



universität
wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation /Title of the Doctoral Thesis

Mining and Visualization of Customer Journey Processes in Information Systems

verfasst von / submitted by
Dipl.-Ing. Marian Lux BSc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Doktor der Technischen Wissenschaften (Dr. techn.)

Wien, 2023 / Vienna 2023

Studienkennzahl lt. Studienblatt /

degree programme code as it appears on the student record sheet: UA 786 880

Dissertationsgebiet lt. Studienblatt /

field of study as it appears on the student record sheet::

Betreut von / Supervisor:

Informatik

Univ.-Prof. Dipl.-Math. Dr. Stefanie Rinderle-Ma

Abstract

The understanding of customer journey processes helps across many industries to win in the market. Customer journeys contain all steps of user interaction with companies for their products and services. Business processes, which are also processes of the customer journey, reflect user behavior. Most of the customer journey processes are highly individual and therefore hard to discover and visualize. For example, for information systems with a keyword based search functionality for products and services, nearly endless options of user entered keywords are possible, through which the user behavior is just hard to reveal. However, the understanding of the user behavior can help, for example, in marketing to improve suggestions for keywords or even search results to increase sales. With process mining techniques, customer journey processes can be discovered, analyzed and enhanced. In this work, various novel techniques are introduced, which show how to discover customer journey processes in information systems, how to assess the quality of customer journey processes in information systems and how to visualize highly individual customer journey processes in information systems. Moreover, this cumulative dissertation covers the whole cycle from the data preparation till the visualization of meaningful process models as follows: First, it shows how to prepare companies with existing information systems by pre-processing their available data for process mining techniques and furthermore, questions which can be answered by analyzing customer journey processes. After that, a metric is introduced to measure search processes in terms of complexity and clustering – which were discovered through process mining – on different granularities, like on nodes, paths or even to compare whole process models. Together with the novel metric, ontology support on search process models is introduced as well, which is capable to simplify search process models in terms of complexity and clustering on customer journeys by showing recommendations for search terms during the search and by pooling search terms on the event log data before an actual process mining algorithm is performed. Additionally, for search process models, a metric is developed which reveals different search patterns to reflect the users intention during the usage of the search functionality which takes not only the user entered data, but also the results from the system - which is the number of search results - into account. Finally, a novel clustering algorithm is introduced which helps to reduce the cognitive load of visualized processes, by evenly distributing elements among their clusters but considering a low variance in each cluster. The cluster algorithm can be used on use cases beyond this work as well, like for visualizing choropleth maps. The developed concept for pre-processing data, discovering and visualizing highly individual processes was evaluated on real world applications, like in the tourism domain.

Zusammenfassung

Das Verständnis von Customer-Journey-Prozessen hilft über viele Branchen hinweg, sich auf dem Markt durchzusetzen. Customer Journeys beinhalten alle Schritte der Nutzerinteraktion mit einem Unternehmen für deren Produkte und Dienstleistungen. Geschäftsprozesse, die auch Prozesse der Customer Journey sind, spiegeln dabei das Nutzerverhalten wider. Die meisten Customer Journey-Prozesse sind hoch individuell und daher nur schwer zu erfassen und zu visualisieren. Beispielsweise sind bei Informationssystemen mit einer schlagwortbasierten Suchfunktion für Produkte und Dienstleistungen nahezu unendliche Möglichkeiten bei der Eingabe von Schlagwörtern durch die Nutzer möglich, wodurch das Nutzerverhalten in weiterer Folge nur schwer zu erfassen ist. Das Verständnis des Nutzerverhaltens kann beispielsweise im Bereich Marketing helfen, Vorschläge für Suchbegriffe während der Suche anzuzeigen oder Suchergebnisse zu verbessern, mit dem Ziel, den Umsatz eines Unternehmens zu steigern. Mit Process Mining können Customer-Journey-Prozesse entdeckt, analysiert und verbessert werden. In dieser Arbeit werden verschiedene neuartige Techniken im Kontext von Process Mining vorgestellt, die dabei helfen, Customer-Journey-Prozesse in Informationssystemen zu entdecken, die Qualität von Customer Journey-Prozessen in Informationssystemen zu bewerten und hochindividuelle Customer Journey-Prozesse in Informationssystemen zu visualisieren. Darüber hinaus deckt diese kumulative Dissertation den gesamten Zyklus von der Datenaufbereitung bis hin zur Visualisierung von aussagekräftigen Prozessmodellen wie folgt ab: Zu Beginn wird gezeigt, wie Unternehmen mit bestehenden Informationssystemen, durch den Einsatz von Datenaufbereitung, für Process Mining vorbereitet werden können und welche Fragestellungen durch die Analyse von Customer-Journey-Prozessen beantwortet werden können. Anschließend wird eine Metrik vorgestellt, mit der die durch Process Mining entdeckten Suchprozesse hinsichtlich Komplexität und Clustering auf verschiedenen Granularitäten, wie z.B. auf Knoten, Pfaden oder auch zum Vergleich ganzer Prozessmodelle, gemessen werden können. Zusammen mit der neuartigen Metrik wird auch eine Ontologieunterstützung für Suchprozessmodelle eingeführt, die in der Lage ist, diese Prozessmodelle in Bezug auf Komplexität und Clustering für Customer Journeys zu vereinfachen. Dabei können Vorschläge für Suchbegriffe während der Suche angezeigt werden, beziehungsweise können bereits aufgezeichnete Suchbegriffe in Logdaten thematisch zusammenfasst werden, bevor der eigentliche Process-Mining-Algorithmus durchgeführt wird. Darüber hinaus wird für Suchprozessmodelle eine neue Metrik entwickelt, die verschiedene Suchmuster aufzeigt, um die Absichten der Nutzer während der Nutzung der Suchfunktionalität zu reflektieren, wobei nicht nur die vom Nutzer eingegebenen Daten, sondern auch die Ergebnisse des Systems, d.h. die Anzahl der Suchergebnisse, berücksichtigt werden. Abschließend wird ein neuartiger Cluster-Algorithmus vorgestellt, der die kognitive Belastung von visualisierten Prozessmodellen verringert, indem er die Elemente gleichmäßig auf ihre Cluster verteilt, aber eine geringe Varianz in jedem Cluster berücksichtigt. Der Cluster-Algorithmus kann auch abseits der dargestellten Anwendungsfälle dieser Arbeit verwendet werden, z.B. für die Visualisierung von Choropleth-Karten. Alle entwickelten Konzepte, beginnend bei der Aufbereitung von Daten bis hin zur Entdeckung und Visualisierung von hochindividuellen Prozessen wurden in realen Anwendungen, wie z.B. im Tourismusbereich, evaluiert.

Acknowledgements

First, I would like to express a big thank you to my supervisor, Prof. Stefanie Rinderle-Ma, who helped me to steer my research direction during my whole doctoral study. I always felt absolutely professionally supported during my whole study, which I take not for granted. Thus, she helped me to grow as a researcher where I learned a lot from her, especially how to write scientific papers and the roots how to develop strong proven concepts and research questions. Especially I appreciate her openness to individual approaches from her doctoral students without restricting them. That gave me the self-confidence and the courage to develop novel approaches. Further, I am really grateful for my family, my wife Beate and my daughter Lucy who always gave me the strength when I needed it. Beate advised me, when strong decisions had to be made and helped me where she could with her smart mind. Lucy on the other side, showed me the life through her childlike perspective which helped me to compensate for long, strenuous working days and to develop innovative concepts through different angles. Additionally, I want to thank my family, as well my parents-in-law and my friend Christoph, who is also my co-founder and co-owner of the company LuxActive – where I make a living of – for their support and understanding when I needed time and room for studying new concepts or writing papers for my scientific research. Last, but not least important, I particularly want to thank my parents. They supported me on my life-long journey of learning and they did whatever they could to give me always the opportunity to do so, because it is an essential part of me. I am grateful for my mother that she corrected me – since I was a child – when I was not able to precisely formulating anything which helped me especially later on during scientific research. And regarding my father, the year 2023 had been very challenging for me and my family because he passed away too early after a long and serious illness. I am really grateful for the time in my live I had with him. He learned me to never fear anything and to never give up, as he did not when he was fighting for his life till the end. The honor is to him and that's why I wanted to close these acknowledgements in his name. Rest in peace, my father, Helmuth.

Contents

| | | |
|----------|--|------------|
| 1 | Introduction | 1 |
| 2 | Process Mining on Customer Journey Processes | 3 |
| 2.1 | Research Gaps and Research Questions | 6 |
| 2.1.1 | Preparation of Information Systems for Process Mining to Discover Highly Individual Customer Journey Processes | 6 |
| 2.1.2 | Assessment of Highly Individual Discovered Processes from Information Systems on Customer Journeys | 7 |
| 2.1.3 | Visualization of Highly Individual Customer Journey Processes in Information Systems | 8 |
| 2.1.4 | Research Questions Revisited | 9 |
| 2.2 | Methodology | 11 |
| 3 | Publications | 13 |
| 3.1 | Preprocessing Data from Highly Individual Customer Journey Processes and Answering Questions through Process Mining | 14 |
| 3.1.1 | Motivation | 14 |
| 3.1.2 | Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System | 15 |
| 3.2 | Assessment of Customer Journey Processes in Information Systems and Reduction of Complexity in Highly Individual Processes | 28 |
| 3.2.1 | Motivation | 28 |
| 3.2.2 | Assessing the Quality of Search Process Models | 30 |
| 3.2.3 | Analyzing User Behavior in Search Process Models | 47 |
| 3.3 | Visualization of Customer Journey Processes to Reduce the Cognitive Load and Gaining Insights | 60 |
| 3.3.1 | Motivation | 60 |
| 3.3.2 | DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling | 61 |
| 4 | Discussion | 101 |
| 4.1 | Research Questions Answered | 101 |
| 4.1.1 | Questions Answered on <i>RQ1</i> | 101 |
| 4.1.2 | Questions Answered on <i>RQ2</i> | 101 |
| 4.1.3 | Questions Answered on <i>RQ3</i> | 101 |
| 4.2 | Conclusion and Outlook | 103 |
| 4.3 | Limitations and Threats to Validity | 105 |
| 4.4 | Findings and Results | 107 |
| | References | 108 |

List of Figures

| | | |
|---|--|----|
| 1 | Customer Journey | 2 |
| 2 | Mined Highly Individual Search Process Model from <i>oHA</i> using the Process Mining Tool <i>DISCO*</i> | 4 |
| 3 | Customer Journey Process Mining | 5 |
| 4 | Customer Journey Process Mining with Research Questions and related Publications . | 10 |

List of Acronyms

| | |
|---|-----|
| IoT Internet of Things | 1 |
| CJ Customer Journey | 1 |
| CJM Customer Journey Map | 3 |
| CJP Customer Journey Process | 3 |
| KPI Key Performance Indicator | 3 |
| PM Process Mining | 3 |
| ETL Extract, Transform, and Load | 6 |
| PPM Paper: Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System | 9 |
| PM1 Paper: Assessing the Quality of Search Process Models | 9 |
| PM2 Paper: Analyzing User Behavior in Search Process Models | 10 |
| PCL Paper: DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling | 10 |
| spm Search Process Quality Metric | 28 |
| sbm Search Behavior Metric | 28 |
| DDCAL 1d distribution cluster algorithm | 60 |
| XES eXtensible Event Stream | 105 |
| MXML Mining eXtensible Markup Language | 105 |

1 Introduction

The amount of worldwide created, captured, copied, or consumed data has grown drastically during the last decade. A forecast even predicts an increase from 64.2 zettabytes of data in 2020 to 180 zettabytes in 2025. The highest increase of total produced data showed up between the year 2019 and 2020. A reason for this increase is, that through the *COVID-19* pandemic, which started in 2020, the growth of data was even higher than expected because more people used digital services at home, like for work, education or entertainment.¹ But in general, there is an ongoing trend and exponential increase of yearly produced data worldwide, which started at least in 2010 through emerged devices that collect information, like various mobile devices, *Internet of Things (IoT)* or software logs. (Berisha, Mëziu, & Shabani, 2022). Through the digital shift, which gives rise to big data, new possibilities for value creation of companies emerge. On the other hand, companies operate today through a fast changing world of “*volatility, uncertainty, complexity and ambiguity*”. Also, new demands on companies arise for utilizing available data for creating value and improving their processes. Therefore, the study of a *Customer Journey (CJ)* becomes important to gain insights from customers about their pain points, preferences and usage patterns. (Hansson, Angel, Mannhardt, & Kvale, 2021) A *CJ* is defined as a collection of touchpoints/interactions between a customer and a company (Bernard & Andritsos, 2019), (Richardson, 2010). Fig. 1 depicts a customer journey for customers who have several touchpoints with a company’s product or service offerings, which is described in detail as follows:

The graphic shows that the company, which is placed in the center, has many touchpoints for customers according to their offered products or services. These touchpoints have certain stages. For example, stages can be classified in *BEFORE*, *DURING* and *AFTER* the usage of a product or service, where on every stage, different touchpoints are in focus, e.g., in the stage *BEFORE*, customers show *interest* for a product or service, where companies can provide, for example, incentives for customers and marketing actions to attract customers for their offerings. Different touchpoints are depicted in this graphic for the state *DURING*, because there are different chronological touchpoints involved, e.g., a *search* phase, where customers can use, for example, an information system to find the right offering. And, after that, a customer can *buy* and *use* a product or service. During and after the usage of products or services, the stage *AFTER* collects touchpoints for *supporting* activities on them. The depicted phases from *BEFORE* to *AFTER* show different color gradients – from light blue at stage *BEFORE* to dark blue at stage *AFTER*. These gradients indicate the not exclusive but possible chronological order of the different stages for customers. For example, in the stage *DURING*, the customer may already has gained interest for the product (i.e., stage *BEFORE*), searched for the product through a, by the company, provided information system and bought it (i.e., stages *DURING*). Because the same customer went through these stages, more customer related data (e.g., event logs from different software systems) could be collected from different touchpoints. Therefore, on later stages there are more data available than on earlier stages, because later stages can use data from earlier stages as well.

In general, the analysis of *CJs* – together with their underlying touchpoints to customers – helps companies to understand their customers, making more revenue with them and thus, to win in the market (Lemon & Verhoef, 2016), (Richardson, 2010), (Maechler, Neher, & Park, 2016). This dissertation focuses on data driven methods to understand and visualize *CJs* and their related processes in information systems, where the further structure of the work is as follows: Sect. 2 introduces the key techniques which are covered by this work, and represent the scope of it as well. The section shows how process mining can be applied to reveal customer journey processes, which research gaps exist and what related research questions can be derived from the current state-of-the-art. Also, the used methodology in this work is outlined. Next, in Sect. 3, the publications of this cumulative dissertation are presented, where each publication is embedded in a particular topic, and for each topic,

¹<https://www.statista.com/statistics/871513/worldwide-data-created/>, accessed 2023-06-28

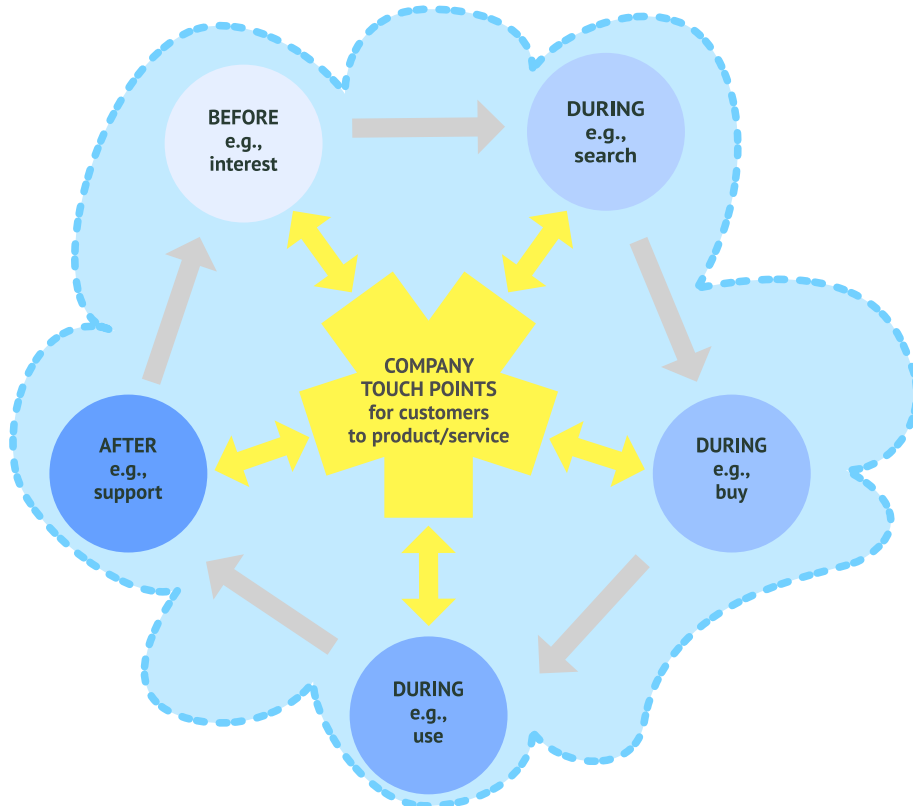


Figure 1: Customer Journey

a motivation is introduced and the tackled research questions are presented. There are three main topics, where the first treats how to prepare data for process mining and how to answer questions through process mining techniques to better understand customer journeys. The second topic covers the assessment of customer journey processes in terms of quality and user behavior. Finally, the last topic handles the visualization of customer journey processes with the goals, to reduce of the complexity on highly individual processes, to reduce the cognitive load on visualized process models and to gain insights from customer journey process models. The work closes with a discussion in Sect. 4. It covers, how this work answered the given research questions, concludes this dissertation and gives an outlook on further research topics, also beyond of the scope of this work. Furthermore, limitations are shown and finally, the work at hand finishes with findings and discovered results.

2 Process Mining on Customer Journey Processes

There exist several data driven methods for understanding *CJs*. One of them, which is very popular, is the development of a *Customer Journey Map (CJM)*, where the method is used to better understand the customer’s journey for consumed services of companies. It shows several touchpoints, which help to put the service provider (i.e., the company) in the “customer’s shoes”. For presenting a *CJM*, a visual chart can be used where the x-axis represents the ordering of the sequence of touchpoints and the y-axis lists the touchpoints in alphabetical order. Then, dots are visualized and connected with a smooth line to represent a particular *CJ*. (Bernard & Andritsos, 2019)

Another data driven method for companies to understand their *CJs* is to analyze related business processes, where a business process model is used to reflect the user behavior of customers. Such a process model, aka *Customer Journey Process (CJP)* (Terragni & Hassani, 2019), can, for example, visualize an end-to-end purchase process from customers or visualize different types for advertising touchpoints for attracting visitors from a corporate website. (Weijs & Caron, 2022) *Process Mining (PM)* offers well proven techniques, such as *Process Discovery*, *Conformance Checking* and *Process Enhancement* (Van Der Aalst, 2016), which leads to visualize, evaluate and enhance business processes. In other words, the main “idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today’s systems” (Van Der Aalst, 2016). For example, the discovering of (*CJPs*) by using *PM* techniques on system event logs of digital systems can help to identify particular *CJ* paths for optimizing a *Key Performance Indicator (KPI)* for a company. (Terragni & Hassani, 2019) Today, *PM* is used in many fields to discover *CJPs* (Bernard & Andritsos, 2019), (Weijs & Caron, 2022) and the usage of *PM* techniques together with *CJPs* is the main focus of this work. Case studies from different domains show that *PM* techniques for discovering *CJPs* – by using logs from digital systems – can be applied successfully, for example, in the telecommunication domain (Hansson et al., 2021), in the entertainment domain², in banking³, on pension investment companies (Syed, Leemans, Eden, & Buijs, 2020) and in the tourism domain (Lux & Rinderle-Ma, 2017), (Lux, Rinderle-Ma, & Preda, 2018), (Lux & Rinderle-Ma, 2019). For the last listed domain, real world cases, data sets and platform instances were accessible for the author of this work, because he is shareholder and *CEO* of the company *LuxActive KG*⁴, which provides a digital Web App platform for tourists as guest portal that contains touristic information, like point of interest or GPS tours, and also bookable products or services, like local wines or guided experience tours. The platform is called *oHA* and is sold to tourism companies, like accommodations or tourism associations in white label. But *oHA* is not just limited to the tourism domain because through the public funded H-2020 project *AURORAL*, new pillars for the mobility-, the public sector- and the energy domain arose⁵. (Gómez-Carmona et al., 2023)

CJPs are often discovered by using *PM* from information systems that provide a search functionality to customers for searching products or services, offered by a company. The data used for *PM* are event logs, where an event log entry consists at least of *a*) a user entered search query containing one or more search terms, usually in string format, *b*) a date and time which indicates when and in which order the event actually happened, e.g., in timestamp format, and *c*) a case identifier which represents what to track with a process, for example, a user, a device or a session. Because of the high variety of possible user entered search terms, such discovered process models lead to a paramount complexity, which are also known as *spaghetti processes* (Van Der Aalst, 2016). (Lux et al., 2018) An example of such a highly individual search process model is shown in Fig. 2.

From such highly individual process models, it is hard to gain insights for improving a *CJ*. Basically, two problems can be identified which are, 1) to do the right selection and preparation of the

²<https://fluxicon.com/blog/2018/02/case-study-customer-journey-mining/>, accessed 2023-05-03

³<https://www.youtube.com/watch?v=qJ2NcdZSxA4>, accessed 2023-05-03

⁴<https://www.luxactive.com>, accessed 2023-05-04

⁵<https://www.auroral.eu>, accessed 2023-05-04

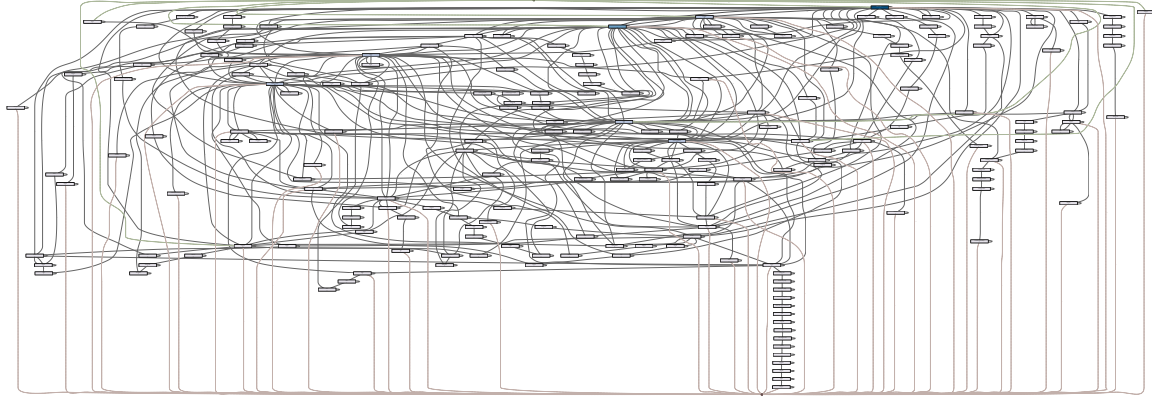


Figure 2: Mined Highly Individual Search Process Model from *oHA* using the Process Mining Tool *DISCO**

*<https://fluxicon.com/disco>, accessed 2023-05-04

available data for *PM* and 2) to assess the *CJP*-models to visualize simple but meaningful enough models to gain insights for improvements of *CJs*.

Fig. 3 shows the involved data driven steps and components to improve *CJs* based on *PM*, where one component is the *CJ* itself (cf., Fig. 1), which is embedded in this figure and was already introduced in Sect. 1.

First, the involved digital systems have to be identified and logs from these systems may be extracted and stored in a data warehouse. Furthermore, there may be several pre-processing steps involved to improve and prepare event logs for *PM*. With *PM* algorithms on those logs, process models are discovered and visualized. Different metrics help *a)* to assess the quality of mined process models and *b)* to reveal patterns from the user behavior. In addition, there exist several techniques to simplify complex process models without losing necessary information from these models, like the usage of semantic technologies in combination with preparation tasks on event logs. Also, visualization methods can be applied on discovered process models to reduce the cognitive load, e.g., by displaying nodes and edges with different colors or sizes. As a result, such *CJPs* help to improve the *CJ* on different stages and thus, to improve the *KPIs* of a company to win in the market, as pointed out earlier (cf., Sect. 1).

