Discovering Customer Journeys from Evidence: a Genetic Approach Inspired by Process Mining

Gaël Bernard and Periklis Andritsos

[1] University of Lausanne, Faculty of Business and Economics (HEC), Switzerland
gael.bernard@unil.ch
[2] University of Toronto, Faculty of Information, Toronto, Canada,
periklis.andritsos@utoronto.ca

Abstract. Displaying the main behaviors of customers on a customer journey map (CJM) helps service providers to put themselves in their customers' shoes. Inspired by the process mining discipline, we address the challenging problem of automatically building CJMs from event logs. In this paper, we introduce the CJMs discovery task and propose a genetic approach to solve it. We explain how our approach differs from traditional process mining techniques and evaluate it with state-of-the-art techniques for summarizing sequences of categorical data.

Keywords: customer journey mapping, process mining, customer journey analytics, genetic algorithms

1 Introduction

We aim to summarize customer journeys. A customer journey is a collection of interactions (or touchpoints) between a customer and a firm. A journey can be as simple as a single activity (e.g., 'looking at a product'), but can also involve complex interactions. Concretely, a challenge faced by many practitioners is to make sense of the–potentially infinite–combination of activities that exist in order to consume a service. As a response, new methods to understand, design, and analyze customer journeys are emerging from the industry and are becoming increasingly popular among researchers. One of these methods, which is the focus of this paper, is the Customer Journey Map (CJM). A CJM is a conceptual tool used for better understanding customers' journeys when they are consuming a service. A CJM depicts typical journeys experienced by the customers across several touchpoints. To represent a CJM, we use a visual chart showing the basic components of a CJM; namely, the touchpoints, and the journeys. It possesses two axis: the y-axis lists the touchpoints in alphabetical order, while the x-axis represents the ordering of the sequence of touchpoints. Dots connected with a smooth line represent a journey.

The left part of Fig. 1 display a CJM containing all the observed journeys from customers while the right part shows how our algorithm helps in reducing a CJM's complexity by building three representative journeys that best summarize the entire dataset. Summarizing thousands of journeys using few representatives

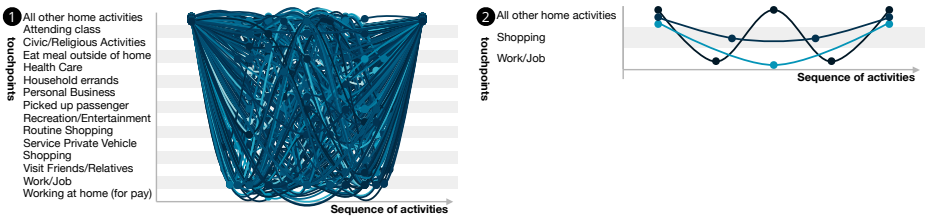

Fig. 1. ❶ A fraction of the dataset –612 sequences out of 123'706– displayed on a CJM, and ❷, a CJM that summarizes the entire dataset using three representatives.

has many compelling applications. First, it allows a business analyst to discover the common customers' behaviors. Second, the representative journeys extracted from data can also serve as a basis to fuel the discussion during workshops and complement strategic CJM built by internal stakeholders. Third, by assigning each journey to its closest representative, we turn a complex type of input data into a categorical one. The latter can be used to complete the traditional types of data that are used to perform behavior segmentation (e.g., usage volume, loyalty to brand).

Our work contributes by: (1) clarifying the customer journey discovery activity which has, to the best of our knowledge, never been defined, (2) proposing ground truth datasets, which are particularly suited for evaluating this activity, and (3) introducing a genetic algorithm to discover representative journeys. Using the proposed datasets and existing cluster analysis techniques, we demonstrate that our approach outperforms state-of-the-art approaches used in social sciences to summarize categorical sequences.

The paper is organized as follows. Section 2 defines the customer journey discovery activity while Section 3 provides an overview of existing techniques closely related to this task. Section 4 introduces our genetic algorithm. In Section 5, we evaluate the results. Finally, Section 6 concludes the paper.

2 Customer Journey Discovery

The customer journey discovery activity can be described with the following definition: *given a set of actual journeys, find a reasonable amount of representative journeys that well summarize the data.*

Definition 1 (Touchpoint): a touchpoint is an interaction between a customer and a company's products or services [2]. 'Sharing on social network' or 'ordering a product on the website' are two typical examples of touchpoints in a retail context. Let t be a touchpoint, and let T be the finite set of all touchpoints.

