On Inferring a Meaningful Similarity Metric for Customer Behaviour

Sophie van den Berg and Marwan Hassani[0000−0002−4027−4351]

Eindhoven University of Technology, The Netherlands
`sophievandenberg4@gmail.com`, `m.hassani@tue.nl`

Abstract. In omnichannel customer service environments, where no real process is enforced, a wide variety of customer journey variants exists. This variety makes it complex to find process improvement opportunities. Modeling the journeys as traces is an essential step before discovering an explainable model of various behaviours. Trace clustering helps improvement efforts by separating the journeys into homogeneous subsets in terms of behaviour and purpose. For this, a one-size-fits-all distance metric has been used so far in the literature. This paper shows that a domain-informed similarity metric will improve customer journey clustering compared to a generic one. We propose SIMPRIM framework, which uses clustering quality metrics to develop a similarity metric that maximizes the separability of the journeys in a low dimensional space while agreeing with existing process knowledge. Experimental evaluation on real life use cases of a large telecom company and a benchmark dataset show that, compared to a generic metric, respectively a 46% and 39% improvement can be obtained in terms of the internal clustering quality while keeping the external clustering quality equal. We also show that the inferred metric can be useful for prediction applications.

Keywords: Similarity Metric, Customer Journey Clustering

1 Introduction

In today's business environment, delivering a superior customer experience is becoming a priority to compete. Customer experience is the result of every interactions a customer has with a business, from navigating the website to using the product or talking to a customer service agent. The sequential steps and interactions, i.e. touchpoints, that a customer goes through for accessing or using a product, is referred to as a customer journey. Analyzing customer journeys is an extremely useful exercise for companies that aim to understand and improve the customer experience for their users as interactions do not occur in isolation. Nowadays, many companies are adopting a data-driven way of working in which a lot of data about customer contact is collected. However, data-driven methods for customer *journey* analysis are still very limited. Recently, [1] highlighted that process mining techniques are suitable for exploring customer behaviour. Such techniques aim to extract useful information about process execution by analyzing event logs. In the scope of customer journey analysis, the process can be considered the sequence of touchpoints in a customer journey for which at least its timestamp, activity and a journey identifier are stored. Traditional process mining approaches expect processes to be well-structured and limited in scope [8]. Customer journeys, on the other hand, are often derived from processes that have opposite characteristics. Customers can usually operate in very *flexible* environments where no process is enforced. To offer the best possible customer experience, companies provide omnichannel customer service which yields high customer journeys variability: for reaching each goal, a customer has countless ways. There are inherent problems of applying process mining techniques to flexible environments like these. The corresponding event log is very diverse as it captures a wide spectrum of behaviour, both in terms of topic and journey length, meaning journeys vary significantly from each other. This typically yields so-called spaghetti-like process models that are large, highly unstructured and essentially useless for further analysis [12]. To overcome this issue, [12] propose to use Trace Clustering in which the event log is split into homogeneous subsets. Separately, these processes are significantly more structured than the complete process and thus yield more interpretable process models. Using this approach, journeys could be clustered based on *customer intent*, i.e. the purpose of and the behaviour behind customer contact. Due to the high journey variety, both aspects are required to find clusters that reflect similar journeys. To cluster journeys in a meaningful way, an appropriate notion of journey similarity is of a critical importance. However, current approaches to trace clustering solely make use of standard distance metrics (e.g. Euclidean, Cosine or Jaccard). They are assumed to create meaningful clusters for journeys belonging to a wide spectrum of processes ranging from hospital patients to customer webclicks. This assumption seems to be violated in practise where for instance different pieces of information are relevant in varying percentages (cf. Fig. 1). Therefore, the suitability of a certain similarity function for a given business problem depends on the characteristics of the data and nature of the problem. Hence, we challenge the current usage of 'naive' metrics and hypothesise that a meaningful similarity metric is much more suitable. Such a metric can be further useful in any

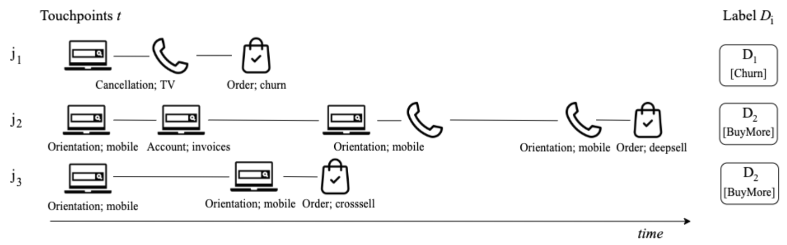

Fig. 1: Schematic overview of touchpoints t from 3 example journeys j_i in telecom journey log L (cf. Def. 1) and the corresponding journey labels D_i (cf. Def. 9). Standard distance metrics will fail in deciding which of j_i are closest to each other. The perspective V_i (cf. Def. 2) from which one looks at j_i matters too.

prediction and/or recommendation task beyond clustering. As such, inferring a meaningful similarity metric is essential for advanced journey analysis in general.

In this paper, we propose a methodology for the development of a domain-informed similarity metric for customer journeys from *flexible* environments. Predominantly focusing on, but not limited to, *customer service* environments from which the trivial examples in Fig. 1 are illustrated. The main use case of this work comes from a leading telecommunication company that we will refer to as Anonycomm. The core of the framework is based on unsupervised learning (clustering), while domain knowledge in the form of journey annotation is used to support the development of the metric along the way as much as needed. This approach is hypothesized to yield journey analysis results that could not be found by domain experts but are in line with their existing knowledge. We can assume that a rough journey categorization is available since information of this kind is available in almost every data science problem or can be created using automatic labelling techniques. However, this categorization is often imperfect or totally wrong. Often, only a high-level journey categorization (e.g. topic-based) is available in which high variability of journeys is expected. Therefore, the similarity metric should not overfit on that domain knowledge.