Those aforementioned involved components and steps build as well the foundation of this dissertation, whereas the associated research questions are being dealt in the next section.

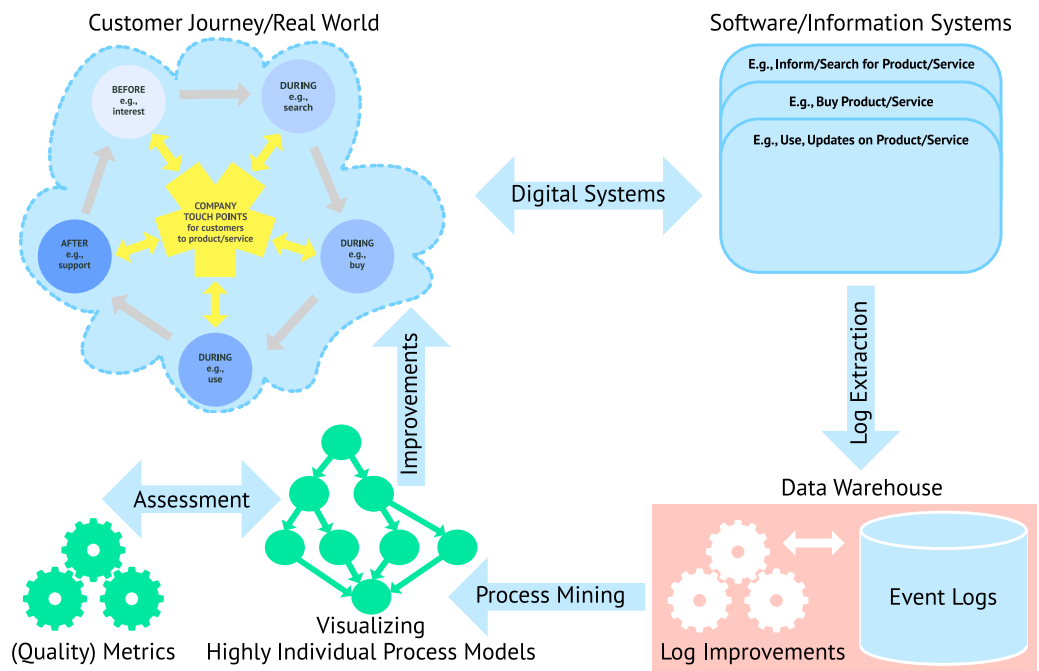


Figure 3: Customer Journey Process Mining

2.1 Research Gaps and Research Questions

This work at hand addresses research questions, having the overriding goal, to improve *CJs* by using process mining techniques to discover and visualize *CJPs*. In the following subsections, in alignment to the overriding goal, research gaps are outlined from state-of-the-art, where the *RQs* are then derived.

2.1.1 Preparation of Information Systems for Process Mining to Discover Highly Individual Customer Journey Processes

PM has become a well proven technique to answer user behavior related questions based on event logs from information systems (Van Der Aalst, 2016) (Van Der Aalst, 2011) (Van Der Aalst et al., 2012). A crucial precondition for applying *PM* is the access of high quality event logs (Van Der Aalst et al., 2012) from information systems. It is even more challenging to provide them from existing systems. (Bose, Mans, & Van Der Aalst, 2013) A literature review shows that many approaches and best practices exist, which show how to design a data warehouse and how to implement *Extract, Transform, and Load (ETL)* phases as well. (Eder, Olivotto, & Gruber, 2002), (Gupta, 2014), (Nabli, Bouaziz, Yangu, & Gargouri, 2015), (Stolba, 2007). Generally, the importance of data warehouses is shown through surveys, e.g., (Aftab & Siddiqui, 2018). In the particular context of *PM*, various data preparation and data warehouse approaches exist, where event logs are treated. (Jareevongpiboon & Janecek, 2013) (Dijkman et al., 2020) Also, the research topic to consider *CJs* together with *PM* is already established (Terragni & Hassani, 2019) (Hansson et al., 2021) (Weijs & Caron, 2022), where even the popular technique of *CJMs* (Bernard & Andritsos, 2019) (Bernard & Andritsos, 2017) is brought together with *PM* (Bernard & Andritsos, 2018). Even the usage of the data driven method to discover business processes from *CJ* log data to receive *CJPs*, which help companies to improve their *KPIs*, is covered in literature. (Terragni & Hassani, 2019) In addition, *CJPs* have in many cases the characteristic of being highly individual (Hansson et al., 2021).

But there is still a research gap when considering data preparation and data warehouse development together with *CJs* in the context of *PM*. There was no literature found, which covers all these topics together, with the exception of (Buijs, Bergmans, & El Hasnaoui, 2019), where a data hub called “data core” is developed, which copies data from different systems of the *Algemene Pensioen Groep*, the Netherlands largest pension fund provider, for using the data with the process mining tool *Celonis*⁶ to discover *CJs*. But in this publication, no extensive details about the data hub development, the architecture (e.g., the database schema) and the data preparation details (e.g., quality improvement methods of data) are provided. In the lessons learned of this publication, the following is pointed out: “*our data core approach requires attention in detail*” (Buijs et al., 2019). Furthermore, this publication, as well as the literature in general, lacks approaches to prepare existing information systems for *PM* to answer questions about *CJs*. Especially, to show how such questions can look like, business cases from a real world domain are needed. In addition, according to such business cases, application landscapes need to be developed for storing and preparing event logs for further applied *PM* in the context of *CJs* with highly individual processes.

Thus, the following *RQs* are derived:

RQ1 How to prepare information systems for process mining to discover highly individual customer journey processes?

[RQ1a] Which analysis questions on customer journeys can be answered by applying process mining techniques on logs in information systems?

[RQ1b] How to process data for applying process mining on logs in existing information systems?

⁶<https://www.celonis.com>, accessed 2023-06-28

2.1.2 Assessment of Highly Individual Discovered Processes from Information Systems on Customer Journeys

When event logs from information systems of, e.g., *CJs* are accessible, the mature and efficient technique *PM* algorithms (Van Der Aalst, 2016) can be applied on them to discover process models. *CJPs* generally tend to be highly individual (Hansson et al., 2021), especially on search processes, which are a special type of *CJPs*, where search terms in queries are defined by users in an arbitrary manner (Silverstein, Marais, Henzinger, & Moricz, 1999). Those process models lead to a paramount complexity, where such complex process models are also referred to as *spaghetti processes* (Van Der Aalst, 2016). Therefore, various methods, like (Fahland & Van Der Aalst, 2013), exist to simplify discovered process models. But after processes are simplified, the question arises, how to measure the process quality of a simplified process model in contrast to the original process model? There exist quality metrics from literature, which are, e.g., the *fitness* of the process conformance (Rozinat & Van der Aalst, 2008) or graph metrics (Mendling & Strembeck, 2008) like *size*, *diameter*, *structuredness*, *separability* and *cyclicity*. In addition, in (Vanderfeesten, Reijers, & Van der Aalst, 2008), *coupling* and *cohesion* have been proposed for measuring the relation or connection between activities. There exists also a *cross-connectivity* metric, which assigns weights to nodes and edges to reflect their connectivity (Vanderfeesten, Reijers, Mendling, van der Aalst, & Cardoso, 2008).

But by evaluating the quality of discovered *CJPs*, one could argue, that one quality metric alone cannot assess all quality aspects of a process model at the same time because such metrics might be contradicting, e.g., a model shows all details vs. a model is abstracting them (i.e. there exists a trade-off between *underfitting* and *overfitting* of process models and there is a balancing needed for the four quality dimensions *fitness*, *simplicity*, *precision*, and *generalization* (Van Der Aalst, 2016)). Furthermore, by considering a search process, which is discovered from an information system on a *CJ*, *coupling* and *cohesion* may not be meaningful in this context because no information objects are considered for search processes. Therefore, further research is needed to fill the gap to assess the quality of processes in information systems and in particular for search process models.

Even if *CJPs* are simplified and can be assessed in terms of quality by using quality metrics, the reflected user behavior from *CJs* has to be revealed as well from these models to improve, for example, *KPIs* of a company. (Terragni & Hassani, 2019). In particular, a search process captures the search behavior (Kuhlthau, 1991), where each activity in the process model represents a search query entered by users on a search system, which was, e.g., provided by a company. In web search analysis, logged search terms and the number of search results are taken into account (Silverstein et al., 1999). From Information Retrieval, in (Borisov, Wardenaar, Markov, & De Rijke, 2018), single interaction events, such as clicks, are modelled and predicted, where subsequently click sequences are proposed as well. In addition to these observations, a literature research showed, that the topic of search process models together with search behavior along paths was still not covered yet.

Therefore, the following derived *RQs* summarize the outlined aforementioned research gaps:

RQ2 How to assess customer journey processes in information systems?

[RQ2a] How to develop new metrics and which characteristics in those metrics are necessary in order to measure the quality of search processes from customer journey processes in information systems?

[RQ2b] How to develop new metrics and which characteristics in those metrics are necessary in order to reveal the user behavior of search processes from customer journey processes in information systems?

2.1.3 Visualization of Highly Individual Customer Journey Processes in Information Systems

Since *CJPs* tend to be highly individual (Hansson et al., 2021), methods are needed to simplify discovered process models. Generally, various methods exist (Fahland & Van Der Aalst, 2013). Some of them by constantly improving algorithms (Dixit, Buijs, van der Aalst, Hompes, & Buurman, 2015) but even by improving the event log quality (Ly, Indiono, Mangler, & Rinderle-Ma, 2012), (Ingvaldsen & Gulla, 2012). For the latter, semantic technologies (De Medeiros et al., 2007) (Van Der Aalst, 2016), e.g., ontology support can be applied (Dunkl, 2013). For example in (Koschmider & Oberweis, 2005), ontologies are used to reduce the complexity of process models by dealing with synonyms, hierarchies, reasoning or constraints. But to apply them on different domains in industrial solutions (Thomas & Fellmann MA, 2009), they are either too domain specific (Fernandez & Ponnusamy, 2016) or too complex and thus, burdensome (Pedrinaci, Domingue, & Alves de Medeiros, 2008). In comparison to the aforementioned literature, which considers predefined labels (Ingvaldsen & Gulla, 2012), an approach is missing, which addresses search processes, where arbitrary entered search terms from users are handled. Furthermore, on search processes, operational ambiguities could be the major source of data quality issues, which are produced through the description of process activities on different abstraction levels (Ingvaldsen & Gulla, 2012), by different used languages or based on homonyms and synonyms (Thomas & Fellmann MA, 2009). To use semantics together with a keyword based search is already covered by (Sugumaran & Gulla, 2011), but without log preparation and by using ontologies, which are easy to adapt with a minimum of effort on wide ranging domains and on business cases. Thus, for search process models on *CJPs*, approaches are needed to simplify process models by using ontology support.

