Incremental Preference Elicitation for Horizontal Collaboration in Transport Operations

Federico Toffano
Insight Centre for Data Analytics, School of Computer Science and IT
University College Cork, Ireland
federico.toffano@insight-centre.org

Nic Wilson
Insight Centre for Data Analytics, School of Computer Science and IT
University College Cork, Ireland
nic.wilson@insight-centre.org

ABSTRACT

We consider a system computing collaborative delivery plans to deliver a set of orders, by sharing trucks between the companies involved. Collaborative journeys are evaluated by the decision-makers considering multiple Key Performance Indicators (KPIs) such as cost, distance and time. Knowing the decision-makers' preferences over the KPIs would enable us to compute personalised solutions that increase the chance of finding attractive collaborations. With this work we address two problems. Firstly, we show how we can learn the decision-makers' preferences over the KPIs by observing their interactions with the system. Secondly, we show how to estimate the improvement or disimprovement in the KPIs arising from the collaboration.

KEYWORDS

Incremental Preference Elicitation; Collaborative Transport; Intelligent Logistics; Shapley Value

ACM Reference Format:
Federico Toffano and Nic Wilson. 2021. Incremental Preference Elicitation for Horizontal Collaboration in Transport Operations . In *Proc. of the 1st Workshop on Multi-Objective Decision Making (MODeM), Online, July 14-16, 2021*, IFAAMAS, 8 pages.

1 INTRODUCTION

Horizontal collaborations are inter-organisational relationships between two or more companies whose purpose is sharing resources to achieve a common objective. LOGISTAR *(Enhanced data management techniques for real time logistics planning and scheduling)* is an EU research project to develop a system to support horizontal collaborations of transport operations between competitors, and one of the main purposes of this project is the computation of collaborative delivery plans sharing trucks between the companies involved. Collaborative deliveries can be very beneficial, not only economically but also environmentally, since the optimisation of the truck-load reduces the number of journeys and the total CO_2 emissions. In fact, companies cannot always fill their trucks, especially during the return journey, when the truck could be completely empty. Therefore sharing the truck capacity with other companies can improve the efficiency of a network of transports with benefits for all the partners involved.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Proc. of the 1st Workshop on Multi-Objective Decision Making (MODeM), Hayes, Mannion, Vamplew (eds.), July 14-16, 2021, Online. © 2021 Copyright held by the owner/author(s).

The horizontal collaboration we focus on in this paper involves two companies, P and N, with whom we have defined the problem requirements and who will be involved in the testing of our software. P and N will then be the two decision-makers sharing some of their trucks to find collaborative delivery plans. In this context, the collaborative journeys proposed by the LOGISTAR system are computed by an optimiser and evaluated by the decision-makers considering multiple Key Performance Indicators (KPIs) such as cost, distance and time. The optimiser needs then to consider such KPIs during the optimisation of a delivery plan for a set of input orders, and each decision-maker may have different preferences with respect to the KPIs. Thus a fair objective function for the computation of collaborative journeys should consider the preferences of all partners involved.

Our goal is to learn the preferences of the decision-makers by observing their decisions with respect to the collaborative journeys proposed by the system. We assume a parameterised preference model for each decision-maker, with the domain of the parameters being the set of all the feasible preferences. The preference model considered is the weighted sum of the KPIs evaluating the journeys, and the weights of the KPIs vector define the decision-maker preferences. The general idea is to estimate the preference model of each decision-maker and merge them to create a common preference model used as the objective function for the optimiser computing the collaborative journeys. Although the details on how the optimiser computes the journeys are out of the scope of this paper, it is important to note that the goal is to find a collaborative solution that performs better than non-collaborative solutions and not a collaborative solution that reduces the costs for all the decision-makers involved.

Our preference learning method is based on the observation of the acceptance or rejection of the collaborative journeys proposed by the system. Each collaborative journey is associated with two KPIs vectors. On the one hand, we have the KPIs of the collaborative journey computed by the optimiser; on the other hand, we have an estimated KPIs vector for a non-collaborative delivery of the orders planned in the collaborative journey. The decision-maker is also asked to input a motivation for their decisions from a predefined list, and we generate constraints on the preference model of the decision-maker if the motivation is related to the two KPIs vectors. The constraints generated from all the past decisions will then be used as input for a Support Vector Machine (SVM) algorithm to estimate the weights vector of the decision-maker.

The two main contributions of this work are the following. Firstly, we propose a method to learn the decision-makers' preferences by observing the motivation of the decisions taken in the system. In

particular, we allow each decision-maker to express the importance of a specific KPI, and we show how to translate this preference information into constraints for the preference model. Secondly, we present a method based on the Shapley value to split the KPIs of a journey among the corresponding orders. This is a crucial step of our preference learning approach since we need to compare a collaborative KPIs vector associated with a single collaborative journey with a non-collaborative KPIs vector of the same orders composing the collaborative journey. However, the orders of a collaborative journey may be part of multiple non-collaborative journeys along with other orders. Thus, supposing, for example, two companies, we may not have two non-collaborative journeys delivering the same orders of a collaborative journey. Then we cannot directly compare the corresponding KPIs vectors. On the other hand, having a KPIs vector evaluating every single order, we could evaluate any set of orders by combining the corresponding KPIs vectors.

The rest of the paper is organised as follows. Section 2 provides a brief literature review of relevant works. Section 3 defines the key concepts and goals of our work. Section 4 presents our preference learning approach based on the observation of the interaction of the decision-makers with the system. Section 5 defines how to estimate the KPIs of a generic set of orders using the KPIs of the journeys planned by the optimiser. Section 6 shows how to split the KPIs of a journey among the corresponding orders with a method based on the Shapley value. Section 7 presents some computational results showing the time performances of the methods proposed in this work. Section 8 summarises our work, including some related future works.