The paper is organized as follows: Section 2 formalizes the addressed problem. Section 3 introduces the proposed framework to develop a meaningful customer journey similarity metric. In Section 4 we demonstrate the usefulness of the framework using two real-life event logs. In Section 5 we extensively discuss related work before concluding the paper in Section 6.

2 Problem Definition

Below, we introduce some definitions and subsequently we formally define the problem of inferring a meaningful similarity metric out of customer behaviour. Def. 1 & 2 are adopted from [8] but adjusted to the customer journey application.

Definition 1 (Journey log). *Let A be a set of attribute names. Let T be the set of all touchpoint identifiers. $a(t)$ is the value of attribute $a \in A$ for touchpoint $t \in T$. Typically, the following attributes are present in touchpoints: activity(t), time(t), channel(t), see examples in Fig 1. Other touchpoint attributes can be the cost, resource or activity outcome. Journeys, like touchpoints, have attributes. Let J be the set of all journey identifiers. $a(j)$ is the value of attribute $a \in A$ for journey $j \in J$. Each journey has a mandatory attribute 'trace': trace$(j) \in T^*$. A trace is a finite sequence of touchpoints $\sigma \in T^*$. If t or j do not have a, then $a(t) = \perp$ or $a(j) = \perp$ (null value). A journey log is a set of journeys $L \subseteq J$. A log is complex if it contains journeys from a less structured process, i.e. no enforced process, ('structured' defined in [18]), where the activity (and channel) set is large, due to which L contains many journey variants.*

Definition 2 (Perspective, journey profiles). *Let $V = \{V_1, V_2, ..., V_H\}$ be a set of perspectives, views on a journey. map$_V : J \rightarrow \mathbb{R}^n$ denotes the function that maps a journey to an n-dimensional vector according to perspectives V.* $\mathbf{p_V}(\mathbf{j})$

denotes the projection of journey $j \in L$ to perspectives V. Furthermore, we let $\mathbf{p_V(j)} = p_{\{V_1,V_2,...,V_h\}}(j) = map_{V_1}(j) \mid\mid map_{V_2}(j) \mid\mid ... \mid\mid map_{V_h}(j)$, i.e. $\mathbf{p_V(j)}$ is a journey profile vector in which all perspectives are concatenated.

Definition 3 (Feature set). *Let $F(n) = \{f_1, f_2, ..., f_n\}$ be the full set of features in journey profile $\mathbf{p_V(j)}$. $F(m) \subseteq F(n)$ denotes a feature subset that contains m most relevant features according to a specific feature selection technique where $m \leq n$. Journey profiles reduced to this subset are denoted by $\mathbf{p_{F(m)}(j)}$.*

Definition 4 (Similarity metric). *We define a similarity function $S(j, j', w)$, S in short, is a linear function parameterized by a set of weights $\mathbf{w}$ that defines pairwise similarities of customer journeys j in a journey log $L \subseteq J$. Here, $\mathbf{w} = (w_1, w_2, ..., w_m)$ is a set of weights where $w_i \in [0, 1]$. S operates over journey profile vectors $\mathbf{p_{F(m)}(j)}$ and has the following properties:*

- *For the sake of interpretability, S is bounded between 0 and 1.*
- *When two journeys j and j' have exactly similar profiles, then $S(j, j') = 1$;*
- *If journey j is more similar to j' than to j'', then $S(j, j') > S(j, j'')$;*
- *S is symmetric, i.e. $S(j, j') = S(j', j)$.*

Definition 5 (Universal similarity metric). *A universal similarity metric S_U is a standard similarity metric, e.g. Cosine similarity, that uses a set of uniform weights $w_i = 1$ and operates over $F(n)$.*

Definition 6 (Trace clustering). *A trace clustering $TC = \{TC_1, TC_2, ..., TC_k\}$ is a set of k trace clusters over journey log L. We assume a hard clustering, i.e. every journey is part of exactly one trace cluster. TC is the result of a convex-based clustering algorithm with a pre-defined number of clusters k. In theory, other clustering techniques such as hierarchical clustering could also be used but these would not make use of the initial guess from domain experts about k that we have in our problem (Def. 9) and would yield other parameter problems.*

Definition 7 (Internal trace clustering quality). *Internal quality of a trace clustering is measured using an internal index validity statistic γ that expresses the quality of a clustering TC in terms of the cohesion of traces within the same cluster and the separation between traces in different clusters. For a clustering TC' that has a higher internal quality than TC, we write $\gamma(TC') > \gamma(TC)$.*

Definition 8 (External trace clustering quality). *External quality of a trace clustering is measured using an external validity statistic δ that expresses the quality of a clustering TC based on the correspondence between TC and a labeling D (cf. Def. 9). For a clustering TC' that has more over overlap with labeling D than TC, we write $\delta(TC') > \delta(TC)$.*

Definition 9 (Domain knowledge). *Domain knowledge provides a journey labelling $D = \{D_1, D_2, ..., D_b\}$ that assigns a journey $j \in L$ to exactly one label. This labeling represents domain experts' intuition about a (high-level) clustering structure of L. The domain knowledge also informs about the possibility of journeys being separated over much more labels than b, namely g, but $g << |J|$. Due* *to the existence of many journey variants, D_i could still include very different behaviour. Finally, it also specifies δ_{min}, the lowest acceptable $\delta(TC)$ with respect to the similarity metric learning problem.*

Definition 10 (Meaningful similarity metric). *An optimal similarity metric S^* maximizes $\gamma(TC)$ for $k = b$ clusters (here b is derived from D). This metric is considered meaningful if a TC' with a number of clusters k up to g is of comparable (or better) quality, i.e. $\gamma(TC')$ is comparable to $\gamma(TC)$.*

Definition 11 (Optimal feature set). *A feature set $F(m)$ is considered optimal if journey profiles $\mathbf{p_{F(m)}}(\mathbf{J})$ maximizes $\gamma(TC)$ while keeping $\delta(TC) > \delta_{min}$. This set is denoted by $F(*)$.*

This work aims at inferring a meaningful similarity function S^* that defines similarities between customer journeys in a complex journey log $L \subseteq J$ based on *customer intent.* Using S^*, a trace clustering TC can be created. For a specific clustering algorithm, S^* maximizes $\gamma(TC)$ for $k = b$ clusters while $\delta(TC) > \delta_{min}$.