A different angle to view is how process model elements itself are visualized. In (Van Der Aalst, 2016) it is noted that visualization concepts, like size or color, can be applied when constructing business processes. By utilizing, for example, colors to visualize the frequency of occurred activities, which are visualized as nodes in process models, similarities to cartography (Van Der Aalst, 2016), and in particular to choropleth maps (Tobler, 1973), are given, but with the difference, that those are not using geographic areas. By using choropleth maps for color mapping to nodes, two options exist: *a*) to use unclassified colors, which have the advantage to visualize a “raw accuracy”, or *b*) to use classed colors, which are easier to process for humans. (Tobler, 1973) This phenomenon is due to the fact that a few number of distinct colors to recognize helps to reduce the cognitive load and in addition, a legend lists the ranges of values which are corresponding to each color. (Dobson, 1973), (Dobson, 1980) Therefore, the approach to classify colors based on properties, like frequencies, on highly individual process models looks promising, but cluster algorithms are needed, like the algorithm *Jenks natural breaks*, which was developed to map colors on choropleth maps by producing classifications for a pre-defined number of classes on one-dimensional data, which can be applied, e.g., on frequencies. But this algorithm considers only to minimize the variation within each class. (Jenks, 1967), (Coulson, 1987) Moreover, the dealing with highly individual process models in the context of clustering has never been covered in literature, especially, when outliers occur in data sets, which may lead to overemphasizing them, which leads to build not even distributed data points over all clusters and therefore, some clusters are too small and others are too big. Thus, a requirement analysis for such process models and in further consequence, a comparison of existing algorithms – like *Jenks natural breaks* (Jenks, 1967), *k-means* (Arthur & Vassilvitskii, 2006) including variations of the algorithm, *DBSCAN* (Ester, Kriegel, Sander, Xu, et al., 1996), *head/tail breaks* (Jiang, 2013), *Kernel density estimation* (Scott, 2015), *Gaussian mixture model* (Reynolds, 2009), *mean shift* (Comaniciu & Meer, 2002) – is needed in this context and if none of them suits the requirements for highly individual process models, a new cluster algorithm has to be developed. In addition, the same clustering and visualization method can be applied as well on edges in process models, by visualizing them with different thicknesses or different distances based on their importance, e.g., determined through the frequency of the corresponding causal dependency between two activities, represented as nodes. (Van

Der Aalst, 2016)

In summary, the following research questions arise based on the previous outlined need for further research.

RQ3 How to visualize highly individual customer journey processes in information systems?

[RQ3a] Which methods can be used to simplify highly individual customer journey processes in information systems?

[RQ3b] How to develop new algorithms for reducing the cognitive load of visualized customer journey processes?

2.1.4 Research Questions Revisited

The derived research questions from the sub chapters before are listed together in the following:

RQ1 How to prepare information systems for process mining to discover highly individual customer journey processes?

[RQ1a] Which analysis questions on customer journeys can be answered by applying process mining techniques on logs in information systems?

[RQ1b] How to process data for applying process mining on logs in existing information systems?

RQ2 How to assess customer journey processes in information systems?

[RQ2a] How to develop new metrics and which characteristics in those metrics are necessary in order to measure the quality of search processes from customer journey processes in information systems?

[RQ2b] How to develop new metrics and which characteristics in those metrics are necessary in order to reveal the user behavior of search processes from customer journey processes in information systems?

RQ3 How to visualize highly individual customer journey processes in information systems?

[RQ3a] Which methods can be used to simplify highly individual customer journey processes in information systems?

[RQ3b] How to develop new algorithms for reducing the cognitive load of visualized customer journey processes?

The before introduced *RQs* are also embedded in Fig. 4, which shows the scope of this dissertation in the depicted light orange square, whereby this figure is an extension of the previously introduced Fig. 3 (cf., Sect. 2). The *RQs* and their related *papers*, which answer these *RQs*, are anchored at the particular position of the depicted scope. *RQs* are represented in magenta rhombuses and *papers* are shown in orange triangles with their abbreviations inside, which are explained in the following. To tackle *RQ1*, several data preparation tasks for companies are needed and analysis questions have to be formulated as well for further *PM* in order to obtain *CJPs* from information systems. This is handled in “*Paper: Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System (PPM)*”, where the development of a data warehouse and log preparation steps are treated before different *PM* algorithms can be applied on those data. Also, results from *PM* algorithms are shown, which answer different analysis questions on a *CJ* on a real world case from the tourism domain. Thus, the aforementioned publication builds the foundation of this work and for *RQ2–RQ3* as well. According to *RQ2*, metrics for the assessment of mined *CJPs* have to be developed, where in “*Paper: Assessing the Quality of Search Process Models (PM1)*”, a metric is developed, which measures the quality of search process models and its

influence by applying ontology support during the search and afterwards, on event log data through post-processing. In “*Paper: Analyzing User Behavior in Search Process Models (PM2)*”, user behavior patterns from search processes are gained by considering the number of returned search results. With *RQ3*, a visualization approach of highly individual *CJPs* is needed, where in “*Paper: DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling (PCL)*”, a cluster algorithm is developed to simplify highly individual *CJPs* by reducing the cognitive load on process visualization through, for example, different predefined colors for nodes in process models. *PM1* shows in addition, how to simplify *CJPs* through ontology support to prevent, for instance, *spaghetti processes* (cf., Sect. 2).

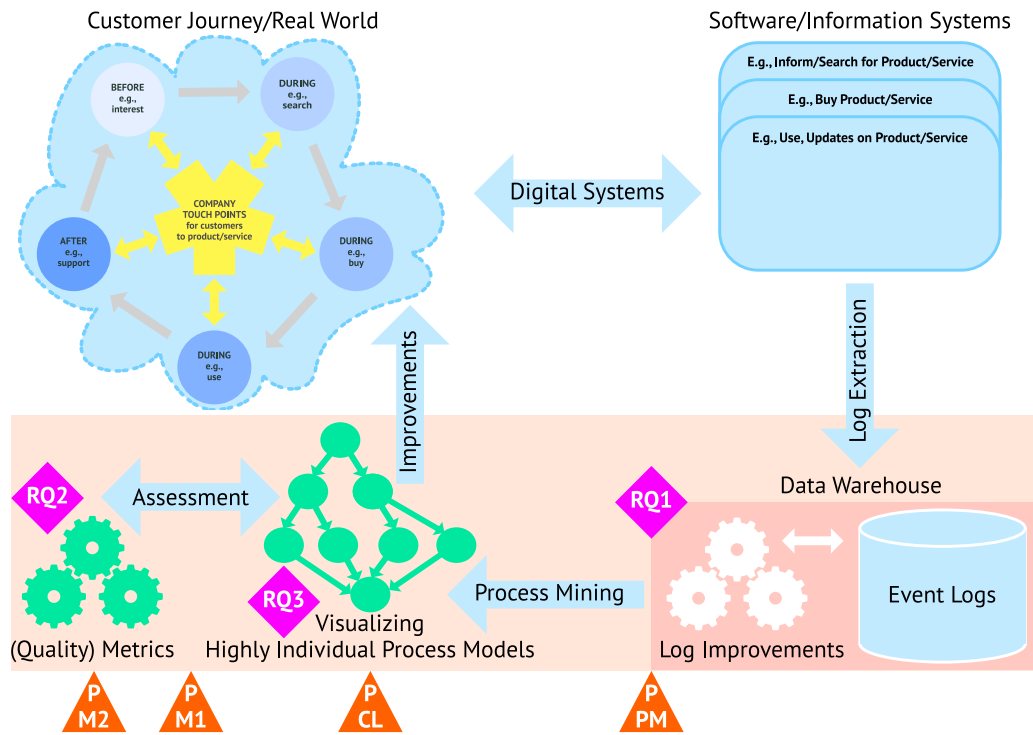


Figure 4: Customer Journey Process Mining with Research Questions and related Publications

2.2 Methodology

This work was primary processed through the methodology *design science research* (Wieringa, 2014) in information systems (A. R. Hevner, March, Park, & Ram, 2004), which follows generally the guidelines that are outlined in the following, with associated examples, which were performed during research:

1. “*Design as an Artifact*”, e.g., the development of a method which improves *CJPs* in the tourism domain by understanding the user behavior.
2. “*Problem Relevance*”, e.g., getting a clear evidence of the problem by discovering what *CJ* quality improvement means in that context.
3. “*Design Evaluation*”, e.g., the usage of metrics to measure the quality of *CJs* on real world data or by conducting associated experiments.
4. “*Research Contributions*”, e.g., when developing a new clustering algorithm for process visualization or a novel metric to reveal the user behavior in *CJPs*, those concepts are novel and filling a research gap. Furthermore, the concepts of those artifacts are well documented and the results produced by them are documented and reproducible as well.
5. “*Research Rigor*”, e.g., when developing a new clustering algorithm to improve the visualization of *CJPs*, a comprehensive research of state-of-the-art cluster algorithms and evaluation metrics in this context is performed beforehand. Therefore, existing foundations and methodologies can be used for development and the cluster algorithm is rigorously defined formally or in pseudo code.
6. “*Design as a Search Process*”, e.g., by iteratively improving existing algorithms, metrics and as well the *RQs*, because trough a constructed quality metric – to measure the complexity for *CJPs* – a new problem space may raise and new *RQs*, like how to reduce the complexity on those processes, emerge.
7. “*Communication of Research*”, e.g., to communicate the research results on international conferences and through international journals.

But sometimes, when dealing with complex systems and the development of artifacts on those systems – as they appear in information systems for *CJs*, which contain *a)* multiple touchpoints on different levels, *b)* highly individual processes, *c)* search functionalities with arbitrary search terms entered by the users, and *d)* different visualization options for presenting suggestions of suitable search terms or the visualization of *CJPs* – *action research* was used as well. This methodology is used for introducing changes into their complex processes and afterwards, for observing the effects of these changes (A. Hevner & Chatterjee, 2010). In particular, when metrics and algorithms were developed, based on real world data sets, *action research* was performed. For example, when a search process quality metric was developed, an experiment was conducted by using a touristic information system, containing real world data, where different event logs were collected from different subject groups. One group used an online ontology support for suggestions of search terms and the other group was not supported with search terms. The results from the generated logs helped to develop ontology support to reduce the complexity of search process models and to develop a quality metric to measure the complexity of search process models.

Evaluation of the developed concepts and artifacts: For evaluating the developed concepts and artifacts, which are presented in the research papers of this work, and to address *RQ1–RQ3*, the

accessible real world application *oHA* (cf. introduced in Sect. 2) was heavily used to gain evidence from a real world domain, in this case, the tourism domain. Therefore, real world event logs from *oHA* helped to build the main foundation of the research and were used massively (cf., *PPM, PM1, PM2*), but also experiments were conducted, which used the application *oHA* as well to produce event logs in a controlled real world setting by making changes in the application for different subject groups (cf., *PM1*). Of course, metrics were also used for evaluation, but when a comparison of a concept with many different data sets and existing methods was needed, also *simulation* was used – together with metrics – as evaluation method (cf., *PCL*).

3 Publications

The current section lists all the publications of this cumulative dissertation. Each subchapter covers particular research questions, which were defined in Sect. 2.1.

3.1 Preprocessing Data from Highly Individual Customer Journey Processes and Answering Questions through Process Mining

3.1.1 Motivation

As a prerequisite to answer questions through analysis tasks, such as *PM* on *CJs*, data from related software systems has to be extracted and stored properly. Answering such questions can help companies to develop new products or services, marketing activities or even to find their niches. To do so, several challenges arise, like the architecture of a data warehouse to store the data, the preparations of data to improve the quality, and the consideration and visualization of a typical *CJ* for the particular domain, i.e., a *CJ* visualization with its different stages, for example, from the tourism domain. The publication *PPM* handles all those challenges and shows a best practice approach for how to apply *PM* in the tourism domain by using the tourist information system *oHA*, which is provided as platform to tourism companies and used as guest portal by tourists. The paper acts as the foundation of this work as well – and also for the following introduced papers – which helped to prepare and provide the necessary research environment. One crucial requirement is to have high quality logs for further *PM*. To reach that requirement, log extraction has to be performed from software systems which are related to the particular *CJ*. Data is preferably stored in a relational data base in contrast to the file system – because on file systems, modifications of data is more difficult – where data can be improved through micro services which are responsible for, e.g., pooling of synonyms or building term clusters by using semantic technologies like an ontology. Then, process mining algorithms are used to show *CJPs*, which answer questions on different granularities (also known as *cases*), for example in the tourism domain, “what is a typical search process of a guest in an information system on a *CJ*?”, based for a user session, for a particular device or an accommodation. After all, lessons learnt are shown, like to use loose coupling between different systems and instances which report to a central data warehouse. Another lesson learnt is the physical separation of the data warehouse from the live system, because analytical operations on a data warehouse can be very time consuming and thus, may hinder the live system in its performance. The separation also has security advantages because the data warehouse system has its own protection and security measures. Furthermore, the use of non-time-critical micro services helps to enrich the logs and thus, to improve the quality of the stored data. Apart from technical aspects, it is also recommended to identify relevant cases and analysis questions before starting with *PM*, where analysis questions can be asked from different viewpoints, e.g., for a region, an accommodation or even a user session. *PM* and analysis results from the platform *oHA* are shown based on discovered search processes, revealed peak periods or most used services – also visualized in process models – on different *cases*.

