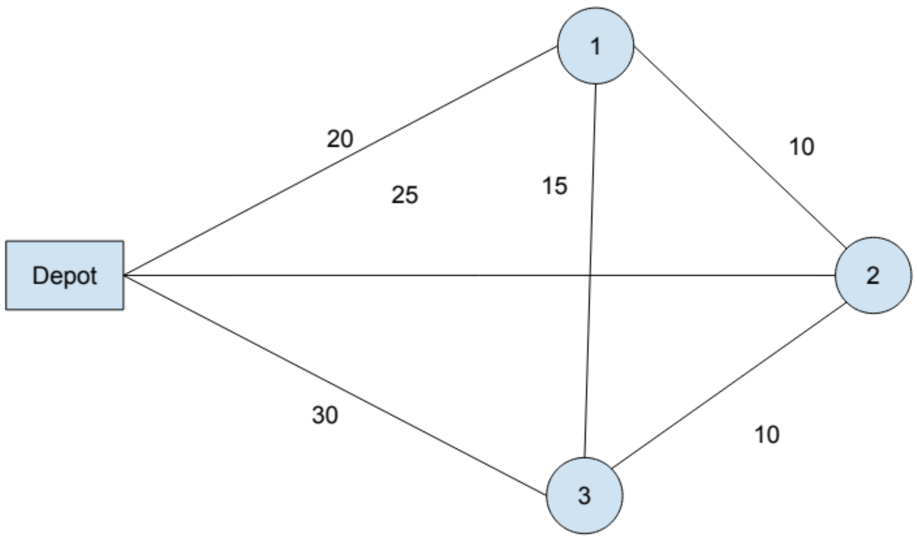


Figure 3.1: Example with depot and three customers

Allocating costs to a customer solely based on their marginal cost may lead to an unfair cost division, as the cost of incorporating the customer may differ based on the point in time the customer places an order, and the customers who are already present at time of order placed by a new customer. Shapley values mitigate this problem by considering all possible orders of arrival, thereby considering all $n!$ combinations of arrivals. For each possible order of arrival, the marginal cost of including customer i is then computed and averaged to yield the Shapley values. This allows to remove biased cost allocations, and consequently makes the Shapley values suitable for the current problem.

Although the Shapley values are promising in terms of fair cost allocation, its implementation in Traveling Salesman Problems is associated with a considerable burden in computational effort. This is attributable to the 2^n number of subsets for which the TSP

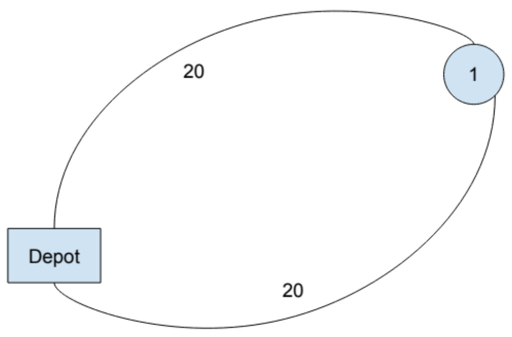


Figure 3.2: Route when customer 1 is first to join the route

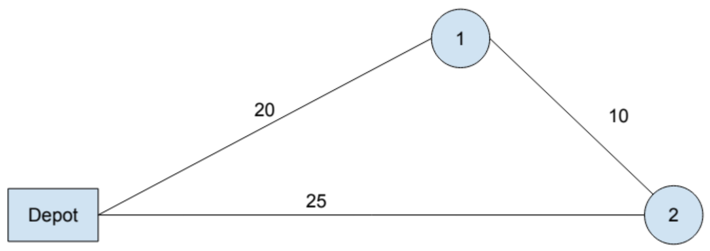


Figure 3.3: Route when customer 1 is second to join the route

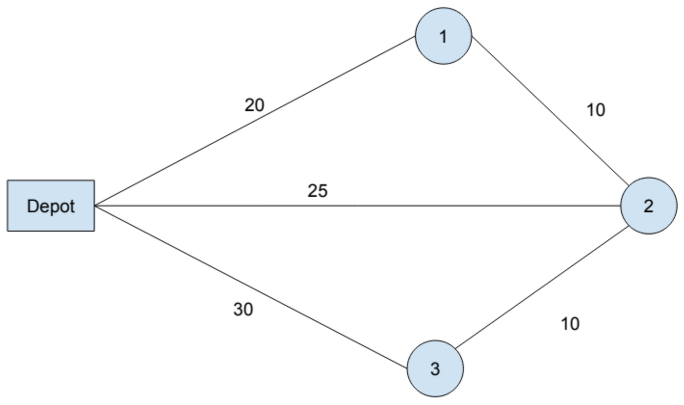


Figure 3.4: Route when customer 1 is third to join the route

needs to be solved for, in addition to the $n!$ permutations which need to be considered to compute the Shapley value. Solving TSP's are regarded as NP-hard problems, therefore computing them for 2^n subsets may not be solvable in polynomial time. To overcome this challenge, Castro et al. (26) propose the ApproShapley algorithm to obtain an unbiased estimate of the Shapley value (21). This algorithm is known to be a sampling method, which draws uniformly a random permutation of customers arrivals for a specified number of times. For each drawn permutation, the marginal cost of each customer i is calculated. The reformulated Shapley values based on the ApproShapley algorithm is then calculated as follows in Equation 3.7:

$$\Phi_i = \frac{1}{n!} \sum_{\pi \in S_n} (c(\pi_i \cup \{i\}) - c(\pi_i)), \tag{3.7}$$

where π_i is the subset of customers in N which precede location i in a set of |n|! permutation orders of customer arrivals, and S_n defines the set of all permutations given n customers. A permutation is randomly selected (π), and the marginal costs for each customer are calculated based on the cost of subset before and after the arrival of customer i.

The ApproShapley algorithm is promising in terms of yielding a baseline approximation for the Shapley values. However, it also leads to large computational burden in larger and realistically sized games. Therefore, during the remainder of this research this algorithm will be utilised to compare the performance of the implemented proxies in Sections 3.3 and 3.4. The speed and accuracy of the ApproShapley algorithm with smaller scale instances are illustrated in Chapter 4.

3.2 TSP Games and variants

The Traveling Salesman Problem is a well-known optimisation problem within combinatorial optimisation. As it is such a significant and burdensome problem, as well as the variety of practical considerations with businesses, it consists of variants which have been analysed by various researchers. Such variants are discussed in a survey by (27), where the TSP problem is altered for numerous use cases. In this survey, categories of TSP variants are reviewed, namely being the profit-based and time window based TSP. The profit based TSP is a simplification of the traditional TSP problem, where it is not a requirement to visit all locations, and visiting a location comes with a profit. The TSP with time windows requires all customers to be visited within a certain time window. Although these existing variants address various aspects of the TSP, this thesis will consider two TSP games. The traditional TSP game constitutes visiting a set of locations and constructing a route such that the resulting cost is minimal and resulting in assigned costs to each customer. In such a variant only customers pay a share of the total cost. In addition to this variant, this thesis will also explore TSP games where the depot also becomes a collaborative player and is assigned costs. The two TSP games will be explored for both a TSP tour, and a TSP path - when the vehicle does not return to the depot after visiting the last location. The TSP games when the depot is also a collaborative player have not been adequately addressed within literature, and presents opportunities in uncovering different patterns and characteristics of cost allocation.

Different games are used and explored since the traditional problem of solving a TSP tour when the depot is not a player, and subsequently obtaining the Shapley values are demanding. Adapting the problem to a TSP path, and incorporating the depot also as a collaborative player allows to reduce the complexity of the problem. Section 3.2.1 will briefly introduce and motivate the variant of solving a TSP path when the depot is a player. Followingly, Section 3.2.2 will discuss the TSP tour when the depot is a player. Finally, Section 3.2.3 will discuss the traditional TSP, where the depot is not assigned costs. The complexity of the variants are increasing. Performing an analysis of algorithms for variants of increasing complexity allows to demonstrate how additional constrains and real-world considerations impact the problem.

3.2.1 TSP path - depot as a player

The TSP path is a variant to the traditional TSP, and is modified in a manner such that the vehicle starts its route from the depot, but is not required to return back. Instead, the vehicle ends its route at the last customer visited. This variant is relevant to open vehicle routing problems (OVRP), where a vehicle similarly does not return to the depot after servicing the last client on a route (28). These sorts of problems have been encountered in practice in home deliveries of packages. Therefore, observing a TSP path can be subsequently relevant. To adapt the traditional TSP route into the TSP path, the most expensive edge - the longest distance between any two locations - is removed. This variant is adapted since it is relevant to some of the proxies developed to approximate the Shapley values, which will be elaborated in Section 3.4.

The conventional manner of allocating costs in a TSP problem comprises of assigning a share of the total payoff to each customer. This is customary, since it is generally only the customers who bear the transportation costs, and the depot is viewed solely as a hub or starting point, with no burden of financing costs. When the depot is also a collaborative player, they are assigned a portion of the costs of the route, along with the customers.

This approach identifies the depot as a fundamental part of the transportation network, and consequently allots a cost to them as well. This approach can also be considered to be representative of the real-world, as the position of the depot with respect to the customers can play a role in all costs incurred. Moreover, considering the depot as a player attributes the depot's impact on the total travel costs, and promotes a possibly more fair distribution of costs among all customers and the depot.

Observing how costs are allocated when the problem consists of a TSP path, and allowing to allocate costs also to the depot allows to simplify the traditional TSP problem. Implementing proxies to approximate the Shapley values in a scenario where everyone contributes to the total costs is a less demanding task, in particular when the TSP to be solved is a path rather than a tour.

3.2.2 TSP tour - depot as a player

To make the problem more complex yet representative, the problem of allocating costs for a TSP tour when the depot is a player is additionally analysed. Once again, the depot is also assigned costs, which simplifies the problem context, since every player is allocated a portion of the total payoff. However, in this setting the Shapley values are observed when the TSP concerned is a tour rather than a path, signifying that the vehicle returns to its starting point - the hub - once it has visited all customers.

3.2.3 TSP tour - depot not as a player

This variant is also known as the traditional TSP, however it is also the most complex of the three settings discussed. It is nevertheless essential to observe this variant, and allocate costs to customers in this setting, as it is a scenario which represents real-life problems well, as this can directly affect the pricing and service quality in logistics and transportation. Therefore, understanding the true cost contributions of each customer helps in fair pricing and better resource allocation.

In regards to the methods implemented particularly in Section 3.4, the TSP path problem when the depot is not a player will additionally be analysed, and the proxies implemented will be evaluated based on the Shapley values for these settings, as the path is a minor simplification of the traditional TSP.

In order to review the performance of the proxies implemented and discussed in Sections 3.3, 3.4 and 3.6, the Shapley values will be computed for both the TSP path and tour. Moreover, the Shortcut distance algorithm introduced in Section 3.3 and the MST algorithm discussed in Section 3.4 are adapted to approximate the Shapley value for both cases when the depot is a collaborative player, and when it is not.

3.3 Naive proxies

This section describes three naive proxies, which were inspired from the experiments of (21). Namely, the depot distance, shortcut distance and the rerouted margin.

3.3.1 Depot distance

The depot distance considers the distance d_{i0} of a customer to the depot, and is regarded as a proxy for the Shapley value δ_i - the shorter the distance between the depot and the customer, the less costs they pay. This value is normalised by dividing the cost by distances of all customers to the depot:

$$\delta_i^{\text{Depot}} = \frac{d_{i0}}{\sum_{j\in L} d_{j0}} \tag{3.8}$$

This proxy has the advantage of being solvable in polynomial time, irrespective of the size of the problems concerned. However, it may also cause the problem of overestimating the true Shapley values, especially when the customers are equidistant from the depot (21). Therefore it is a computationally efficient, yet ineffective in ensuring accuracy for the true Shapley values.

3.3.2 Shortcut distance

The shortcut distance proxy is yet another naive proxy which can be used as an adequate proxy for the Shapley values, as the computation is simple in terms of the steps required. However, this proxy does require the shortest path given a route of customer locations. The shortcut distance is the marginal cost savings of skipping a customer within the optimal route. Given an optimal route $[x_1, x_2, x_3, ..., x_n]$ where x_i denotes the order of the customers in the optimal tour, the shortcut distance algorithm is defined as:

$$\text{Shortcut}_i = d_{i-1,i} + d_{i,i+1} - d_{i-1,i+1}, \tag{3.9}$$

where $d_{i,j}$ denotes the distance between customers i and j. To scale this proxy to a value between 0 and 1 - for a fair comparison between proxies - the Shortcut distance value is divided by the sum of all shortcut distance values for all customers in the route:

$$\delta_i^{\text{Shortcut}} = \frac{\text{Shortcut}_i}{\sum_{j \in N} \text{Shortcut}_j} \tag{3.10}$$

Although this proxy is efficient in terms of ease of computation once an optimal route is established, similarly to the Depot distance proxy, it is ineffective in establishing an accurate proxy for the true Shapley values. In addition, this proxy requires an optimal route given a set of customers, which may be NP-hard to solve depending on the number of customers in the problem. However, this proxy is regarded as beneficial in capturing essential patterns in routes, such as customers being 'on a line', which will be further elaborated in Section 3.5.

3.3.3 Rerouted margin

The rerouted margin consists of the marginal cost of including a customer in a route, and is defined as:

$$\text{Reroute}_i = c(N) - c(N \backslash i) \tag{3.11}$$

Similarly to the other aforementioned naive proxies, this proxy is also scaled to be compared by dividing over all rerouted margin costs:

$$\delta_i^{\text{Reroute}} = \frac{\text{Reroute}_i}{\sum_{j \in N} \text{Reroute}_j} \tag{3.12}$$

For smaller instances, the Reroute Margin proxy yields identical results as the Shortcut Distance proxy. However, this proxy is not favourable in comparison with the Depot Distance and Shortcut Distance proxies, as this method consists solving the TSP problem $n+1$ times - once to get the total cost of the route, and once for every customer. Therefore, as the problem size gets bigger, the computational time increases and is consequently an inefficient option to approximate the true Shapley values.

3.4 Advanced proxies

The current section describes advanced heuristics implemented as a proxy for the Shapley values of a TSP game. The Christofides' algorithm is introduced, based on the method by Aziz et al. (21), although it is not implemented due to the computational burden of the problem. Novel approaches are introduced in this section, by utilising a minimum spanning tree. Namely, the MST and the Modified Kruskal algorithms are defined which allocate costs to customers during the construction of the tree. The MST algorithm is applied by (29) to allocate costs to players in a minimum cost spanning tree game, however this thesis defines this method as a proxy to approximate the Shapley values of customers for a TSP problem as a novel approach. These methods are advanced as the steps taken to allocate costs are involved, while being solvable in polynomial time.

3.4.1 Christofides' algorithm

The Christofides' algorithm is an involved method of solving the Travelling Salesman Problem (30). It is a favourable approach for an approximate TSP solution, as the algorithm ensures that its solutions are within 1.5 times the optimal solution length (31), given that the triangle inequality holds.

The idea behind this proxy is that the TSP problem is solved using the Christofides algorithm, and the ApproShapley sampling method described in Section 3.1 is applied to approximate the Shapley values. The algorithm is as follows:

- The Minimum Spanning Tree of the graph is constructed
- Vertices with the odd number of degrees are found in order to construct a minimum-weight perfect matching, which consists of connecting pairs of odd-degree vertices with the least possible total weight, using the edges not present in the MST
- The edges of the MST and the minimum-weight perfect matching are combined to form a connected graph, which in turn is the Eulerian circuit, as it is guaranteed to visit every edge exactly once
- Shortcutting is performed by removing redundant edges in order to minimise the total length of the circuit.

Once the problem is solved, the ApproShapley algorithm is utilised to sample randomly from the list of permutations a specified number of times, defining the order of arrivals of customers in the route. The marginal cost of including customer i in the route given the permutation order is then calculated and normalised after repeating the sampling process. For a small number of instances this method is effective in both efficiency and accuracy, however for larger instances above 15 customers, this method may also become computationally burdensome. Although this method performs well in terms of accuracy, the goal of this research is to implement algorithms that can provide reliable results in an efficient manner. Therefore, this algorithm is not regarded as suitable for large problem sizes, as it is computationally burdensome even for smaller instances.

3.4.2 MST algorithm

A minimum spanning tree (MST) is a subset of the edges of a connected, weighted graph which connects all vertices together with the minimum possible total edge weight (31). Their property of being solvable within polynomial time with a complexity of $O(N^2log(N))$ - where N denotes the number of nodes in the graph - makes them a relevant method to use as a baseline to develop heuristics to solve the traveling salesman problem, such as the Christofides' algorithm. Minimum spanning trees are relevant and interesting especially within the context of travelling salesman problems. MST's offer a lower bound to the optimal TSP route (31), as it joins all nodes together with the minimal cost, whereas a TSP path can be considered to be a tree with no branches. The cost of the TSP path is always at least the cost of an MST, thereby providing a lower bound. Moreover, the Christofides' algorithm described in Section 3.4.1 makes use of a MST to construct a TSP tour. The Christofides algorithm is known to provide a solution which is always at most $\frac{3}{2}$ longer than the optimal TSP length. Therefore, MST's are relevant within the context of travelling salesman problems. This makes them an attractive method to employ to assign costs to customers.

In addition to constructing the minimum cost spanning tree, it is essential to address the problem of allocating costs to each node when constructing the tree. This problem is known as a minimum cost spanning tree game (29), where an MST is constructed to connect all nodes such that the created network has minimal cost, and associated costs are distributed to each node.

Given an MST problem, a vertex oriented allocation vector γ can be obtained as the tree is constructed, which provides the allocated costs to each node in an MST game. Kruskal's algorithm is utilised to construct the MST game, which will in turn characterise the vertex oriented allocation vector as the sum of the cost contributions obtained in each step of the tree construction (29). This vertex oriented allocation vector γ directly translates into the Shapley values for an MST game. The novelty of the this thesis is to utilise the MST algorithm - which will be formulated for two variants - and obtain a vertex oriented cost allocation vector. The results from this vector will then be utilised as a proxy for approximating the allocated costs to customers in a TSP problem.

The MST algorithm is implemented for two variants: when the depot is a player, and when it is not. When the depot is also a player, costs are also allocated to this player. The resulting algorithm is a simplified model of the traditional problem. Sections 3.4.2.1 and 3.4.2.2 describe each variant of the MST algorithm. For each variant, cost contributions will be attained for each customer during the construction of the minimum spanning tree. The exact mechanism and procedure to obtain cost contributions will be described for both variants.

3.4.2.1 MST algorithm - depot as a collaborative player

As introduced in Section 3.2, it is of interest to compute the Shapley values of customers when the depot is a collaborative player, and is assigned costs. This would allow for a more realistic yet simplified setting, when the position of the depot with respect to customer locations can impact the costs allocated to all players involved. In such a setting, every player is assigned costs. Accordingly, the method of assigning costs to players using the MST algorithm is undemanding yet intuitive.

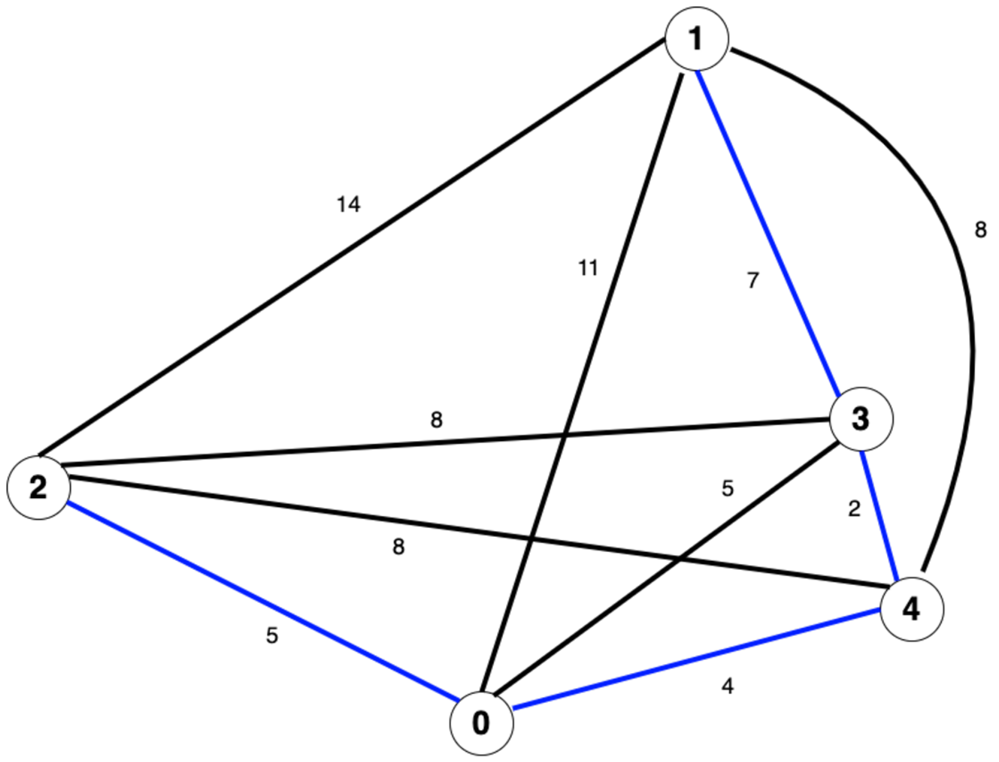


Figure 3.5: Minimum spanning tree - resulting tree highlighted in blue

Consider the minimum cost spanning tree problem with source node 0 and location set $L = \{1, 2, 3, 4\}$ as presented in Figure 3.5. The resulting tree from this graph is highlighted in blue. It is assumed that the triangle inequality holds for the fully connected graph, and each value on the edges represents the cost of the connection. Consequently, the MST problem can be solved using the construct and charge algorithm, in which the