Definition 2 (Journey): A journey J is a sequence of touchpoints. For instance, $J = \{\langle$'Visiting the shop', 'Testing the product', 'Sharing on social network'$\rangle\}$ is a journey with three touchpoints. For the sake of brevity, we replace the touchpoints with alphabetical characters so that J becomes $\langle$ABC$\rangle$.

Definition 3 (Actual Journey): An event log J_a is the set of all actual journeys observed from customers.

Definition 4 (Event Logs): Let $J_{\mathcal{A}}$ be an event log, the set of all actual journeys observed by customers.

Definition 5 (Representative Journey): A representative journey, J_r, is a journey that summarizes a subset of actual journeys $\in J_{\mathcal{A}}$.

Definition 6 (Customer Journey Map): A CJM summarizes customer journeys through the use of representative journeys. Let a customer journey map $J_{\mathcal{R}}$ be the set of all the J_rs summarizing $J_{\mathcal{A}}$. Let $k_{\mathcal{R}}$ denotes the number of journeys in a map (i.e., $|J_{\mathcal{R}}|$). Typically, the part ❷ of Fig. 1 is a CJM, $J_{\mathcal{R}}$, containing three representative journeys, J_r ($k_{\mathcal{R}} = 3$), summarizing an event log $J_{\mathcal{A}}$.

Discovering $J_{\mathcal{R}}$ from $J_{\mathcal{A}}$ is an unsupervised clustering task that entails interesting challenges. First, there is no work in the literature that deals with the optimal number of $k_{\mathcal{R}}$. Second, the sequence that best summarizes its assigned actual journeys needs to be found.

3 Related Work

In [1], we propose a web-based tool to navigate CJMs, which is called CJM-ex (CJM-explorer). It uses a hierarchical structure so that the first layers show only the most representative journeys, abstracting from less representative ones.

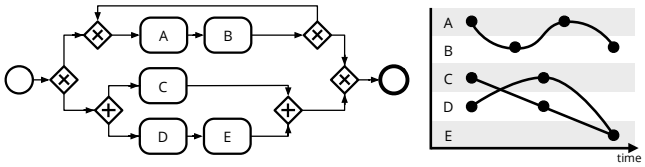

Fig. 2. Expected BPM and CJM when L = [⟨ABAB⟩, ⟨DCE⟩, ⟨CDE⟩].

In [2], we show that the process mining framework as well as the input data are relevant in a customer journey analytics context. Similar to the process discovery activity in process mining, this work focuses on the *discovery* of a model from *event logs*. Our work was inspired by the approach in [3, 6, 14] where the authors propose discovering business process models (BPMs) using a genetic algorithm. A BPM and a CJM is not used for the same purpose. Fundamentally, the goal of a BPM is to find models that best describe how the processes are handled. In contrast, the aim of a CJM is to help internal stakeholders to put themselves in their customers' shoes. Fig. 2 shows that these models convey different type of information. A CJM depicts journeys as experienced by customers while a BPM shows the available combination of activities using advanced constructs such as exclusive or parallel gateways. Overall, CJMs are used to supplement but not to replace BPMs [12].

In fact, our work is closer to the summarization of categorical sequences used in social sciences. In particular, in [8, 9], Gabadinho et al. propose summarizing a list of observed sequences (i.e., actual journeys) using representative (i.e., representative journey). They define a representative set as *"a set of non-redundant 'typical' sequences that largely, though not necessarily exhaustively, cover the spectrum of observed sequences"* [8]. We argue that their definition matches our definition of a representative sequence summarizing a set of sequences. The authors propose five ways to select a representative. The *'frequency'* (1), where the most frequent event is used as the representative. The *'neighborhood density'* (2), which consists of counting the number of sequences within the neighborhood of each candidate sequence. The most representative is the one with the largest number of sequences in a defined neighborhood diameter. The *'mean state frequency'* (3): the transversal frequencies of the successive states is used to find a representative using the following equation:

$$MSF(s) = \frac{1}{\ell} \sum_{i=1}^{\ell} f_{si} \tag{1}$$

where

$s = s_1 s_2 ... s_l$: Sequence of length ℓ
$(f_{s_1}, f_{s_2}, ... f_{s_l})$: Frequencies of the states at time t_ℓ

The sum of the state frequencies divided by the sequence length becomes the mean state frequency. Its value is bounded by 0 and 1 where 1 describes a situation where there is only one single distinct sequence [8]. The sequence with the highest score is the representative. The *'centrality'* (4): the representative – or medoid – can be found using the centrality. Being the most central object, the representative is the sequence with the minimal sum of distances to all other sequences. Finally, the *'sequence likelihood'* (5): the sequence likelihood of a sequence derived from the first-order Markov model can also be used to determine the representative. In the evaluation section, we compare our genetic approach with these five techniques using their own implementation of the package Traminer available in R [7].