Thus, the following research questions are tackled with *PPM*:

RQ1 How to prepare information systems for process mining to discover highly individual customer journey processes?

[RQ1a] Which analysis questions on customer journeys can be answered by applying process mining techniques on logs in information systems?

[RQ1b] How to process data for applying process mining on logs in existing information systems?

Summarized, according to *RQ1a*, *PPM* shows questions on *CJs* that can be answered by using logs from information systems for applying *PM*. And according to *RQ1b*, the same publication shows how to build a data warehouse architecture, which can be used on already existing systems as well, to process data for applying *PM* on logs.

3.1.2 Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System

Authors and Contributions: The authors are Marian Lux and Stefanie Rinderle-Ma. The concept was developed and evaluated by Marian Lux and the main writing of the paper was performed by him as well. Stefanie Rinderle-Ma contributed with supervision during concept development, refining and reviewing the paper.

Publication Status: Published (Lux & Rinderle-Ma, 2017): Lux, M., & Rinderle-Ma, S. (2017). Problems and challenges when implementing a best practice approach for process mining in a tourist information system. In M. Brambilla & T. T. Hildebrandt (Eds.), Proceedings of the BPM 2017 industry track co-located with the 15th international conference on business process management (BPM 2017), barcelona, spain, september 10-15, 2017 (Vol. 1985, pp. 1–12). CEUR-WS.org.

The final publication is available at <https://ceur-ws.org/Vol-1985/BPM17industry01.pdf>

Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System^{*}

Marian Lux and Stefanie Rinderle-Ma

Faculty of Computer Science, University of Vienna
{marian.lux,stefanie.rinderle-ma}@univie.ac.at

Abstract. The application of process mining techniques for analyzing customer journeys seems promising for different stakeholders in the tourism domain, i.e., the tourism providers are enabled to, e.g., find nice offers or partner services and the guests can improve their holiday experience. One precondition for mining processes (high quality) logs. This paper reports on experiences in implementing a data warehouse component for storing process logs in the tourism information system *oHA*. It shows which analysis questions can be answered by applying process mining and analysis on the logs. Finally, lessons learned are discussed.

Keywords: process mining, customer journey, data warehouse, tourism-information system

1 Introduction

This business case shows how we designed a sustainable and scaleable data warehouse architecture as well a log concept for the tourism-information system "*oHA*" (online Holiday Assistant)¹, which provides information and digital services for tourists. In more detail, *oHA* is a digital e-service, accessible for the tourist in form of a web app, mostly in a public WiFi, which is designed for tourism agencies and hotels to make more revenue with guests and provide a better service level to their guests. Fig. 1 shows three examples how the web app *oHA* looks like for a tourist guest. In the following we explain some technical details about *oHA* and go through the three displayed screenshots. This is important for understanding our application terminology and thus for understanding our log concept, later in this work.

The first screenshot shows the main menu of *oHA* with possible menu items to be selected by a guest. Every menu item corresponds to at least one digital service in *oHA*. There are lots of services in *oHA* and to name some of them, a

^{*} "M. Brambilla, T. Hildebrandt (Eds.): BPM 2017 Industrial Track Proceedings, CEUR-WS.org, 2017. Copyright 2017 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors."

¹ <https://www.luxactive.com/>

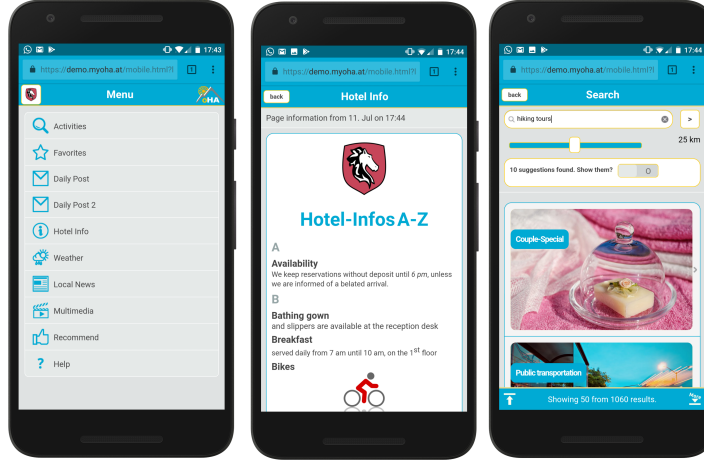


Fig. 1: *oHA* web app for tourist (using MockUPhone <https://mockuphone.com>).

service can be a hotel information (second screenshot), a activity search (third screenshot), a daily post, a regional news, the weather, or a GPS navigation. For instance on the first screenshot, by selecting *Hotel Info*, the second screenshot and by selecting *Activities*, the third screenshot shows up. Each of the displayed screenshots shows a different view in the application which has technically a place name for the current displayed view and we name the statistics behind that, *place usage*. The first screenshot shows, e.g., the place name *HomePlace* and the third *SerachActivityPlace*. On the third screenshot, a semantic search function for touristic activities is provided. The tourist can search for location based and time related *activities* like events near by, POIs (points of interest) or tours to navigate with *oHA*. We record the user entered *search terms* and try to generate processes out of the users search behavior (*search process*) with our stored data which will be covered in more detail later.

Analyzing the guest behavior is an opportunity to distinct *oHA* from competitive tourism information systems. For this reason the *CustPro*² project was initiated between the company LuxActive and the University of Vienna. Some of our presented concepts and techniques are already blueprinted and developed and others are still in development. With this work, we will show how we solved the main challenges when starting to implement process mining in a tourist information system. As first step, we designed a data warehouse for storing and preparing logs, to discover further research fields like process mining, aiming to analyze the customer journey process through the tourism platform *oHA*. Fig. 2 shows the different stages of a customer journey in the tourism domain and how it could be interoperated with process mining.

² <http://cs.univie.ac.at/project/custpro>

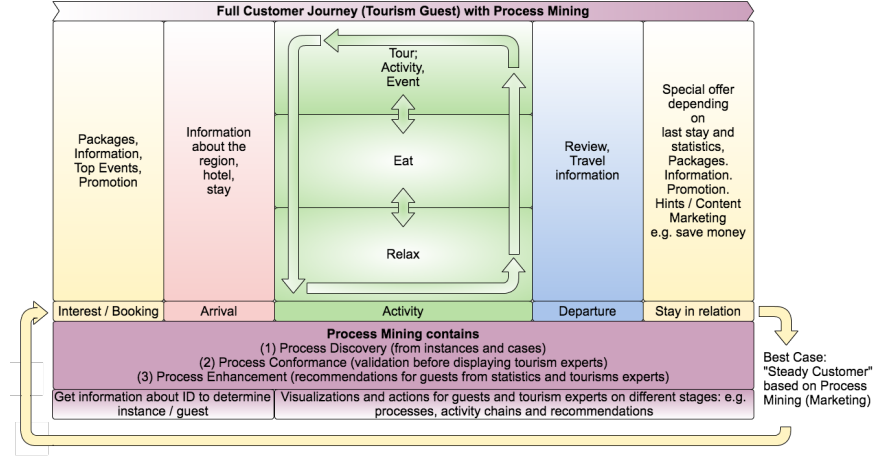


Fig. 2: Implementing process mining along the customer journey with different stages.

The first stage *Interest / Booking* bears the challenge to figure out preferences of the guests for booking a stay. The next stage *Arrival* is for providing all relevant information for the stay which is relevant for the individual guest. In stage *Activity* it is important to provide suggestions for individual activities and an easy way for consuming and booking them. In the stages *Departure* and *Stay in relation* it is important to get feedback about the stay and to encourage the guest and his surrounding people for booking again. For the latter, individual content marketing can be one method for achieving recurring bookings. Today *oHA* focuses strongly on the stages *Activity* and *Departure* but in future we want to cover all stages of the customer journey with *oHA*. As described before, every stage has different characteristics which require research and implementation. Also recorded logs from the different stages may influence each other. For example, recorded logs from the stage *Activity* may have influence to the stage *Interest / Booking* by serving the right information for promotion, out of historical data from guests.

The first step is to answer the following business process related questions based on the stages *Activity* and *Departure* for customers of *oHA* as the information can be useful for tourism companies when searching for niches, business partners, or increasing their revenue by providing new activities for tourists.

- Which digital services are used by guests mostly?
- Which searched activities like tours, events or POIs are most interesting for guests at the stay and after stay?
- What is a typical search process of a guest (cf. Fig. 3) and how to display it?
- When are the guests searching for services and activities and what are their peak periods?

- Which services and activities are missed by the guests?
- Which services and activities are mostly liked by the guests?
- Who will be a best fitting strategic partner for providing services and activities, e.g., a tour guide?

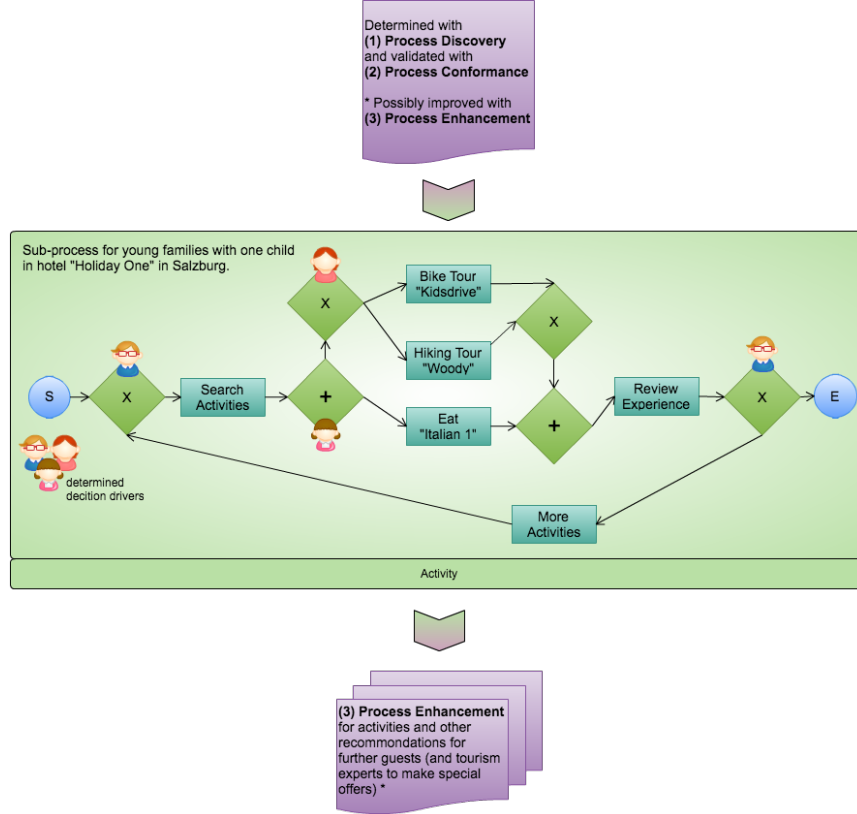


Fig. 3: A sample case on the stage *activity*.

The problem is that the original log implementation in *oHA* had no cases to answer questions on different levels and views. Hence it was not possible to mine processes from the level of individuals, because we could not distinguish between different guest devices. Furthermore, all logs were distributed in files on different local file systems. Thus, log preparations and modification tasks were time consuming and the logs were hard to access due to security restrictions on different servers. Also state changes in our system, which might influence the user behavior, were not recorded and thus taken into account by the logs. Such changes could be for instance hidden or shown menu items in the main menu or

new data sources for *oHA*. Lastly, we had no high-quality maturity level of our logs, which is recommended for instance by the process mining manifesto [1], before starting process mining with logs.

2 Related Work

A literature review suggested process mining [2, 1] as promising technology for answering user behavior related questions as described in the introduction. The preconditions for applying process mining techniques are (high quality) process logs which are challenging to provide in existing systems [3]. Observing data from multiple perspectives has been suggested by work on multidimensional structures in process mining (cf. e.g., [8]). Different approaches and best practices on how to design a data warehouse and how to implement ETL phases, also in the context of process logs, exist, e.g., [4], [7], [5], [12]. How important data warehouses are, is also shown in surveys. [10]. Further research and implementations on process mining in the data warehouse, would be to simplify discovered process models [9] and to improve the quality of process logs [3]. Regarding to our employed relational database, further security [11] or process mining approaches [6] can be researched, evaluated and implemented.