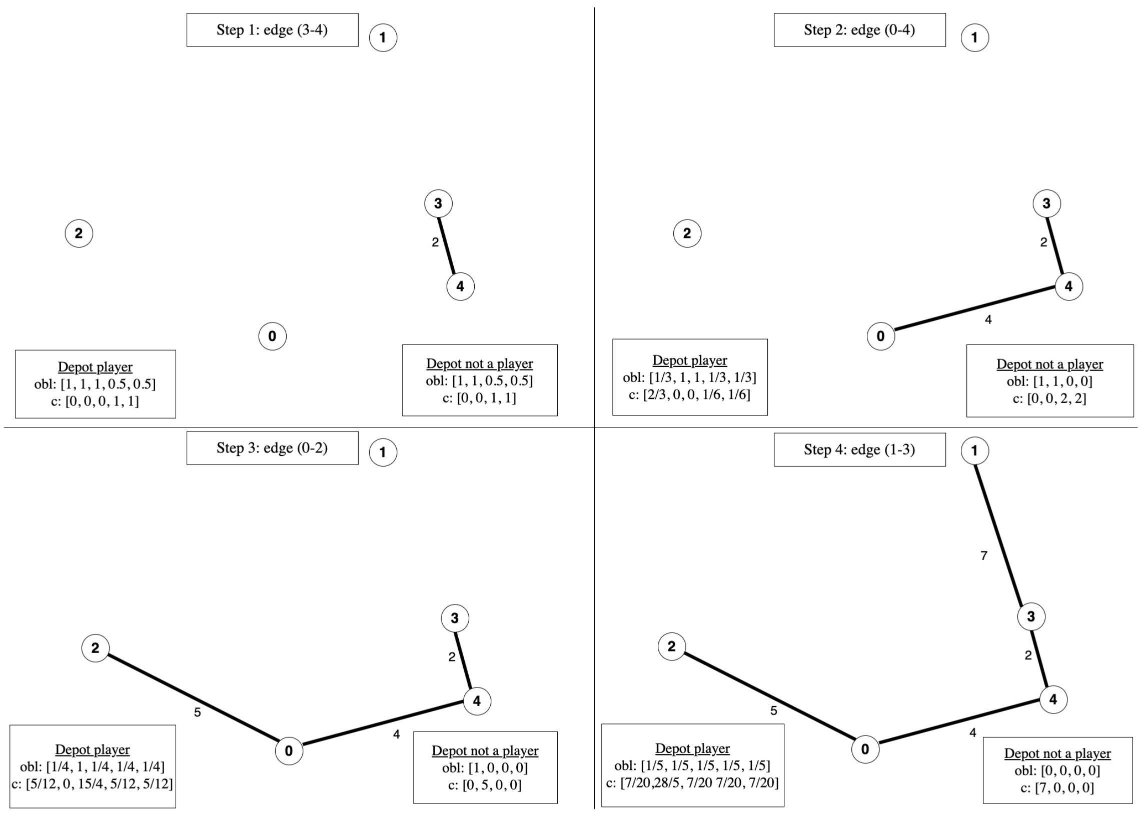


Figure 3.6: Constructing an MST using Kruskal's algorithm

costs of the connections are assigned to each location for each step. In order to obtain a cost allocation vector, the following assumptions are made. Each player has to pay in total one unit of edges of the tree constructed - namely, obligations. Moreover, for each step, the players in the newly formed component will contribute to the costs of the edge constructed. Finally, all players in the newly formed component must have equal remaining obligations (29). Figure 3.6 illustrates the steps of constructing the minimum cost spanning tree following Kruskal's algorithm. Initially, all locations have an initial obligation vector $obl^0 = [1, 1, 1, 1, 1]$. All players including the depot have obligations. This figure shows the remaining obligations and costs assigned to both variants when the depot is a collaborative player which is described in the current section, as well as the variant when the depot is not a player, which will be introduced in Section 3.4.2.2

In Step 1, the edge {3,4} is constructed. Since the newly formed component consists of two players, the remaining obligations of both components will be $\frac{1}{2}$, and both players will contribute to the cost of the connection. The resulting cost vector for the current step is then $c^1 = [0, 0, 0, 1, 1]$, and remaining obligations $obl^1 = [1, 1, 1, \frac{1}{2}, \frac{1}{2}]$. In Step 2, the edge {0,4} is included, resulting in the newly formed component consisting of players [0,3,4]. Since the newly formed component now consists of three players, the remaining obligations vector is of the form $obl^2 = [\frac{1}{3}, 1, 1, \frac{1}{3}, \frac{1}{3}]$. To obtain the contribution of each player to the new edge, the difference between the current and previous obligations vectors are taken: $contribution_i = obl^{i-1} - obl^i$. This is based on the cost allocation method for minimum spanning tree games by (29). Therefore, the contribution of each player to the newly added edge in this step is $[1, 1, 1, \frac{1}{2}, \frac{1}{2}] - [\frac{1}{3}, 1, 1, \frac{1}{3}, \frac{1}{3}] = [\frac{2}{3}, 0, 0, \frac{1}{6}, \frac{1}{6}]$. Subsequently, the costs assigned to each player is then the contributions, multiplied by the weight of the edge: $c^2 = 4 \cdot [\frac{2}{3}, 0, 0, \frac{1}{6}, \frac{1}{6}] = [\frac{8}{3}, 0, 0, \frac{2}{3}, \frac{2}{3}]$. In Step 3, the edge {0,4} is included, resulting in the newly formed component to be consisting of players [0,2,3,4]. All players in the newly formed component have equal remaining obligations, resulting in $obl^3 = [\frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]$. By obtaining the contribution of each player to the new edge as in the previous step, the cost $c^3 = [\frac{5}{12}, 0, \frac{15}{4}, \frac{5}{12}, \frac{5}{12}]$ is obtained. In the final step, all players are in the same component, resulting in equal remaining obligations of $\frac{1}{5}$. Since all players are in the same component, and have the equal remaining obligations equal to $\frac{1}{n}$, n denoting the number of locations including the depot, this obligation is considered negligible, as any remaining edge costs would be equally attributed to each player. The final costs assigned to each player is $c^4 = [\frac{7}{20}, \frac{28}{5}, \frac{7}{20}, \frac{7}{20}, \frac{7}{20}]$.

Summing the four cost allocation vectors, the vertex oriented allocation vector γ will be obtained, corresponding to the cost associated with each player, including the depot.

$$\begin{aligned} c^1 + c^2 + c^3 + c^4 &= [0, 0, 0, 1, 1] \\ &+ \left[\frac{8}{3}, 0, 0, \frac{2}{3}, \frac{2}{3}\right] \\ &+ \left[\frac{5}{12}, 0, \frac{15}{4}, \frac{5}{12}, \frac{5}{12}\right] \\ &+ \left[\frac{7}{20}, \frac{28}{5}, \frac{7}{20}, \frac{7}{20}, \frac{7}{20}\right] \\ &= \left[\frac{103}{30}, \frac{28}{5}, \frac{41}{10}, \frac{73}{30}, \frac{73}{30}\right] \end{aligned} \tag{3.13}$$

To make a fair comparison between different instances and proxies, the results are normalised by dividing the cost of each player with the sum of the total costs - 18 - thereby resulting in the following approximated Shapley values:

$$\hat{\gamma} = [0.19, 0.31, 0.22, 0.13, 0.13] \tag{3.14}$$

In Equation 3.14, the value 0.19 is the allocated costs for the depot, signifying that 19% of the total route cost is allocated to the depot. Allocating such costs to the depot can additionally provide perspective and potential strategies on where to place a hub with respect to a company's customers. Therefore, this method can be insightful to logistics companies.

3.4.2.2 MST algorithm - depot not as a player

An adaptation of the MST algorithm is when the depot is not a player, and is not assigned any costs. The resulting algorithm is similar to that described in Section 3.4.2.1. However, a key distinction between this variant of the MST algorithm is that once players are connected to the source node (the depot), the depot will not contribute to the cost of this connection. Rather, the obligations of the players in the component who are connecting with the depot will be depleted. Consider the same example as before. Now the obligations and cost vectors are of length 4, with every player except the depot having obligations.

All customers have an initial obligation vector $obl^0 = [1, 1, 1, 1]$. The tree is constructed in the same order as before. In the first step, the edge {3,4} is constructed. Since the newly formed component consists of two players, the remaining obligations of both components will be $\frac{1}{2}$, and both players will contribute to the cost of the connection, thereby making the cost vector for the current step $c^1 = [0, 0, 1, 1]$, and obligations vector $obl^1 = [1, 1, \frac{1}{2}, \frac{1}{2}]$. In the next step, the connection between the edges {0,4} are made, thereby connecting the components {3,4} with the source, resulting in the newly formed component [0,3,4]. A key difference between the costs and obligations in this step compared to when the depot is a player is that the obligations of customers 3 and 4 are now depleted, and will have no remaining obligations. In the previous step, these players had remaining obligation $\frac{1}{2}$, therefore based on the contributions, the costs vector formed in this step are $\mathrm{c}^2 = [0, 0, 2, 2]$, with remaining obligations $obl^2 = [1, 1, 0, 0]$. In the third step, the connection between the source 0 and player 2 is made, thereby constructing the new component {0,2,3,4}. Since the remaining obligations of players 3 and 4 are already depleted, player 2 will solely pay for the cost of the new connection. Therefore the costs vector for the third step will be $c^3 = [0, 5, 0, 0]$, and $obl^3 = [1, 0, 0, 0]$. In the final step, the connection between the edges {1,3} will be made, and the newly formed component will consist of all edges. Since player 1 is the only player with remaining obligations, the cost vector for the final step will be $\mathrm{c}^3 = [0, 0, 0, 7]$, and all obligations will be depleted.

In the same manner as before, the four cost contribution vectors are summed to yield the vertex oriented allocation vector γ, which corresponds to the cost allocated to each player in the tree.

$$c^1 + c^2 + c^3 + c^4 = [0,0,1,1] + [0,0,2,2] + [0,5,0,0] + [7,0,0,0] = [7,5,3,3] \tag{3.15}$$

Moreover, the costs are once again scaled to be between 0 and 1 by dividing the cost of each player with the sum of all costs, thereby yielding a normalised vertex oriented allocation vector:

$$\hat{\gamma} = \left[\frac{7}{18}, \frac{5}{18}, \frac{3}{18}, \frac{3}{18}\right] \tag{3.16}$$

This method is regarded as a more advanced proxy for approximating the Shapley values for a travelling salesman problem, and its usage in combination with naive proxies will be later discussed in Chapter 4.

A further extension of this algorithm, MST+ was considered, in which the Christofides' algorithm would also be utilised. The resulting TSP tour from the Christofides algorithm would be used as a basis, and the most expensive edge would be removed to result in a TSP path. Edges present within the resulting MST algorithm but absent from the TSP path of the Christofides algorithm would be removed, and the costs and obligations of the relevant players would be redistributed. Subsequently, the edges present within the resulting TSP path which are absent in the resulting tree from the MST algorithm would be included in the MST algorithm, and costs would be assigned to players who have had their obligations redistributed. However, this method did not seem promising and experimentations showed poorer performance in comparison to the MST algorithm. Therefore, this algorithm is excluded from this thesis.

3.4.3 Modified Kruskal path and tour

Section 3.4.2 introduces the MST algorithm, in which a minimum spanning tree is constructed using Kruskal's algorithm, and costs are assigned to players based on the added edges and the components constructed. Kruskal's algorithm is a greedy method which sorts all edge costs in an ascending order, and includes the edges one by one if including the edge will not result in a cycle in the spanning tree. The result of this algorithm will be a spanning tree with minimal weight. An observation made with the MST algorithm is that branching nodes from the resulting tree - nodes who have more than two edge connections - were more likely to overestimate the normalised true Shapley values, whereas leaf nodes - nodes who have only one edge connection - were more likely to underestimate the true values. Moreover, the resulting tree in the MST algorithm would not be fully representative of a TSP path or tour due to the property that a spanning tree tends to branch, whereas a path or tour would consist of nodes with degree at most 2.

In order to mitigate this problem, Kruskal's algorithm is modified in the following manner: the edges are again ordered in non-descending order, and the edges are included to the tree one by one. If including the edge results in a cycle, or if it results in a node having more than two degrees - leading to branching - then the edge will not be included. Costs are allocated in the same manner as described in Section 3.4.2.2, with the assumption that the depot is not a player and consequently not assigned any costs.

To illustrate the modification of the Kruskal algorithm, consider a scenario with 10 customers. The resulting MST by applying Kruskal's algorithm can be observed in Figure 3.7. By applying this algorithm, it can be observed that customers 3, 6 and 8 are branching. This sort of route would typically not occur in a TSP path or tour. Therefore, the modified Kruskal algorithm ensures that branching does not occur. The resulting network from this modification can be observed in Figure 3.8. The edge between customers 3 and 6 are not included, as this edge resulted in both customers being branched. As an alternative, the edge between customers 2 and 10 are constructed, which leads to a longer edge connection, but eliminates branching. Similarly, customer 8 is branching, due to its connection to customer 7. Instead of this connection, customers 7 and 9 are connected together. When each edge is included, the costs are allocated in the same manner as before according to the MST algorithm for when the depot is not a collaborative player.

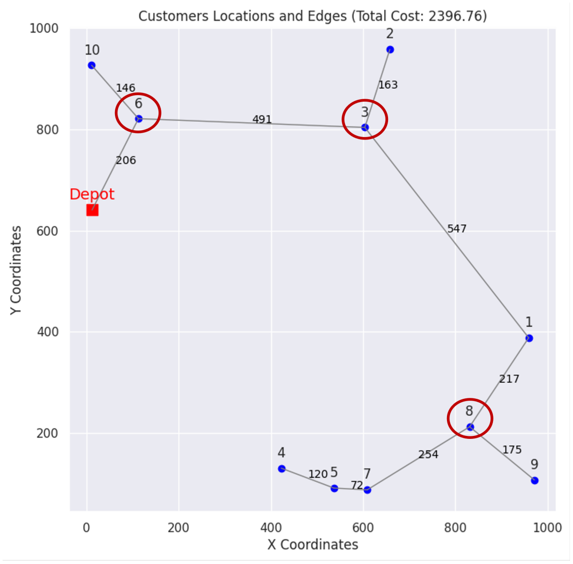


Figure 3.7: MST results using Kruskal's algorithm

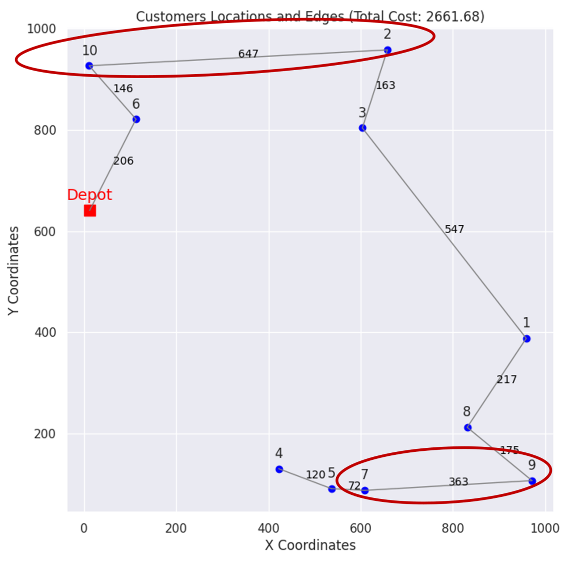


Figure 3.8: MST results using modified Kruskal's algorithm

The resulting network in Figure 3.8 is then not a tree, but a TSP path. This can be a more representative network than a tree, since it can resemble the route a vehicle can take. In order to make the resulting path into a tour, the two remaining leaf nodes are connected, and the cost of this edge is distributed to customers according to the Depot distance algorithm described in Section 3.3.1.

3.5 Observing patterns

The naive proxies introduced in Section 3.3 - in particular the Shortcut distance and Depot distance proxies, in addition to the MST algorithm introduced in Section 3.4 comprise of various properties that can allow to capture the general framework allowing to approximate the Shapley values in a suitable manner. Sections 3.5.1 and 3.5.2 provide two separate TSP instances and will discuss observations made using the aforementioned proxies.

3.5.1 Example 1 - 4 customers

Table 3.1: Normalised Shapley values and proxy results of Figure 3.9

Customer	Shapley value	MST	Depot distance	Shortcut distance
1	0.6460	0.6267	0.5054	0.8375
2	0.0195	0.0537	0.0433	0.00017
3	0.11325	0.1597	0.1906	0.00015
4	0.2211	0.1597	0.2606	0.1621

Figure 3.9 illustrates a simple TSP route, consisting of a depot and 4 customers. The normalised true Shapley values, and the proxy results for depot distance, shortcut distance and MST can be observed in Table 3.1. A few observations can be made when observing the optimal TSP route - computed using Concorde.

- Customer 1 is relatively distant from the depot and the other customers - it can be considered to be remote
- Customer 2 is very close to the depot
- Customer 2 is on the way to customer 4 in the optimal route
- Customers 3 and 4 are relatively close to each other
- Customer 3 is on the way to customer 1 from customer 4 in the optimal route

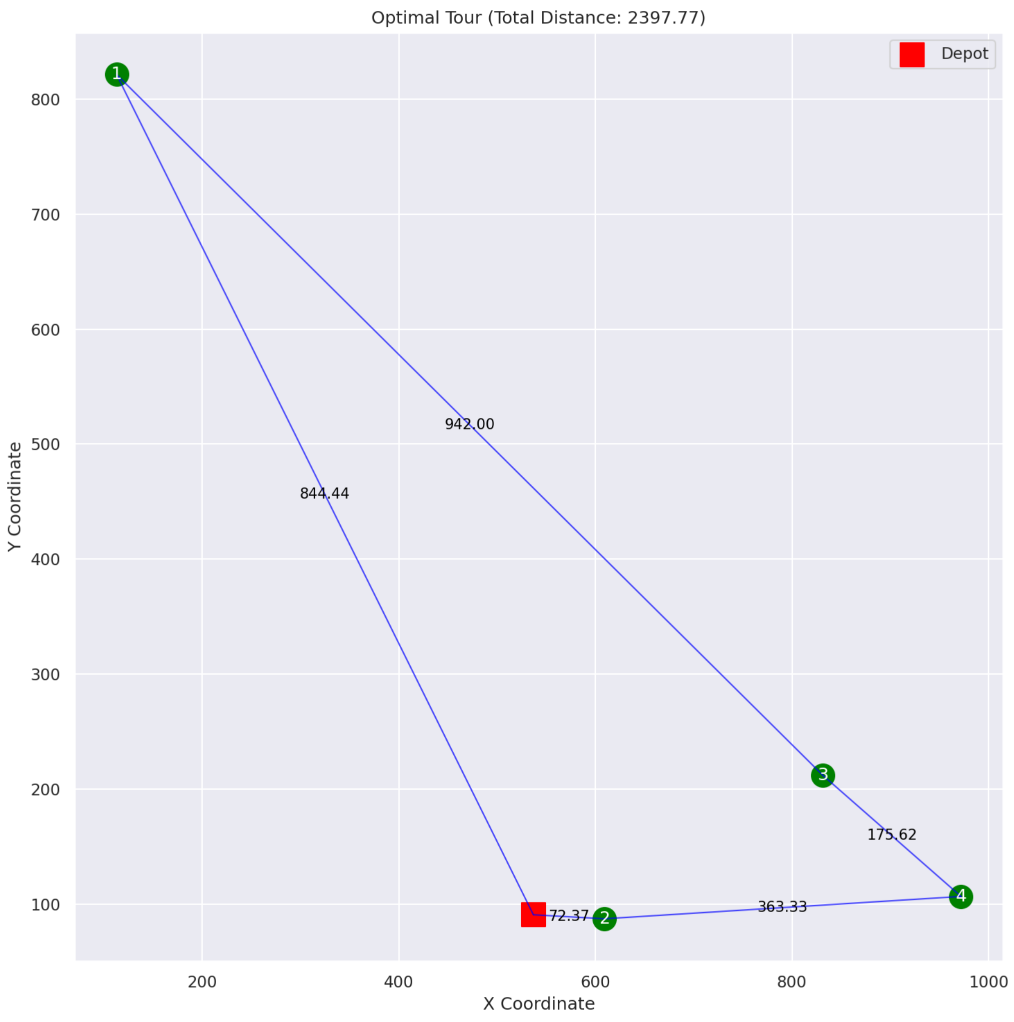


Figure 3.9: TSP instance with 4 customers

When analysing the results from Table 3.1, it can be seen that the Shapley values of the Shortcut distance proxy for customers 2 and 3 are very close to 0. This can be attributed to their properties of being 'on the line' - if customer 2 was removed from the route, the vehicle would still traverse through the same route. Consequently, including customer 2 in this route would not result in additional costs for the truck driver. However, this observation is not fully true, as we can observe from the Full Enumeration results that in particular for customer 3, costs are allocated - specifically, 11% of the total route cost. From this, it can already be suggested that having a customer on a line is solely not sufficient to approximate the true Shapley values.

Another key observation from this table is that the Shortcut distance value for customer

1 is relatively high, with 83.5% of the costs allocated to them. Contrary to the previous observation made, customer 1 is both distant and not 'on a line'. From the optimal route it can be seen that the truck would have to 'go out of its way', meaning it would incur additional travel costs to serve customer 1. This could in turn lead to significantly higher costs allocated to them. Although when comparing this value with the true values, it can be suggested once again that the property of being on a line is solely not sufficient to approximate the Shapley values well. If there are two groups of customers clustered together, and there is a single, remote customer between the two clusters, the sole customer tends to have a cost close to zero. This is because the clustered customers would share the cost of the route with each other. However, in the case of no clusters, such as the case with Figure 3.9, sole customers on a line such as customer 3 would still be assigned costs. If customers 1 and 4 had other customers clustered around them, then the true Shapley value of customer 3 would be closer to zero.

When observing the results from the MST algorithm, it can be seen that customers 3 and 4 are allocated the same costs, with 15.9% of the costs, suggesting that they share the costs of the route. This observation can be attributed to the characteristics of the MST algorithm, which utilises Kruskal's algorithm - adding edges with the lowest cost such that the edge will not cause the construction of a cycle. Since the distance between customers 3 and 4 appear to be the closest with respect to other customers, the MST algorithm assigns equal costs to them when the other edges are included in the resulting tree. However, it can be observed from Table 3.1 that this is not fully accurate, as customer 4 is assigned twice the costs of customer 3. It can be suggested that although the customers are close to each other, 3 has a lower Shapley value due to being 'on the line', as observed from the values of the shortcut distance algorithm. However, as will be observed in Figure 3.10 and Table 3.2, the property of clustering nearby customers and assigning the same cost to them can be a beneficial attribute in certain cases.