Initially, domain knowledge D is helpful for the similarity metric development but at some point it becomes questionable. The journey labeling D is not blindly trusted, it is not considered the ground truth. This is a typical setting for a semi-supervised approach where the domain knowledge is considered a good starting point. The solution approach aims to collaborate with the existing knowledge but also extends this knowledge (in particular for $k > b$).

3 SIMPRIM Framework

We propose SIMPRIM, a framework for Similarity Metric learning for Process Improvement, that simultaneously learns a similarity metric S and a journey clustering TC. By integrating metric optimization techniques with trace clustering and domain knowledge in a joint framework, S maximizes the separability of the journeys in a low dimensional space while agreeing with existing domain knowledge D. Since clustering results are heavily influenced by the metric that is used, the quality of S is approximated with the quality of the corresponding TC. The methodology is formalized in Algorithm 1. For each step, a set of suitable techniques is selected to experiment with, such that no assumptions have to be made about the effectiveness of a technique for a specific dataset. This makes the framework applicable to manifold applications contexts.

3.1 Journey Log to Journey Profiles

As a first step, journeys $j \in L$ are translated into a format on which similarity can be calculated. SIMPRIM adopts the abstract representation approach from [12] in which journey profiles $p_V(J)$ are used. While traditional approaches only describe a journey from the control-flow perspective [7], this approach also allows for the inclusion of other trace perspectives V. Fig. 1 shows the necessity of including different perspectives to create a meaningful journey separation.

Algorithm 1 SIMPRIM: Finding a domain-informed similarity metric for customer journeys and a meaningful journey clustering

```
1: Determine journey perspectives $V$ and find profiles $\mathbf{p_V}(\mathbf{J})$;    ▷ Cf. Sec. 3.1
2: Find $S_U{}^*$ that maximizes $\gamma(TC)$ for $\mathbf{p_V}(\mathbf{J})$;    ▷ Cf. Sec. 3.2
3: $F(*) \leftarrow FeatureSection(\mathbf{p_V}(J),\ D,\ S_U{}^*)$;    ▷ Cf. Sec. 3.3
4: $\mathbf{w}^*, TC^* \leftarrow WeightOptimization(\mathbf{p}_{F(*)}(J),\ D,\ S_U{}^*)$;  ▷ Cf. Sec.3.4 & Alg.2
5: $S^* = S(\mathbf{p_{F(*)}}(\mathbf{j}),\ \mathbf{p_{F(*)}}(\mathbf{j'}),\ \mathbf{w}^*)$;
6: Qualitatively evaluate $S^*$ and $TC^*$ and adjust previous steps if
   required.    ▷ Cf. Sec 3.5
```

Perspectives can be based on both journey and touchpoint attributes. Common perspectives are the Activity, Originator, Transition, Event-Attributes, Case-Attributes and Performance perspective [12] [15]. Custom perspectives can be added if preferred. Carefully designing the journey profiles can improve the quality of TC and thus S.

SIMPRIM uses function $map_V : J \rightarrow \mathbb{R}^n$ to map the journeys into journey profiles (Def. 2). Most perspectives V mark the presence of a certain attribute (hot-encoding). For the mapping function, two representation techniques are compared within the framework: Bag-Of-Activities (BOA) [2], and Set-Of-Activities (SOA), indicating a real and a binary attribute count respectively. For complex journey logs, this mapping typically yields numerical trace vectors that are very high dimensional and sparse. This is caused by the large set of attributes available of which journeys often only contain a very small subset.

3.2 Measuring Similarity

To express the similarity between any two journey profiles that are represented as n-dimensional vectors, a number of universal similarity metrics S_U can be used. The *weighted* Jaccard similarity [4] and Cosine similarity are considered most suitable for our problem. They operate efficiently on sparse and high-dimensional vectors, are bounded between $[0, 1]$ and proven effective in existing work [8] [12]. We define $Cosine\ similarity(j, j') = \frac{\mathbf{p_{F(m)}}(\mathbf{j}) \cdot \mathbf{p_{F(m)}}(\mathbf{j'})}{||\mathbf{p_{F(m)}}(\mathbf{j})||_2 ||\mathbf{p_{F(m)}}(\mathbf{j'})||_2}$ and $Jaccard\ similarity(j, j') = \frac{\sum_{i=1}^{m} min(\mathbf{p_{F(i)}}(\mathbf{j}),\ \mathbf{p_{F(i)}}(\mathbf{j'}))}{\sum_{i=1}^{m} max(\mathbf{p_{F(i)}}(\mathbf{j}),\ \mathbf{p_{F(i)}}(\mathbf{j'}))}$ which is set to 1 if the denominator is 0. Their different notion of similarity makes them interesting to compare. While Jaccard similarity is originally designed for binary sets, here we use the *weighted* version. No actual weights are assigned to features but it means that the metric can also be used on non-negative real vectors $\mathbb{R}$. This ensures that there is a difference in measuring the similarity using the BOA versus SOA trace representation.