3 Methods and Techniques

This section presents design decisions and methods used for enabling process mining and analysis in *oHA*.

The previous situation in *oHA* was, that all logs including the user behavior were distributed over different file systems and in different file formats. So there was no possibility for tracking the user behavior of the tourists in an efficient way, without time-consuming manual interventions in logs on different file systems. Such interventions include manually gathering the logs, modifying them by removing outliers or test data, and converting them into a format which can be used for statistical calculations or process mining. This was overcome by implementing a central hub for our logs, which acts as data warehouse in our application landscape. An overview of our data warehouse architecture with its main workflow is depicted in Fig. 4.

We opted for an extra physical server environment with its own web server for managing and handling the logs in the data warehouse due to several reasons. Processing logs can be resource consuming and our production systems should not be impacted with performance issues because of resources like memory running out. A data warehouse database is designed to answer complex queries rather than performing a high throughput for updating transactions [4]. To use a web server in front of the database brings also advantages in security, because all the data traffic is encrypted and only authorized applications are able to log data and consume them via a defined API. Using a relational database in general for storing the data makes modifying logs easy to perform and data preparation tasks can be carried out with only a few steps by using, e.g., SQL queries. More

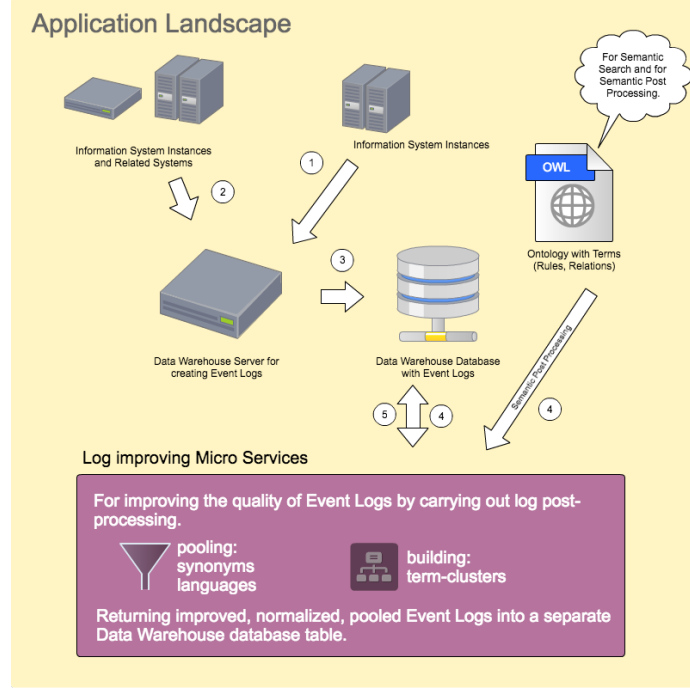


Fig. 4: Overview of the application landscape of *oHA* and the data warehouse. All server-applications (1), (2) send their log data to the data warehouse web server. The web server stores the data in a relational database (3). Log improvements are carried out by micro services or other web servers on the data warehouse, which use semantic technologies (4) and store the enriched log data back into the data warehouse (5).

technically, we use a Java web server, which is responsible for storing and processing the logs into the database. Our ETL (Extract, Transform, Load) process to the data warehouse is kept simple, because we have the full control over all systems which are logging. So we can also modify our systems which are logging, to fit the needs of the data warehouse. In future, we plan to include also external data from an open world environment like a weather API, tourism databases or a web crawler which is gathering important events nearby. The most notable approach in our current case is the following: If a web client logs events from a tourist, it sends the logs from the client to its responsible server. After that, the server sends the logs to the data warehouse. Due to the mentioned security reasons before, we try to keep our system secure, but with modest effort. For that, we disallow to send the log data directly from a client to the web server of the data warehouse. Only our servers are allowed send log data in JSON format to the data warehouse via our developed REST API with HTTPS.

Regarding the presence of (high quality) logs, in *oHA* some important data was missing, i.e., different cases from a session level to a region, the users language, additional timestamps, and search results of activities from tourists. Thus as shown in Fig. 4 for data transformation and data enrichment processes, the raw log data is processed by separate web servers or micro services. This is also an advantage for a loosely coupled architecture as implemented with our data warehouse web server, which is responsible for the whole data management and communicates via API calls. New web servers or micro services are easy to integrate now into the data warehouse. For instance we currently implement a web service, which uses semantic technologies for handling synonyms and different languages of logged search terms and converts them into a normalized form for improving the quality to further carried out process mining. The processed data is stored back in an extra database table in the data warehouse.

The *oHA* data warehouse database consists of the tables shown in Fig. 5. Because we use an iterative development approach which is still ongoing, not all following presented details are implemented now.

| Table Name | Columns | Indexes |
|---------------------------------|--|---------|
| client_actions | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) case_device VARCHAR(45) config_actions_last_modified DATETIME action_time DATETIME last_modified DATETIME action VARCHAR(45) item VARCHAR(256) place VARCHAR(45) language VARCHAR(45) | |
| place_usage | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) case_device VARCHAR(45) config_actions_last_modified DATETIME action_time DATETIME last_modified DATETIME place VARCHAR(45) place_count INT(11) language VARCHAR(45) | |
| server_requests | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) case_device VARCHAR(45) config_actions_last_modified DATETIME action_time DATETIME last_modified DATETIME service_name VARCHAR(45) language VARCHAR(45) | |
| search_terms | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) case_device VARCHAR(45) config_actions_last_modified DATETIME action_time DATETIME last_modified DATETIME search_string VARCHAR(256) kilometer_distance INT(11) search_result_count INT(11) | |
| config_actions | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) config_color1 VARCHAR(45) config_color2 VARCHAR(45) config_datasources VARCHAR(45) config_menu_items VARCHAR(45) action_time DATETIME last_modified DATETIME | |
| search_terms_postpromine | id INT(11) case_region VARCHAR(45) case_hotel VARCHAR(45) case_device VARCHAR(45) config_actions_last_modified DATETIME action_time DATETIME last_modified DATETIME search_string VARCHAR(256) kilometer_distance INT(11) search_result_count INT(11) original_uid INT(11) mined_term VARCHAR(256) pooled_term VARCHAR(256) pooled_query VARCHAR(256) | |

Fig. 5: Relational database tables from data warehouse.

Every table which contains logs has stored cases for region (*case_region*), a customer (*case_hotel*) and a device (*case_device*). For the latter, we implemented a client based solution to store a unique id in the local storage of the clients device which is mostly owned by a tourism guest. This id represents the case for the device and can be also used to identify a session. A session can be calculated together with timestamps of executed actions from the client. E.g. if there is no action with the same device for more than 10 minutes, we can infer a session. Identified sessions can be very useful for process mining. [2] Also current configuration states of the system are recorded with a timestamp field (*config_actions_last_modified*) in every relevant log table. Every time, a system state changes, the timestamp and the system new state will be recorded in the table *config_actions*. A state change is a system configuration change, which impacts the user behavior and thus the recorded logs. The most relevant state changes are currently the change of the displayed menu items, sources for searchable activities or color schemes of the web app *oHA*. So, if e.g., the source for *hiking tours* will be disabled by a tourism provider in the CMS (Content-Management-System) of *oHA*, which is called “*oHA Base*”, no tourist can see results after searching for activities which are related with *hiking tours* any more. By executing a SQL query together on both tables, i.e., the table which contains the system states (*config_actions*) and a table of interest for the logs (e.g. *serach_terms*) and by comparing the before mentioned timestamps for a given period, we can identify, in which state the system was, when the logs with the table of interest were created. Thus, this concept enables further improvements in regards to quality and meaningfulness of our logs.

Every client related log table also contains a language field which seems important for performing further analysis tasks. The data warehouse also includes a universal log table *client_actions*, which should log every action from the client in future implementations. It contains three relevant elements. The first is an action type *action*, which could be a selected button or focused text field. The second is the content of the action *item*, which contains, e.g., the title of a selected activity or an entered text. Finally, the third element contains a unique view name, like the *place usage*, from the client *place* for identifying where the action has taken place.

4 Results

With the enriched log concept and the central data storage, most popular search activities and services by tourists in *oHA*, on different locations and in different regions can now be determined. Moreover, analysis questions can be answered from different views due to the different implemented cases, i.e., regions, tourism companies (where every company has its own *oHA* instance), guest devices, and guest sessions. Fig. 6 shows, how mined processes for used digital services in *oHA* on different cases can look like. The first process shows the user behavior of a single user session, the second process shows the same user along the period of

one month and the third process shows, how all users in a hotel used the system in an one month period.

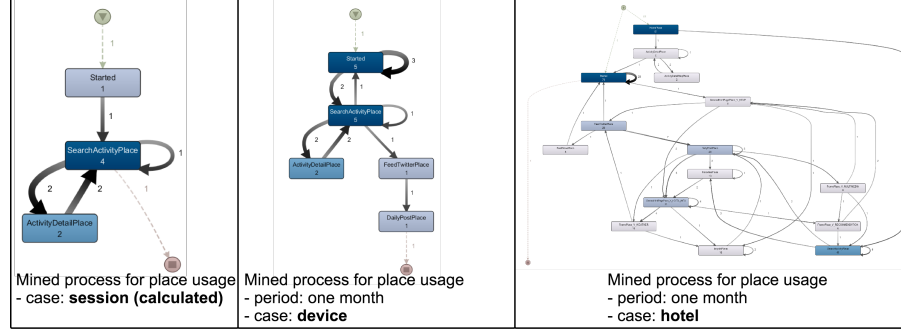


Fig. 6: Different cases for used services in *oHA* (using DISCO <https://fluxicon.com/disco/>).

We can also show which digital services are interesting for the guests on different cases (cf. Fig. 7). The first chart shows statistics from one device, the second shows the same type of statistics from the case of a hotel and the third one from the case of a region. In the first pie chart, the user was most interested in searching for activities. Looking for hotel news was less important. Indirect assumptions about missing services can be derived as well.

In the following, Fig. 8 (left) shows an example for a mined search process of a device which identifies a single guests behavior. One path of the process shows, that the user first searched for a tour and then for different variation of sights. We can also identify peak hours of a day, where guests are demanding different services in our system. This can be a useful information for coordination tasks in service for tourism companies. Fig. 8 (right) shows such an example how peak hours can look like on a hotel, after investigating logs for one month in the data warehouse. At noon, there was most demand of the service *oHA* and thus guests were looking for information.

5 Lessons Learned and Future Work

This work reported on the implementation of a data warehouse in a tourist information system. The primary goal was to improve the quality of logs for analysis tasks such as process mining and to finally understand the customer journey for tourism companies. This can help them in developing attractions, marketing activities, and finally finding their niches.

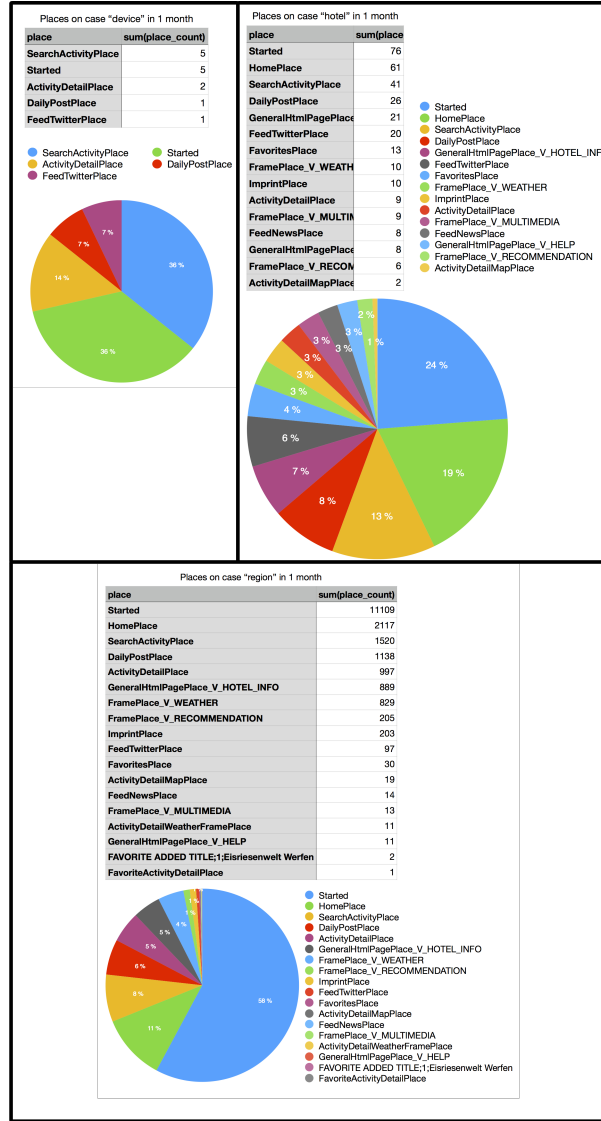


Fig. 7: Different cases for the most used services in *oHA*.