Considering the values of the Depot distance proxy, it can be observed that although the distance of a customer may have some role in determining the Shapley values, there is a significant discrepancy between the proxy values, and the true values.

3.5.2 Example 2 - 6 customers

In slightly expanded instances, certain patterns can be more prominent. Figure 3.10 illustrates the TSP route for an instance with 6 customers, and Table 3.2 displays the resulting Shapley values, in addition to the proxy results. Similarly to observations made in Section 3.5.1, a couple of remarks on customers can be made for this instance:

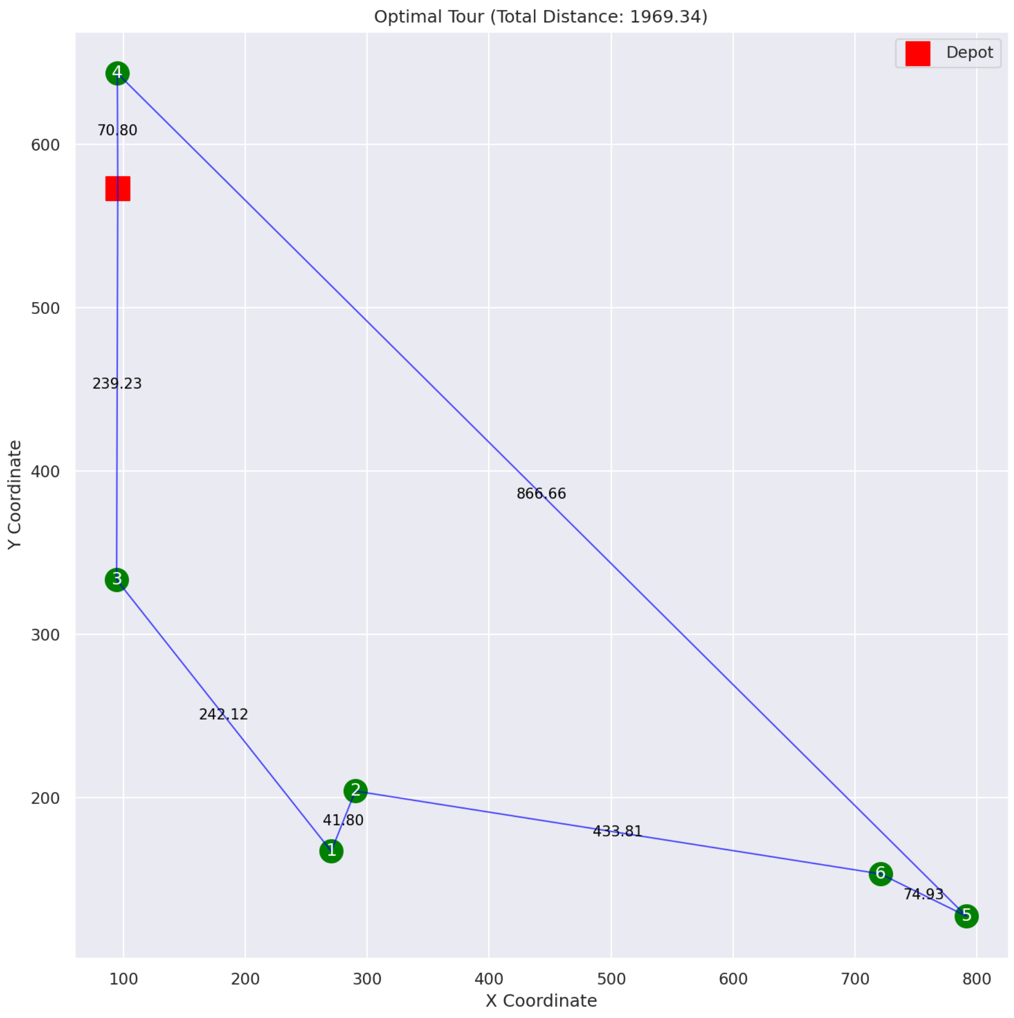


Figure 3.10: TSP instance with 6 customers

Table 3.2: Normalised Shapley values and proxy results of Figure 3.10

Customer	Shapley value	MST	Depot distance	Shortcut distance
1	0.1259	0.1276	0.1607	0.1319
2	0.1082	0.1276	0.1517	0.0664
3	0.0741	0.2156	0.0870	0.1063
4	0.0615	0.0646	0.0257	0.2980
5	0.3519	0.2322	0.3006	0.3924
6	0.2782	0.2322	0.2740	0.0046

- Customers 1 and 2, and customers 5 and 6 can be considered to be clustered due to their close proximity

- Customer 4 is relatively close to the depot
- Customer 6 is on the way to customer 5

Considering the values of the MST proxy, it can be seen that customers 1 and 2, and 5 and 6 are assigned the same costs, with 12.7% and 23.2% of the total route cost, respectively. As suggested in Section 3.5.1, close customers are clustered together in the MST algorithm, leading to shared costs. When comparing these values to the true Shapley values, the MST proxy is successful in approximating the costs for customers 1 and 2 - since they have similar costs with 12.5% and 10.8% respectively. However, despite being close in proximity, customers 5 and 6 do not share the costs, with customer 5 being assigned 35.1% of the costs, and 27.8% for customer 6. From here, it can be suggested that although being clustered may help in approximating the true Shapley values, more properties are required and it is not solely sufficient in obtaining an accurate estimation.

Regarding the values for the Shortcut distance proxy, a striking pattern is that for customers 2 and 6, their contribution is close to 0. As mentioned in the observations, customer 6 is on the way to customer 5 in the optimal tour. Similarly, customer 2 is close in proximity to customer 1, therefore the vehicle would not require to divert significantly from the route to visit customer 2. Subsequently, they are assigned low costs. Although as can be seen from the results of the true Shapley values and observations made in Section 3.5.1, being on a line is an effective pattern to capture, however it is not sufficient to make a good proxy for the true Shapley values.

When observing the values of the depot distance, it can be noticed that customers further away from the depot are assigned higher costs. Although for customers 3 and 6, this approximation is accurate, the proximity of a customer to the depot is exclusively not the only element which plays a role in approximating the Shapley values accurately.

To conclude this section, it can be stated that the naive proxies Depot distance and Shortcut distance, as well as the advanced proxy, MST algorithm encapsulate useful properties relating to the route and the relations between customers. Although solely they are not precise in approximating the Shapley value, it is of belief that a hybrid proxy combining these elements can result in an improved estimation of the Shapley values.

3.6 Blended models

The naive and advanced proxies discussed in Sections 3.3 and 3.4 are subsequently employed to build blended models which can predict the Shapley value. This method is implemented by employing Decision Trees, Linear Regression models, Random Forests and Linear Trees. The underlying motivation for this method is to observe whether decision support models such as decision trees and linear trees, ensemble methods like random forests as well as statistical models such as linear regression models can allow to uncover linear or nonlinear patterns in predicting the Shapley values in the naive and advanced proxies by combining them. As previously discussed in Section 3.5, the Shortcut distance algorithm efficiently captures the 'customers on a line' property of a tour - customers who are on the way to other customers tend to be allocated less costs than expected, as the vehicles do not need to steer into a different direction to visit the customer. Moreover, the MST algorithm can effectively capture the notion of clustered customers; as customers who are close to each other have similar allocated costs. The depot distance proxy can also capture how the distance of a customer to the depot can vary the allocated cost to the customer.

The combination of these algorithms and their properties can allow for an improved prediction for the true Shapley values. Due to ease in implementation and interpretation, as well as the minimal data preparation required, the focus for blended models consist of decision trees, random forests, linear regression model and linear trees. In addition, a weighted average model of the form $x \cdot A + (1 - x) \cdot B, x \in (0, 1)$ will be formed to get a linear combination between the MST proxy and one of the naive proxies. The experiments conducted and the performance of these models will be discussed in Chapter 4. Sections 3.6.1, 3.6.2, 3.6.3 and 3.6.4 provide a brief description of each model utilised. The weighted average model will be introduced in Section 3.6.5.

3.6.1 Decision trees

Decision trees are decision support models which use a tree-like model in order to make decisions based on input features. These models consist of nodes, branches and leaves, in which the nodes represent the features of the dataset, and leaves represent the outcome of the decision tree. Decision tree models were selected as one of the models to make a prediction for the true Shapley values by combining various proxies, due to its ease of interpretability. Moreover, decision trees are adept in making feature selection by splitting the dataset based on the most significant proxies. Their property of capturing non-linear relationships between features and the target variable makes them a flexible option for combining proxies.

3.6.2 Random forests

Random forests are an ensemble method, which operates by constructing numerous decision tree models during the training process. For a classification problem, the output of the random forest model is the class which is selected by most trees, whereas for a regression task this is the average prediction of the individual trees. The advantage of this method in comparison to the decision trees is the increase in accuracy due to the construction of multiple decision trees. However, this increase in accuracy comes with the cost of reduced interpretability. In a decision tree, the decision-making process can be observed since a single tree is constructed, however with a random forest model this is more complex due to the multitude of decision trees. However, since decision trees are already a good choice for interpretability, the random forest model is also employed for the problem of predicting the Shapley values for a TSP tour and path due to ease of implementation and better performance in terms of accuracy.

3.6.3 Linear regression models

Linear regression models are statistical models employed to model the linear relationship between features and the target value, and has the aim of predicting the target value by utilising the feature values. The model consists of intercepts and coefficients in the following form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \epsilon \tag{3.17}$$

The objective in linear regression models is to find the best fitting line through the data points, and the coefficients determine the weight of the features in predicting the dependent variable. Linear regression models were also utilised in order to build a hybrid proxy, due to its simplicity in implementation. Especially its feature in ease of interpretation makes this a desirable option to build a hybrid proxy, as the model provides clear insights into the influence of each feature (proxy) into predicting the target value (true Shapley value).

3.6.4 Linear trees

Linear trees combine the attributes of both linear models and decision trees in order to create a hybrid model which is known to improve predictions in addition to having improved interpretability (32). Instead of having a constant approximation at each leaf node of the decision tree, linear trees fit a linear model on each leaf node. Therefore, each leaf contains a local linear model that predicts the target feature based on a subset of features relevant to the leaf. This model has the advantage of combining both decision trees and linear models, thereby enhancing the predictive power of the model. This is due to its effectiveness in capturing both global and local patterns in the data, which can in turn improve the predictive accuracy. Despite its enhanced features, linear models are able to have the quality of being easily interpretable. Consequently, this model was also considered to be a suitable choice to implement blended proxies.

3.6.5 Weighted average model

A weighted average model is a simple yet intuitive method of blending two proxies together. As noted in Section 3.5, the Shortcut distance, Depot distance and MST proxies capture useful properties regarding customers and a route. Therefore combining them can be beneficial. A weighted average model consists of the form $x \cdot A + (1 - x) \cdot B$, where x denotes the weight assigned to proxies A and B. This model is implemented to approximate the ApproShapley values for the TSP path and tour, for when the depot is not a collaborative player.

For x values ranging from 10% and 90%, the MST proxy is blended with either the Shortcut distance or Depot distance proxies, and their performance is evaluated against the ApproShapley values by observing the MSE and RMSE scores. To illustrate, 20 instances consisting of 10 customers will be employed. To approximate the TSP tour, the proxies selected are MST and Depot distance, whereas for the TSP path, MST and Shortcut distance proxies are utilised.

The performance of the blended model consisting of the MST algorithm and Depot distance proxy in comparison to the ApproShapley value of the TSP tour can be observed in Figure 3.11. This Figure was obtained by observing 20 benchmark instances from (21) consisting of 10 customers. For each instance, the ApproShapley values are computed for each customer. The MST algorithm is blended with the Depot distance algorithm with different values of x, ranging from 10% to 90%. For each variant, the weighted average model is compared against the ApproShapley values in terms of the MSE and RMSE scores. Figure 3.11 illustrates the average MSE and RMSE scores of the weighted average model for varying x values. From the MSE and RMSE values, it can be observed that the average error is lowest when the ratio of MST/Depot distance proxies utilised are defined as 30/70.

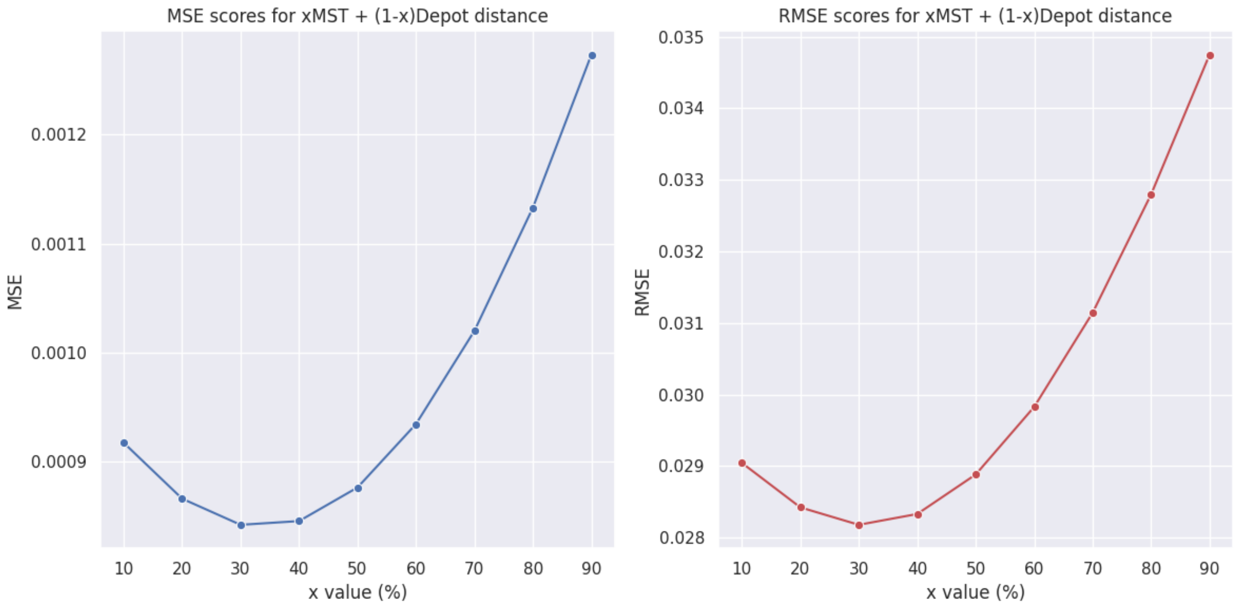
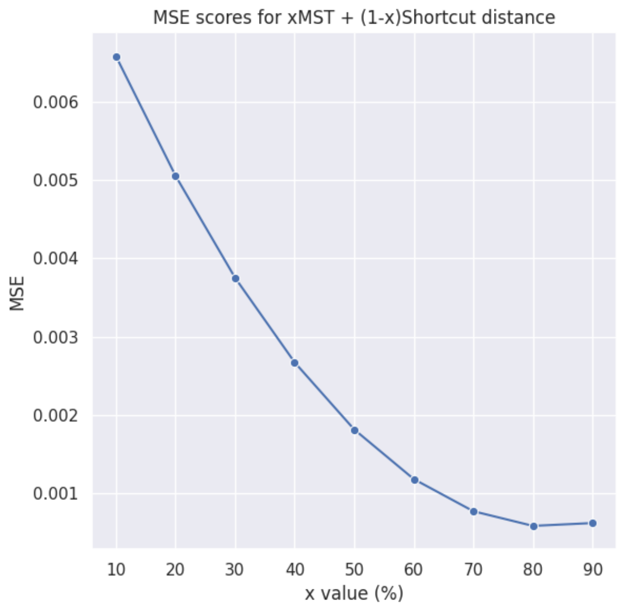


Figure 3.11: MSE & RMSE scores for varying x values with respect to the TSP tour

For the blended model consisting of the MST algorithm and Shortcut distance proxy, the performance against the ApproShapley value of the TSP path can be observed in Figure 3.12. It can be observed that the RMSE is lowest when x is 80%, therefore the weighted average model is composed in the form of $0.8 \cdot MST + 0.2 \cdot \textit{Shortcut distance}$.

In Section 4.3, benchmark instances based on Aziz et al. (21) will be utilised to evaluate the performance of the implemented proxies against the ApproShapley values. The results obtained from the weighted average model will also be discussed. It should be noted that for different customer sizes, the ratio of the proxies blended are susceptible to change. Therefore, for each customer size the proxies utilised and their corresponding weights during blending are included in Appendix A. In general, it was observed that these x values obtained are not instance dependent, but vary marginally based on the number of customers in the route. For increasing number of customers, it was observed in Appendix A that the weighted average model performs better when it places a higher weight on the MST model.



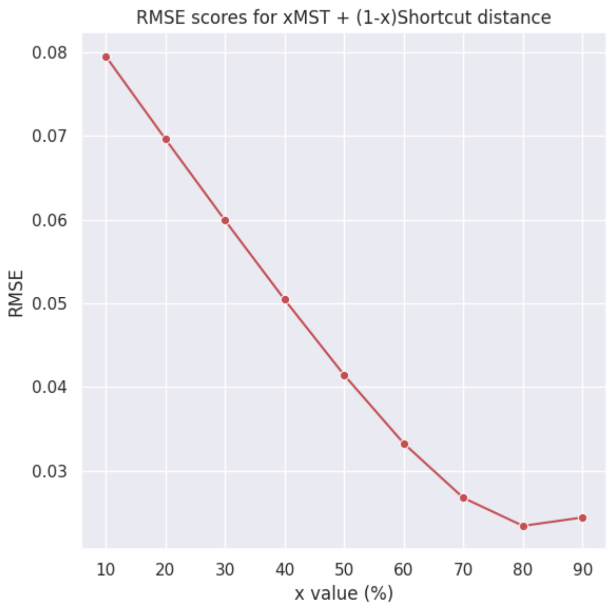


Figure 3.12: MSE & RMSE scores for varying x values with respect to the TSP path

4

Results

The current section presents the findings of this finding in four components. The study aimed to implement proxies that can approximate the Shapley values of a Travelling Salesman Problem (TSP) in a fair and efficient manner. The results will introduce an instance with 15 customers, where observations for the proxies will be made in a similar manner as in Section 3.5.

The rest of this section will be structured as follows: Section 4.2 will discuss the performance of AppроShapley as an unbiased estimator for true Shapley values (21) in terms of speed and accuracy. Subsequently, the experimental setup for evaluating the performance of the naive and advanced proxies in addition to the blended algorithms will be described in Section 4.3. Benchmark instances ranging from 10 to 35 customers from (21) are utilised to obtain allocated costs for customers in different scenarios, and are compared against ApproShapley values in terms of accuracy and speed. Section 4.4 will introduce a case study by utilising data from The Driving Force. Finally, Section 4.5 will evaluate the robustness of the MST algorithm.

4.1 Observing patterns - 15 customers

In order to illustrate the features and patterns certain proxies captured, Section 3.5 utilised two small-scale instances, with one consisting of four customers, and the other consisting of six customers. Prior to evaluating the performance of the implemented proxies on benchmark instances, the current section will illustrate an example consisting of 15 customers, and the performance of the MST, Depot distance and Shortcut distance proxies, which will be evaluated against the ApproShapley values, which were obtained by sampling 5000 times. For the current example, true Shapley values are not calculated due to computational complexity. This example further amplifies the complexity of approximating the Shapley values, thereby the necessity in blending proxies to capture key features of a TSP.

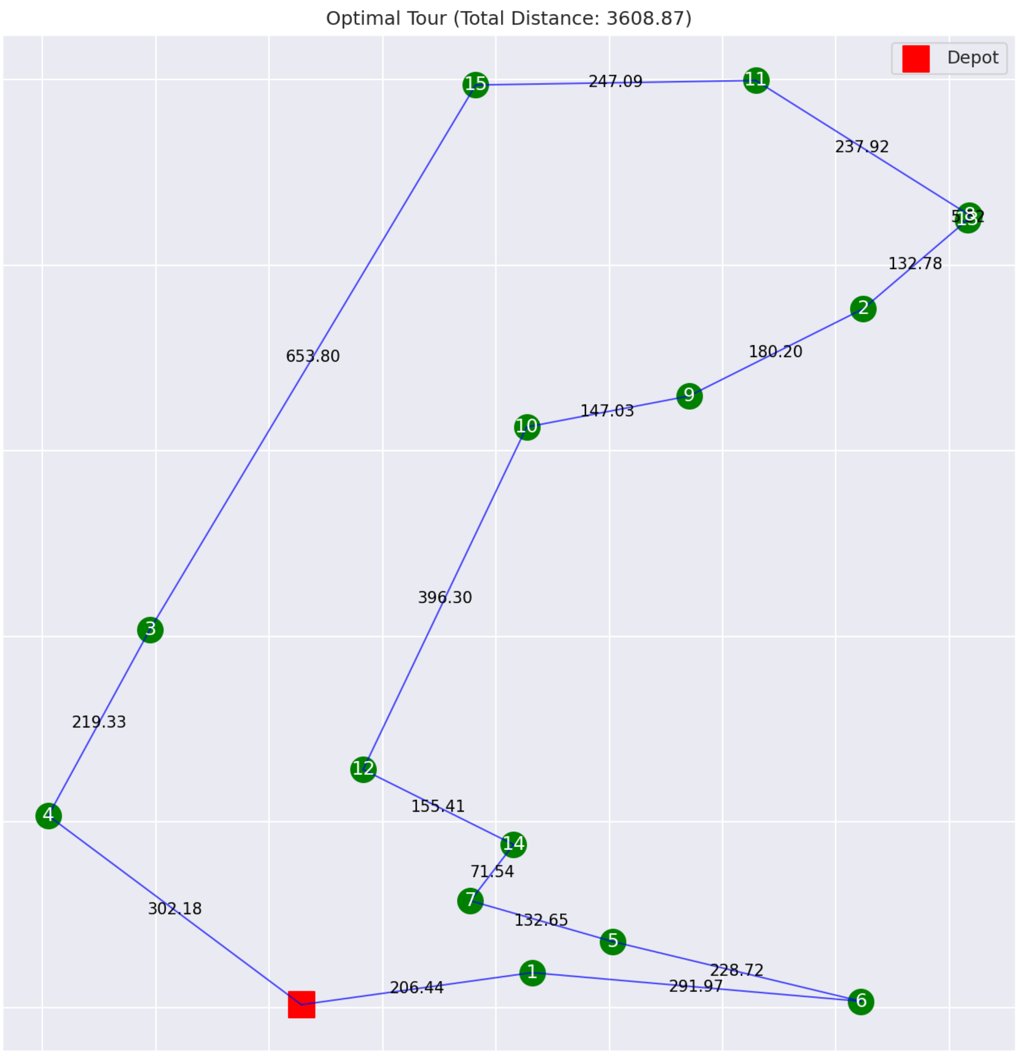


Figure 4.1: TSP instance with 15 customers

Figure 4.1 shows the resulting TSP tour for an instance consisting of 15 customers. The normalised ApproShapley values for each customer, in addition to the approximated Shapley values using the MST, Depot distance and Shortcut distance algorithms can be observed in Table 4.1.