3.3 Dimensionality Reduction

Many issues arise when applying clustering and weight optimization techniques on high dimensional vectors. Therefore, *feature selection* is applied to reduce a journey profile to the most effective feature subset. Such techniques are found more suitable here than *feature extraction* techniques (e.g. PCA) since the interpretability of the metric is considered highly important for the business use case. SIMPRIM allows for experimentation with a variety of techniques to find $F(*) \subseteq F(n)$ on which S will operate. In theory, any *filter* or *wrapper* feature selection method can be incorporated. However, techniques that are *embedded* in a clustering algorithm are not suitable for this framework as no dependence on specific techniques is desired. Besides supervised techniques, we suggest experimenting with unsupervised techniques since the quality of of labeling D might be poor. Besides, an optimal clustering from a label perspective, i.e. $TC = D$, could have a very low $\gamma(TC)$ in which case γ and δ are competitors. In that case, supervised feature selection might not lead to the desired optimization.

Feature Set Size Restrictions To be able to develop a *meaningful* similarity metric, the size of feature set m is restricted by a lower bound. A very small m yields a non-meaningful metric because (1) journey profiles $\mathbf{p_{F(m)}(J)}$ might contain too little information to separate journeys correctly for $k > b$, and (2) it can yield a large set of exactly similar journey profiles which causes all these journeys to be put in the same cluster $TC_i \in TC$ that cannot be separated by increasing k since for these profiles $S = 1$. Therefore, SIMPRIM puts an indirect lower bound on the feature set size using Def. 12. On the other hand, when dimensionality increases the difference between the min and max pairwise similarity, i.e. *similarity contrast*, becomes really small which makes the similarity values indistinctive. If similarity is only expressed in a small part of the $[0, 1]$ range, its interpretation is not intuitive and S would not be a useful stand-alone product. Therefore, Def. 13 puts a minimum on the similarity contrast which indirectly upper bounds m. All remaining values for m are considered for finding $F(*)$.

Definition 12 (Optimal Feature Set Lower Bound). *A feature set $F(m)$ can only be optimal, i.e. $F(*)$, if the corresponding number of exactly similar journey profiles $\mathbf{p_{F(m)}(j)}$ for journey $j \in L$ does not exceed $max(\alpha|D_i|)$ for $D_i \in D$ where $\alpha \in [0, 1]$ is a framework parameter.*

Definition 13 (Optimal Feature Set Upper Bound). *Let $H \in J$ be the set of journey with pairwise distances in J that lay within two standard deviations from the mean distance between any two points in J, i.e. $\mu_J \pm 2 * \sigma_J$. We define the similarity contrast c as $max(H) - min(H)$. Now, a feature set $F(m)$ can only be optimal, i.e. $F(*)$, if $c > \beta$ where $\beta \in [0, 1]$ is a framework parameter.*

3.4 Co-learning of Metric Weights and Journey Clustering

Unweighted metrics assume all features are of equal importance to distinguish traces, while in reality their importance for finding a good clustering structure differs. Handpicking weights in a meaningful way is not a trivial task. To tune distance metrics automatically, feature weighting techniques can be used that learn a set of weights $\mathbf{w}^*$ in domain W that optimizes a given objective function.

Algorithm 2 Weight optimization using an SBMO method [10]

```
Input: uniform weights $w$, surrogate model $f$, objective function $O$, acquisition function $\mathit{\Gamma}$, stopping criteria
Output: optimal weights $\mathbf{w}^*$
1: $R \leftarrow \{\}$
2: while stopping criteria are not met do
3:     $\mathbf{w} \leftarrow SMBO(R)$        {Fit $f$ and maximize $\mathit{\Gamma}$}
4:     $\lambda_\mathbf{w} \leftarrow O(TC_\mathbf{w})$        {Evaluate weight set}
5:     $R \leftarrow R \cup \{(\mathbf{w}, \lambda_\mathbf{w})\}$    {Add to results}
6: $\mathbf{w}^* \leftarrow argmax_{(\mathbf{w},\lambda_\mathbf{w}) \in R}\ \lambda_\mathbf{w}$
7: return $\mathbf{w}^*$
```

We use the clustering quality as objective such that the weights and the clustering are optimized simultaneously. We refer to this as a *co-learning* approach.

Weight Optimization For weight optimization we use Sequential-Model-Based-Optimization (SMBO), i.e. Bayesian optimization. SMBO is commonly used for hyper parameter tuning and is very efficient for expensive objective functions O [10]. Besides, it can take any objective function and the resulting weights w^* can be added to an arbitrary metric to adapt the scaling of dimensions. The SBMO algorithm is described in Algorithm 2. An acquisition function $\mathit{\Gamma}$ is used to maximize over a cheap surrogate model and find a new set of candidate weights. Because of our expensive objective (clustering) function, informed weight sampling could yield great efficiency by finding the optimum with as few evaluations as possible. The algorithm stops when (1) x sequential function calls did not improve the clustering quality or (2) after a specified maximum number of calls.

We aim to develop a similarity metric S^* that maximizes $\gamma(TC)$ while keeping $\delta(TC)$ at an acceptable level. Therefore, both quality criteria are included in the optimization objective (Eq. 1). The contribution of δ is parameterized by $\theta \in [0, 1]$, and can be determined based on the quality of journey labeling D.

$$O(TC_\mathbf{w}) = \theta \cdot \delta(TC_\mathbf{w}) + (1 - \theta) \cdot \gamma(TC_\mathbf{w}) \tag{1}$$

The surrogate model should reflect the actual objective as much as possible. Within SIMPRIM, we experiment with two surrogate models: (1) a Gaussian Process (GP) (most common) [10] and a Gradient Boosting Regression Trees (GBT) model [3]. Both are available in the `scikit-optimize` Python library.