For storing logs, we would always prefer a relational database over a file system. It is much easier to deal with outliers or excluding test data on productive instances. Log modifications become easier and less time consuming as well. Also, transferring the logs from the data storage into a process mining tool

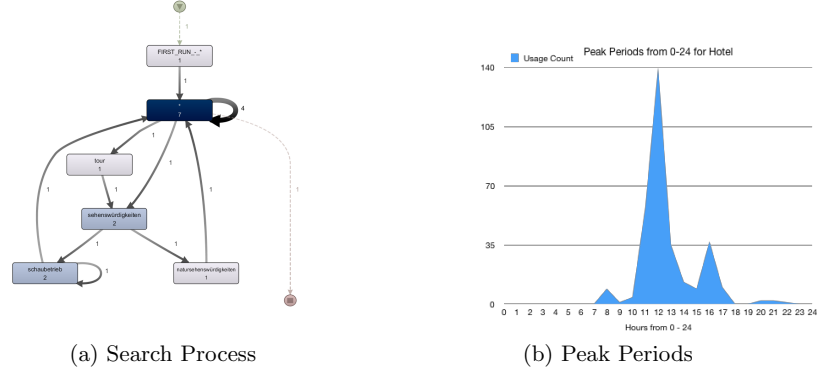


Fig. 8: Analysis Results on Search Processes and Peak Periods

can be done faster now. The database in *oHA* is realized by a service running on a (web-)server, which is responsible for managing the data and receiving the logs from different systems. Doing so we have achieved a loose coupling between different systems and instances, which are reporting to a central data warehouse. The decision to separate the data warehouse server physically from the log generating applications bears advantages with respect to security, because there is only one server to protect. This is particularly challenging with respect to data from user applications where different regulations for different countries exist. Another recommendation is to create a scaleable architecture to be prepared for answering further questions and to integrate new systems. It was also useful to design non-time-critical micro services in the data warehouse for enriching and processing the stored log data, e.g., for mining and visualizing search processes. Finally, automating the log processing task reduces the failure rate with respect to conclusions on the guest behavior. Apart from technical aspects we recommend to identify relevant cases and to define analysis questions before starting to mine processes. The more cases are identified, the more expressive the questions can be as most questions can be asked from different viewpoints, e.g., for a region, a hotel, a device, or a user session.

One future goal in *CustPro* refers to improving the quality of the mined models based on semantic technologies in terms of, e.g., complexity. We also want to study the transferability to other industries.

References

1. van der Aalst, W., et al.: Process mining manifesto. pp. 169–194. Springer (2012)
2. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
3. Bose, R., Mans, R., van der Aalst, W.: Wanna improve process mining results? In: Computational Intelligence and Data Mining. pp. 127–134. IEEE (2013)

4. Eder, J., Olivotto, G., Gruber, W.: A data warehouse for workflow logs. *Engineering and Deployment of Cooperative Information Systems* pp. 117–121 (2002)
5. Gupta, G.: *Introduction to data mining with case studies*. PHI Learning Pvt. Ltd. (2014)
6. de Murillas, E.G.L., van der Aalst, W.M., Reijers, H.A.: Process mining on databases: Unearthing historical data from redo logs. In: *International Conference on Business Process Management*. pp. 367–385. Springer (2015)
7. Nabli, A., Bouaziz, S., Yangui, R., Gargouri, F.: Two-etl phases for data warehouse creation: Design and implementation. In: *East European Conference on Advances in Databases and Information Systems*. pp. 138–150. Springer (2015)
8. Ribeiro, J.T.S., Weijters, A.J.M.M.: Event cube: Another perspective on business processes. In: *On the Move to Meaningful Internet Systems*. pp. 274–283 (2011)
9. San Pedro Martín, J.d., Carmona Vargas, J., Cortadella Fortuny, J.: Log-based simplification of process models. In: *Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31-September 3, 2015: proceedings*. pp. 457–474. Springer (2015)
10. Schamp, E.E.E., Schamp, E.: Status quo of big data analysis in small and medium size enterprises in Austria
11. Singh, A., Umesh, N.: Implementing log based security in data warehouse. *International Journal of Advanced Computer Research* 3(1) (2013)
12. Stolba, N.: Towards a sustainable data warehouse approach for evidence-based healthcare. *na* (2007)

3.2 Assessment of Customer Journey Processes in Information Systems and Reduction of Complexity in Highly Individual Processes

3.2.1 Motivation

By using *PM* algorithms to discover highly individual *CJPs*, the problem very often arises, that the resulting process models are of high complexity, which are then hard to analyze. In further consequence, it is difficult to gain insights from such process models to improve the related *CJ*. For example, as discussed before, when mining a search process based on event logs from an information system, containing individual user entered search terms, in many cases, *spaghetti processes* are resulting (as shown in Fig. 2). Therefore, different kinds of metrics and methods are needed. Metrics, to assess the quality of mined *CJP*-models and to reveal the user behavior. And methods, to reduce the complexity of highly individual process models. *PM1* addresses two analysis questions, where in the context of *CJs* the first asks, “*what is the typical customer search behavior*” and the second asks, “*what are critical customer touchpoints*”. For answering these analysis questions, a definition for a search process model is given first and based on that, a novel *Search Process Quality Metric (spm)* is defined, that rates the complexity in search process models, which are discovered by process mining techniques, combined with the assessment of the existence of clusters. The metric can be applied on a single node, a particular path or a whole process model. It considers in terms of *complexity* the number of edges incident to a node (aka activity) and the number of distinct activities. In terms of *clustering*, the metric considers the number of executions of an activity and the number of all occurred events in a log, where the process model was generated from. A higher value, where values range between $[-1; 1]$, indicates an important node or path, and a process model with a high value indicates that the model contains clear paths and is therefore less complex. In the same publication, also post processing on event logs is applied – for pooling user entered search terms to reduce the complexity on process models. The pooling concept was developed as micro service in a data warehouse (as described in *PPM*). For the post processing step, an ontology is used and the same ontology is applied for online ontology support to provide suggestions during the search for users in an information system. An experiment was conducted, where process models were generated with and without ontology support, as described before. The evaluation was performed with the *spm* metric and with regular process model quality metrics like *size* and *diameter* as well. Also, a statistical comparison of logs was conducted by applying a *t-test*. Especially the *spm* metric showed, that ontology support helps to improve the quality by reducing the complexity and improving clusters of mined search process models from information systems, which usually have a high level of complexity. If high quality search process models exist, its assessed user behavior yields a competitive edge for companies by providing, e.g., specific offers along a search process in an information system. To tackle such an assessment, in *PM2*, a definition of a search process model with search results is given, and based on this definition, a novel data-oriented *Search Behavior Metric (sbm)* – which is based on the number of retrieved search results per search activity – is introduced. This metric offers a novel approach because it is to employ data and data values, where in a nutshell, it considers the number of search results for an activity of interest and puts it into relation with the overall success of the search in a search process model. Additionally, the metric weights the search results by the relative number of executions of the activity of interest. This enables to differentiate whether, e.g., a high number of search results has been produced by a single activity execution or by several executions. This mechanic allows to identify search patterns along paths in a search process model and assesses the search behavior where different patterns are observed from the search strategies of narrowing and broadening.

All introduced metrics and methods were evaluated with real world data from the tourism domain out of the application *oHA*. The following research questions are tackled with the mentioned publications in this subchapter:

RQ1 How to prepare information systems for process mining to discover highly individual customer

journey processes?

[RQ1a] Which analysis questions on customer journeys can be answered by applying process mining techniques on logs in information systems?

[RQ1b] How to process data for applying process mining on logs in existing information systems?

RQ2 How to assess customer journey processes in information systems?

[RQ2a] How to develop new metrics and which characteristics in those metrics are necessary in order to measure the quality of search processes from customer journey processes in information systems?

[RQ2b] How to develop new metrics and which characteristics in those metrics are necessary in order to reveal the user behavior of search processes from customer journey processes in information systems?

RQ3 How to visualize highly individual customer journey processes in information systems?

[RQ3a] Which methods can be used to simplify highly individual customer journey processes in information systems?

In summary, *PM1* shows and answers analysis questions through process mining on search processes, which addresses *RQ1a*. To answer these analysis questions, ontologies are used for user support and for pooling of search terms (cf., *RQ1b*) to simplify mined highly individual search process models, according to *RQ3a*. Furthermore, a novel quality metric for search process models is introduced in the same publication, which helps to assess the quality of these process models, based on *RQ2a*, e.g., to compare them between the simplified process models with ontology support and process models without ontology support. Finally, according to *RQ2b*, in *PM2*, a novel search process quality metric is developed, which reveals user behavior patterns – by incorporating the number of search results – from users.

3.2.2 Assessing the Quality of Search Process Models

Authors and Contributions: The authors are Marian Lux, Stefanie Rinderle-Ma and Andrei Preda. The concept was developed and evaluated by Marian Lux and the main writing of the paper was performed by him as well. Stefanie Rinderle-Ma contributed with supervision during concept development, refining and reviewing the paper. Andrei Preda was just helping Marian Lux with the provision of the software infrastructure for the performed experiment for evaluating the developed concept.

Publication Status: Published (Lux et al., 2018): Lux, M., Rinderle-Ma, S., & Preda, A. (2018). Assessing the quality of search process models. In M. Weske, M. Montali, I. Weber, & J. vom Brocke (Eds.), Business process management - 16th international conference, BPM 2018, sydney, nsw, australia, september 9-14, 2018, proceedings (Vol. 11080, pp. 445–461). Springer.

The final publication is available at https://link.springer.com/chapter/10.1007/978-3-319-98648-7_26

Assessing the Quality of Search Process Models

Marian Lux^{1,2}, Stefanie Rinderle-Ma¹, Andrei Preda²

¹University of Vienna, Faculty of Computer Science, ds:UniVie, Vienna, Austria
marian.lux@univie.ac.at, stefanie.rinderle-ma@univie.ac.at

²LuxActive KG, Vienna, Austria, office@myoha.at

Abstract. Search processes are highly individual business processes reflecting the search behavior of users in search systems. The analysis of search processes is a promising instrument in order to improve customer journeys and experience. The quality of the analysis results depends on the underlying data, i.e., logs and the search process models. However, it is unclear what quality means with respect to search process logs and models. This paper defines search process models and revisits existing process model and log quality metrics. A metric for search process models is proposed that assesses their complexity and degree of common behavior. In order to compare metrics for search process models different logs and search processes are generated by using ontologies for user guidance during search process execution and for post processing of the logs. Based on an experiment with users in the tourism setting different logs and models are created and compared.

1 Introduction

It is a big asset for companies to know and understand their business processes. Process mining offers a bundle of promising techniques for discovering and analyzing business processes [1]. Business processes are ubiquitous and vary in their nature ranging from short-running and rather rigid administrative processes to highly individual processes such as patient treatment processes [12] and customer journeys describing the user interactions with the company [26]. Recent case studies show that process mining techniques can be successfully applied in order to derive customer journey processes from system logs, e.g., in the entertainment domain [25], in banking [3], and in the tourism domain [16]. Customer journey processes often imply search activities by the users, such as searching for activities when planning a trip. The search behavior can be captured as a *search process* [14] where each of the activities represents a search term a user has looked for through the search system provided by the company. Analyzing such search processes can provide valuable insight for companies [16] and answering the following **analysis questions (AQ)** through search processes:

- What is the typical customer search behavior?
- What are critical customer touch points?