2 RELATED WORK

The problem of computing collaborative journeys delivering orders of two or more e-logistics operators is a collaborative Vehicle Routing Problem (VRP). See, e.g., [8] for a survey on this topic. In general, collaborative VRPs are classified as centralised or decentralised. In centralised systems the total profit is maximised jointly (see, e.g., [5, 7, 19]); on the other hand, in a decentralised system, collaborations are evaluated separately by the partners sharing a limited amount of information (see, e.g., [4, 6, 14]). Our work could be of interest for both of these settings in a multi-objective context since we first learn the preferences of every single partner over a set of KPIs and afterwards we combine them to define a joint preference model. Thus, the preferences of every single partner could be used in a decentralised system to define personalised objective functions not shared among the collaborators; the joint preference model instead could be used for the computation of a fair collaborative solution with a centralised system.

To the best of our knowledge, there is no work on collaborative VRP based on an incremental multi-objective preference elicitation procedure to estimate the preferences over a set of KPIs of the partners involved. The purpose of incremental preference elicitation is to iteratively reduce the uncertainty of the decision-makers' preferences by observing their actions. In the literature we can find several incremental approaches for preference elicitation. These methods are often based on a preference model represented by a parameterised value function with the parameters representing the possible preferences of the decision-makers. Common examples are such as Bayesian approaches [2, 11, 18] and works involving a qualitative uncertainty representation [1, 16, 17]. Bayesian approaches represent the user preferences with a probability distribution over the feasible parameters of the value function, and the input preference information is used to update the probability distribution. Qualitative methods instead reduce the uncertainty of the value function by reducing the set of feasible parameters with hard constraints. Bayesian methods are usually more robust since they allow inconsistent preference information; on the other hand, they are much more computational demanding than qualitative methods. Ordinal regression [9, 10] is an alternative approach without a predefined structure for the value function; in this case the preference information is used to define a set of constraints defining the admissible parameterised value functions.

3 FORMAL SETTINGS

Order: : Let O be the input set of orders of N and P. In our context, an *order* $\alpha \in O$ is an atomic element representing a set of items that has to be transported from a pickup location to a drop-off location.

Journey: We define a *journey* (of a truck) as a non-empty set of orders $J \subseteq O$. A journey is defined to be *collaborative* if it contains at least one order of N and at least one order of P. Otherwise it is defined to be *non-collaborative*. Each journey J is evaluated with a vector $K(J) = (K_1(J), \ldots, K_5(J)) \in \mathbb{R}^5$ representing the following KPIs:

- Monetary cost, which includes the driver compensation and the fuel cost.
- Total journey distance in kilometres.
- Total journey time in hours.
- CO2 emissions of the journey.
- Empty running, i.e., distance travelled in kilometres without any load.

Preference model: We define

$$u_w(x) = w \cdot x = \sum_{i=1}^{5} w_i x_i \tag{1}$$

to be the value function of a decision-maker parametrised with respect to the *weights vector* w, where x is a KPIs vector and $w \in \mathcal{U}$ and

$$\mathcal{U} = \{w \in R^5 : w_i \geq 0, \sum_{i=1}^{5} w_i = 1\}. \tag{2}$$

The (unknown) weights vector w defines the trade-offs of a decision-maker with respect to the KPIs. An estimation of the weights vector w defines then a value function $u_w(x)$ which can be used as objective function for an optimiser to compute personalised solutions. Note that the goal is to minimise the KPIs and thus to minimise the objective function.

Goal 1: We want to learn two weights vectors w^n and w^p, one for N and one for P, and a common weights vector w^c which considers the preferences of both N and P.

Solution: We define a *solution* as a non-empty set of journeys. A solution is defined to be *collaborative* if it is optimised with an objective function considering the weights vector and the input

orders of all the partners involved. A solution is defined to be *non-collaborative* if it is optimised considering the weights vector and the input orders of a single partner. The objective function of the optimiser is the weighted sum of the sum of the KPIs associated with the journeys planned in a solution. In our context, an optimiser computes a collaborative solution for N and P, using the weights vector w^c, and two non-collaborative solutions, one for N and one for P, using the weights vectors w^n and w^p respectively.

Goal 2: Let $\mathcal{Z}^c$ be the set of collaborative journeys of the collaborative solution, i.e., the set of journeys of the collaborative solution containing at least one order from N and one order from P. The system will show to the decision-makers only the collaborative journeys $\mathcal{Z}^c$. For a collaborative journey $J \in \mathcal{Z}^c$, we want to compute an estimated non-collaborative KPIs vector $E(J) = (E_1(J), \ldots, E_5(J)) \in \mathbb{R}^5$ to estimate the improvement or disimprovement in the KPIs arising from the collaborative journey J. The estimated non-collaborative KPIs vector $E(J)$ will then be shown to the decision-makers along with $K(J)$ to facilitate the decision, and the decision-makers' preference between $K(J)$ and $E(J)$ will be used to learn their weights vector.

4 PREFERENCE LEARNING

The preference learning method considered in this work is based on the observations of the preferences of the decision-makers with respect to the KPIs vectors evaluating the collaborative journeys. For each collaborative journey $J \in \mathcal{Z}^c$ computed by the optimiser, the system shows to the decision-makers the details of the journey, a KPIs vector $K(J)$ evaluating the journey, and an estimated KPIs vector $E(J)$ evaluating a non collaborative delivery for the orders in J. The purpose of the latter KPIs vector is to evaluate the improvement or disimprovement in the KPIs arising from the collaboration. The preferences of the decision-makers arising from the comparison of $K(J)$ with $E(J)$ will be used to generate constraints for an SVM algorithm estimating the value functions of the two decision-makers.

Constraints generation: The two decision-makers N and P need to either accept or reject each collaborative journey, and also express a motivation for their decision from a predefined list (see Table 1). We want to distinguish three scenarios with the selected motivation:

(1) The decision was made considering the KPIs.
(2) The decision was made considering mainly a specific KPI.
(3) The decision was made for other reasons.