From this figure, the following observations can be made:

- Customer 3 is on the way between customers 4 and 15

Table 4.1: Normalised Shapley values and proxy results of Figure 4.1

Customer	ApproShapley tour	ApproShapley path	MST	Depot distance	Shortcut distance
1	0.022214	0.023875	0.041169	0.023341	0.004716
2	0.071822	0.062941	0.069771	0.101634	0.002640
3	0.054250	0.066709	0.087775	0.048106	0.000106
4	0.069966	0.078404	0.087775	0.034165	0.100064
5	0.028378	0.030254	0.041169	0.031969	0.000114
6	0.120232	0.122530	0.087213	0.055839	0.460821
7	0.017108	0.021178	0.039849	0.021094	0.071024
8	0.100825	0.074066	0.045507	0.117035	0.003868
9	0.060350	0.069135	0.075501	0.083632	0.004292
10	0.052176	0.058358	0.075501	0.073880	0.050744
11	0.116443	0.096249	0.099432	0.121385	0.028546
12	0.034197	0.048238	0.061635	0.029303	0.105920
13	0.094495	0.073946	0.045507	0.116434	0.000506
14	0.025998	0.033720	0.039849	0.028791	0.059137
15	0.131546	0.140397	0.102346	0.113393	0.107502

- Customer 5 is on the way between customers 6 and 7
- Customers 2 and 9 are on the way between customers 10 and 13
- Customers 8 and 13 are very close in proximity
- Although close to the depot, customer 6 seems to be out of route for a vehicle to traverse

When observing the normalised ApproShapley values alongside the proxy approximations, a few remarks can be made. The MST algorithm clusters close customers together, such as the pairs of customers 1,5, 3,4, 7,14, 8,13, 9,10. Consequently, the MST proxy assigns the same costs to these pairs of customers. Although similarities in costs can be observed in the ApproShapley values for pairs 8,13 and 1,5, the MST algorithm tends to over or underestimate these costs at times. However, this proxy is proficient in capturing the feature of clustered customers, and allocates similar costs to them. This is representative of a real life situation, as customers who live close together are likely to share the cost of the route.

For the approximations of the depot distance algorithm, it can be seen that customers closer to the depot tend to have lower costs. However, this solely is not sufficient to allocate costs to customers, as there are discrepancies between this proxy and the ApproShapley values - for instance customer 6 has an ApproShapley tour value of 0.12023 - indicating that 12% of the total tour costs are allocated to this customer. The depot distance algorithm assigns 5.5% of the costs. Although this customer is relative close to the depot, the combinations of other customers in the route determine the trajectory. Since the vehicle would need to go out of its way to visit this customer, it would yield higher costs than necessary. Therefore, the depot distance is a practical proxy, yet is not sufficient in capturing the full essence of the Shapley values.

Finally, considering the allocated costs to customers using the Shortcut distance proxy, it can be observed that almost 50% of the entire route costs are assigned to customer 6. This is due to being out of route for a vehicle to traverse, given the other customers in the route. Whereas for customer 2,3,5 and 9 the costs are almost 0 - this is due to their properties of being 'on the line'. As discussed in Section 3.5, this proxy captures the feature of customers being on the way very efficiently, however in terms of allocating costs in a fair manner, it fails to provide a systematic approach.

Observing the different proxies and the various practical properties they capture, the significance of combining them is once again emphasized.

4.2 Performance of ApproShapley

In Section 3.1, Shapley values and the sampling method of ApproShapley was introduced. In order to assess the performance of ApproShapley with respect to the true Shapley values, five instances consisting of 10 customers were utilised. For each instance, the Shapley values were computed using:

- True Shapley values using full enumeration to solve the TSP
- Shapley values using the TSP solver Concorde
- Sampling method (ApproShapley) using 5000 samples

The performance of the ApproShapley algorithm is compared against both the Shapley values obtained via full enumeration, and via the TSP solver Concorde. The runtime of each method and instance can be observed in Table 4.2. For each of the instances, ApproShapley was employed by sampling 5000 permutations from the list of permutations possible with a route consisting of 10 customers. It can be suggested that even with small customer instances such as 10, the ApproShapley algorithms performs faster than both the direct enumeration, and the Concorde solution, while simultaneously resulting in an average RMSE value of 0.0017. Due to its speed and efficiency in providing a baseline estimate for the true Shapley values, the remainder of the experimental setup will consist of comparisons of the proxies and blended models with the ApproShapley values.

Table 4.2: Runtime of Shapley values for instances with 10 customers (in seconds)

Instance	Full enumeration	Concorde	ApproShapley
1	112.8	78.1	8.31
2	111.8	75.9	10.57
3	110.1	76.1	11.03
4	110.9	75.0	10.87
5	113.02	75.39	7.78
Average	**111.73**	**76.09**	**9.71**

In order to assess the number of samples required for the ApproShapley algorithm to perform effectively, 24 randomly generated instances consisting of 10 customers were utilised. For each instance, the true Shapley values are computed using full enumeration - where the cost of the TSP routes are calculated manually. Moreover, Shapley values are calculated by solving the TSP routes using a TSP solver, Concorde. In addition, the ApproShapley algorithm was employed, ranging from 10 to 8000 samples, with replacement. For each sample size and instance, the RMSE score was calculated. The performance of the ApproShapley algorithm can be visualised in Figure 4.2. Between 10 and 1000 iterations, it can be observed that the RMSE score drops sharply, although more samples are required to stabilise. Around 5000 samples, the RMSE score appears to stabilise to a minimum, with minor fluctuations. Between 5000 and 8000 samples, the difference in average RMSE can be considered negligible, as the fluctuations reduce and the error approaches to a value close to a 0. With increasing sampling size, the error between the Shapley and ApproShapley values converge to 0 (21). However, increasing the sampling size comes with a cost of computational burden. From this figure, it can be suggested that in order to ensure a good comparator to the true Shapley value while simultaneously saving computation time, 5000 is a reasonable sample size. Therefore, the remaining Sections of this thesis will discuss ApproShapley values which will have been sampled 5000 times.

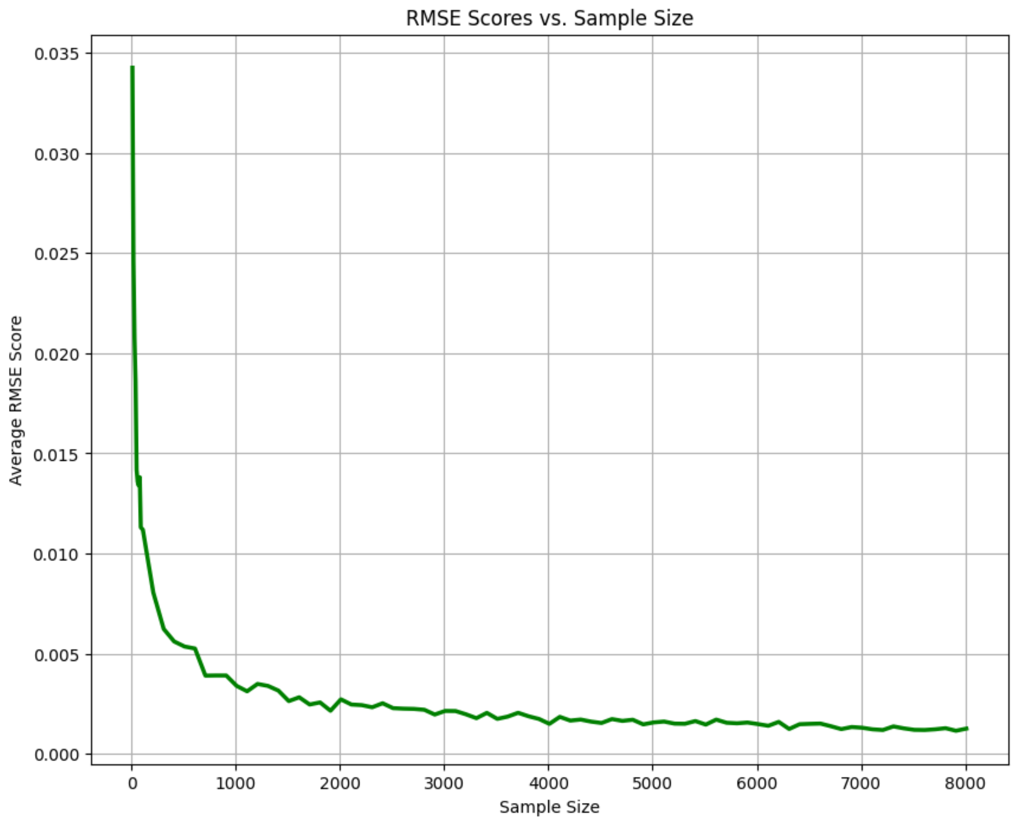


Figure 4.2: Average RMSE score of ApproShapley with increasing sample size

4.3 Synthetic data - Aziz et al.

The performance of the naive and advanced proxies, in addition to the blended models are evaluated using synthetic data made available by Aziz et al. (21), where Euclidean TSG instances are initialised by sampling randomly from the uniform distribution in a square with a dimension of 1000. Instances of sizes ranging between 4 and 35 are provided. In this Section, the results will be discussed with customers of sizes ranging from 10 to 35. The performance of the implemented heuristics when the depot is a collaborative player will be discussed in Section 4.3.1. Subsequently, the heuristics implemented for approximating ApproShapley values when the depot is not a player will be evaluated and compared against algorithms implemented in literature and will be discussed in Section 4.3.2.

4.3.1 Evaluation - Depot as a collaborative player

This section discusses the TSP variant when the depot is a collaborative player, and is assigned a cost during the route and the tour. As discussed in Section 3.2, incorporating the depot as a player in the cost allocation for the Travelling Salesman Problems and Shapley values ensures that all entities involved in the TSP carry a share of the overall costs, which may lead to a more impartial distribution of costs. Moreover, when the depot is included in the cost allocation, it provides an incentive for businesses to strategically locate their depots, as it can motivate businesses to position their depot in a location such that the total travel costs are minimised, thereby improving the efficiency of the logistics network. Consequently the MST and Shortcut distance algorithms have been adjusted to take into account the depot also as a collaborative player. This section will discuss the performance and efficiency of the adjusted MST and shortcut distance proxies - as well as the weighted average model combining the two - in approximating the ApproShapley value of both the TSP tour and path for instance sizes ranging from 10 to 35.

Tables 4.3 and 4.4 show the resulting approximation errors of the MST, Shortcut distance and weighted average MST for sizes varying from 10 to 35. For both approximations of the ApproShapley TSP path and TSP tour, the blended/weighted average model consists of the form $0.8 \cdot MST + 0.2 \cdot \textit{Shortcut distance}$. The average RMSE score for each method - MST, Shortcut distance and weighted average MST - is calculated in order to review the performance of the algorithms in approximating the ApproShapley values of customers for a TSP tour and TSP path. These resulting RMSE's can be observed in Tables 4.3 and 4.4. These values are additionally visualised in Figures 4.3 and 4.4.

When comparing the two Figures, it can be observed that the MST and blended MST models tend to perform better with respect to approximating the ApproShapley values for a TSP path. This is expected, since the MST model does not have a closed loop but rather a tree, therefore it represents the TSP path better than a TSP tour. Nonetheless, the models tend to perform generally effectively, with low RMSE values. The egalitarian method - where all customers pay a fraction of $\frac{1}{n}$ the total costs, with n being the number of customers - is also included in the performance overview for fair comparison. It can be seen that for a higher number of customers, the average RMSE is decreasing. This is expected since the costs of a route/path become more evenly distributed with increasing number of customers, resulting in a better approximation for all proxies, including the Egalitarian method. From both Figures 4.3 and 4.4, it is observed that the MST and blended MST models significantly outperform the Shortcut distance algorithms. It is interesting to note that creating a hybrid proxy by combining the MST and Shortcut distance algorithms results in a better performance than the MST model. Although the difference in performance is decreasing for larger number of customers, it is reasonable to expect improved performance by intuitively combining two methods. As was described in Section 3.5, the

MST and Shortcut distance algorithms captured useful properties regarding allocation of costs to customers. These observations hold true when the depot is also a collaborative player. The Shortcut distance algorithm can capture the property of customers being on a line, whereas the MST algorithm captures the property of clustered customers, who are likely to share costs. Combining the two proxies allow for a fairer cost allocation, with improved performance in approximating costs to customers efficiently.

Table 4.3: Average RMSE Scores TSP tour: sizes 10-35, when depot is a player

Instance size	MST	Shortcut distance	Weighted average MST
10	0.030761	0.077348	0.023136
15	0.023730	0.054405	0.031845
20	0.017245	0.046525	0.014521
25	0.014013	0.039326	0.011773
30	0.011637	0.033534	0.010301
35	0.010139	0.028725	0.009017

Table 4.4: Average RMSE Scores TSP path: sizes 10-35, when depot is a player

Instance size	MST	Shortcut distance	Weighted average MST
10	0.024012	0.07947	0.017583
15	0.018585	0.055386	0.013646
20	0.017245	0.046525	0.014521
25	0.011877	0.038988	0.009501
30	0.010107	0.032895	0.008491
35	0.009195	0.027883	0.007589

The runtime of the implemented algorithms in addition to the time taken to approximate the costs of customer via the ApproShapley algorithm for a TSP path and tour can be observed in Figure 4.5. Moreover, a closeup observation of the algorithm runtime for the MST and Shortcut distance algorithms can be seen in Figure 4.6. Upon observation of Figure 4.5, it can be seen that approximating costs for ApproShapley values is increasing immensely for higher number of customers, with the TSP path algorithm utilising a runtime exceeding of one hour for an instance size of 35 customers. Whereas for the TSP tour, the average time taken to allocate costs to customers by using the ApproShapley algorithm is 50 minutes for a single instance. From Figure 4.6, it can be observed that the runtime of the MST and Shortcut distance algorithms are all under a second, thereby allocating

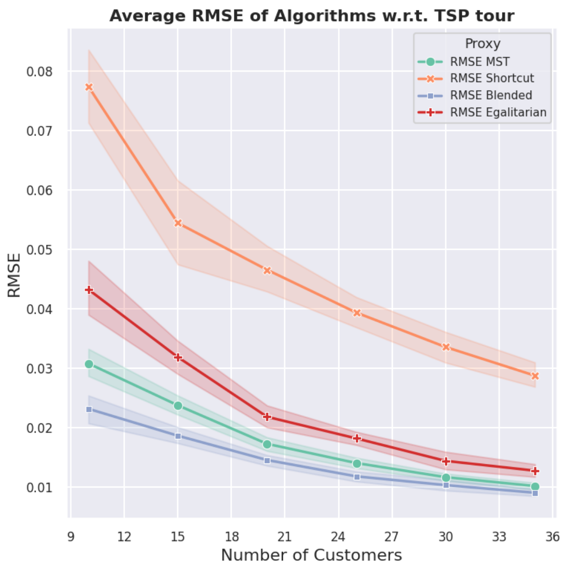


Figure 4.3: TSP tour average RMSE - Depot as a player

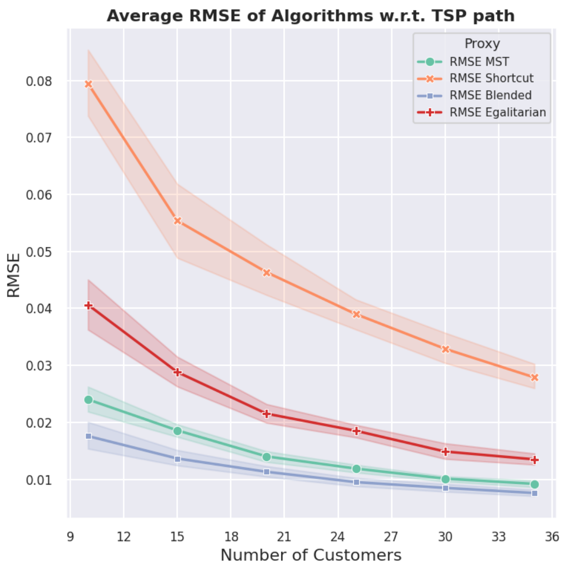


Figure 4.4: TSP path average RMSE - Depot as a player

costs to customers efficiently. The discrepancy in cost allocation runtime for the MST algorithms for instances consisting of 25 customers can be attributed to a faulty run in a particular instance, thereby increasing the runtime. However, in general it can be seen that the runtime of these methods are almost instant. From Figures 4.3 and 4.4, it was also observed that the MST algorithm performs effectively, with RMSE values being less than 0.02 for all customer sizes ranging from 10 to 35. The runtime for the blended MST algorithm is not included, as the calculation of this proxy is instant once the MST and Shortcut distance algorithm results are obtained.

Based on the accuracy/runtime comparison of the algorithms, it can be stated that the MST and blended MST models are both efficient and effective in fair cost allocation to customers, when the depot is a collaborative player. The algorithms result in an average RMSE score less than 0.02 for all customer sizes, and are effectively able to allocate costs in under a second.

4.3.2 Evaluation - Depot not a collaborative player

In the traditional setting when only customers are assigned costs for a route, it is essential that customers can be allocated a cost efficiently. There has been research on fair cost allocation, with researchers such as Aziz et al. (21) and Levinger et al. (23) implementing methods to approximate the Shapley values. In Sections 3.3, 3.4 and 3.6, naive approaches,

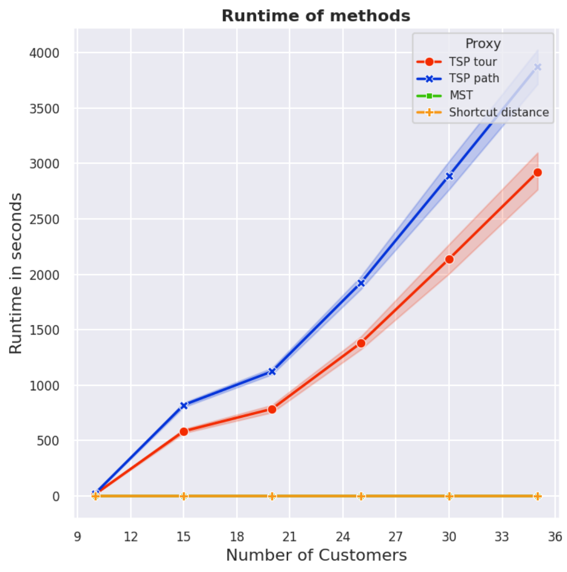


Figure 4.5: Runtime of algorithms when the depot is a player

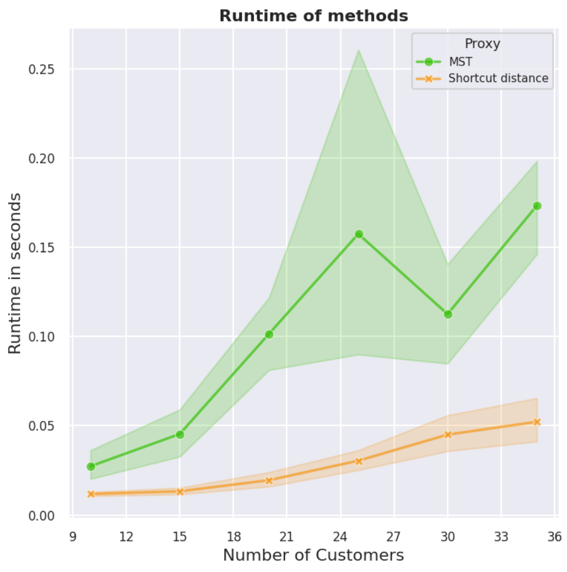


Figure 4.6: Runtime of algorithms when the depot is a player - closeup

as well as novel advanced approaches and blended methods have been introduced and discussed. The naive proxies have been inspired from Aziz et al. (21), whereas novel methods have been introduced by utilising a minimum cost spanning tree to assign costs. Naive and advanced proxies were combined to produce blended proxies. These proxies will be evaluated by utilising benchmark instances for customers of sizes ranging from 10 to 35. Moreover, the SHAPO algorithm recently developed by Levinger et al. (23) will be used to compare the performance of the novel heuristics introduced.

Table 4.5: Average RMSE Scores TSP tour: sizes 10-35

Instance size	Depot distance	Shortcut distance	SHAPO	MST	Weighted average MST	Modified Kruskal tour
10	0.030024	0.089422	0.038147	0.036963	0.028181	0.03125
15	0.023958	0.060865	0.028056	0.028306	0.022031	0.025344
20	0.018293	0.049048	0.02351	0.022296	0.033961	0.019658
25	0.016055	0.040798	0.018471	0.017925	0.015285	0.016393
30	0.013081	0.034841	0.015752	0.015151	0.011935	0.013974
35	0.0125	0.029821	0.014407	0.013604	0.012235	0.012539

The average RMSE scores of instances ranging from size 10 to 35 for approximating the TSP tour and path can be seen in Tables 4.5 and 4.6. These values can be visualised in Figures 4.7 and 4.8. In addition to the proxies implemented, the line charts include the

Table 4.6: Average RMSE Scores TSP path: sizes 10-35

Instance size	Depot distance	Shortcut distance	SHAPO	MST	Weighted average MST	Modified Kruskal tour
10	0.032244	0.089563	0.036601	0.029029	0.031252	0.02507
15	0.024148	0.06068	0.026387	0.021485	0.017748	0.019193
20	0.017095	0.050559	0.020819	0.014681	0.012796	0.01367
25	0.014871	0.041787	0.016943	0.012252	0.010625	0.012027
30	0.012735	0.034902	0.014529	0.010509	0.009208	0.0103
35	0.012053	0.029662	0.013418	0.009719	0.008575	0.009419

performance of egalitarian cost allocation for comparison of scaling. This cost allocation consists of assigning equal costs to every customer, based on the formula $\frac{1}{n}$, where n defines the number of customers in the route. At a general glance at both figures, it can be observed that the average RMSE scores for all algorithms are decreasing for increasing number of customers. This is also the case with a naive method as Egalitarian cost allocation.