Clustering While any convex-based clustering algorithm can be used, k-Medoids is selected based on its simplicity and speed. It is preferred over k-Means as its clusters are represented by actual journeys, which allows for more intuitive interpretation. Throughout the optimization, both the clustering algorithm and k remain fixed. This allows to optimize TC by varying S merely.

3.5 Evaluation

The quality of S is approximated with the quality of its corresponding TC. Many clustering validation indices exist. Based on [11], we use the revised validity index (S_Dbw) for γ, a metric that resembles the well-known Silhouette index. S_Dbw is a summation of inner-cluster compactness and inter-cluster separation. A lower value indicates a better quality. For δ, the adjusted rand index ARI and adjusted mutual information AMI are commonly used (a higher value indicates better quality). In the assessment of a more fine-grained cluster output ($k > b$), for which no labelled data exists, we test the robustness of the approach by involving domain experts. This yields a *qualitative evaluation* instead of a quantitative one.

4 Experimental Evaluation

The usefulness of SIMPRIM is demonstrated by applying it to a real-life use case with a leading telecom provider in Sec. 4.1. Additionally, we evaluate our framework over another real world benchmark dataset in Sec. 4.2 which is hosted by its owners here: `https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204`. Our Python implementation of SIMPRIM is accessible via `https://github.com/sophievdberg/SIMPRIM`.

4.1 Customer Service Process at Anonycomm

Application Scenario Our work is performed in collaboration with a large telecom provider and is inspired by their data. Since they would like to stay anonymous, we will refer to them as Anonycomm. Hundreds of thousands of customers interact with Anonycomm each month using multiple channels. Offering a superior customer experience is a high priority for Anonycomm as competitors have increasingly similar service offers and devices. As such, they aim to improve their business process towards a more efficient and self-service user portal. However, the unstructured nature of their customer support process allows for a wide variety of journey types, which makes it very difficult to find behavioural improvement opportunities. A meaningful journey clustering could provide a more complete picture of the average journey per customer intent. This allows for better sizing and prioritization of improvement efforts.

Journey log L used for this research is a collection of customer contact moments (touchpoints t) in a 3-month period. A customer is getting into contact for many different reasons ($activity(t)$), such as having a question about a service, acquiring a new subscription or the installation of a new piece of hardware. They do so using a contact type such as call, mechanic visit or order placement ($channel(t)$). Finally, for all touchpoints t we have a more detailed description/reason of $activity(t)$ stored in $eventtype(t)$, e.g. 'disruption after installation'. The log is considered *complex* because of the large number of channels, activities and event-types (18, 80 and 767 respectively) and heavily varying journey lengths (2-183 touchpoints). Journeys with only one touchpoints are removed. In total, this leaves us with around half a million journeys in the dataset.

Journey labeling D distinguishes 53 journey categories. For example, 'WiFi disruption' or 'Extra TV subscription' or 'Move'. The labeling is done by domain experts based on a large set of business rules that assign a touchpoint to a journey (=class based on the presence of specific channels, activities, event types and time). Domain experts are not completely sure about their labeling. Besides, the labeling is performed from a reason/topic-based perspective on journeys which motivated us to find similarities of journeys amongst two axes: topic and behaviour. This indicates that the metric S should not overfit on the labeling D (i.e. relatively small δ_{min} and limited contribution of δ in the weight optimization process). The distribution of labeling D is skewed and since some of the categories are still relatively big, domain experts indicated that dividing the journeys up to 100 clusters could be meaningful ($g \approx 100$).

Implementation Details Journeys are clustered using the Partitioning Around Medoids (PAM) algorithm. Since this clustering algorithm is sensitive to the set of initial medoids, we run each experiment 5 times and report the average clustering quality. Furthermore, a label-based medoid initialization is used (1 journey per class). This initialization technique outperformed the kmeans++ initializer for our data. For the external cluster validation, AMI index is used since it better suits the unbalanced labeling D (small clusters exist). We experiment with 2 supervised feature selection techniques, **Fast-Based-Correlation-Filter** and **L1-regularization**, and with 2 unsupervised techniques, **Variance** and **Laplacian score**. These are selected since they are common and relatively efficient. Note that SIMPRIM allows for the usage of other feature selection techniques too. The journey profiles are scaled in the range $[0, 1]$. Since the unsupervised techniques do not take correlation into account, features with a covariance ≥ 0.8 are removed. A regularization of 0.1 is used for $L1$.

We have recommendations about parameters: we set $\alpha = 0.5$ (Def. 12), $\beta = 0.2$ (Def. 13) and $\theta = 0.3$ (Eq. 1). A lower value for θ is used due to the limitations of labeling D. Stopping criteria are 60 iterations without an improvement or a maximum of 150 iterations. Note that neither the scope of the paper nor the space capacity allow us to do an extensive parameter sensitivity evaluation.

The similarity metric is optimized on 3 training sets to assess overfitting and tested on 3 test sets to evaluate its stability. Splits are stratified on class labels. Due to memory issues, clustering could only be done on 20.000 journeys simultaneously using our machine (2.3GHz Intel Core, 16GB RAM). Other than that, with this implementation we did not experience any run out of memory issues or freezing of the experimentation. However, if one decides to use another technique and SIMPRIM is selecting multiple iterations of the same task, then the user should select an efficient model: our framework does not make the approach more efficient. The total running time was around 20 hours.

Experimental Results For Step 1 in Alg. 1, we translate journeys in L into numerical journey profiles using 4 perspectives V. V_1 is the Activities profile that hot-encodes event and sub-event types. V_2 is the Event Attribute profile

Table 1: Clustering quality results for universal metrics S_U in the Anonycomm dataset. Error indicates the σ over the training and test sets (i.e. stability).