Scenario (1). Suppose that a decision-maker accepts a collaborative journey selecting *better collaborative KPIs overall* as motivation. Since the motivation is related to the KPIs vector, we assume that their associated weights vector w satisfies the constraint

$$w \cdot (K(J) - E(J)) \leq 0, \tag{3}$$

(using the notation introduced in Equation 1) where $K(J)$ is the KPIs vector associated with the collaborative journeys, $E(J)$ is the estimated non-collaborative KPIs vector, and w is the (unknown) weights vector of the decision-maker. If, instead, the decision-maker rejects the collaborative solution selecting *better non-collaborative KPIs overall* as motivation, then we learn the constraint $w \cdot (E(J) - K(J)) \leq 0$.

Scenario (2). In this case, we learn the same constraint as scenario (1) (since the decision was made considering the KPIs) and a further set of constraints to highlight the importance of the main KPI considered. Suppose that a decision-maker accepts a collaborative journey and that the main KPI considered has index i. The constraints we learn in this case are Equation 3 (scenario (1)), and

$$w_i(K_i(J) - E_i(J)) + w_j(K_i(J) - E_j(J)) \leq 0 \text{ for all } j \neq i. \tag{4}$$

If the decision-maker rejects the collaborative journey, we learn the same constraints but exchanging $K(J)$ and $E(J)$. Intuitively, the set of constraints given by Equation 4 means that the gain of the i-th KPI is more important than any other eventual loss in the remaining KPIs.

Scenario (3). In this case we do not learn any constraint since we suppose that the decision was not related to the KPIs vector.

Example 4.1. suppose we have only three KPIs, e.g., cost in euro, distance in kilometres and time in hours, and suppose that a decision-maker accepts a collaborative journey selecting *better collaborative cost*. Let $K(J) = (1000, 100, 2.5)$ and $E(J) = (1200, 80, 3)$. Then we learn:

(a) $(w_1, w_2, w_3) \cdot (1000, 100, 2.5) \leq (w_1, w_2, w_3) \cdot (1200, 80, 3)$
(b) $(w_1, w_2) \cdot (1000, 100) \leq (w_1, w_2) \cdot (1200, 80)$
(c) $(w_1, w_3) \cdot (1000, 2.5) \leq (w_1, w_3) \cdot (1200, 3)$

Constraint (a) corresponds to Equation 3. Constraints (b) and (c) corresponds to Equation 4. Constraint (b) can be interpreted as: the collaborative KPIs are preferred even if we ignore the loss in time of $E(J)$ with respect to $K(J)$. Constraint (c) does not reduce the set of possible weights vectors since it is always verified.

Estimation of the weights vectors: To estimate the weights vector of a specific decision-maker, we use a SVM algorithm named Ranking SVM [12, 13]. Roughly speaking, the idea is to compute the hyperplane containing the origin that produces the largest distance from a set of preference points representing input preference constraints. In our context, the preference points are derived from the inequalities learned observing the decision-maker's preferences concerning the KPIs shown by the system, and the normal vector of the output hyperplane represents the estimated decision-maker's trade-offs between the KPIs. Ranking SVM also includes slack variables to allow the violation of the constraints derived from the input preference information, resulting in a cost for the objective function. This means that we can estimate a decision-maker's weights vector even if the intersection of the constraints associated with the decision-maker's decisions leads to an empty set of feasible weights vectors. This can happen when the decision-maker expresses a preference over the KPIs $K(J)$ and $E(J)$ conflicting with their real weights vector.

Formally, the optimisation problem we want to solve is the following.

$$\arg\min_{\omega, \xi_k} = \frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{5} \xi_k \tag{5a}$$

subject to

$$1 + \omega \cdot \lambda_k - \xi_k \leq 0 \quad \forall k \tag{5b}$$

$$\xi_k \geq 0 \quad \forall k \tag{5c}$$

$$\omega_i \geq 0 \quad \forall i \in \{1, \ldots, 5\}. \tag{5d}$$

Table 1: Possible motivations shown to the user for their acceptance or rejection of a collaborative journey, and the corresponding scenarios described in Section 4.

Acceptance	Rejection	Scenario
Better collaborative KPIs overall	Better non-collaborative KPIs overall	(1)
Better collaborative cost	Better non-collaborative cost	(2)
Better collaborative distance	Better non-collaborative distance	(2)
Better collaborative time	Better non-collaborative time	(2)
Better collaborative CO2 emissions	Better non-collaborative CO2 emissions	(2)
Better collaborative empty running	Better non-collaborative empty running	(2)
Other reasons	Other reasons	(3)

This method is very similar to that described by Equation 13 of [13]. The differences are in the normal ω of the hyperplane, which in our case is non-negative (Equation 5d) since we do not want to penalise any KPI, and in the sign of λ_k (Equation 5c) since our goal is to minimise the value function of the decision-maker (and not to maximise it). ξ_k is the slack variable used to assign the penalty to the k-th constraint when it is violated. C is a constant scaling the penalty of inconsistent constraints; a larger value of C corresponds to assigning a higher penalty. λ_k is the coefficients vector of the k-th constraint derived from the input preference information of the decision-makers as described below.

To generate the values λ_k, we first need to normalise the values range of every component of all the KPIs vectors $K(J)$ and $E(J)$, dividing it by the corresponding maximum value. This step is required otherwise the maximisation of the margin performed by the SVM algorithm could focus mainly on the KPIs with a larger scale. We consider then a vector $M = (M_1, \dots, M_5) \in \mathbb{R}^5$ where M_i is the maximum value of the i-th KPI of a database of previous instances. Thereafter, we define the vector λ_k by scaling the i-th value of the k-th constraint learned (whose structure is defined by Equation 3 or Equation 4) by the reciprocal of M_i, for all $i \in \{1, \dots, 5\}$.