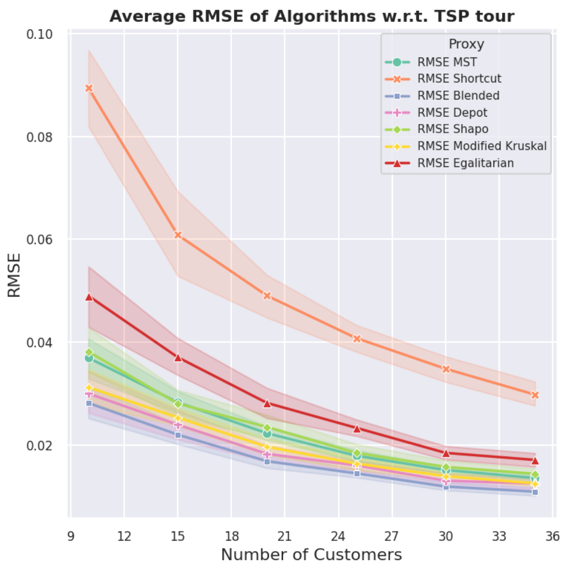


Figure 4.7: TSP tour RMSE

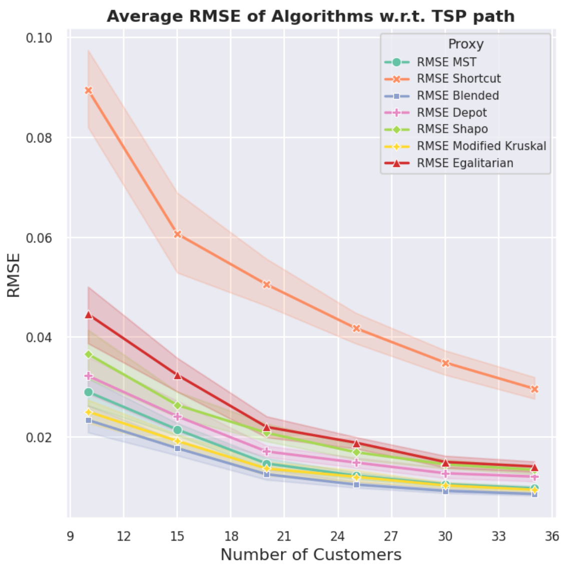


Figure 4.8: TSP path RMSE

When observing Figure 4.7, it can be observed that the blended (weighted average) algorithm tends to outperform all other algorithms for all customer sizes. Although the Modified Kruskal and MST algorithms tend to outperform the Shortcut distance and SHAPO algorithm, it tends to perform slightly poorer than the depot distance algorithm in terms of approximating the TSP tour. This was expected, as the MST and modified Kruskal algorithms are based on a construction of a tree or path. Although the modified Kruskal algorithm incorporates the Depot distance algorithm as a final step to form a route, it tends to not be adequate in approximating the TSP tour. However, it can be seen that combining the MST and depot distance algorithms to result in the blended algorithm yields competent, as it tends to perform better than all algorithms. The properties of the MST algorithm - clustering close neighbours - and the usage of proximity via the depot distance allows for a simple, yet intuitive and effective model performance.

When observing the performance of the algorithms in approximating the TSP path in Figure 4.8,it can be seen that the blended model outperforms all algorithms for all customer sizes once again. In this variant, the blended model consists of the MST and Shortcut distance algorithms. It was discussed in Sections 3.5 and 4.1 that the Shortcut distance algorithm has useful properties, where it can capture customers on a line. This property, in addition to the property of distributing equal costs to nearby customers through the MST algorithm yields an effective performance. In approximating the ApproShapley values for the TSP path, the modified Kruskal algorithm tends to outperform all algorithms except for the blended algorithm. It was expected that this proxy would perform better than the MST algorithm, as this proxy results in a path rather than a tree, and allocates costs to relevant customers during each step. This simple yet intuitive algorithm tends to outperform simple heuristics such as the depot distance and shortcut distance algorithm, in addition to the SHAPO algorithm. In general, it can be observed that blending the MST algorithm with one of the naive proxies yields successful in approximating the allocated costs to customers in a fair and efficient manner.

The runtime of the ApproShapley algorithms for the TSP tour and path, in addition to the runtime of the implemented proxies in seconds can be observed in Figures 4.9 and 4.10. Figure 4.10 is an evaluation of the proxies' runtimes in seconds for increasing number of customer sizes. In this Figure, the shortest, longest and average runtime across 20 instances can be seen. As a general remark, it can be seen from Figure 4.9 that the ApproShapley algorithm of providing a baseline approximation for the costs allocated in a TSP tour and path requires excessive computational time, even for small number of customer sizes such as 35. In Section 4.2, the performance of the ApproShapley algorithm with respect to the true Shapley values were evaluated in terms of speed and accuracy. Although for smaller problem instances the ApproShapley is promising and effective in approximating the true Shapley values, for increasing problem sizes it can be seen that the computational time increases substantially. This observation hence emphasises the need for heuristics which can allocate costs to customers in an efficient manner. In Figure 4.10, it can be observed that for all customer sizes and algorithms, the costs are allocated to customers in under a second. The runtime of the blended MST algorithm is not included here, as the approximations for this proxy can be maintained once the two proxies - MST and either the Depot distance, or Shortcut distance algorithms - are calculated. It can be seen that the depot distance algorithm requires the least amount of computational time, whereas the Shortcut distance algorithm requires the most time. The Shortcut distance algorithm requires solving the TSP route once, hence the higher computational time with respect to the other algorithms. The SHAPO algorithm has a low computational time with under 0.02 seconds. The MST and modified Kruskal algorithms require a relatively stable amount of computational time of under 0.01 seconds for all problem sizes.

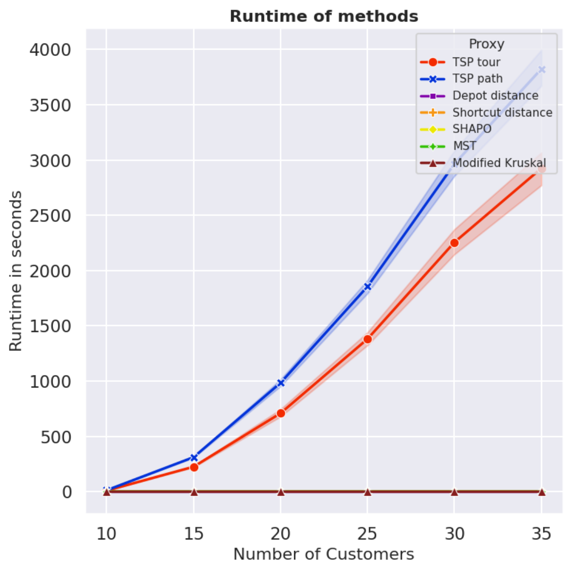


Figure 4.9: Runtime of algorithms when the depot is not a player

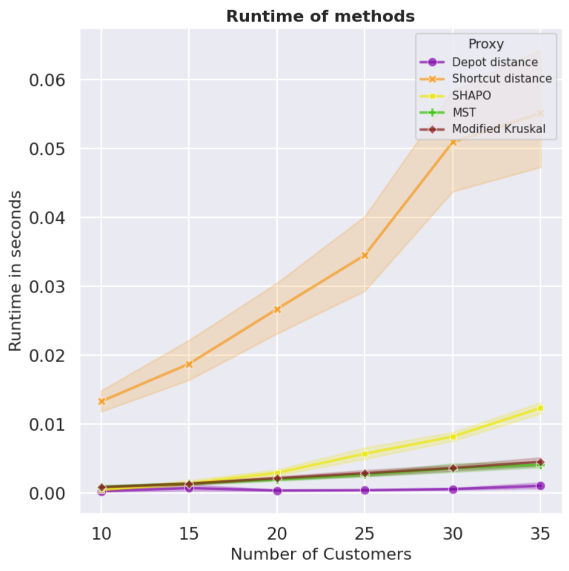


Figure 4.10: Runtime of algorithms when the depot is not a player - closeup

When analysing the performance of the implemented proxies in terms of both speed and accuracy, it can be suggested that the modified Kruskal, MST and blended MST models outperform the other proxies. The Depot distance algorithm has low computational time and a low average RMSE score when comparing the performance with the normalised ApproShapley values for the TSP tour and path. However, this performance is surpassed by the modified Kruskal and blended MST algorithms. The SHAPO algorithm also performs well in terms of speed and accuracy, however this algorithm is also surpassed by the MST algorithm in terms of approximating the allocated costs to customers. The MST, modified Kruskal and blended (weighted average) methods can be regarded as the leading algorithms, as they are effective in allocating costs to customers in a fair manner, while ensuring that these costs are allocated in under half a second.

4.3.2.1 Evaluation of relative Shapley values

The results obtained and discussed in Sections 4.3.1 and 4.3.2 are evaluated based on normalised Shapley values. The values of the ApproShapley algorithm, in addition to the proxies implemented are scaled to be between 0 and 1, which translate into the fraction of the route cost which should be allocated to each customer. All proxies were normalised for a fair comparison and assessment of performance. However, it is also essential for relevant stakeholders to know not just the proportion of costs that should be allocated to customers, but rather the actual costs. Therefore, this section will analyse the relative Shapley values for two instances - one consisting of 20 customers, and the other consisting of 30 customers. These instances are utilised from the synthetic Euclidean games provided in the literature of Aziz et al. (21). For each instance, the total cost that should be allocated for each customer is determined via the ApproShapley algorithm for the TSP path and tour. Furthermore, the costs that are allocated to each customer via the proxies mentioned in Section 4.3.2 will be provided, and their performance in accurately estimating the Shapley values will be compared by observing the root mean squared error (RMSE). Moreover, the best and worst case approximations of each algorithm will be analysed, along with the average difference in the true versus approximated costs.

Relative Shapley values - Instance with 20 customers A synthetic Euclidean TSP game is utilised from Aziz et al. (21), consisting of 20 customers. The optimal route of the problem instance can be visualised in Figure 4.11. The total cost of this route is determined to be approximately 3914, by utilising the Pyconcorde solver. The relative costs are obtained for each algorithm by solving each proxy, without scaling the costs in order to review the total cost assigned to each customer, rather than the percentage of costs assigned. In order to provide an illustrative example, Table 4.7 displays a sample of 5 rows from the full data. This subset is representative of the overall structure of the total cost assigned to each customer for each algorithm.

The performance of each algorithm is compared by calculating the RMSE score between the TSP tour and the algorithms. These RMSE scores can be observed in Figure 4.12. Similarly, the performance of the algorithms in fair and accurate cost allocation with respect to the TSP path can be observed in Figure 4.13. These RMSE scores are used to measure the accuracy of the heuristics implemented in approximating the true Shapley

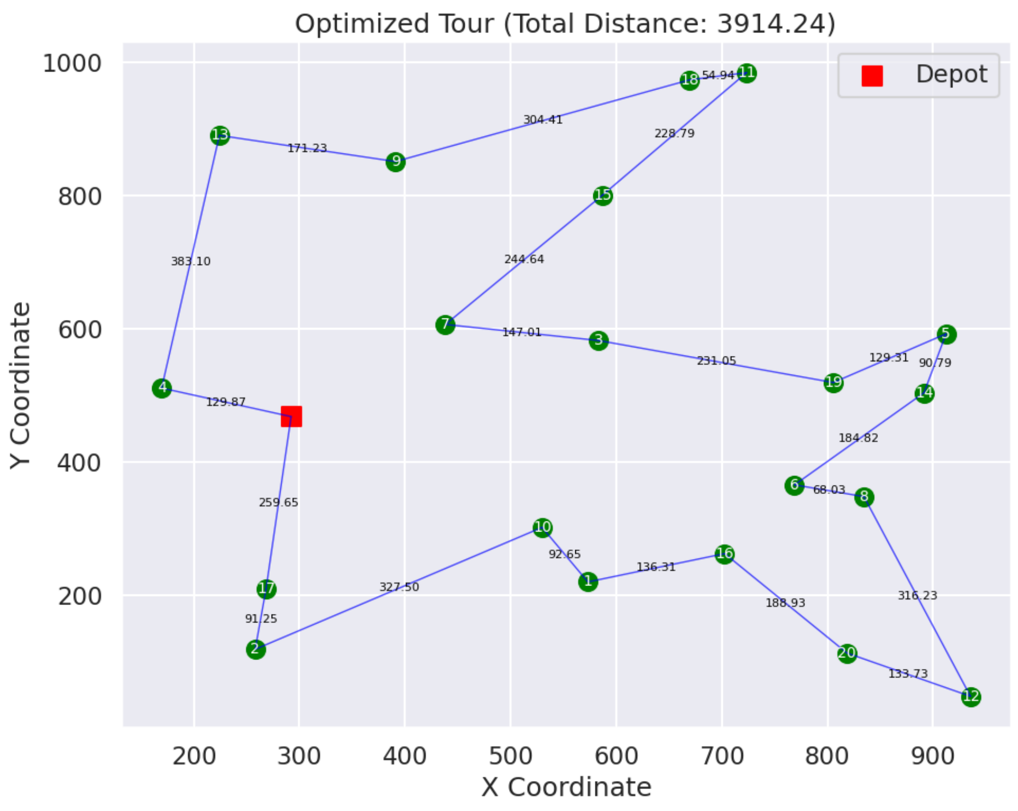


Figure 4.11: Optimal tour for an instance consisting of 20 customers

Table 4.7: Subset of data showing total costs assigned to customer per algorithm

Customer	TSP Path	TSP Tour	MST	Depot distance	Shortcut distance	SHAPO	Modified Kruskal
1	120	102	126	347	51	388	179
2	236	274	175	350	141	519	226
3	189	157	174	312	0.56	363	206
4	143	158	129	129	85	559	143
5	199	226	121	632	132	520	202
Total	**3532**	**3915**	**2977**	**9144**	**1369**	**11524**	**4604**

values for cost allocation to customers. Lower RMSE values indicate a closer approximation to the Shapley values. When observing the RMSE scores for approximating the costs for the TSP tour, it can be observed that the Modified Kruskal algorithm achieved the lowest score of 73.7, indicating the highest accuracy in approximating the Shapley values. This suggests that in this instance, the modified Kruskal heuristic effectively captures the distribution of costs to customers in a fair manner. The MST algorithm follows with the second heuristic with the lowest accuracy. In contrast, the Depot Distance and SHAPO algorithms yielded higher RMSE values of 294.6 and 405.8, respectively. In estimating the costs for the TSP tour, the weighted average (blended) MST algorithm also results in higher estimation errors.

When observing the RMSE scores of the algorithms for approximating the TSP path in Figure 4.13, it can be observed that the MST, weighted average MST and Modified Kruskal algorithms yield the lowest error, with an error of 64, 65 and 76, respectively. The Shortcut distance algorithm yields an average RMSE score of 123, whereas the Depot Distance and SHAPO algorithms result in the highest error values once again.

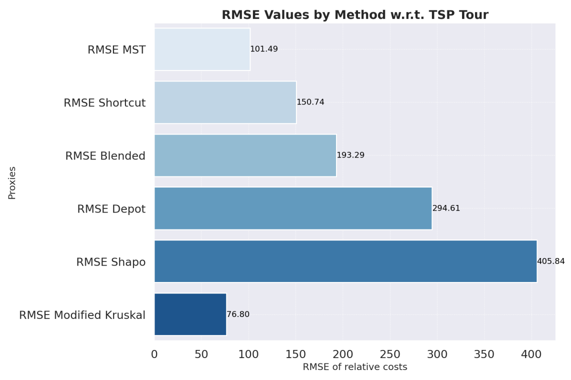


Figure 4.12: TSP tour RMSE

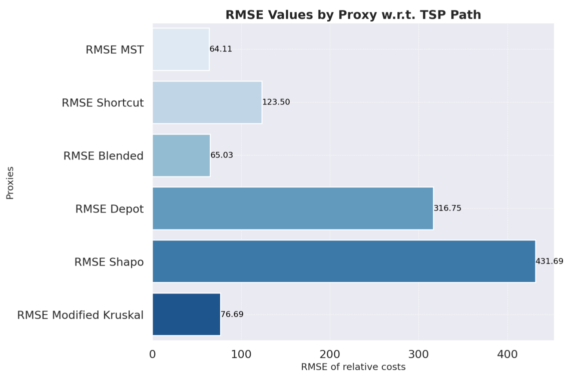


Figure 4.13: TSP path RMSE

In addition to the RMSE values, the approximation errors of the algorithms are compared based on their best-case, worst-case and average absolute differences. This comparison is visualised in Figures 4.14 with respect to the TSP tour and 4.15 with respect to the TSP path.

The bar charts are obtained by calculating the difference in costs allocated between the ApproShapley values for the TSP/tour and the proxies. The best case scenario is considered to be where the difference between the ApproShapley value and a proxy is smallest. The worst case scenario is when the difference between the two is highest. The mean absolute difference is considered to be the average absolute difference between the ApproShapley value and proxy, also known as the Mean Absolute Error (MAE).

Observing the results for the performance of proxies in approximating the ApproShapley values for the TSP tour for best-case, worst-case, average and mean absolute difference scenarios in Figure 4.14, it can be noticed that the modified Kruskal algorithm has the lowest mean absolute difference, with a value of 66. In the best-case scenario, this proxy

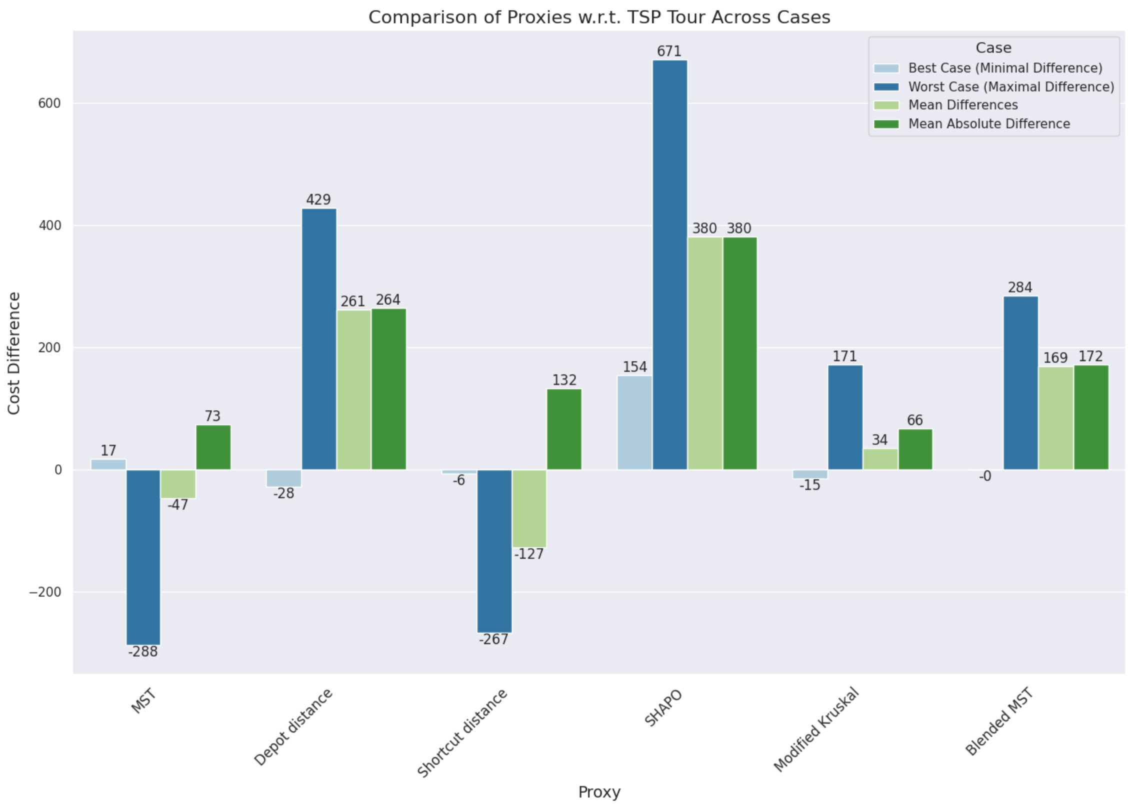


Figure 4.14: Best, worst and average difference cases between TSP tour and proxies for size 20 instance

underestimates the costs to be allocated to a customer by -15, whereas in the worst-case scenario, the deviation between the ApproShapley TSP tour and modified Kruskal is 171. This can be interpreted as follows: the proxy overestimates the cost should be paid by a customer by 171. Although this can be considered as a substantial difference, it can be seen that this proxy performs well with a relatively low average difference. Moreover, this proxy has the lowest worst-case scenario value with respect to the other proxies: the SHAPO algorithm overestimates the cost of a particular customer with 671 in the worst-case scenario, whereas the MST algorithm underestimates the cost by -288 in the worst-case scenario. In short, the modified Kruskal algorithm does not deviate excessively from the ApproShapley values in the worst-case scenario when compared with the other proxies.