Metric	Trace Type	S_Dbw (γ)	AMI (δ)
Cosine	**BOA**	**1.078** ± 0.015	**0.526** ± 0.010
	SOA	1.049 ± 0.030	0.531 ± 0.045
Jaccard	BOA	1.046 ± 0.054	0.403 ± 0.062
	SOA	1.051 ± 0.089	0.482 ± 0.005

that does the same thing for the contact types of a journey (i.e. *channel*). V_3 is the Transition profile that only includes 2-grams for channel types, not for activities and event types as they are extremely diverse and dimensions would explode. Finally, the Performance perspective V_4 includes the duration of the journey, the number of different channel types and the number of touchpoints in a journey. This mapping yields journey profiles with $n = 2756$ features.

To find the optimal universal similarity metric ${S_U}^*$ for the resulting profiles $\mathbf{p_V}(\mathbf{J})$, in Step 2 of Alg. 1 we compare the quality of the Jaccard and Cosine metric on both BOA and SOA journey profiles (4 metrics). Table 4 shows that, in terms of γ, all metrics have similar performance (differences within error bandwidth) but the Cosine metrics have slightly better stability. Furthermore, the Cosine metrics yield better results in terms of δ. The BOA representation is preferred as it shows a slightly more stable quality than SOA and it is preferred by domain experts since it aligns with their current view on journeys. We therefore find ${S_U}^*$ to be based on the Cosine metric and BOA journey profiles. Now that we have found ${S_U}^*$, we can start optimizing it. First, we reduce the dimensions it is operating on by evaluating the feature selection techniques (Step 3 in Alg. 1). Fig. 2 visualizes the clustering results on different feature sets F. The largest evaluated set has $m = 400$ since the results stabilize from that point. We find that L_1-regularization and Variance are not yielding any candidate feature sets for $m \leq 400$ (dashed lines). Additional evaluation indicates they yield too many similar journey vectors and thus do not meet Def. 12. Based on Fig. 2, the Laplacian feature set with 60 features can be considered $F(*)$. Using this feature set, γ is improved with 41% compared to the baseline ${S_U}^*$ and the clustering remains stable over the different train and test sets used. The external clustering quality δ for $F(*)$ remains comparable to ${S_U}^*$: only a 2.6% reduction is observed, which is still considered to overlap sufficiently with D. Only the supervised technique FCBF was able to slightly improve ${S_U}^*$ in terms of δ.

Using only the features in $F(*)$, we can optimize the weights (w) of S (Alg. 1, Step 4). Fig. 4 shows how the weight optimization techniques converge to a final set of weights. The GP surrogate model is able to minimize the objective function (Eq. 1) best, indicating it is better able to approximate the expensive clustering objective than the GBT model. Also, it shows a very efficient optimization process as it reaches its optimum very quickly. However, Table 2 indicates that the GBT weight sets yield better clustering quality when evaluated on the test sets. GP weights seem to suffer from slight overfitting on the training set, while this is

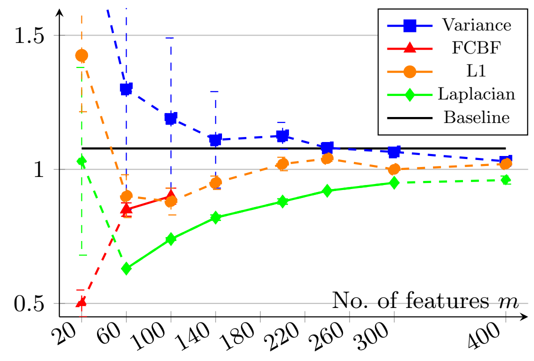

Fig. 2: Clustering quality γ for S operating over different $F(m)$ in Anonycomm dataset. Error bars indicate the σ over the training and test sets. Dashed lines indicate that the corresponding $F(m)$ does not fulfill Def. 12&13 and thus cannot be considered $F(*)$.

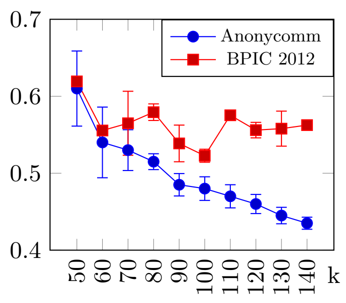

Fig. 3: Clustering quality γ for different k using S^*. For each k, 10 clusterings with different initial medoids sets are compared. Note: for the BPIC 2012 dataset, the x-axis should be divided by 10, i.e. read as $5 \geq k \leq 14$.

Table 2: Clustering quality γ and δ before (surrogate=None) and after weight optimization using two surrogate models in Anonycomm dataset.

Surrogate	Train γ	Test γ	%Δ	Train δ	Test δ	%Δ
None		0.634 ±0.011			0.520 ±0.010	
GP	0.594 ±0.049	0.641 ±0.032	-1.1	0.507 ±0.028	0.493 ±0.031	-5.2
GBT	0.597 ±0.021	**0.583** ±0.013	**+8.0**	0.518 ±0.025	**0.523** ±0.026	**+0.6**

not the case for GBT. A possible explanation for this could be that GP shrinks more weights to zero than GBT, for all training sets, while for generalization it is more safe to not completely remove them. Although the feature space is reduced, the feature set requirements are still met (Def. 12 & 13). The optimal metric S^* is thus based on GBT weights and operates on only 36 out of the original 2756 features. Overall, optimizing the weights yields an additional improvement of γ of 8%, while keeping δ at a comparable quality. In total, compared to ${S_U}^*$, S^* is able to improve the $\gamma(TC)$ with 46%, while maintaining a similar $\delta(TC)$. Fig. 3 shows that S^* is indeed *meaningful* as it shows a better γ for k up to g clusters. Around k =90-100, γ stabilizes, indicating this could be a more natural number of clusters (Elbow method). Finally, the results are qualitatively evaluated (last step of Alg. 1). In collaboration with a domain expert it is assessed whether journey clustering TC obtained with S^* makes sense. In general, the results were appreciated. Most clusters in TC show very similar behaviour and deal with the same problem type, especially for a more fine-grained clustering ($k = 100$). To improve the incorrect clusters, further experimentation can be performed by manually adding features that include information for better separation.