Example 4.2. Suppose we have shown to a decision-maker two collaborative journeys J' and J'', from which we derived two constraints $w \cdot (K(J') - E(J')) \leq 0$ and $w \cdot (E(J'') - K(J'')) \leq 0$. Then $M_i = \max(K_i(J'), K_i(J'), E_i(J''), E_i(J''))$ and the constraints for the SVM problem represented by Equation 5c are

$$1 + \omega \cdot \left(\frac{K_1(J') - E_1(J')}{M_1}, \dots, \frac{K_5(J') - E_5(J')}{M_5} \right) - \xi_1 \leq 0 \quad (6a)$$

and

$$1 + \omega \cdot \left(\frac{E_1(J'') - K_1(J'')}{M_1}, \dots, \frac{E_5(J'') - K_5(J'')}{M_5} \right) - \xi_2 \leq 0. \quad (6b)$$

Let $M^{-1} = (\frac{1}{M_1}, \dots, \frac{1}{M_5})$. The normal ω of the hyperplane computed by the SVM algorithm is related to the constraints defined on the rescaled KPIs. Thus the estimated weights vector w of a decision-maker defining their value function $u_w(x)$ for a generic KPIs vector x can be computed with the following transformation.

$$w = f(\omega) = \frac{\omega \circ M^{-1}}{\omega \cdot M^{-1}} \quad (7)$$

where $\omega \circ M^{-1}$ is the pointwise product (i.e., the Hadamard product) between ω and M^{-1}. The normalisation $\frac{1}{\omega \cdot M^{-1}}$ of Equation 7 is required to obtain a weights vector w with $\sum_{i=1}^{5} w_i = 1$, so that $w \in \mathcal{U}$.

The weights vectors w^n and w^p of the two decision-makers N and P will then be estimated by transforming with Equation 7 the two outputs ω^n and ω^p of our SVM algorithm executed considering the constraints derived by the preferences of N and P, respectively.

The joint weights vector w^c considering the preferences of two decision-makers is estimated by transforming with Equation 7 the midpoint of the segment connecting ω^n and ω^p:

$$w^c = f\left(\frac{(\omega^n + \omega^p)}{2} \right). \quad (8)$$

The resulting weights vector can be interpreted as a fair set of trade-offs among the KPIs considering the preferences of the two decision-makers involved. Thus the corresponding value function $u_{w^c}(\cdot)$ can be used as the objective function to compute collaborative solutions.

Without any preference information, i.e., when we do not have received any feedback from the decision-makers regarding the KPIs vectors, we compute the collaborative and non-collaborative solutions with a weights vector minimising the cost.

5 COMPUTATION OF $E(J)$

Our preference elicitation method learns the decision-makers' weights vector based on their preferences between the KPIs vector $K(J)$ evaluating a collaborative delivery J, and an estimated KPIs vector $E(J)$ evaluating a corresponding non-collaborative delivery plan for the orders in J. The idea is that the KPIs vector $E(J)$ will be considered by the decision-maker to evaluate the improvement or disimprovement in the KPIs arising from the collaboration. However, the computation of $E(J)$ is not straightforward since the non-collaborative deliveries for the orders in J planned in the non-collaborative solutions may be assigned to several non-collaborative journeys also delivering other orders not in J. Thus, since the KPIs vector computed by the optimiser evaluates journeys and not single orders, we may not have a set of non-collaborative KPIs vectors that can be directly used to evaluate a non-collaborative plan for the orders in J.

Example 5.1. Suppose we have two orders $N1$ and $N2$ from N, and two orders $P1$ and $P2$ from P. Suppose that the optimiser finds two distinct collaborative journeys, J_1 delivering $N1$ and $P1$ with KPIs $K(J_1)$, and J_2 delivering $N2$ and $P2$ with KPIs $K(J_2)$. The question is: how can we evaluate the value of the collaboration

of each single journey? Consider, for example, the collaborative delivery J_1. We want to estimate a KPIs vector $E(J_1)$ for a non-collaborative delivery for the orders in J_1 and evaluate the collaboration by comparing $K(J_1)$ with $E(J_1)$. Suppose that the computation of a non-collaborative solution leads to two non-collaborative KPIs evaluating a non collaborative delivery for $N1$ and $N2$, and a non-collaborative delivery for $P1$ and $P2$. The issue is that we are interested in a non-collaborative KPIs vector evaluating $N1$ and $P1$, but we do not have such KPIs vector since these orders are delivered by two distinct non-collaborative journeys along with other orders, i.e., $N2$ and $P2$.

Our idea is to split the KPIs of the non-collaborative journeys among the corresponding orders to evaluate a non-collaborative plan for a generic subset of orders, even if they belong to different non-collaborative journeys. In fact, having a KPI vector for every order, we can estimate the KPIs vector of any subset of orders by summing up the individual KPIs vectors.

Recall that $\mathcal{Z}^c$ is the set of collaborative journeys of the collaborative solution computed by the optimiser with objective function u_{w^c}. Our goal is then to estimate a non-collaborative KPIs vector $E(\alpha) \in \mathbb{R}^5$ for each $\alpha \in J$ and $J \in \mathcal{Z}^c$, and to compute $E(J)$ as:

$$E(J) = \sum_{\alpha \in J} E(\alpha). \tag{9}$$

Let $\mathcal{Z}^{nc}$ be the union of the journeys of the two non-collaborative solutions for N and P computed by the optimiser based on objective functions u_{w^n} and u_{w^p}, respectively. Suppose that the delivery of every order planned with the collaborative journeys in $\mathcal{Z}^c$ is also planned in a non-collaborative journey in $\mathcal{Z}^{nc}$. We then compute $E(\alpha)$ as follows

$$E(\alpha) = \mu(\alpha) K(J^{nc}(\alpha)). \tag{10}$$

Where $J^{nc}(\alpha)$ is the unique non-collaborative journey in $\mathcal{Z}^{nc}$ delivering α and with KPIs vector $K(J^{nc}(\alpha))$, and $\mu(\alpha)$ is a non-negative weight, defined such that $\sum_{\gamma \in J^{nc}(\alpha)} \mu(\gamma) = 1$. We call $\mu(\alpha)$ the *Shapley weight* of α, and it is computed using a method based on the *Shapley value* described in Section 6. Roughly speaking, the weights computed by this method fairly split the cost of a journey among the corresponding orders considering the extra kilometres required to deliver each order $\gamma \in J^{nc}(\alpha)$.