4 Genetic Algorithm for CJM Discovery

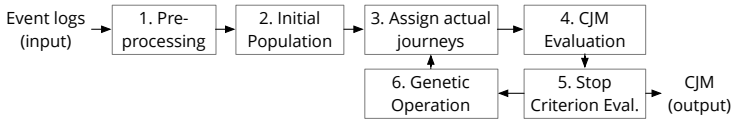

Fig. 3. Overview of the genetic algorithm to discover CJMs

As an introduction, Fig. 3 provides intuition on how the genetic algorithm builds a CJM such as the one visible in Fig. 1 (part ❷). A set of actual journeys, J_a, is provided to the algorithm. Then, during *Generation 1*, we build p number of $J_\mathcal{R}$, where p *is the population size*; i.e., the number of CJMs that will be evaluated after each generation. In our example $p = 3$. Each $J_\mathcal{R}$ is evaluated and we keep the e best $J_\mathcal{R}$s, called the *elite set* while we discard the other $(p - e)$ $J_\mathcal{R}$s. Then, we move to *Generation 2*, where we keep an untouched version of the e number of $J_\mathcal{R}$s in the elite set. For instance, $J_{\mathcal{R}2}$ was kept in Generation 2 because it has the best average quality in Generation 1, i.e., $J_{\mathcal{R}2}$ is intact in Generation 2. We then apply some transformation (to be discussed next) to generate $(p - e)$ new $J_\mathcal{R}$s that will, in turn, be evaluated. In our case, we generate $J_{\mathcal{R}4}$ and $J_{\mathcal{R}5}$. We recursively transform and evaluate the p number of $J_\mathcal{R}$s until a stopping criterion is met. Once a stopping criterion is met, we return the best $J_\mathcal{R}$. *The best $J_\mathcal{R}$ can be interpreted as the best set of representative journeys, J_r, representing a set of journeys from $J_\mathcal{A}$ that have been found given acertain evaluation criterion.* The next section describes how we generate the initial population, what various types of operations we apply on each $J_\mathcal{R}$ to transform them, and how we evaluate each one of them given a set of journeys in $J_\mathcal{A}$.

4.1 Preprocessing

To gain in efficiency, we make the assumption that $J_\mathcal{R}$ will be close to the frequent patterns observed in J_a. Let Top_{ℓ_n} be the n most occurring pattern of length ℓ and $Top_n \supseteq Top_{\ell_{[2,m]}}$ be the superset of all the most occurring patterns of lengths 2 to m. Top_n is used later to form the initial population of $J_\mathcal{R}$ (described in Section 4.2), and to add a random journey to $J_\mathcal{R}$. Using Top_n we avoid generating journeys by picking a random number of touchpoints from T. According to our experiments, using Top_n reduces the execution time by two to get an output $J_\mathcal{R}$ of the same average quality.

4.2 Initial Population

The initial population is generated by adding a sequence randomly picked from Top_n (defined in Sect. 4.1).

4.3 Assign Actual Journeys

The quality of a representative journey can only be measured when knowing which actual journeys it represents. Hence, a first step toward evaluating the quality of $J_\mathcal{R}$ is to assign each journey $J_a \in J_\mathcal{A}$ to its closest journey in $J_r \in J_\mathcal{R}$. To characterize the closeness between J_a and J_r, we use the Levensthein distance borrowed from [11]. It is a metric particularly well suited to measure the distance between sequences. The Levensthein distance counts the number of edit operations that are necessary to transform one sequence into another one.

There are three types of operations: deletions, insertions, and substitutions. For instance, the distance between $\langle$ABC$\rangle$ and $\langle$ACCE$\rangle$ is 2 since one substitution and one insertion are required to match them. We define the closest representative as the one having the smallest Levensthein distance with the actual journeys. Note that if a tie occurs between multiple best representatives, we assign the J_a to the J_r having the smallest amount of actual journeys already assigned to it. Once each actual journey has been assigned to its closest representative, we can evaluate $J_{\mathcal{R}}$ using the criteria described in the next section.