When observing the values in Figure 4.15, it can be seen that the MST, Blended MST and modified Kruskal proxies have the lowest mean absolute difference, with values of 46, 47 and 58, respectively. In the best-case scenario, the blended MST proxy has a deviation of less than 1 from the ApproShapley TSP path values, whereas in the worst-case scenario, a deviation of -177 was observed, indicating that the proxy underestimated the cost of

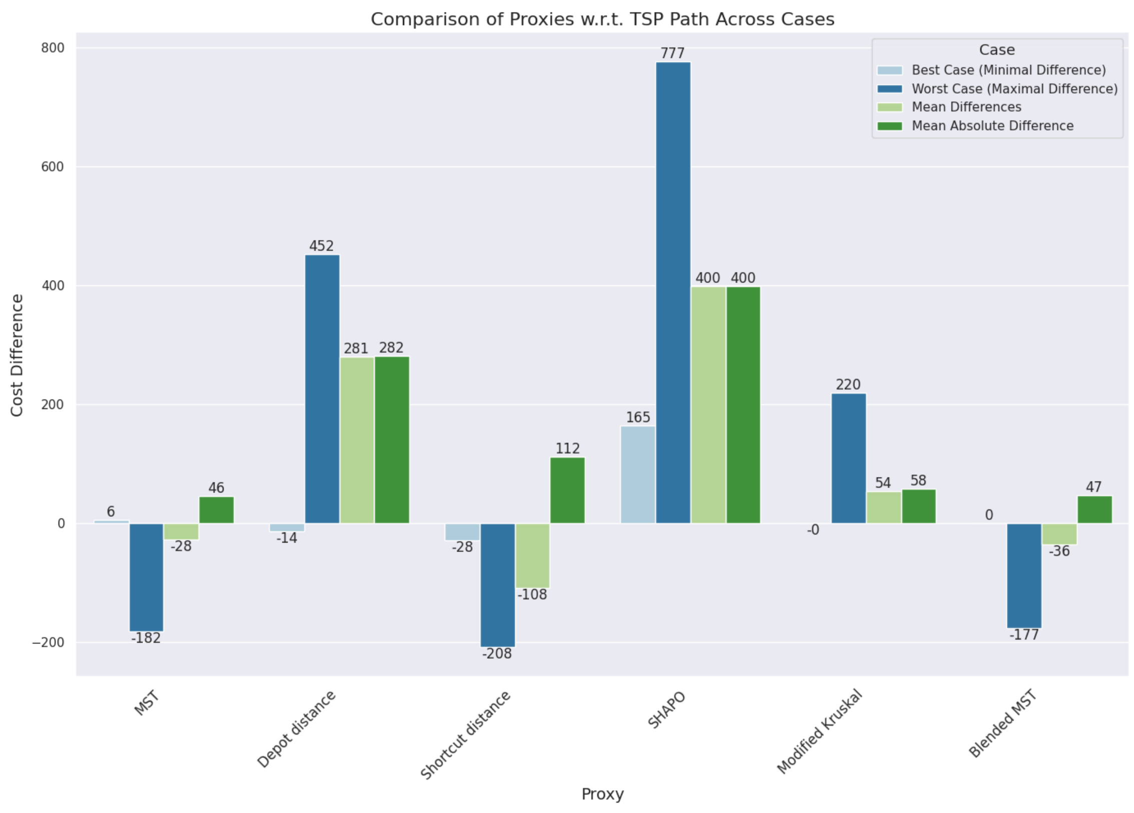


Figure 4.15: Best, worst and average difference cases between TSP path and proxies for size 20 instance

the customer by 177 units of currency. The Depot distance and SHAPO algorithms are observed to be the proxies with the highest mean absolute error, with an discrepancy of 281 and 400, respectively.

Relative Shapley values - Instance with 30 customers To assess the performance of the algorithms in approximating the relative Shapley values, a moderately higher instance is considered, consisting of 30 customers. Similarly to the steps taken as with the 20 customer instance, costs are allocated to customers based on each algorithm, and are compared against the true costs assigned based on the ApproShapley values for the TSP tour and path. The RMSE scores showing the errors between the algorithms and TSP tour/path can be observed in Figures 4.16 and 4.17.

With Figure 4.16, it can be seen that once again, the Modified Kruskal algorithm yields the lowest error, with a value of 73. The MST algorithm tends to perform proficiently, with an RMSE score of 89. The Depot distance and SHAPO algorithms result with the highest costs, with error values of 503 and 435, respectively. For Figure 4.17, the MST,

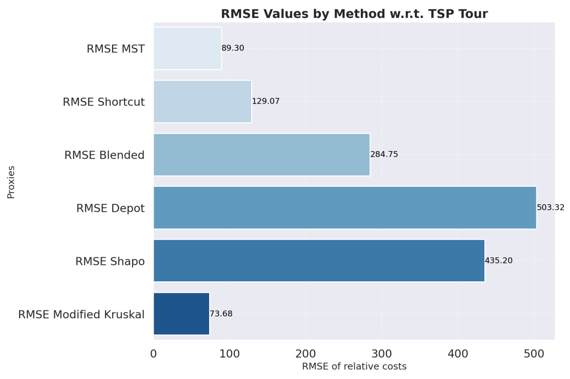


Figure 4.16: TSP tour RMSE

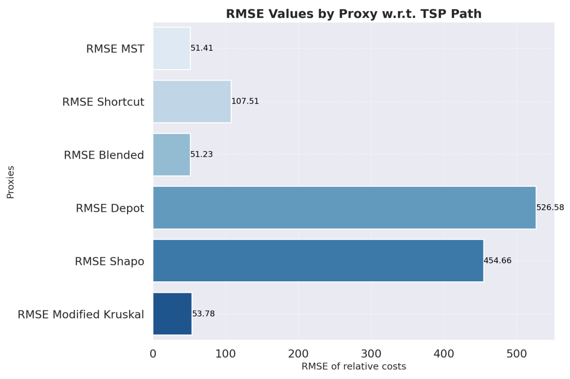


Figure 4.17: TSP path RMSE

Blended MST and modified Kruskal algorithms tend to outperform the other algorithms from literature. The MST and weighted average MST algorithms share an equal RMSE score of 51, whereas the modified Kruskal algorithm yields a slightly higher error of 53. The Depot distance and SHAPO algorithms yield the highest approximation error of the TSP path with a cost error amounting to more than 400.

Figures 4.18 and 4.19 once visualise the best-case, worst-case and average difference performance of the proxies with respect to the ApproShapley values for the TSP tour and TSP path, respectively.

Upon first glance to Figure 4.18, it can be seen that the Modified Kruskal algorithm tends to perform better than the other proxies, with the best case scenario having a deviation from the ApproShapley TSP tour value of 0.67. In the worst-case scenario, the modified Kruskal algorithm underestimates the true ApproShapley values with -227. This occurs for a customer who should pay a cost of 636, but is assigned a cost of 409 by using the Modified Kruskal algorithm. The average absolute difference between this proxy and the ApproShapley TSP tour is 55. This means that on average, the approximated costs for customers by using the Modified Kruskal algorithm deviate from the ApproShapley values by 55 units of cost. The total cost allocated to all customers in this instance according to Modified Kruskal is 5207. Therefore, an average deviation of 55 units can be considered an inexpensive amount, as the average deviation from the ApproShapley costs is only 1% of the total cost. The Depot distance algorithm tends to have the highest average difference with respect to the ApproShapley values, thereby having an ineffective cost allocation to customers.

When observing Figure 4.19, it can be seen that the MST, modified Kruskal and blended MST algorithms have the lowest average absolute difference with respect to the ApproShap-

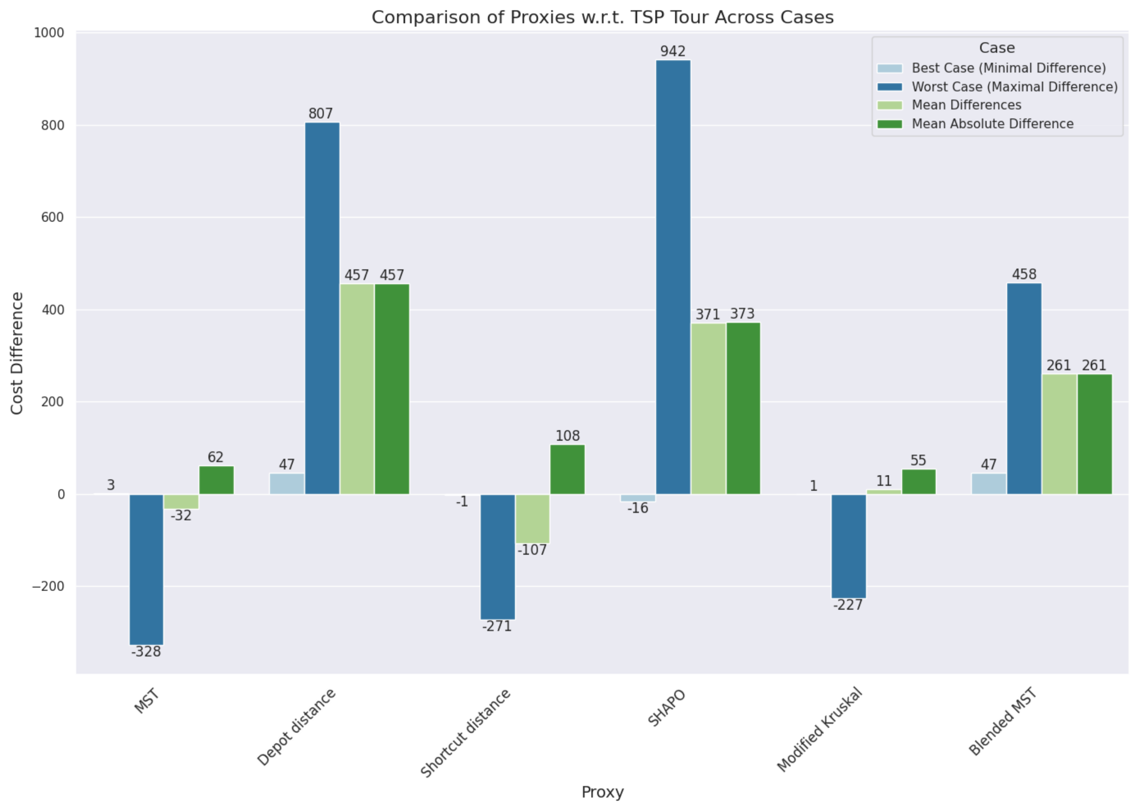


Figure 4.18: Best, worst and average difference cases between TSP tour and proxies for size 30 instance

ley values for the TSP path. In the best-case scenario, the MST algorithm is able to accurately allocate the costs to a customer, whereas in the worst-case, it tends to underestimate the cost to be allocated with a value of -173. In such a scenario, the MST algorithms assigns a cost of 308 to a customer, where the ApproShapley algorithm assigns a cost of 481. The MST algorithm assigns a total cost of 3907 in this instance. Therefore, the worst-case approximation deviates from the ApproShapley values with an error of 4% of the total cost, which can be considered low.

Before finalising the evaluation of the implemented proxies in terms of the ApproShapley values for the TSP tour and path when the depot is not a collaborative player, a few concluding remarks will be made. As it can be seen from Figures 4.7 and 4.8, the Depot distance and SHAPO algorithms tended to perform well when comparing the performance of the algorithms with the ApproShapley TSP tour and path in terms of their normalised values. However, in Section 4.3.2.1, it was seen that when the total costs of customers were compared with those of the ApproShapley values rather than the normalised costs, these algorithms performed poorly, with high RMSE values. Moreover, these algorithms

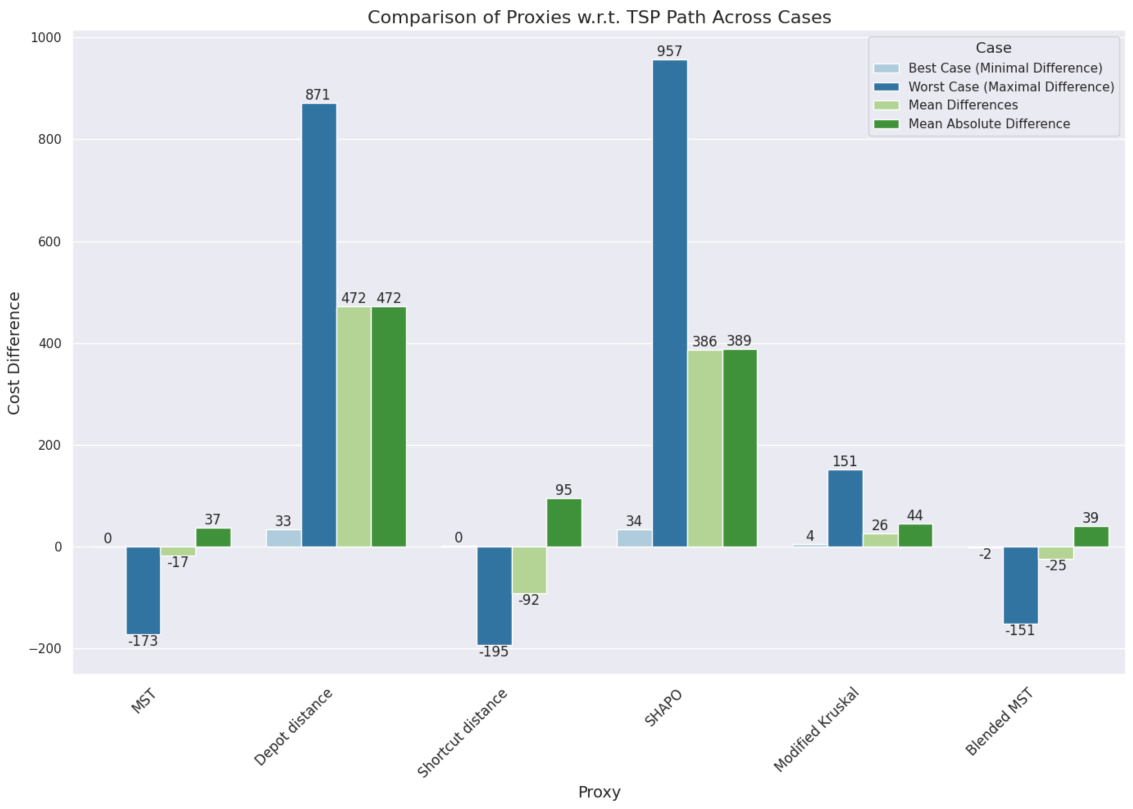


Figure 4.19: Best, worst and average difference cases between TSP path and proxies for size 30 instance

tend to overestimate the costs of customers substantially, with high mean absolute errors. The MST and modified Kruskal algorithms were observed to be effective not only in approximating the normalised ApproShapley values, but also the global values. Therefore, these algorithms are not only efficient in determining what fraction of the total cost a customer should receive, but also the total cost they should receive. Therefore, by experimenting on benchmark instances by Aziz et al. (21), it can be concluded that based on a accuracy/speed performance, the MST, modified Kruskal and blended MST algorithms are effective when compared with other heuristics in literature.

4.3.2.2 Evaluation of machine learning models

In Section 4.3.2, the performance of the implemented algorithms introduced in Sections 3.3 and 3.4 were discussed for cases when the depot is not a collaborative player, and does not burden the cost of paying for a route. These algorithms were also compared with the weighted average model introduced in Section 3.6.5. The current section will evaluate the performance of the decision tree, random forest, linear regression and linear tree models on predicting the ApproShapley values for the TSP tour and TSP path. The features used for the implemented machine learning models consist of the naive and advanced proxies introduced in the Methodology Section.

The input data used for training and testing consists of costs allocated to customers in instances consisting of 10, 20 and 30 customers from the synthetic Euclidean games by Aziz et al. (21) - these instances were utilised to assess the performance of the naive and advanced proxies. For each customer size, 15 of the 20 instances were utilised for training, and the remaining 5 were used for testing. Each instance consists of the cost allocated to each customer in the route based on the proxies. As a result, the training data consisted of 900 data points, whereas the test set consisted of 300 data points.

In total, four machine learning variants were implemented:

- utilising normalised costs (stating the fraction of costs to be paid by a customer), to predict the ApproShapley value for a TSP tour
- normalised costs to predict the ApproShapley for a TSP path
- utilising global costs (stating the total costs to be paid by a customer), to predict the ApproShapley for a TSP tour
- global costs to predict the ApproShapley for a TSP path.

For each machine learning variant, the costs from the depot distance and shortcut distance algorithms were utilised, as well the advanced proxies of the MST and modified Kruskal algorithms as input features. In addition, the number of degrees (edges connected) of each customer from the resulting MST tree in the MST algorithm were utilised as a feature. Through trial and error and performance assessment of the training data, the features utilised for each machine learning variant can be observed in Table 4.8. All ML model variants utilise the Depot distance, Shortcut distance and modified Kruskal algorithms. To predict the global/total ApproShapley values of the TSP path, the MST algorithm is additionally utilised.

In addition to the test set, all model variants are assessed by predicting the values of the ApproShapley values for instances consisting of 15 and 35 customers. Instances consisting of size 15 are utilised to observe how each model performs in predicting the cost of a customer when the model was not trained on this size, but is within the range of customer sizes trained upon. Instances consisting of size 35 are utilised to assess how accurate the models are in predicting the cost of customers when the customer size is larger than

Table 4.8: Features utilised for training ML models

ML model to predict normalised ApproShapley		**ML model to predict total ApproShapley**	
TSP tour	**TSP path**	**TSP tour**	**TSP path**
Depot distance	Depot distance	Depot distance	Depot distance
Shortcut distance	Shortcut distance	Shortcut distance	Shortcut distance
Modified Kruskal	Modified Kruskal	Modified Kruskal	Modified Kruskal
			MST

those the model has trained upon. Therefore, this customer size was utilised to analyse the scalability of the machine learning models. This is essential as the cost of customers become more evenly distributed as the number of customers in a route increases.

Test Set - Customer Sizes		**Predicting TSP tour**		**Predicting TSP path**	
	ML Model	**RMSE**	**R^2 Score**	**RMSE**	**R^2 Score**
10 + 20 + 30	DT	0.01505	84.9%	0.01275	86.49%
	RF	0.0134	88.05%	0.01134	89.33%
	LR	**0.0116**	**91.01%**	0.01011	91.51%
	LT	0.01194	90.53%	**0.00987**	**91.91%**
15	DT	0.01797	77.49%	0.01719	73.29%
	RF	0.01619	81.73%	0.0136	83.31%
	LR	**0.01426**	**85.84%**	**0.01199**	**87%**
	LT	0.01543	83.41%	0.01283	85.13%
35	DT	0.0096	69.06%	0.00789	69.46%
	RF	0.00884	74.09%	0.0069	76.14%
	LR	**0.0085**	**75.90%**	**0.00641**	**79.79%**
	LT	0.0087	74.69%	0.0066	78.4%

Table 4.9: Prediction results of TSP tour and TSP path for normalised costs

Table 4.9 shows the resulting RMSE and R^2 scores of the machine learning models implemented to predict the normalised ApproShapley values for the TSP tour and path. As mentioned earlier, the ML models were trained using the normalised allocated costs from instances with customer sizes of 10, 20 and 30 combined, and evaluated on the test set consisting of instances with the same customer sizes. Moreover, the performance of the models were evaluated on instances consisting of 15 and 35 customers, without having been trained on them.

When analysing the performance of the ML models on the test set consisting of 10, 20 and 30 customers combined, it can be observed that the linear regression model tends to outperform the other models when predicting the cost of customers in a TSP tour, with an R^2 score of 91%. Whereas for predicting the normalised cost of customers for a TSP path, the linear tree model tends to perform slightly better, with an accuracy of 91.91%.

When analysing the performance of the models on instances which the models were not trained on, it can be seen that for instances consisting of 15 customers, the linear regression models tend to perform relatively accurately, with an average RMSE score of 0.014 and 0.011 in predicting the ApproShapley values for the TSP tour and path, respectively. However, it can be observed that for instances consisting of 35 customers, the accuracy of all models are lower compared to other instances. This can be justified to the difficulty of scaling the distribution of costs among customers when the problem instance consists of larger number of customers. Nevertheless, the linear regression models tend to perform well, with an R^2 score of 75.9% and 79.8% for predicting the normalised ApproShapley values for the TSP tour and TSP path, respectively.

Table 4.10 shows the resulting RMSE and R^2 scores of the machine learning models implemented to predict the relative/global ApproShapley values for the TSP tour and path. The relative/global values denote how much cost a customer should be paying, rather than the fraction of cost. When observing the performance of the models on the test set consisting of 10, 20 and 30 customers combined, it can be seen that the Random Forest model outperforms the other models for predicting the ApproShapley values for the TSP tour, with an R^2 score of 79.7%. The Linear Regression and Linear Tree models perform similarly well, with R^2 scores of 78.57% and 78.11%, respectively. Whereas for predicting the ApproShapley values for the TSP path, the Linear Regression model results in the highest performance, with an R^2 score of 35.5. This can be interpreted as following: the model predictions deviate on average by a cost of 35 from the true costs allocated to customers.

When observing the performance of the models on instances consisting of 15 customers, it can be seen that for both the TSP tour and TSP path, the Linear Regression model outperforms the other models, with an R^2 of 80% and 85.3%, respectively. It can be seen that the Linear Regression models performs better on data it has not seen before. This can be attributed since 15 is within the range of customer sizes the model was trained on. Therefore, the model is capable of scaling well for predicting allocated costs to customers within the same range of the training set. When analysing the results for instances consisting of 35 customers, it can be seen that the model performance drops, with an R^2 score

Test Set - Customer Sizes		**Predicting TSP tour**		**Predicting TSP path**	
	ML Model	**RMSE**	**R^2 Score**	**RMSE**	**R^2 Score**
10 + 20 + 30	DT	63.10	73.8%	42.46	75.39%
	RF	**55.5**	**79.7%**	36.99	81.32%
	LR	57.1	78.57%	**35.53**	**82.76%**
	LT	57.72	78.11%	37.67	80.63%
15	DT	72.97	69.61%	47.05	77.35%
	RF	61.22	78.61%	40.20	83.47%
	LR	**59.14**	**80.00%**	**37.87**	**85.33%**
	LT	59.71	79.6%	38.70	84.68%
35	DT	55.42	57.02%	35.59	69.12%
	RF	48.03	67.72%	30.47	77.37%
	LR	**46.45**	**69.80%**	**27.78**	**81.19%**
	LT	50.77	63.93%	28.89	79.66%

Table 4.10: Prediction results of TSP tour and TSP path for global costs

of 69.8% for the Linear Regression model in predicting the ApproShapley values for the TSP tour. However, this model is capable of obtaining a good performance in predicting the TSP path, with an R^2 score of 81.19%.

Despite obtaining intuitive and interpretable results from the Linear Regression models in predicting the total ApproShapley values for a TSP tour and path, obtaining insight onto the steps and coefficients can be insightful. In particular, identifying sub-populations from the data based on different patterns can allow more appropriate prioritisation for different proxies. In order to do this, observing Linear Tree models can be beneficial. As described in Section 3.6.4, Linear Tree models fit a linear model on each leaf node of the resulting decision tree. Consequently, each leaf node has different weights assigned to the features utilised for training the model.