In our work we also had to address the following issue. The delivery of an order planned with a collaborative journey in $\mathcal{Z}^c$ may not be planned in any of the non-collaborative journeys in $\mathcal{Z}^{nc}$. This can happen for several reasons; for example, there may not be enough trucks to deliver all the orders or there is not a feasible solution matching all the delivery time windows of the input set of orders. Thus we may not be able to compute $E(J)$ with Equation 9. To overcome this issue we designed a simple fix described as follows. Let $J_x \subseteq J$ be the set of orders scheduled in the non-collaborative solution, i.e., the set of orders $\alpha \in J$ such that there exists $J^{nc} \in \mathcal{Z}^{nc}$ containing α. We compute the estimated KPIs vector $E(J)$ evaluating the non-collaborative deliveries for the orders in J as the rescaled sum of the estimated KPIs of the non-collaborative deliveries of the orders $\alpha \in J_x$:

$$E(J) = \frac{|J|}{|J_x|} \sum_{\alpha \in J_x} E(\alpha) = \frac{|J|}{|J_x|} \sum_{\alpha \in J_x} \mu(\alpha) K(J^{nc}(\alpha)). \tag{11}$$

The scale factor $\frac{|J|}{|J_x|}$ is used to estimate the KPIs of non-collaborative deliveries of $|J|$ orders having the estimation of only $|J_x|$ orders. We suppose that a scenario with $J_x = \emptyset$ will not occur; however, in our implementation we also cover the case $J_x = \emptyset$ by computing $E(J)$ as the average KPIs vector of the non-collaborative journeys, i.e., $E(J) = \frac{1}{|\mathcal{Z}^{nc}|} \sum_{J^{nc} \in \mathcal{Z}^{nc}} K(J^{nc})$. This is just to cover all the possible scenarios, but an average of the KPIs of the non-collaborative journeys may not return realistic values for $E(J)$.

Example 5.2. Consider three input orders $P1$, $P2$ and $P3$ from P, and three input orders $N1$, $N2$ and $N3$ from N. Suppose that the optimiser computes a collaborative solution with the following collaborative journeys evaluated with KPIs cost in euro, distance in kilometres and time in hours:

- Journey J_1 with KPIs $[900, 80, 2]$ delivering $P1$ and $N1$.
- Journey J_2 with KPIs $[800, 100, 3]$ delivering $P2$ and $N2$.
- Journey J_3 with KPIs $[1000, 115, 4]$ delivering $P3$ and $N3$.

Also, suppose that the optimiser computes the following non-collaborative solution:

- Journey J_1^{nc} with KPIs $[900, 100, 2.5]$ delivering $P1$ and $P2$.
- Journey J_2^{nc} with KPIs $[1000, 70, 2]$ delivering $N1$.
- Journey J_3^{nc} with KPIs $[800, 100, 3]$ delivering $N2$ and $N3$.

Note that we are supposing that the optimiser didn't find a non-collaborative journey for order $P3$.

The first step of our procedure is to compute the Shapley weight of the orders of the non-collaborative journeys J_k^{nc} with $k \in \{1, 2, 3\}$. (In this example we do not show how to compute the Shapley weights; see Section 6 for an example). Then we use Equation 10 to estimate the KPIs vectors $E(\alpha)$ evaluating a non-collaborative delivery for the orders α planned in the non-collaborative solution:

- $\mu(P1) = 0.4; E(P1) = [0.4 \cdot 900, 0.4 \cdot 100, 0.4 \cdot 2.5] = [360, 40, 1]$
- $\mu(P2) = 0.6; E(P2) = [0.6 \cdot 900, 0.6 \cdot 100, 0.6 \cdot 2.5] = [540, 60, 1.5]$
- $\mu(N1) = 1; E(N1) = [1 \cdot 1000, 1 \cdot 70, 1 \cdot 2] = [1000, 70, 2]$
- $\mu(N2) = 0.4; E(N2) = [0.4 \cdot 800, 0.4 \cdot 100, 0.4 \cdot 3] = [320, 40, 1.2]$
- $\mu(N3) = 0.6; E(N3) = [0.6 \cdot 800, 0.6 \cdot 100, 0.6 \cdot 3] = [680, 60, 1.8]$

Finally, we estimate the KPIs vectors $E(J_k)$ for each collaborative journey J_k using Equation 11:

- $E(J_1) = [\frac{1}{1}(360+1000), \frac{1}{1}(40+60), \frac{1}{1}(1+1.5)] = [1360, 100, 2.5]$
- $E(J_2) = [\frac{1}{1}(540+320), \frac{1}{1}(40+60), \frac{1}{1}(1.5+1.2)] = [860, 100, 2.7]$
- $E(J_3) = [\frac{2}{1}(680), \frac{2}{1}(60), \frac{2}{1}(1.8)] = [1360, 120, 3.6]$

6 SPLITTING THE KPIS OF A JOURNEY AMONG THE CORRESPONDING ORDERS

A well-known method used to split the cost of a collaboration with n participants is the *Shapley value* (see, e.g., [15]). This method is based on a cost function $v : 2^N \to \mathbb{R}$ with $v(\emptyset) = 0$, which defines the total cost of any collaboration of the set N of participants. Our intention is to divide the cost of a journey among the orders planned for delivery within the journey itself. Thus, we interpret the set of orders defining a non-collaborative journey J^{nc} as the set of participants for the computation of the Shapley value. Regarding the cost function, we consider a function $\mathrm{TSP}(t, I)$ which returns the total journey distance computed solving a travelling salesman problem with same start and end point t (which is the location of the truck assigned to J^{nc}), and with input locations defined as pick

up and drop location of the orders $I \subseteq J^{nc}$. On the computation of $\text{TSP}_{J^{nc}}(t, I)$, we consider precedence constraints for the pick up and drop location of each order. However, we ignore the capacity constraint of the corresponding assigned truck.