4.4 CJM Evaluation Criteria

This section introduces the evaluation criteria used to determine the quality of each $J_{\mathcal{R}}$, namely, (1) the fitness, (2) the number of representatives, and (3) the average quality.

Fitness. The fitness measures the distance between each sequence of activities J_a and its closest representative $J_{\mathcal{R}}$ using the Levenshtein distance [11].

$$Fitness(J_a, J_{\mathcal{R}}) = 1 - \frac{\sum_{i=1}^{|J_a|} min_{j=1}^{|J_{\mathcal{R}}|}(Levensthein(\sigma_{\mathcal{A}_\mathrm{i}}; \sigma_{\mathcal{R}_\mathrm{j}}))}{\sum_{i=1}^{|J_a|} Length(\sigma_{\mathcal{A}_i})} \tag{2}$$

where

$\sigma_{\mathcal{A}_\mathrm{i}}$	: i$^{\mathrm{th}}$ actual sequence observed in event logs
$\sigma_{\mathcal{R}_\mathrm{j}}$	: j$^{\mathrm{th}}$ representative contained in $J_{\mathcal{R}}$
$Levenshtein(x_1, x_2)$	: Levensthein distance between two sequences
$Length(x)$	: Length of the sequence of activity x

A fitness of 1 means that the representative journey perfectly catches the behavior of the actual journeys assigned to it. In contrast, a fitness close to 0 implies that many edit operations are necessary to match the sequences.

Number of Representatives. If we maximize the fitness without trying to keep a low $k_{\mathcal{R}}$, the CJM will become unreadable because too many representative journeys will be displayed in it. In other words, $J_{\mathcal{R}}$ overfits. Hence, the goal is to find a $k_{\mathcal{R}}$ that offers a good compromise between underfitting and overfitting. Finding the optimal number of clusters is a recurrent challenge when clustering data. We propose integrating traditional ways of determining the optimal number of clusters, such as the Bayesian information criterion [13], or the Calinski-Harabasz index [5]. The idea is to evaluate a range of solutions (e.g., from 2 to 10 journeys) and to keep the best solution. Let k_h be the optimal number of clusters returned by one of the techniques mentioned above. By integrating k_h into the evaluation, we can guide the solution toward a $k_{\mathcal{R}}$ that is statistically relevant. To evaluate the quality, we measure the distance between $k_{\mathcal{R}}$ and k_h. To do this, we propose the following distribution function:

$$NumberOfRepresentatives(k_{\mathcal{R}}, k_h, x_0) = \frac{1}{1 + (\frac{|k_{\mathcal{R}} - k_h|}{x_0})^2} \tag{3}$$

where

$k_{\mathcal{R}}$: Number of J_r journeys on $J_{\mathcal{R}}$ (i.e., $|J_{\mathcal{R}}|$)
k_h : Optimal number of journeys (e.g., using the Calinski-Harabasz index)
x_0 : x value of the midpoint

The parameter x_0 determines where the midpoint of the curve is. Concretely, if $x_0 = 5$, $k_{\mathcal{R}} = 11$ will result in a quality of 0.5 because the absolute distance from k_h is 5. We set $x_0 = 5$ for all our experiments. Intuitively, x_0 guide the number of representatives that will be found. We set it to 5 as we believe that it is a reasonable amount of journeys to display on a single CJM. Because the number of representatives is not the only criteria to assess the quality of a CJM, the final CJM might contain more or less journeys if it increases the average quality.

Average Quality. We assign weights to the fitness and the number of representatives qualities to adjust their relative importance. According to our experiments and in line with the work from Buijs et al. [3], the results tend to be best if more weight is given to the fitness quality. Typically, weights $w_f = 3$, and $w_{k_h} = 1$ lead to the best results. Then, we get the overall quality by averaging the weighted qualities using the arithmetic mean. In the next section, we determine the stopping criterion of the algorithm.