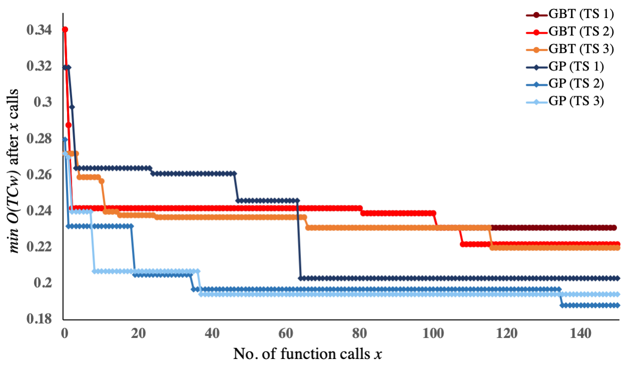

Fig. 4: Weight convergence per training set (TS) for weight optimization based on Gaussian Process (GP) or Gradient Boosting Trees (GBT) (Anonycomm).

Next Event Prediction The added value of S^* can also be demonstrated using a *model-based* approach. For any model that uses some notion of similarity, including meaningful similarity information might improve the prediction quality. For Anonycomm, we experimented with a model that predicts the contact type of next touchpoint t for a running journey j, to be able to get ahead of mistakes and prevent them beforehand ('on the fly') [16]. A baseline prediction model trained on all journeys is compared to a cluster-based approach in which one model is build per cluster. A prediction is then made by the model that corresponds to the cluster closest to the journey at that point in time according to S^*. This approach is hypothesized to have higher predictive abilities since the selected model selected is trained on journeys with similar behaviour.

To predict the next contact type of t_{i+1}, we can include all available information up until t_i. We included the information of the previous 4 touchpoints (i.e. 4 'lagged' features). A Random Forest model is used (for the baseline and cluster-based approach) and its predictive capability is expressed using the weighted F_1 score. Table 3 shows that the cluster-based approach does not improve the prediction quality when S^* is used. This could indicate that the clusters in TC are not highly discriminative in terms of contact types. Since S^* operates over a very small feature set ($m = 36$), it might not contain sufficient information regarding contact types. Hence, the performance of a S with more features $m = 140$, i.e. S_{140}, is also tested. Table 3 shows that this increases the prediction quality with 1.4%, which is small but significant. Note that the performance could be further improved by properly tuning the model but this is out of the scope of this paper.

4.2 BPIC 2012 Real Dataset

Since the data of Anonycomm is confidential, the SIMPRIM methodology is replicated on the publicly available BPIC 2012 event log, using Annonycom

Table 3: Weighted F-scores of the next contact type prediction models, comparing a baseline with cluster-based approaches (CBA). An 80/20 train-test split is made and 5-fold CV is used for the train scores. (Anonycomm dataset).

Prediction Model	Baseline	CBA with S^*	CBA with S_{140}
Training set	0.571	0.563	0.579
Test set	0.580	0.579	**0.594**

Table 4: Optimal clustering results of Step 2-3 in Alg. 1 (BPIC 2012 dataset).

	S_Dbw (γ)	AMI (δ)	Techniques
${S_U}^*$ `(Step 2)`	0.777 $\pm$0.016	0.639 $\pm$0.005	Cosine, BOA
S on $F(*)$ `(Step 3)`	0.704 $\pm$0.002	0.645 $\pm$0.003	Variance, $m = 50$

implementation details. This real-life event log contains 13087 journeys in the loan application process of a Dutch Financial Institute. It should be mentioned that SIMPRIM adds the most value for journeys from very flexible environments, while this event log has a much more sequential (structured) nature than Anonycomm's event log. Since no labeling D for this log exists, classes are derived based on the loan application outcome. We distinguish 6 classes: applications that are (1) accepted directly (only 1 offer); (2) accepted after some optimization of the offer; (3) rejected straight away, (4) rejected after an offer was drafted, (5) cancelled before an offer was drafted and (6) cancelled after an offer was drafted. D is heavily imbalanced: most applications are rejected straight away. The separation between with or without offer is created as corresponding journeys show very different journeys based on the absence or presence of "O_" states. Journeys without an event related to being accepted, rejected or cancelled are considered 'running' journeys and thus incomplete. Hence, they are removed from the dataset (2%). In the first step of SIMRPIM (Alg. 1), the journey profile mapping is done similarly as on the Anonycomm dataset. We only have *resources* instead of *contact types* in V_2 and since the number of activities is significantly smaller we also include 2-grams for these in V_3. This yields a total of $n = 670$ features. Only 197 of these features are considered relevant, i.e. not fully correlating and a variance score below 1. We make use of one training set (70%) and two test sets (15% each). Table 4 shows the sequential optimal results for Step 2-3 in Alg. 1. As can be seen, ${S_U}^*$ is again based on Cosine similarity and a BOA feature representation. Variance is found to be the most suitable dimensionality reduction technique, with $F(*)$ consisting of 50 features. We also tested $m = 100$ and $m = 150$. Table 5 shows the weight optimization results (Step 4 in Alg. 1). The metric that uses GP-based weights can be considered S^*. The metric operates on 42 features, the other 8 were shrank to zero. Again, the test score is slightly better than the training score, which could indicate overfitting but to a smaller degree than on the Anonycomm dataset. The weight optimization here was expected to suffer less from overfitting since a larger percentage of journeys is used to train the weights on. Again, for GBT, that does not shrink any feature

Table 5: Clustering quality after weight optimization using two surrogate models (BPIC 2012 dataset). $\%\Delta$ relates to the *test* results of Step 3 in Table 4.