The Shapley value of an order $\alpha \in J^{nc}$ is then defined as:

$$\phi(\alpha) = \sum_{I \subseteq J^{nc} \setminus \{\alpha\}} \frac{|I|!(|J^{nc}| - |I| - 1)!}{|J^{nc}|!} (\text{TSP}(t, I \cup \{\alpha\}) - \text{TSP}(t, I)) \tag{12}$$

The Shapley value of an order $\alpha \in J^{nc}$ can be interpreted as the average increase of total distance caused by the inclusion of α in journeys delivering the permutations of the set of orders $J^{nc} \setminus \{\alpha\}$. A fundamental property derived from the theory behind the Shapley value is that $\text{TSP}(t, J^{nc}) = \sum_{\alpha \in J^{nc}} \phi(\alpha)$. Roughly speaking, the higher the Shapley value of an order is, the higher the extra distance that a truck has to travel to include the delivery of the order in the journey. The pseudocode of our implementation to compute $\phi(\alpha)$ for all the orders of a journey J^{nc} with starting point t is shown in Algorithm 1.

Once we have computed the Shapley value for every order α of a non-collaborative journey J^{nc}, we compute the corresponding Shapley weight $\mu(\alpha)$ as:

$$\mu(\alpha) = \frac{\phi(\alpha)}{\sum_{\beta \in J^{nc}} \phi(\beta)} \tag{13}$$

The Shapley weights are then used in Equation 10 to split the KPIs of a non-collaborative journey J^{nc} among the corresponding orders.

Algorithm 1 Shapley Value

procedure $SV(J^{nc}, t)$
  $n \leftarrow |J^{nc}|$
  $\mathcal{S}_0 \leftarrow \{\emptyset\}$
  $\mathcal{S}_i \leftarrow \emptyset$ for each $i \in [1, \dots n-1]$
  $L(\emptyset) \leftarrow 0$
  $\phi(\alpha) \leftarrow 0$ for each $\alpha \in J^{nc}$
  for $m \in [0, \dots n-1]$ **do**
    $c \leftarrow \frac{m!(n-m-1)!}{n!}$
    for $I \subseteq J^{nc}$ with $|I| = m+1$ **do**
      $\mathcal{S}_{m+1} \leftarrow \mathcal{S}_{m+1} \cup \{I\}$
      if $\exists \alpha, \beta \in I$ with same pick up and drop location **then**
        $L(I) \leftarrow L(I \setminus \{\alpha\})$
      else
        $L(I) \leftarrow (\text{TSP}(t, I))$
    for $I \in \mathcal{S}_m$ **do**
      for $\alpha \in J^{nc} \setminus I$ **do**
        $\phi(\alpha) \leftarrow \phi(\alpha) + c(L(I \cup \alpha) - L(I))$
  return ϕ

Example 6.1. Consider for example a non-collaborative journey composed by three orders $J^{nc} = \{\alpha_1, \alpha_2, \alpha_3\}$ with corresponding pick up and drop locations (p_i, d_i) shown in Figure 1. Suppose that t is the start and end location of the truck assigned to J^{nc}, and suppose that the distance between two locations of the map shown in Figure 1 is the corresponding Manhattan distance. We then get the following values of $\text{TSP}_{J^{nc}}(t, I)$ for each subset I of J^{nc}:

$$\begin{aligned}
\text{TSP}_{J^{nc}}(t, \emptyset) &= 0 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_1\}) &= 14 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_2\}) &= 14 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_3\}) &= 14 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_2\}) &= 14 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_2, \alpha_3\}) &= 20 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_3\}) &= 20 \\
\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_2, \alpha_3\}) &= 20
\end{aligned} \tag{14}$$

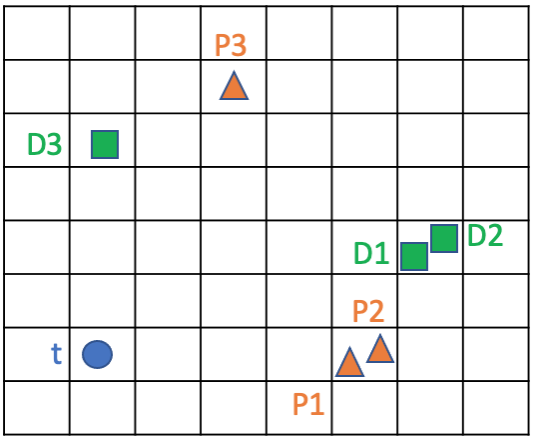

Figure 1: Delivery map of three orders $J^{nc} = \{\alpha_1, \alpha_2, \alpha_3\}$ with start location t (circle) corresponding pick up (triangle) and drop (square) location (p_i, d_i).

Following Equation 12 we get:

$$\begin{aligned}
\phi(\alpha_1) = \phi(\alpha_2) &= \frac{1}{3}\text{TSP}_{J^{nc}}(t, \{\alpha_1\}) \\
&+ \frac{1}{6}(\text{TSP}_{J^{nc}}(t, \{\alpha_2, \alpha_1\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_2\}) \\
&+ \frac{1}{6}(\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_3\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_3\}) \\
&+ \frac{1}{3}(\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_2, \alpha_3\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_2, \alpha_3\}) \\
&= \frac{14}{3} + \frac{0}{6} + \frac{6}{6} + \frac{0}{3} = \frac{34}{6}.
\end{aligned} \tag{15}$$

$$\begin{aligned}
\phi(\alpha_3) &= \frac{1}{3}\text{TSP}_{J^{nc}}(t, \{\alpha_3\}) \\
&+ \frac{1}{6}(\text{TSP}_{J^{nc}}(t, \{\alpha_2, \alpha_3\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_2\}) \\
&+ \frac{1}{6}(\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_3\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_1\}) \\
&+ \frac{1}{3}(\text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_2, \alpha_3\}) - \text{TSP}_{J^{nc}}(t, \{\alpha_1, \alpha_2\}) \\
&= \frac{14}{3} + \frac{6}{6} + \frac{6}{6} + \frac{6}{3} = \frac{52}{6}.
\end{aligned} \tag{16}$$

Thus, following Equation 13 we get:

$$\mu_1 = \mu_2 = \frac{\phi(\alpha_1)}{\phi(\alpha_1) + \phi(\alpha_2) + \phi(\alpha_3)} = 0.283, \tag{17}$$

$$\mu_3 = \frac{\phi(\alpha_3)}{\phi(\alpha_1) + \phi(\alpha_2) + \phi(\alpha_3)} = 0.433. \tag{18}$$

Note that with a straightforward method to compute the weights such as $\mu_i = \frac{\text{TSP}_{J^{nc}}(t,\{\alpha_i\})}{\sum_{j\in\{1,2,3\}} \text{TSP}_{J^{nc}}(t,\{\alpha_j\})}$ we would have got the same weight for each order in this example.