For the linear tree model to predict the total ApproShapley values for the TSP tour, the resulting decision tree from training the dataset consisting of instances consisting of 10, 20 and 30 customers combined can be observed in Figure 4.20. The decision rules can be followed by observing the decision tree, with each leaf node representing a separate subset of data to construct a linear model.

For each leaf node, a linear model is fitted. The efficiency of the Linear Tree model derives from the possibility of obtaining the decision rules and corresponding coefficients for each node. Figure 4.21 illustrates the weight given to each feature based on the decision rules and hence leaf node. By querying the linear model, the impact of each variable in the leaves can be seen (33).

From this Figure it can be observed that based on different rules allocated by the decision tree, the features utilised for training - consisting of the naive and advanced proxies - are assigned a weight. Observing the coefficients for leaf node 10, it can be seen that the Shortcut distance receives a high weight, whereas the Modified Kruskal and Depot Distance algorithms are assigned a weight close to 0.

Figure 4.22 displays the coefficients and decision rules for leaf node 10. When observing the decision rules for this node, it can be seen that the Shortcut distance algorithm gets a higher coefficient and hence weight when these values are observed to be relatively small. Moreover, the Depot distance algorithm is observed to be getting a negative coefficient, although this weight is close to 0. From the decision rule it can be seen that the Depot distance algorithm should be assigned this weight whenever it is observed to assign a cost of more than 838 to a customer. From these observations and the patterns observed in Section 3.5, it can be suggested that based on the decision rules for leaf node 10, customers who fall in this leaf tend to be distant from the depot - due to the relatively high costs - and have a small Shortcut distance value. This small value suggests that these customers are on a line, and hence considered to be on the route of the vehicle within the optimal route. Therefore, according to the Shortcut distance algorithm, these customers should not be assigned high costs. However, it was also observed in Section 3.5 that whenever a customer is considered on the line, the Shortcut distance algorithm tends to deviate substantially from the Shapley or ApproShapley values. During training, the Linear Tree model was efficient in observing patterns relating to these observations. Consequently, the linear model does not assign a lot of weight to the Depot distance algorithm - as it tends to overestimate the cost a customer should be assigned when they are far from the depot - and assigns a higher weight to the Shortcut distance algorithm - which tends to substantially underestimate the cost of a customer when they are considered to be on the route.

Utilising the Linear Tree model comes with various advantages, as it allows for high accuracy, as well as a structured overview of the decision path followed by the model training (33). This allows for high interpretability and intuition on the importance of features depending on different scenarios. However, from Tables 4.9 and 4.10 it was observed that in majority of the cases, the Linear Regression models outperformed the Linear Tree models. The marginally lower performance of the Linear Tree model can be attributed to the relatively small size of the training dataset, which consisted of 900 instances. Consequently, the model can be overfitting and learning the patterns of the training data, while failing to capture different patterns on the test sets. It can be considered to observe the performance of the model on bigger training sets to evaluate whether the performance would be better than the other ML models.

As a final remark on the performance of all models on predicting the total ApproShapley values, the following conclusion can be made. Even though the models were trained on instances consisting of 10, 20 and 30 customers, it can be seen based on the model performance for 35 customers that especially the Linear Regression model generalises well for customer sizes outside of the range the model was trained on. This is expected of a linear regression model, as it learns to give weights to certain features during training, which can tend to generalise well in different customer size settings.

4.4 Case study - The Driving Force

In transportation and logistics, case studies provide an opportunity to explore the real-world applications of the algorithms implemented. Section 4.3 described the results of implemented heuristics on benchmark instances consisting of synthetic data. Conducting a case study allows to shed light on the performance of these algorithms in realistic settings. Therefore, this section will discuss a case study on data from The Driving Force, a company focused on digitisation and helping farmers with delivery routes.

The Driving Force is the creator of the BuyFresh platform, who allows small and medium scale food producers to sell their products directly to the market. BuyFresh has various clients such as farmers, who have customers to deliver to. These customers tend to be generally known, therefore the company tends to have previous knowledge on which customers they will need to serve for a given day. In such cases, it is vital for them to know based on a determined set of customers, how much cost should be allocated to them. Such a problem can be solved using the algorithms implemented: MST, Modified Kruskal and Weighted Average models.

The dataset from the Driving Force consists of a company with a set of 90 customers. These customers are located within and around Noord Holland, the Netherlands. For the set of 90 customers, the ApproShapley values are computed for the TSP tour, in order to obtain an estimate for the costs to be allocated to each customer. In earlier experiments in Section 4.3, distances between customers were calculated by using Euclidean distances. In this case study, the distance between customers are evaluated using 'over the road' distances, which provide the distances between customers based on the road networks available. These distances between customers are obtained through Project OSRM (34). The resulting distances between customers are provided as durations, however for this case study these durations will directly translate into the travel costs. The distance matrix provided is not symmetric - signifying that the cost from location A to B may be different than the cost from B to A. For the current case study, it will be assumed that all distances between customers and the depot is symmetric, therefore the distance matrix is symmetrised by taking the average between the distances of all customers A to B and B to A. Moreover, the TSP solver Concorde assumes that distances are symmetric, which is necessary to obtain the ApproShapley values. The allocated costs for customers are calculated also by utilising the proxies MST, Modified Kruskal, Depot distance, and Shortcut distance. The SHAPO algorithm by Levinger et al. (23) is additionally included for an evaluation of the performances of the proxies in approximating the ApproShapley values for the TSP tour.

Figure 4.23 illustrates the customers of the company on a map, in addition to the depot, labelled as a star. Each customer is denoted with varying sizes and colours. The total cost of the route consisting of 90 customers based on their symmetric distances is obtained to be 13376. The size of the customer location denotes the size of the allocated cost to a customer based on the ApproShapley values for the TSP tour - expensive customers have a bigger size on the map, whereas cheaper customers have a smaller size. From Figure 4.23, it can be seen that customers which are clustered together and closer to the depot have smaller sizes, and hence costs. Isolated customers who are distant from the depot are larger in size, and hence are more expensive.

The colour of the customer location denotes the costs that are allocated to customers based on the MST algorithm - customers with smaller costs are green, whereas with increasing allocated costs, the colour range changes between green, yellow, orange and red. From the map, it can be seen that customers close to the depot tend to be green. Moreover, clustered customers are also brighter in colour. Isolated customers tend to be darker in colour, with a customer assigned the colour orange on the east of the map, and a customer assigned the colour red on the north.

When analysing both the colour and size of the customer locations, it can be seen that the MST algorithm consistently captures the costs allocated to customers with respect to the ApproShapley values - cheaper customers are not only smaller in size, but also greener. The opposite also holds true: more expensive customers are not only larger in size, but also darker in colour. Therefore, from this Figure it can be suggested that the MST algorithm is effective and efficient in capturing the ratio of allocated costs to customers, even with bigger customer sizes, and with a different distance metric.

In order to assess and compare the performance of the implemented algorithms in allocating costs to customers with respect to the ApproShapley values for the TSP tour and TSP path, the RMSE score is evaluated, for both the normalised costs, and the global/total costs. The weighted average model is included, with ratios $0.8 \cdot MST + 0.2 \cdot \textit{Depot distance}$ and $0.8 \cdot MST + 0.2 \cdot \textit{Shortcut distance}$ used for the TSP tour and TSP path, respectively.

Figures 4.24 and 4.25 illustrate the performance of the algorithms with respect to the normalised costs for the ApproShapley tour and path in terms of their RMSE scores. From here, it can be observed that the MST, Modified Kruskal and Weighted Average models have the lowest error, with values ranging between 0.003 and 0.004. These models perform much better in comparison to the Shortcut distance, Depot distance and SHAP0 algorithms. It is especially interesting to see that the performance of the Depot distance algorithm has declined in comparison to the performance observed in Section 4.3.2, where this algorithm obtained a comparable performance with respect to the MST algorithm.

Figures 4.26 and4.27 illustrate the performance of the algorithms with respect to the total costs for the ApproShapley tour and path. From these figures it can be observed that the MST, Modified Kruskal and Weighted Average models outperform the other algorithms once again, with an average deviation from true costs with less than 100. The MST and Modified Kruskal algorithm obtain total costs of 11353 and 14648, respectively. Therefore, an average deviation of 100 units of cost is less than 1% of the total cost amount. Therefore, from this it can be suggested that these novel approaches are efficient in allocating costs to customers with relatively small deviations.

From the analysis of RMSE scores of the algorithms with respect to normalised and total ApproShapley values for the TSP tour and path, it can be suggested that the novel approaches introduced in this research are not only efficient in allocating costs to customers in a fair manner, but are also effective. These conclusions hold true not only in synthetic Euclidean games, but also in a case study where data from the Driving Force is utilised, consisting of distances obtained from road networks. These algorithms are effective in approximating the normalised costs to customers - what fraction of the total cost should be allocated to whom - as well as approximating the total costs.

4.5 Robustness

In order to consider the robustness of the MST algorithm, a scenario will be considered where all locations including the depot are collinear. This signifies that all customers are on the same line - this can be a plausible scenario if customers within a route lie within the same street. Consider the scenario as illustrated in Figure 4.28. All customers lie on one side of the depot, and they are all equally distant from each other, with fixed distance d. Within literature, this 1-dimensional scenario is known as the *Airport problem*, where the problem consists of allocating the total cost of building and maintaining an airport runway among its users on a non-cooperative basis (22, 35).

For such a problem, the true Shapley values can be obtained within polynomial time, regardless of the problem size. The Shapley value for the Airport problem is defined as:

$$\delta_i^* = \sum_{j=1}^{i} \frac{2 \cdot d}{n - j - 1}, \tag{4.1}$$

where n denotes the number of customers in the system, and d denotes the fixed distance between customers.

In Section 3.5, it was detected that with the MST algorithm, customers who are close tend to be clustered together, and result in identical allocated costs. This can be useful in certain cases, as it was observed with the true Shapley values that customers who are close in proximity tend to share costs. However, it is of interest to see how the MST algorithm performs when all locations are collinear and equidistant, and whether the model is robust to all possible scenarios.

For this, three experiments are conducted:

- 5 customers, with equally spaced distance of 5 units
- 25 customers, with equally spaced distance of 4 units
- 60 customers, with equally spaced distance of 7 units

All experiments took under a second to run for both the true Shapley values and the MST algorithm. What was observed is that for the true Shapley values, the allocated costs to customers are increasing for each customer. This is expected, as the cost between the depot and customer 1 is not only paid by customer 1, but all customers. Similarly. the distance between customers 1 and 2 is not only paid by customer 2, but also by customers 3,4 and 5. Therefore, a customer i who is distant from the depot is liable not only for the cost between the distance between itself and customer $i-1$, but all other distances between themselves and customers $i-2, i-3, ..., 1$. Therefore, customers close to the depot are considered 'cheaper'. Whereas for the MST algorithm, edges are included in the minimum spanning tree in an iterative manner, with increasing order of edge costs. Since edge costs are equal here, the order of the edges included do not make a difference and the resulting tree will be the same. As a result, all customers pay for exactly one edge. To illustrate, consider the smallest example, with 5 customers. In the first step, the edge between the depot and customer 1 is constructed. Since this customer is now connected with the depot, all remaining obligations will be depleted, and will solely pay for the edge cost of 5. As second step, the edge between customers 1 and 2 is included. The newly formed component consists of customers 1 and 2, and the depot. Since customer 2 is also connected to the depot via customer 1, and customer 1 has no remaining obligations, customer 2 will solely pay for this edge cost of 5. The same process iteratively holds for each customer, resulting in normalised allocated costs of each customer to be equal to $\frac{1}{n}$.

Table 4.11 shows the resulting normalised and global allocated costs of the true Shapley values in comparison with that of the MST algorithm for the first experiment consisting of 5 customers. It should be noted that the MST algorithm results in a tree, therefore the resulting costs would be for a TSP path, whereas the Shapley values obtained are for a TSP tour, thereby justifying the difference between the two in the sum of global costs.

Table 4.11: Shapley values vs MST values for collinear instance consisting of 5 customers

Customer	**Normalised values**		**Global values**	
	Shapley values	**MST values**	**Shapley values**	**MST values**
1	0.04	0.2	4	10
2	0.09	0.2	9	10
3	0.156	0.2	15.6	10
4	0.256	0.2	25.6	10
5	0.456	0.2	45.6	10

When comparing the two algorithms in terms of their normalised values, it can be observed that the MST algorithm assigns every customer 20% of the total cost of the path, whereas for the true Shapley values the assigned costs for customers are increasing in as customers become more distant from the depot. This can also be observed when analysing the global costs assigned to customers - the MST algorithm seems to overestimate the costs for customer 1. For customer 2, the costs assigned are almost identical, however for customers 3,4, and 5 the MST algorithm tends to underestimate increasingly for distant customers.

These experimental results of the MST algorithm have been similarly observed for experiments with larger sizes, where every customer is allocated $\frac{1}{25}$ and $\frac{1}{60}$ of the total costs for experiments with customers of size 25 and 60, respectively.

In Section 4.3, it was observed that the MST algorithm tends to perform generally well, especially with instances with larger number of customers. Customers who are close together are clustered, which is representative of the true Shapley values. However, in the collinear case with equidistant customers, the MST algorithm tends to perform more poorly, failing to capture the distance between customers and the depot. Nevertheless, this proxy tends to perform well and is capable in allocating costs to customers in a manner that can be considered representative of the true allocated costs through Shapley values. The costs are distributed fairly, as close customers are assigned the same costs. Moreover, these costs can be allocated within a second, thereby being efficient in terms of speed. In a collinear case, making use of the depot distance algorithm may yield more efficient, although such scenarios are unlikely to occur in real life settings.

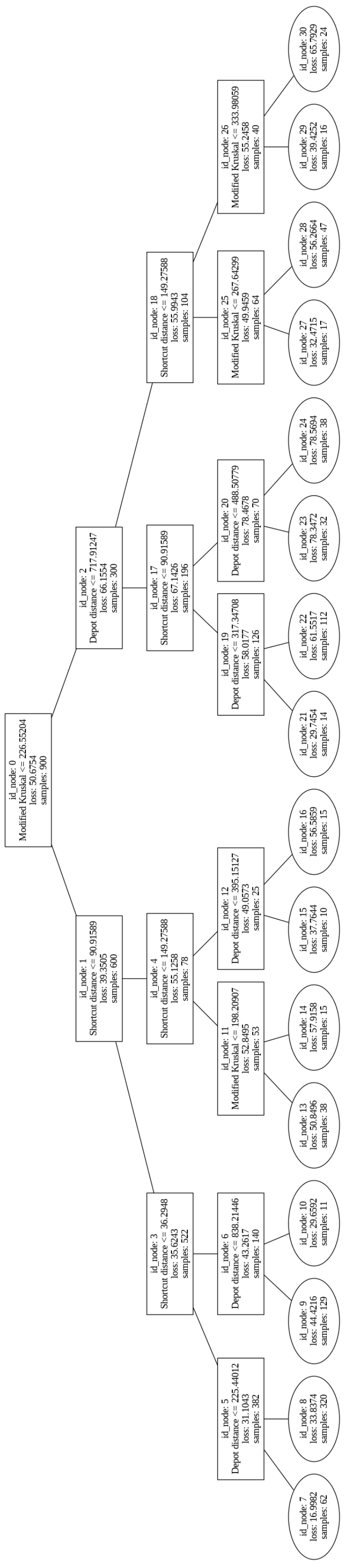


Figure 4.20: Resulting decision tree of the Linear Tree model for predicting the total ApproShapley values for a TSP tour

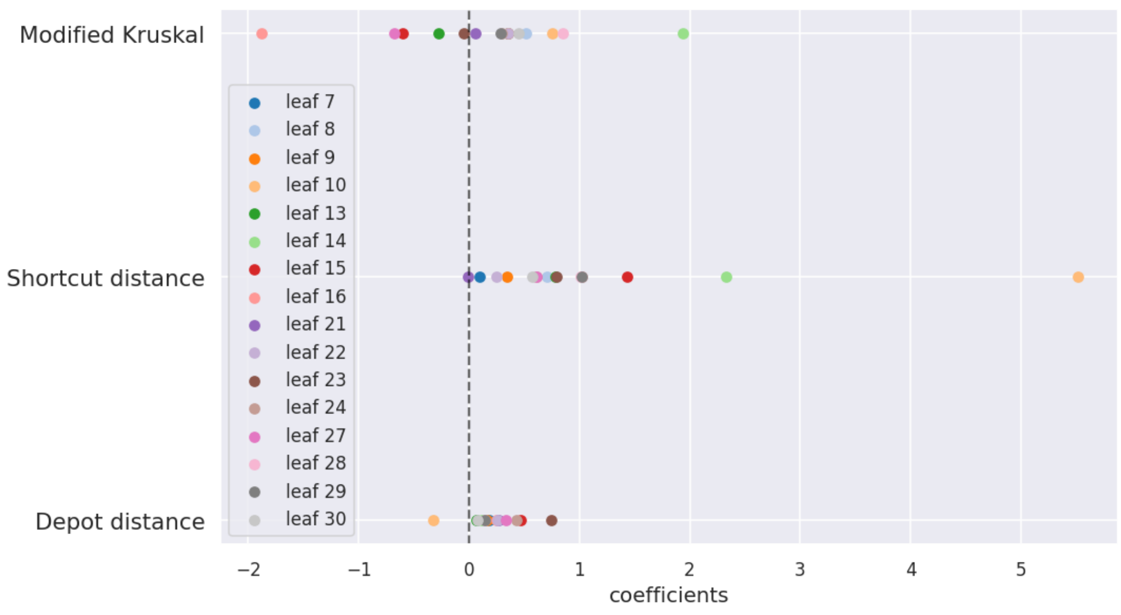
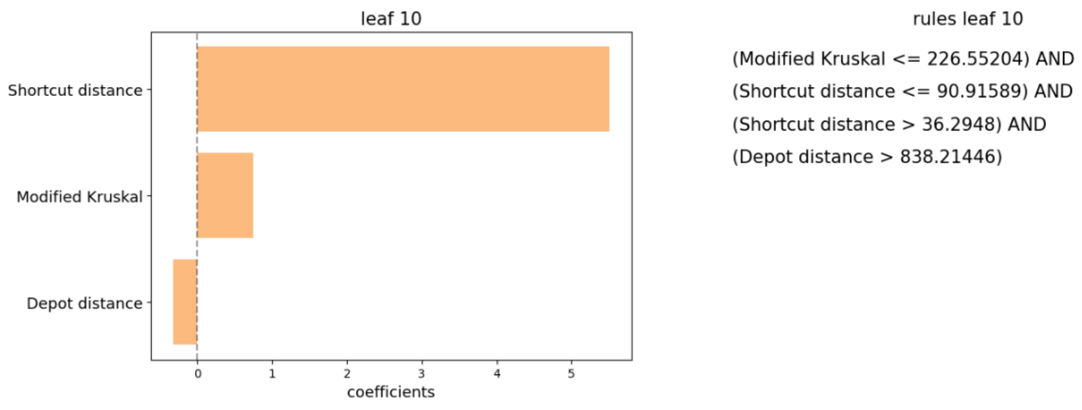


Figure 4.21: Coefficients of each feature based on all leaf nodes in the Linear Tree model



rules leaf 10

(Modified Kruskal <= 226.55204) AND
(Shortcut distance <= 90.91589) AND
(Shortcut distance > 36.2948) AND
(Depot distance > 838.21446)