4.5 Stopping Criterion

Before starting a new generation, we check if a stopping criterion is met. There are three ways we can use to stop the algorithm taken from [3, 6]. (1) The algorithm could stop after a certain number of generations. (2) One could stop the algorithm when a certain number of generations have been created without improving the average quality. (3) We could stop the algorithm when a certain quality threshold is reached for one of the evaluation criteria. Because it is difficult to predict the quality level that can be reached, we believe that stopping the algorithm using a threshold is not advisable. For this reason, we used a combination of approaches 1 and 2 for our experiments. Once the stopping criteria have been evaluated, either the algorithm stops, or, we generate new candidates by applying genetic operators described in the next section.

4.6 Genetic Operations

Once all the CJMs have been evaluated, we rank them by their average quality and copy a fraction (i.e., e) of the best ones in *elite*. Because we keep an untouched version of the e number of $J_{\mathcal{R}}$s, we make sure that the overall quality

will only increase or stay steady. Then, we generate $(p - e)$ new $J_{\mathcal{R}}$s as follows. We pick one random $J_{\mathcal{R}}$ from *elite*, and perform one or multiple of these four operators. *(1) Add a journey:* A sequence is randomly picked from Top_n and added to $J_{\mathcal{R}}$. *(2) Delete a journey:* A random journey is removed from $J_{\mathcal{R}}$. Nothing happens if $J_{\mathcal{R}}$ contains only one journey. *(3) Add a touchpoint:* A touchpoint from T is added to one of the journeys from $J_{\mathcal{R}}$ at a random position. *(4) Delete a touchpoint:* A touchpoint is removed from $J_{\mathcal{R}}$ unless removing this touchpoint would result in an empty set of touchpoints. As described in Fig. 3, once new $J_{\mathcal{R}}$s have been created, we go back to the evaluation phase where the new $J_{\mathcal{R}}$s are evaluated until one stopping criterion is met. Once such a criterion is met, we return the best $J_{\mathcal{R}}$s of the last generation and the algorithm stops.

5 Evaluation

5.1 Datasets

We produced several event logs that simulate journeys. Generating the event logs ourselves means that we know the ground truth represented by the generative journeys and therefore the objective is to recover these journeys from a set of actual ones we produce. *A generative journey is a known list of touchpoints from which we generate the event logs.* Let $J_{\mathcal{G}}$ be a set of $k_{\mathcal{G}}$ generative journeys used to generate a dataset composed of 1,000 actual journeys.

If we were to use only these generative journeys to generate 1,000 journeys, we would obtain only $k_{\mathcal{G}}$ distinct journeys. For instance, if we use $J_{g1} = \langle \text{ABC} \rangle$ and $J_{g2} = \langle \text{ABBD} \rangle$ to generate 1,000 journeys equally distributed, we obtain $J_a = \left\{ J_{g1}^{500}, J_{g2}^{500} \right\}$. A more realistic situation would depict a scenario where each group of customers can be described by a representative sequence of activities, but the actual journeys within the group can deviate from the representative one. Hence, to produce more realistic data, we inject noise for a fraction of the journeys. For instance, if the noise level is set to 50%, $J_a = J_g$ is true for half of the data. For the other half, we add noise by removing or adding touchpoints, or by swapping the ordering of activities.

We generated 8 datasets of varying characteristics. The characteristics are distinct in terms of: number of $k_{\mathcal{G}}$, number of touchpoints, average length of the journeys, and the standard deviation of the number of journeys assigned to each generative journey. For each dataset, we gradually apply 5 levels of noise, resulting in 40 datasets. The datasets, the detailed characteristics of them as well as the procedure followed to produce them are available at the following url: http://customer-journey.unil.ch/datasets/.

5.2 Metrics

We use both external and internal evaluation metrics. On the one hand, the external ones evaluate the results relative to the generative journeys. On the other hand, the internal evaluation uses cluster analysis techniques to assess the

results. The aim is to account for the fact that the ground truth might not be the optimal solution because of the noise that was added. This section introduces these metrics. For the internal evaluation metrics, we borrowed them from [9].