The computational complexity of the Shapley value is exponential with respect to the number of orders composing a Journey. In fact, following Equation 12, for each non-collaborative journey J^{nc} we should solve $2^{|J^{nc}|}$ TSPs, one for each subset of J^{nc}.

In our use case we may have more than ten orders delivered with a single journey, and the computational time required for an exact computation of the Shapley value with this amount of orders is too slow (see Section 7). However, we have several orders with the same pick up and drop location. We thus compute the Shapley values considering orders with the same pick up and drop location as if they were one single order, and equally split the corresponding Shapley value among the orders grouped together. The resulting weights will be different, however dividing the KPIs between stretches of journeys and equally splitting the KPIs among orders with the same pick up and drop location sounds a reasonable approach which could also result in a more desirable solution. The pseudocode of this variation is shown in Algorithm 2.

Example 6.2. Consider again the setup of Example 6.1 shown in Figure 1. We could group the orders α_1 and α_2 and consider them as if they were one unique order α_{12}. The Shapley values would then be:

$$\begin{aligned}\phi(\alpha_{12}) &= \frac{1}{2}\text{TSP}_{J^{nc}}(t,\{\alpha_{12}\}) \\ &\quad + \frac{1}{2}(\text{TSP}_{J^{nc}}(t,\{\alpha_{12},\alpha_3\}) - \text{TSP}_{J^{nc}}(t,\{\alpha_3\})) \\ &= \frac{14}{2} + \frac{6}{2} = 10\end{aligned} \tag{19}$$

$$\begin{aligned}\phi(\alpha_3) &= \frac{1}{2}\text{TSP}_{J^{nc}}(t,\{\alpha_3\}) \\ &\quad + \frac{1}{2}(\text{TSP}_{J^{nc}}(t,\{\alpha_{12},\alpha_3\}) - \text{TSP}_{J^{nc}}(t,\{\alpha_{12}\})) \\ &= \frac{14}{2} + \frac{6}{2} = 10\end{aligned} \tag{20}$$

Thus, we would get $\phi(\alpha_1) = \phi(\alpha_2) = \frac{\phi(\alpha_{12})}{2} = 5$, and the Shapley weights would then be:

$$\mu_1 = \mu_2 = \frac{\phi(\alpha_1)}{\phi(\alpha_1) + \phi(\alpha_2) + \phi(\alpha_3)} = 0.25, \tag{21}$$

$$\mu_3 = \frac{\phi(\alpha_3)}{\phi(\alpha_1) + \phi(\alpha_2) + \phi(\alpha_3)} = 0.5. \tag{22}$$

If also Algorithm 2 was too slow for a real application, one could consider the Monte Carlo approximation for the Shapley value presented in [3].

Algorithm 2 Simplified Shapley Value

procedure $SSV(J^{nc}, t)$
  $\phi(\alpha) \leftarrow 0$ for each $\alpha \in J^{nc}$
  $H(\alpha) \leftarrow \emptyset$ for each $\alpha \in J^{nc}$
  $J^{nc\prime} \leftarrow \emptyset$
  for $\alpha \in J^{nc}$ **do**
    if $\exists \beta \in J^{nc\prime}$ with same pick up and drop location of α **then**
      $H(\beta) \leftarrow H(\beta) \cup \alpha$
    else
      $J^{nc\prime} \leftarrow J^{nc\prime} \cup \alpha$
  $n \leftarrow |J^{nc\prime}|$
  $\mathcal{S}_0 \leftarrow \{\emptyset\}$
  $\mathcal{S}_i \leftarrow \emptyset$ for each $i \in [1, \dots n-1]$
  $L(\emptyset) \leftarrow 0$
  for $m \in [0, \dots n-1]$ **do**
    $c \leftarrow \frac{m!(n-m-1)!}{n!}$
    for $I \subseteq J^{nc\prime}$ with $|I| = m+1$ **do**
      $\mathcal{S}_{m+1} \leftarrow \mathcal{S}_{m+1} \cup \{I\}$
      $L(I) \leftarrow (\text{TSP}(t, I))$
    for $I \in \mathcal{S}_m$ **do**
      for $\alpha \in J^{nc\prime} \setminus I$ **do**
        $\phi(\alpha) \leftarrow \phi(\alpha) + c(L(I \cup \alpha) - L(I))$
  for $\alpha \in J^{nc}$ **do**
    if $H(\alpha) \neq \emptyset$ **then**
      for $\beta \in H(\alpha)$ **do**
        $\phi(\beta) \leftarrow \frac{\phi(\alpha)}{|H(\alpha)|+1}$
      $\phi(\alpha) \leftarrow \frac{\phi(\alpha)}{|H(\alpha)|+1}$
  return ϕ

7 EXPERIMENTAL RESULTS

We briefly summarise some results of our experimental testing. All experiments were performed on a computer facilitated by a Core i5 2.70 GHz processor and 8 GB RAM. We used the Java library Jsprit to compute TSP problems, and we tested our algorithms with instances randomly generated.