Figure 4.22: Coefficients and decision rules of leaf node 10

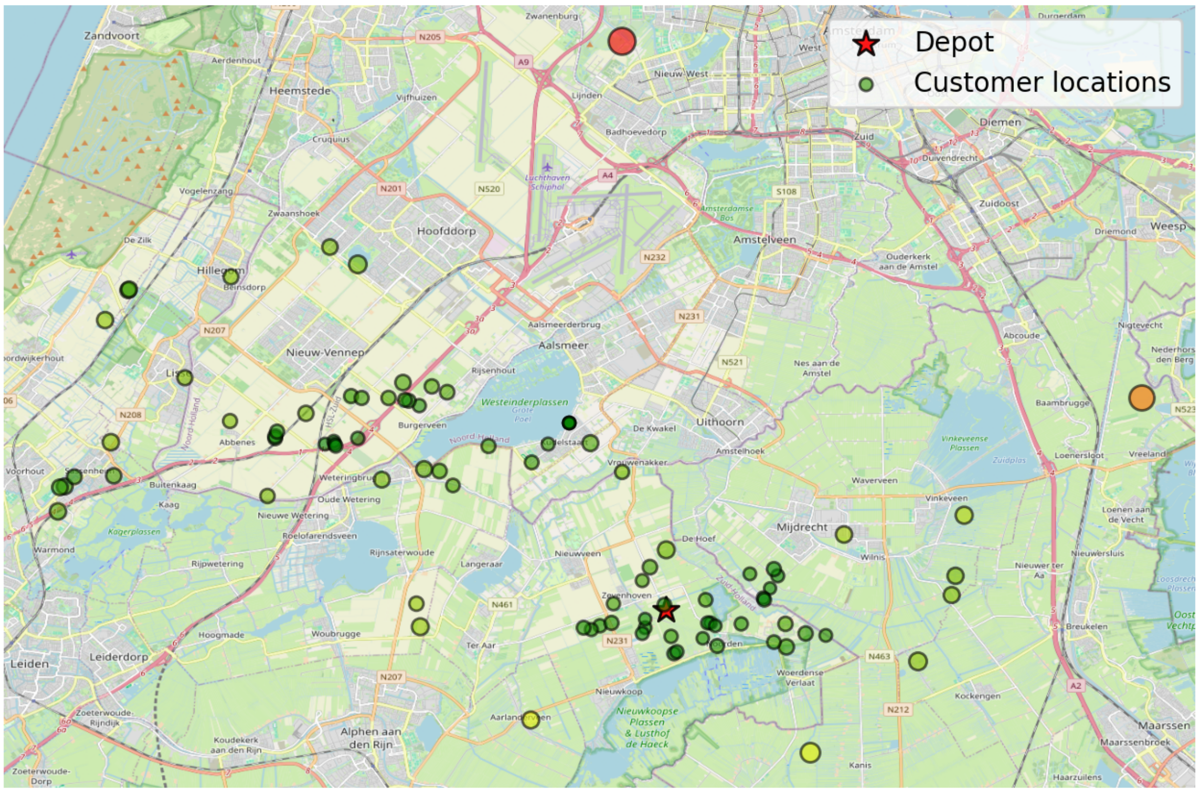


Figure 4.23: Costs allocated to customers of Company 1250

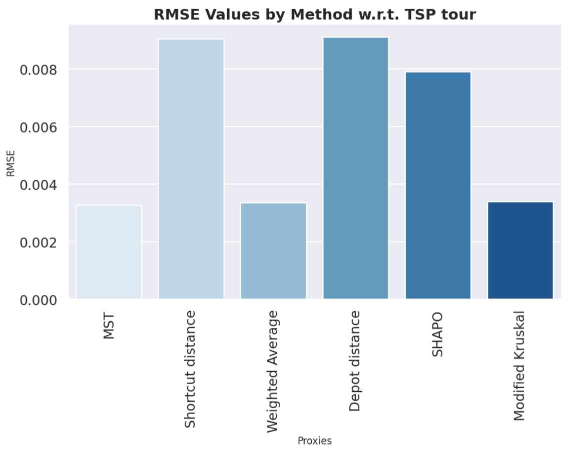


Figure 4.24: RMSE scores for TDF data for normalised TSP tour values

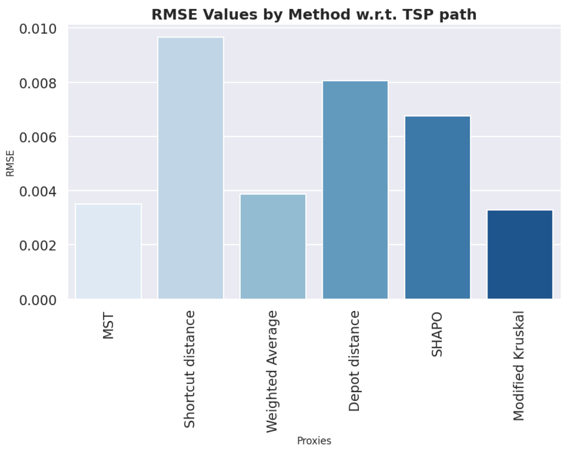


Figure 4.25: RMSE scores for TDF data for normalised TSP path values

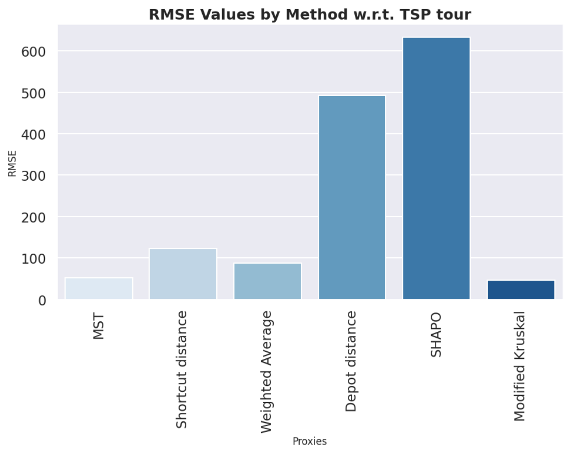


Figure 4.26: RMSE scores for TDF data for total TSP tour values

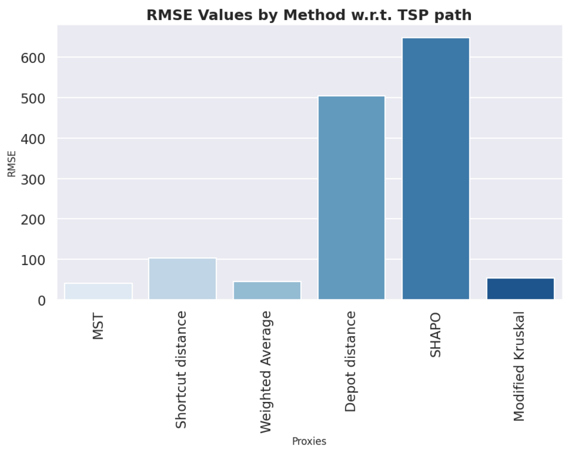


Figure 4.27: RMSE scores for TDF data for total TSP path values

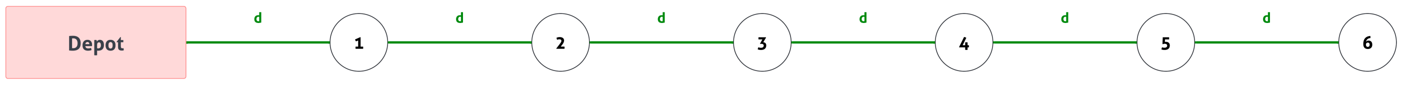


Figure 4.28: Collinearity of customers and depot

5 Discussion

This thesis explored heuristics - both available in literature and novel approaches - in order to approximate the Shapley values for a Travelling Salesman Problem, thereby allocating costs to customers in a fair and efficient manner. In Section 4, the implemented heuristics introduced in Section 3 were evaluated using synthetic data from benchmark instances. In addition to the performance of the heuristics for the normalised ApproShapley values for a TSP tour and path in terms of speed and accuracy, the performance of the methods were evaluated with respect to the total Shapley values. These heuristics were further applied to a case study based on customers of a company from The Driving Force, where over the road distances were applied from road networks. Finally, the robustness of the MST algorithm was analysed by constructing an instance consisting of collinear and equidistant customers.

In Section 4.3.1, the performance of the MST algorithm and Shortcut distance algorithms were evaluated for cases when the depot is a collaborative player and is assigned costs. What was observed is that the MST algorithm was effective in approximating the fraction of costs to be allocated to customers for both the TSP tour and the TSP path. This claim also holds true for the TSP variants where the depot is not a player. The MST algorithm and the Modified Kruskal algorithms were compared against naive algorithms made available by Aziz et al. (21), as well as the state-of-the-art SHAPO algorithm developed by Levinger et al. (23). For approximating the ApproShapley values for both the TSP tour and TSP path, the Weighted Average model consisting of the MST algorithm with one of the naive proxies outperformed all other methods. For approximating the ApproShapley values for the TSP path, the MST, Modified Kruskal and Weighted Average models outperformed all other methods in terms of the average RMSE score. All algorithms were evaluated not only in terms of accuracy, but also in terms of speed. From these evaluations, it was observed that all of the proposed novel heuristics in this paper allocated costs to customers in less than a second.

The aim of this research was to allocate costs to customers in a routing problem, while ensuring an intricate balance between a fair allocation, and an efficient one. The novel approaches presented in this paper - the MST and the modified Kruskal algorithms - were effective in approximating the ApproShapley values for benchmark instances both in terms of speed and accuracy. The modified Kruskal algorithm had a slightly better performance in comparison to the MST algorithm in efficiency and speed. A minor adjustment to the Kruskal algorithm ensured that the resulting tree would be a TSP path and allowed for a fairer allocation of costs to customers during the construction of the path. These results were also observed when analysing the performance of the models in terms of the total costs allocated, rather than the normalised. In such cases, it was observed that despite obtaining low RMSE scores for the normalised cases, the Depot Distance and SHAPO algorithms overestimated the total costs to be allocated, resulting in substantially high RMSE scores with respect to the ApproShapley values. The MST and Modified Kruskal algorithms were efficient in allocating the total costs to customers, with relatively accurate performance. Previous research on fair cost allocation methods have solely observed the performance of heuristics for the normalised Shapley values. However, it is also critical to observe not only the fraction of costs to be allocated, but the total costs. With this research, it was observed that the novel approaches presented were efficient and effective in assigning both normalised and total costs to customers.

In their paper, Aziz et al. (21) present the Christofides' algorithm to approximate the Shapley values for customers in a TSP game. This algorithm also makes use of the ApproShapley values to obtain costs to be allocated to customers. Although this algorithm maintains accurate results, it comes with the trade-off in terms of computational time. The Christofides' algorithm also makes use of MST's, and in this paper MST's are employed to allocate costs to customers without the use of the ApproShapley algorithm. Specifically, the MST and modified Kruskal algorithms only need to be run once to obtain allocated costs to customers. This is considered as a strength and contribution to research on fair cost allocation, as sampling-based methods are not required, and without computational burden these algorithms can allocate costs to customers in a fair manner. This holds true for all TSP games and variants - when the depot is a collaborative player and not, as well as for approximating for the TSP tour or path.

In Section 4.3.2, it was observed that the MST algorithm performs better when approximating allocated costs for customers for a TSP path than for a TSP tour. The same observation holds true for the Modified Kruskal algorithm. This is expected, as the resulting trees from the algorithms result in a path, rather than a tour. When comparing the performance of the heuristics for approximating the ApproShapley values for a TSP tour, it was observed that the Depot distance algorithm performed marginally better in terms of the RMSE scores. However, blending the MST and Depot distance algorithms together for the weighted average model resulted in better performance. Therefore, combining these algorithms can be beneficial in a fair cost allocation for the TSP tour.

Another novelty of this research was to implement machine learning models by incorporating naive and advanced proxies. For this approach, relatively interpretable yet effective models were utilised, namely: linear regression, decision trees, random forests and linear trees. Models were implemented to predict the ApproShapley values for both the TSP tour and the TSP path. Moreover, these models were implemented in terms of both the normalised and total costs. What was observed was that in general, the linear regression models were effective in predicting the ApproShapley values for both the TSP tour and path. The models were also tested on instances of customer sizes that were not within the range of customer sizes the models were trained on. Although some decrease in accuracy was observed, the models were nevertheless successful in allocating costs to customers in a fair manner. Moreover, due to the interpretability of the models, it is possible to observe in which scenarios it is appropriate to give more weight to a certain proxy than another.

The implemented heuristics were tested and analysed in a case study for customers of a company from The Driving Force in the Netherlands, where over the road distances were used. This distance metric allowed for a more realistic setting and thereby evaluation of the heuristics, as this metric allowed to observe the distances between customers based on the road network available within the Netherlands. Utilising the implemented heuristics on this data yielded promising and expected results, with the MST, Weighted Average and Modified Kruskal algorithms outperforming the other algorithms based on literature in terms of the RMSE scores. This experiment showed that the implemented heuristics in this thesis paper can be applied to real life instances.

Finally, the robustness of the MST algorithm was evaluated on collinear instances - when customers are on the same line - and are equidistant. In such instances it was observed that the MST algorithm fails to allocate costs in a fair manner, and results in Egalitarian cost allocation, where every customer pays the same price. Although this example shows undesired performance of the MST algorithm, such scenarios are unlikely to occur. In the likelihood that they do occur, the true Shapley values for such instances can be computed within polynomial time. Therefore, this problem does not decrease the efficiency and effectiveness of the MST algorithm with respect to routing problems.

Upon general observation, it was observed in the Results section that the discrepancy between the allocated costs via the ApproShapley algorithm and the MST algorithm is caused by the MST algorithm underestimating costs for customers. This especially holds true for more costly customers. This underestimation of costs is due to the property of a MST to be a lower bound for the optimal length of a TSP tour (31). For a TSP path this also holds true, as this is considered a spanning tree without branches. This spanning tree is always at least as long as the MST. Consequently, the MST algorithm always has at most the same cost as a TSP path. Due to the MST being a lower bound for the TSP, the costs assigned via the MST algorithm will tend to be lower than those of the ApproShapley algorithm. The Christofides algorithm introduced in Section 3.4 guarantees a solution that is not more than 50% longer than the optimal TSP length (31). Due to this, a recommendation for future research can be to extend the MST algorithm such that the underestimation of costs can be minimised - such as multiplying the allocated costs with a factor of 1.2, or with smarter scaling.

In this thesis, benchmark instances were utilised to allocate costs to customers by utilising various heuristics. In such instances, it was assumed that the demand, and hence the customers within a route are known. Therefore, a static demand assumption was made. This simplifies the general problem of allocating costs, as businesses may not directly know which customers will place orders during a day or time-slot. In some cases the demand may be known, especially within B2B or B2C business settings when customers to be served are determined, yet the problem is to allocate costs to customers. In such cases, this thesis provides promising results. For cases when customer demand is not fixed or known, a possible consideration for further research may be to make use the distributions of demand based on historical data in order to obtain an expected cost to be allocated to customers.

Moreover, this thesis allocates to customers in a Travelling Salesman Problem, where every customer must be visited with a single vehicle. A possible extension for the cost allocation methods can be made for a Vehicle Routing Problem, where multiple vehicles and hence routes may take place. The current problem considers the total cost for the last-mile deliveries as the cost of the route - which consists of distances between all customers. In a more general setting, customers may be assigned higher or lower costs based on other variables, such as the weight of the order. The remaining capacity of a truck may also influence the cost that should be allocated. Therefore, a further consideration for future research may be to collaborate other variable costs to customers and allocate them to each customer, as well as the route cost.

The outcomes of this thesis yield promising results, with a fair and efficient cost allocation by utilising desirable properties of the minimum spanning tree (MST), as well as combining this algorithm with naive cost allocation methods to ensure a methodical approach of allocating costs to customers in a TSP problem. Possible extensions of the introduced methods to more generalised problems will allow for efficient operations for last-mile deliveries in logistics.

6

Conclusion

This thesis formulated the research question of how to allocate costs to customers in a delivery route in a fair and efficient manner. Fairness was defined in terms of Shapley values, where all possible combinations of customer arrival orders are considered and averaged. Efficiency was defined in terms of the speed of the methods. Based on utilisation of minimum spanning trees, novel approaches were introduced to approximate the Shapley values of customers, thereby allocating costs in a fair manner. The results indicate that the novel approaches introduced are effective in approximating the Shapley values, while ensuring these results are obtained with no computational burden. Constructing a case study also allowed for observation of the efficiency of the algorithms with different distance metrics, such as distances based on road networks available.

While the MST and Modified Kruskal algorithms were efficient and effective in allocating costs to customers in a TSP problem, the problems constructed assumed deliveries to be able to fit in a single vehicle. Extending the problem setting to a vehicle routing problem may allow for a more generalised and hence systematic approach. Moreover, it is assumed that demands of customers were known. For future research, costs can be allocated based on information from historical demand and observing demand distributions.

This research has introduced novel approaches to cost allocation. Although approximations to Shapley values exist within literature, this thesis utilised minimum spanning trees to allocate costs to customers in a smart and efficient manner, while ensuring fairness. Previous literature on cost allocation shows an imbalance between the speed and accuracy trade-off: accurate heuristics tended to be burdensome whereas faster algorithms proved to be naive. This thesis improved results from literature while ensuring a subtle balance between efficiency and accuracy.

References

[1] Shelagh Dolan. **The challenges of last mile delivery logistics and the tech solutions cutting costs in the final mile**, 1 2022. 1

[2] Goran Kukolj. **What is last mile delivery? Costs 038; how to optimize**, 3 2024. 1

[3] PricewaterhouseCoopers. **Retail Monitor 2023: Last mile delivery**. 1

[4] Xinan Yang, Arne Strauss, Christine S. M. Currie, and Richard W. Eglese. **Choice-Based demand management and vehicle routing in E-Fulfillment**. *Transportation Science*, **50**(2):473–488, 5 2016. 1, 2, 8

[5] Mario Guajardo and Mikael Rönnqvist. **A review on cost allocation methods in collaborative transportation**. *International Transactions in Operational Research*, **23**(3):371–392, 9 2015. 6

[6] David Fleckenstein, Robert Klein, and Claudius Steinhardt. **Recent advances in integrating demand management and vehicle routing: A methodological review**. *European Journal of Operational Research*, **306**(2):499–518, 4 2023. 6

[7] Katrin Wassmuth, Charlotte Köhler, Niels Agatz, and Moritz Fleischmann. **Demand management for attended home delivery—A literature review**. *European Journal of Operational Research*, **311**(3):801–815, 12 2023. 6

[8] Jean-François Cordeau, Manuel Iori, and Dario Vezzali. **A survey of attended home delivery and service problems with a focus on applications**. *4OR*, **21**(4):547–583, 11 2023. 7

[9] Robert Klein, Jochen Mackert, Michael Neugebauer, and Claudius Steinhardt. **A model-based approximation of opportunity cost for dynamic pricing in attended home delivery**. *OR Spectrum*, **40**(4):969–996, 12 2017. 7, 8

[10] Liana Van Der Hagen, Niels Agatz, Remy Spliet, Thomas Visser, and Leendert Kok. **Machine Learning–Based feasibility checks for dynamic time slot management**. *Transportation Science*, 11 2022. 7, 8

[11] Florent Hernandez, Michel Gendreau, and Jean-Yves Potvin. **Heuristics for tactical time slot management: a periodic vehicle routing problem view**. *International Transactions in Operational Research*, **24**(6):1233–1252, 3 2017. 7

[12] Niels Agatz, Ann Melissa Campbell, Moritz Fleischmann, and Martin Savelsbergh. **Time slot management in attended home delivery**. *Transportation Science*, **45**(3):435–449, 8 2011. 7

[13] Jochen Mackert. **Choice-based dynamic time slot management in attended home delivery**. *Computers Industrial Engineering*, **129**:333–345, 3 2019. 7, 8, 10

[14] Estelle R. S. Kone and Mark H. Karwan. **Combining a new data classification technique and regression analysis to predict the Cost-To-Serve new customers**. *Computers Industrial Engineering*, **61**(1):184–197, 8 2011. 8

[15] Charlotte Köhler and Jarmo Haferkamp. **Evaluation of delivery cost approximation for attended home deliveries**. *Transportation Research Procedia*, **37**:67–74, 1 2019. 8

[16] Akang Wang, Jeffrey E. Arbogast, Gildas Bonnier, Zachary Wilson, and Chrysanthos E. Gounaris. **Estimating the marginal cost to deliver to individual customers**. *Optimization and Engineering*, **24**(4):2409–2447, 1 2023. 8, 10

[17] Ulrich Faigle, Sndor P. Fekete, Winfried Hochstttler, and Walter Kern. **On approximately fair cost allocation in Euclidean TSP games**. *OR spectrum*, **20**(1):29–37, 3 1998. 9

[18] Okan Örsan Özener, Özlem Ergün, and Martin Savelsbergh. **Allocating cost of service to customers in inventory routing**. *Operations Research*, **61**(1):112–125, 2 2013. 9

[19] Nicola Besozzi, Luca Ruschetti, Chiara Rossignoli, and Fernanda Strozzi. **The traveling Salesman game for cost allocation: The case study of the bus service in Castellanza**. *Game Theory*, **2014**:1–6, 12 2014. 9

[20] Alf Kimms and Igor Kozeletskyi. **Shapley value-based cost allocation in the cooperative traveling salesman problem under rolling horizon planning**. *EURO Journal on Transportation and Logistics*, **5**(4):371–392, 12 2016. 9

[21] Haris Aziz, Casey Cahan, Charles Gretton, Philip Kilby, Nicholas Mattei, and Toby Walsh. **A study of proxies for shapley allocations of transport costs**. *Journal of Artificial Intelligence Research*, **56**:573–611, 8 2016. 9, 17, 20, 21, 37, 38, 40, 44, 45, 48, 49, 53, 60, 61, 76, 77

[22] Dan C. Popescu and Philip Kilby. **Approximation of the Shapley value for the Euclidean travelling salesman game**. *Annals of Operations Research*, **289**(2):341–362, 5 2020. 10, 69

[23] Chaya Levinger, Noam Hazon, and Amos Azaria. **Efficient computation and estimation of the shapley value for traveling salesman games**. *IEEE Access*, **9**:129119–129129, 1 2021. 10, 48, 49, 67, 76

[24] Federico Arroyo. **Cost Allocation in Vehicle Routing Problems with Time Windows**. *Junior Management Science (JUMS)*, **9**(1):1241–1268, 2024. 10

[25] L. S. Shapley. *17. A value for N-Person games.* 12 1953. 12

[26] Javier Castro, Daniel Gómez, and Juan Tejada. **Polynomial calculation of the Shapley value based on sampling**. *Computers Operations Research*, **36**(5):1726–1730, 5 2009. 17

[27] K Ilavarasi and K Suresh Joseph. **Variants of travelling salesman problem: A survey**. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pages 1–7. IEEE, 2014. 17

[28] Feiyue Li, Bruce Golden, and Edward Wasil. **The open vehicle routing problem: Algorithms, large-scale test problems, and computational results**. *Computers & Operations Research*, **34**(10):2918–2930, 10 2007. 18

[29] Peter Borm. *Games, cooperative behaviour and economics.* 2020. 21, 23, 25, 26

[30] Kyle Genova and David P. Williamson. *An experimental evaluation of the Best-of-Many Christofides' algorithm for the traveling salesman problem.* 1 2015. 22

[31] Johannes Schneider and Scott Kirkpatrick. *Stochastic optimization.* Springer Science Business Media, 8 2007. 22, 23, 79

[32] Logan Dillard. **The best of both worlds: linear model trees - convoy tech - medium**. 5 2018. 37

[33] Marco Cerliani. **Explainable AI with Linear Trees - Towards Data Science**. 3 2022. 65

[34] **Project OSRM**. 67

[35] M. J. Albizuri, J. M. Echarri, and J. M. Zarzuelo. **A non-cooperative mechanism for the Shapley value of airport problems**. *Annals of operation research*, **235**(1):1–11, 8 2015. 69

Appendix

A Proxies and weights used for Weighted Average Model

Table 6.1: Proxies and weights used for approximating TSP tour in weighted average model

Customer size	Proxy A	Proxy B	Weight Proxy A	Weight Proxy B
10	MST	Depot distance	30%	70%
15	MST	Depot distance	40%	60%
20	MST	Depot distance	30%	70%
25	MST	Depot distance	40%	60%
30	MST	Depot distance	40%	60%
35	MST	Depot distance	40%	60%

Table 6.2: Proxies and weights used for approximating TSP path in weighted average model

Customer size	Proxy A	Proxy B	Weight Proxy A	Weight Proxy B
10	MST	Shortcut distance	80%	20%
15	MST	Shortcut distance	80%	20%
20	MST	Shortcut distance	90%	10%
25	MST	Shortcut distance	90%	10%
30	MST	Shortcut distance	90%	10%
35	MST	Shortcut distance	90%	10%