External Evaluation - Jaccard Index. To evaluate the similarity between the sequences of activities from the generative journeys ($J_{\mathcal{G}}$) and the discovered representative journeys ($J_{\mathcal{R}}$), we propose to use the Jaccard index where a score of 1 indicates a perfect match.

$$JaccardIndex(J_{\mathcal{R}}, J_{\mathcal{G}}) = \frac{|J_{\mathcal{R}} \cap J_{\mathcal{G}}|}{|J_{\mathcal{R}} \cup J_{\mathcal{G}}|} \tag{4}$$

External Evaluation - Distance in Number of Journeys. This metric measures the distance between the number of generative journeys and the number of representative journeys returned by the algorithm. We propose:

$$NbJourneysDistance(k_{\mathcal{G}}, k_{\mathcal{R}}) = abs(k_{\mathcal{G}} - k_{\mathcal{R}}) \tag{5}$$

Internal Evaluation - Mean Distance [9]. The mean distance i returns the average Levensthein distance between the representative sequence i and the sequence of actual journeys that have been assigned to i, k_i being the number of actual journeys assigned to the representative journey i.

$$MeanDistanceScore_i = \frac{\sum_{j=1}^{k_i} D(J_{r_i}, J_{a_{ij}})}{k_i} \tag{6}$$

Internal Evaluation - Coverage [9]. The coverage indicates the proportion of actual journeys that are within the neighborhood n of a representative.

$$Coverage_i = \frac{\sum_{j=1}^{k_i} (D(J_{r_i}, J_{a_{ij}}) < n)}{k_i} \tag{7}$$

Internal Evaluation - Distance Gain [9]. The distance gain measures the gain in using a representative journey instead of the true center of the set, C (i.e., the medoid of the whole dataset). In other words, it measures the gain obtained in using multiple representative journeys instead of a single one.

$$DistGain_i = \frac{\sum_{j=1}^{k_i} D(C(J_a), J_{a_{ij}}) - \sum_{j=1}^{k_i} D(J_{r_i}, J_{a_{ij}})}{\sum_{j=1}^{k_i} D(C(J_a), J_{a_{ij}})} \tag{8}$$

5.3 Settings

We evaluate our genetic algorithm (approach 1) against an approach using Kmedoids clustering (approach 2) and the approaches proposed by Gabadinho et

al.[7] (approach 3). Approach 1 is using our genetic algorithm with a fitness weight set to 3 and a weight for the number of representatives set to 1. Due to the non-deterministic nature of the genetic algorithm, we run it ten times. In approach 2, we build a distance matrix of the edit-distance between sequences. We then create k (found using the Calinski-Harabasz index) clusters using an implementation, [4], of the k-medoids algorithm. Finally, the medoid of each cluster becomes the representative. In approach 3 we build the same distance matrix as then one used in approach 2. Then, we used an agglomerative hierarchical clustering to define k clusters. Finally, we return the best representatives of each cluster using the frequency, the neighborhood density, the mean state frequency, the centrality, or the likelihood using the package Traminer available in R [7].

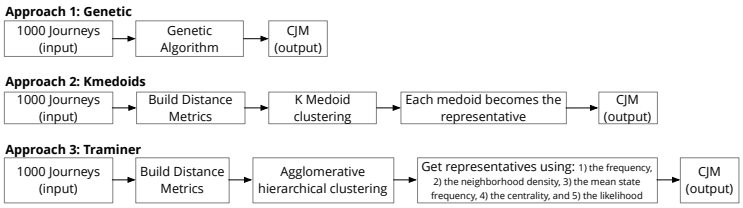

Fig. 4. Three approaches used to find the representative journeys.

5.4 Results

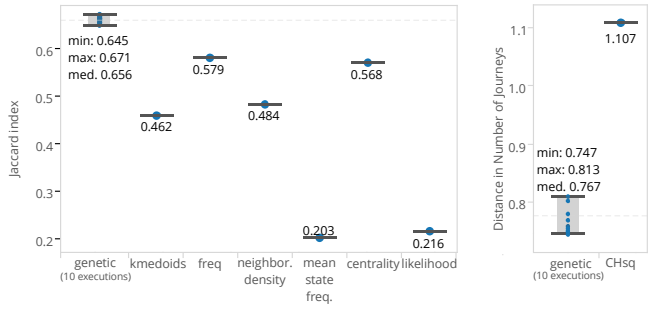

Fig. 5. External Evaluation

We present the results of the external evaluation metrics, then the internal ones and we conclude by mentioning the execution times for several datasets sizes. The external evaluation metrics are shown in Fig. 5. Here, we can see that the solution that reduces the Jaccard index the most and which is closest to

the ground truth in terms of number of journeys is the genetic approach. The internal evaluation in Fig. 6 shows that the genetic algorithm outperforms the other approaches. Finally, the execution time to for 100 actual journeys is much faster using the kmedoids or using the techniques implemented in Traminer [7]. We observe that when we increase the datasets' size the performance of the genetic algorithm tend to be comparable to the kmedoids implementation and faster than the techniques implemented in Traminer.