Table 2 and Table 3 show the time performance of Algorithm 1 and Algorithm 2 with respect to the number of orders and the number of pairs pick up and drop locations. As we can see, the execution time of Algorithm 1 grows exponentially with respect to the number of orders, and it is not very sensitive with respect to the number of pairs pick up and drop locations (which cannot be greater than the number of orders since each order is associated with only one pair of locations). On the other hand, the execution time of Algorithm 2 grows exponentially with respect to the number of pairs of pick up and drop locations, and it is not very sensitive with respect to the number of orders. This is due to the computational burden of our algorithms which is related to the computation of the Shapley values, whose complexity is related to the number of orders for Algorithm 1, and to the number of pairs pick up and drop locations for Algorithm 2. In our context we should not have more than six pairs of pick up and drop off locations for each journey, but we could have more than ten orders. Therefore Algorithm 2 seems

Table 2: Execution time of Algorithm 1 varying the number of orders and the total number of pairs pick up and drop locations

N. orders	N. locations pairs	Time [s]
2	1	2.8
2	2	3.6
4	1	5.1
4	4	5.5
6	1	7.6
6	6	9.3
8	1	23.2
8	8	27.2
10	1	80.2
10	10	86.7

Table 3: Execution time of Algorithm 2 varying the total number of pairs pick up and drop locations and the number of orders.

N. locations pairs	N. Orders	Time [s]
2	2	3.2
2	20	3.6
4	4	4.8
4	40	5.2
6	6	10.1
6	60	11.8
8	8	22.3
8	80	24.3
10	10	83.9
10	100	85.2

to be the best choice since it would improve the time response of the system.

8 CONCLUSIONS

With this work we presented a preference elicitation method for collaborative Vehicle Routing Problem developed within the European project LOGISTAR. In particular, we have shown how we can learn the preferences of the decision-makers involved in collaborative journeys, by observing their interactions with the system, and translating input preference information into constraints for a SVM algorithm estimating a preference model. The preference model considered is the weighted sum of a set of KPIs evaluating the collaborative journeys, and it defines the trade-offs over the KPIs. With our approach we learn a preference model for every partner involved, and also a preference model considering the preferences of all the partners. We also have shown a method based on the Shapley value to evaluate the improvement or disimprovement on the KPIs arising from the collaboration. This method computes an estimated non-collaborative KPIs vector which can be compared with the KPIs vector associated with a collaborative journey with the purpose of evaluating the collaboration.

Future work could involve alternative techniques to translate the motivations of the decisions into preference constraints, the evaluation of different preference learning approaches such as Bayesian methods or robust ordinal regression, and methods for the estimation of joint preference models with more than two partners.

ACKNOWLEDGMENTS

This material is based upon works supported by the Science Foundation Ireland under Grant No. 12/RC/2289-P2 which is co-funded under the European Regional Development Fund, by the LOGISTAR project, which is funded by the European Commission under the Horizon 2020 programme. We are grateful also to our various partners on the LOGISTAR project.

REFERENCES

[1] N. Benabbou, P. Perny, and P. Viappiani. 2017. Incremental elicitation of Choquet capacities for multicriteria choice, ranking and sorting problems. *Artificial Intelligence* 246 (2017), 152–180.
[2] Craig Boutilier. 2002. A POMDP formulation of preference elicitation problems. In *Proceedings of AAAI/IAAI*. 239–246.
[3] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research* 36, 5 (2009), 1726–1730.
[4] Sascha Dahl and Ulrich Derigs. 2011. Cooperative planning in express carrier networks—An empirical study on the effectiveness of a real-time Decision Support System. *Decision Support Systems* 51, 3 (2011), 620–626.
[5] Bo Dai and Haoxun Chen. 2012. Mathematical model and solution approach for carriers' collaborative transportation planning in less than truckload transportation. *International Journal of Advanced Operations Management* 4, 1-2 (2012), 62–84.
[6] Dave de Jonge, Filippo Bistaffa, and Jordi Levy. 2021. A Heuristic Algorithm for Multi-Agent Vehicle Routing with Automated Negotiation. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 404–412.
[7] Elena Fernández, Mireia Roca-Riu, and M Grazia Speranza. 2018. The shared customer collaboration vehicle routing problem. *European journal of operational research* 265, 3 (2018), 1078–1093.
[8] Margaretha Gansterer and Richard F Hartl. 2018. Collaborative vehicle routing: a survey. *European Journal of Operational Research* 268, 1 (2018), 1–12.
[9] Salvatore Greco, Vincent Mousseau, and Roman Słowiński. 2014. Robust ordinal regression for value functions handling interacting criteria. *European Journal of Operational Research* 239, 3 (2014), 711–730.
[10] Salvatore Greco, Roman Słowiński, José Rui Figueira, and Vincent Mousseau. 2010. Robust ordinal regression. In *Proceedings of Trends in multiple criteria decision analysis*. Springer, 241–283.
[11] Shengbo Guo and Scott Sanner. 2010. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 289–296.
[12] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 133–142.
[13] Mojtaba Montazery and Nic Wilson. 2016. Learning User Preferences in Matching for Ridesharing.. In *ICAART (2)*. 63–73.
[14] Christoph Schneeweiss. 2012. *Distributed decision making*. Springer Science & Business Media.
[15] Lloyd S Shapley. 2016. *17. A value for n-person games*. Princeton University Press.
[16] Federico Toffano, Paolo Viappiani, and Nic Wilson. 2021. Efficient Exact Computation of Setwise Minimax Regret for Interactive Preference Elicitation. In *20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*.
[17] Paolo Viappiani and Craig Boutilier. 2009. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 101–108.
[18] Paolo Viappiani and Craig Boutilier. 2020. On the equivalence of optimal recommendation sets and myopically optimal query sets. *Artificial Intelligence* (2020), 103328.
[19] Wenyu Zhang, Zixuan Chen, Shuai Zhang, Weirui Wang, Shuiqing Yang, and Yishuai Cai. 2020. Composite multi-objective optimization on a new collaborative vehicle routing problem with shared carriers and depots. *Journal of Cleaner Production* 274 (2020), 122593.