Surrogate	m	Train γ	Test γ	$\%\Delta$	Train δ	Test δ	$\%\Delta$
GP	42	0.461	**0.471**	**+33**	0.644	0.658	+2
GBT	50	0.593	0.555	+21	0.673	0.614	-5

to zero, no overfitting is seen. In total, γ is improved with 39% while keeping δ similar. Different to the first experiment, the largest improvement of γ here is obtained with the weight optimization. Fig. 3 shows that S^* is *meaningful* for larger k and that $k = 10$ might be a more natural number of clusters than 6.

5 Related Work

Trace clustering in process mining is discussed in several works e.g. [12]. Defining an appropriate feature space and distance metric are still key challenges in trace clustering. The work of [6] and [2] contribute to this by developing syntactical techniques based on which appropriate feature spaces are derived using an edit-based distance. However, our work is focusing on vector-based approaches since syntactic techniques do not yield a standalone metric S that we are looking for. This paper contributes to trace clustering techniques that aim to differentiate business processes rather than reducing the complexity of the underlying process models. Specifically, a contribution is made to *distance-based* approaches to trace clustering. Model-based approaches, such as [5], are not considered suitable for our setting since no similarity between vectors is defined and thus no similarity metric can be tested. The methodology proposed is unique because it integrates metric optimisation techniques with clustering in one framework. A similar setting was discussed recently in [13] but for developing a hierarchical distance metric to measure the similarity between different market baskets where neither the behaviour nor the order of items matter. SIMPRIM adopts a semi-supervised approach to metric learning while existing frameworks, with a specific application to clustering, either are completely optimizing on some sort of ground truth [17] or do not include domain knowledge at all [9]. Most feature weighting methods employ some variation of gradient descent. This works for distance metrics from the Euclidean family. However, for other distance metrics, such as Cosine, this task is not so trivial. Especially when the dimensionality of trace vectors is high, the complexity of the gradient is large due to which this feature weight learning approach will be inefficient and ineffective. The dimensionality reduction techniques used in this paper are widely adopted in the field of data mining but are not typically used in trace clustering literature.

6 Conclusion

In this paper, we proposed SIMPRIM, a framework for inferring an appropriate domain-informed similarity metric that outperforms standard similarity metrics in the clustering task. The developed metric can also be used for further customer journey analysis or to improve the accuracy of other predication or recommendation tasks. A co-learning approach is adopted that simultaneously learns metric weights and optimizes the journey clustering. Several components used in our approach can easily be replaced with others equivalent. SIMPRIM has shown to be useful for two real-life event logs. A 46% and 39% improvement of the internal clustering quality is obtained, while agreeing with existing process knowledge in the form of journey labeling. Furthermore, an acceptable improvement of a next touchpoint prediction model was achieved. An interesting future direction is the added value of the metric to recommender systems that recommend a next best action for a running customer journey [14].

References

1. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. AISE, vol. 1848, pp. 49-–56, 2017.
2. Bose, R., Van der Aalst, W.: Context Aware Trace Clustering: Towards Improving Process Mining Results. SDM, pp. 401-412, 2009.
3. Breiman, L.: Classification and regression trees. Routledge, 1984.
4. Chierichetti, F., Kumar, R., Pandey, S., Vassilvitskii, S.: Finding the Jaccard Median. 21st ACM-SIAM Symposium on Discrete Algorithms, pp 293—311, 2010.
5. De Weerdt, J., Van den Broucke, S., Van Thienen, J., Baesens, B.: Active trace clustering for improved process discovery. TKDE **25**(12), pp. 2708—2720, 2013.
6. Evermann, J., Thaler, T., Fettke, P.: Clustering traces using sequence alignment. BPM Workshops **13**, pp. 179—190, 2015.
7. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. TKDE, **18**(8), pp. 1010-1027, 2006.
8. Hompes, B., Buijs, J., Van der Aalst, W., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering. BNAIC, 2015.
9. Huang, J., Ng, M., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. Pattern Analysis and Machine Intelligence **27**(5), pp. 657-668, 2005.
10. Lacoste, A., Larochelle, H., Laviolette, F., Marchand, M.: Sequential Model-Based Ensemble Optimization. UAI, 2014.
11. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of Internal Clustering Validation Measures. IEEE International Conference on Data Mining, 2010.
12. Song, M., Günther, C., Van der Aalst, W.: Trace Clustering in Process Mining. Lecture Notes in Business Information Processing, vol. 17, pp. 109-120, 2008.
13. Spenrath, Y., Hassani, M., Van Dongen, B., Tariq, H.: Why Did My Consumer Shop? Learning an Efficient Distance Metric for Retailer Transaction Data. In: ECML PKDD 2020, vol 12461, pp 323—338, 2020.
14. Terragni, A., Hassani, M.: Optimizing Customer Journey Using Process Mining and Sequence-Aware Recommendation. Ass. for Computing Machinery, NY (2019).
15. Thaler, T., Ternis, S., Fettke, P., Loos, P.: A Comparative Analysis of Process Instance Cluster Techniques. 2015.
16. Van der Aalst, W., Pesic, M., Song, M.: Beyond Process Mining: From the Past to Present and Future. 2nd International Conference CAiSE, pp. 38-52, 2010.
17. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning with application to clustering with side-information. NIPS, vol. 15, MIT Press (2003)
18. Van der Aalst W. Data Science in Action. In: Process Mining. Springer, Berlin, Heidelberg (2016)