	Avg. mean distance (the smaller, the better)	Avg. coverage (the higher, the better)	Avg. dist-gain (the higher, the better)
genetic (10 executions)	1.37 (median) min:1.37 max:1.38	0.59 (median) min:0.58 max:0.59	0.59 (median) min:0.59 max:0.60
kmedoids	1.75	0.47	0.48
freq	1.74	0.49	0.48
neighbor. density	1.72	0.46	0.48
mean state freq.	3.69	0.27	-0.14
centrality	1.62	0.50	0.51
likelihood	2.15	0.40	0.36

Fig. 6. Internal evaluation. The genetic algorithm perform best.

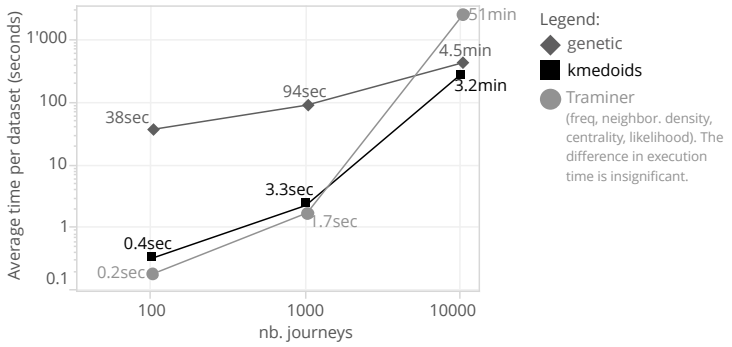

Fig. 7. Comparing the execution time per dataset for 100, 1'000, and 10'000 journeys.

6 Conclusion

In this paper, we propose two basic quality criteria to guide the evolution process of discovering the best representative journeys for a given set of actual journeys that otherwise would be unreadable. We demonstrate that they perform well on synthetic datasets. We show that techniques from social sciences are also useful for studying customer journeys. As suggested by Gabadinho et al., "The methods are by no way limited to social science data and should prove useful in

many other domains" [8]. This present study supports this claim and highlights how research from social science can benefit our understanding of customers. At a time when a customer-centric culture has become a matter of survival according to [10], we anticipate that research at the crossroads between data science, marketing, and social sciences will be key to a full understanding of customer experiences.

References

1. Bernard, G., Andritsos, P.: Cjm-ex: Goal-oriented exploration of customer journey maps using event logs and data analytics. In: 15th International Conference on Business Process Management (BPM2017) (2017)
2. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: Proceedings of the Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017) (2017)
3. Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: A genetic algorithm for discovering process trees. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1–8. IEEE (2012)
4. C, B.: Numpy/scipy recipes for data science: k-medoids clustering[r]. Technical Report (2015), https://github.com/letiantian/kmedoids
5. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**(1), 1–27 (1974)
6. De Medeiros, A.A., Weijters, A.: Genetic process mining. In: Applications and Theory of Petri Nets 2005, Volume 3536 of Lecture Notes in Computer Science. Citeseer (2005)
7. Gabadinho, A., Ritschard, G.: Searching for typical life trajectories applied to childbirth histories. Gendered life courses-Between individualization and standardization. A European approach applied to Switzerland (2013), 287–312 (2013)
8. Gabadinho, A., Ritschard, G., Studer, M., Mueller, N.S.: Summarizing sets of categorical sequences: selecting and visualizing representative sequences pp. 94–106 (October 2009)
9. Gabadinho, A., Ritschard, G., Studer, M., Müller, N.S.: Extracting and rendering representative sequences. In: International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. pp. 94–106. Springer (2009)
10. Goran, J., LaBerge, L., Srinivasan, R.: Culture for a digital age. Tech. rep., McKinsey (July 2017), https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/culture-for-a-digital-age
11. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
12. Olding, E., Cantara, M., Robertson, B., Dunie, R., Huang, O., Searle, S.: Predicts 2016: Business transformation and process management bridge the strategy-toexecution gap. Tech. rep., Gartner (November 2015), https://www.gartner.com/doc/3173020/predicts–business-transformation-process
13. Schwarz, G., et al.: Estimating the dimension of a model. The annals of statistics **6**(2), 461–464 (1978)
14. Vázquez-Barreiros, B., Mucientes, M., Lama, M.: Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. Information Sciences **294**, 315–333 (2015)