These **AQ** can influence the customer satisfaction which helps companies to win in the market to increase their revenue [18].

The high variety in search terms, however, might lead to discovered search process models of paramount complexity (also referred to as spaghetti models [1]). Thus the promise of gaining valuable insight might be repealed by non-interpretable process models. Hence, assessing and improving the quality of discovered search process models is the prerequisite to reach analysis goals in a meaningful way. Further on, measures to reduce the complexity of the discovered search process models through pre- and post-processing of the analyzed search logs can be taken (cf. general data quality issues in process mining [1]). For search processes, operational ambiguities might be one major source of data quality issues as caused by description of process activities at different abstraction levels [11], but also by the usage of different languages or due to homonyms and synonyms [29]. Hence, this paper aims at assessing the quality of discovered search process models with respect to the **AQ**, considering how ontologies can be exploited for improving the quality of discovered search processes. In detail:

- Q1 How to measure the quality of search process models discovered by process mining techniques with respect to the **AQ**?
- Q2 Does the quality of search process models increase when users are supported by an ontology during the search process?
- Q3 Does the quality of search process models increase when using an ontology for log post processing?

This paper has an empirical focus with a concrete application setting, but also necessitates the creation of artifacts. As such the developed concepts are application-independent.

Research method and contribution: The paper follows design science research (cf. [34]). The relevance of the research problem is underpinned by practical applications from tourism [16], entertainment [25], and banking [3] as well as by literature, specifically on process model quality, e.g., [31] and process mining, e.g., [1]. The following artifacts are created to answer RQ 1–3. At first, a notion of search process models as well as a quality metric specifically tailored towards search process models with respect to **AQ** are proposed in Sect. 2 balancing complexity and clustering in search process models. Section 3 introduces the concept of using ontologies during process execution and for post processing of logs. These artifacts are then evaluated based on an experiment with users in Sect. 4: it creates 4 types of logs for different modi operandi, i.e., for executing search processes in the tourism domain with and without using an ontology and combining these logs with or without post processing. The logs are compared statistically and different metrics are applied to assess the effectiveness of using ontologies on the quality of the resulting logs as well as the feasibility of the newly proposed metric. The paper continues with a discussion in Sect. 5 and a related work discussion in Sect. 6. It concludes in Sect. 7.

2 Search Process Models and Quality Metrics

Search Process Models: One type of human-driven and highly individual processes are search processes. The manifestation of real-world search processes are

process logs as, e.g., stored by a tourism platform. Basically, process logs store events that refer to the execution of process activities together with the time stamp of execution and possibly further information such as the originator [1]. The events are grouped for the different process instances based on a case id. One can analyze the logs directly, but as we are particularly interested in typical customer behavior and touch points ($\rightarrow \mathbf{AQ}$) also the models behind these logs are of high interest. To the best of our knowledge no formal definition for search process models exists. In information science, informally, an (information) search process is defined as “*the user’s constructive activity of finding meaning from information in order to extend his or her state of knowledge on a particular problem or topic. It incorporates a series of encounters with information within a space of time rather than a single reference incident.*” [14]. In web (usage) mining, a log-based view is taken: a search or “*query trail qt comprises a user’s query q (consisting of a sequence of terms $\{t_1, t_2, \dots, t_{|q|}\}$ ” [2]. We converge and elaborate both definitions into a (graph-based) search process model:*

Definition 1 (Search process model). Let \mathcal{S} be set of all search terms. A search process model is defined as directed graph $SP := (N, E, l)$ where

- N is a set of nodes
- $E \subseteq N \times N$ denotes the set of control edges
- $l : N \mapsto \mathcal{S}$ denotes a function that maps each node to its label, i.e., $\forall n \in N$ n is a search activity, i.e., the node n represents the search for a certain search term and is labelled with this search term respectively.

Figure 1a depicts a small example for a search process model in the tourism domain. The search terms label the process activities, e.g., *bicycling* or *sports shop*, meaning that – after searching for *active* – a user has searched for *bicycling* followed by searching for *sports shop*.

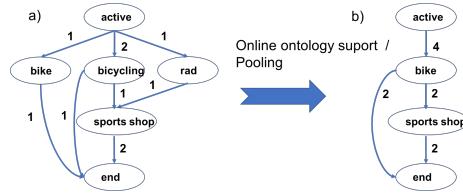


Fig. 1. Example search process model from tourism

One major difference to typical business processes is that a search process is not set out upfront to manage the behavior, but develops individually for each search during runtime. As we aim at ex post analysis of process data, we abstain from defining process instance states for search processes, but rather count the number of executions and annotate the control edges with this information as done for, for example, dependency graphs [33]. For the example shown in Figure 1a, 2 users searched for *bicycling* where one user followed up by searching for *sports shop*. Under the precondition that *active* is the starting point for all

searches, overall 4 search processes (and instances) were conducted in this example. This perception of the number of executions is suitable for, e.g., process analysis regarding question such as “what are the main search paths taken by the users within the platform”.

The example depicted in Figure 1 illustrates the tackled research problem. Even in this small example, 3 different search terms were used in order to describe the same concept, i.e., *bicycling*, *bike*, and *rad* (the latter being the German word for bike). If the goal is to analyze user behavior it could be more interesting to consolidate these terms into one term as depicted in Figure 1b where the 3 aforementioned terms have been pooled into search term *bike*. Here it can be seen more easily that 4 users were searching for a term related to concept *bike* and 2 users followed up looking for *sports shop*.

Quality Metrics: How can a metric assessing the quality of search process models with respect to the **AQ** be defined? We argue that one metric cannot assess all quality aspects of a process model at the same time as they might even be contradicting (e.g., showing all details vs. abstraction). The aim of the metric proposed in the following is to emphasize those properties of the model that relate to the **AQ**. Here, specifically, quality aspects refer to the comprehensibility of and the degree of common behavior in the search process models as well as the semantic enrichment of the process logs. Comprehensibility is tied to the complexity of the search process model by reducing the “spaghetti degree”. The degree of common behavior is reflected by clustering of activities in the model and the log. The quality metric does not refer to other quality aspects such as how well the discovered search process models reflect the underlying process logs (cf. fitness for process conformance [27]). In the following, the metric is constructed by considering existing business process quality metrics for assessing the complexity and metrics for process log quality for assessing the clustering. In Sect. 4 the new metric is then evaluated against selected existing metrics.

Graph metrics can be applied to assess the complexity of a process model [20]. Transferring this to a search process model $SP := (N, E, l)$ one can consider size ($|N|$), diameter (length of the longest path in SP), structuredness (share of nodes in structured blocks), separability (share of cut vertices in SP), and cyclicity (number of nodes in cycles in SP). For measuring the relation or connection between activities in process models coupling and cohesion have been proposed by [30]. Coupling measures “*how strongly the activities in a workflow process are related, or connected, to each other*”. The connection is measured based on the information elements shared by the activities. The cross-connectivity metric assigns weights to nodes and edges to reflect their connectivity [31]: nodes are weighed based on the number of outgoing edges, edges by the product of the weights of source and target nodes. Cross-connectivity seems promising for the envisioned quality metrics in terms of expressing the role of a node in a network and indicating clusters in the process models. Contrary, for search processes, coupling and cohesion are not meaningful in the context of this paper as no information objects are currently considered for search processes. However, such metrics are promising for future analysis.

Process log and model quality is a major concern in process mining [1]. Different techniques have been proposed to deal with “spaghetti degree”, including pre-processing of logs, process mining techniques, and post processing of logs. An example for pre-processing of logs is trace clustering [15] where logs can be clustered along certain criteria, e.g., for a certain process duration or where certain activities were executed. A process mining technique that aims at reducing the complexity of the mined models is the Fuzzy Miner [10]. It employs the principles of aggregation, abstraction, emphasis, and customization. Post processing as suggested by [6] also works with filtering, i.e., abstraction from details, in order to simplify the discovered models. From the principles of Fuzzy Miner and post processing aggregation and abstraction will be chosen for the assessment of search process models with respect to **AQ**. Moreover, the size of the logs will be considered in the proposed metric.

As an outcome of the above discussion, a quality metric for search processes shall incorporate ingredients of process model quality assessing the complexity and connection as well as the existence of clusters as used for process mining, i.e., the frequency of activity execution and the degree of the associate node as well as the number of overall activity executions in order to rate the frequency the activity of interest has been executed. Further on the overall number of events is incorporated to consider the overall diversity of the search process, formally:

Definition 2 (Search Process Quality Metric). *Let $SP = (N, E, l)$ be a search process and let L be a log created by executing instances on SP . Let further $|L|$ be the number of all events contained in L and A be the set of distinct activities having been executed in L . Then the search process quality metric $spm(n)$ for a node $n \in N$ is defined as*

$$spm(n) := 1 - \frac{degree(n) * |A|}{freq * |L|}$$

where $freq$ denotes the number of executions of n .

Search process quality metric $spm(SP)$ for SP turns out as:

$$spm(SP) := \frac{\sum_{n \in N} spm(n)}{|N|}$$

Search process quality metric $spm(SP)$ of a path $p = \langle n_1, \dots, n_k \rangle$ $n_i \in N$ ($k \geq 2$) can be determined similarly, i.e., by

$$spm(SP) := \frac{\sum_{n_i, i=1, \dots, k} spm(n_i)}{k}$$

Note that the metric avoids isolated nodes being considered as paths. By construction, $spm(n) \in [-1; 1]$ holds as $|A| \leq |L|$ and $degree \leq 2 * freq$.

3 Using Ontologies for User Support and Pooling

We consider the usage of ontologies to improve the process model quality of mined search processes. For that, we distinguish between two approaches where the same given ontology could be applied, a) during the search functionality, when users are entering search terms and b) when post processing event logs from search terms which were entered by users through the search process.

For a) – without the support by an ontology – the user has no guidance and in a broader sense no recommendations for entering search terms. When mining resulting search processes in the sequel, the discovered process models might get complex because of the possibly infinite options for search terms. For b), the search system has to possibly process synonyms and different languages which are known to pose challenges on later process mining [13], for example, resulting in different activities which have the same meaning and hence unnecessarily pump up the complexity of the search process models. In order to foster a) and b), we develop a meta concept which is responsible for recommendations when a user is searching and for post processing of event logs on search processes (cf. Fig. 2). Both approaches, a) and b) are implemented for a commercial tourism platform within the CustPro [35] project which aims to analyze the customer journey process of tourists where a) is already used in the live system by tourists and b) is implemented for evaluation purpose of this work but is also on the agenda to be implemented in the live system. The backend is written in *Java* as *RESTful Web-Service* and the ontology support with reasoning was conducted with the *Java* framework *Apache Jena*¹. The frontend was developed in *HTML5* and *JavaScript* as single-page application² and talks to the backend with *AJAX*. Therefore the frontend was accessible in the web browser and the UI was optimized for mobile devices.

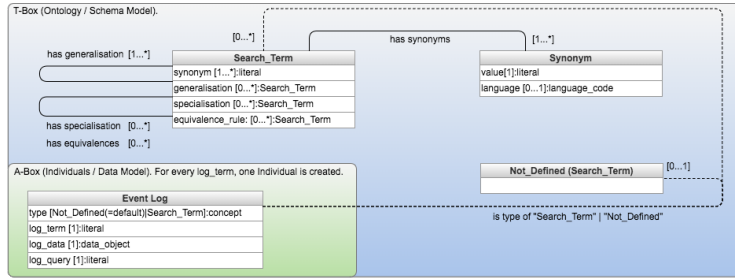


Fig. 2. Ontology with the capability of reasoning and possibilities for post processing

As set out in Fig. 2, a T-Box model [24] is defined which contains the knowledge base for possible search terms in a specific domain. There are only two concepts defined, which makes the ontology easy to implement and maintain. The first concept is called *Search_Terms*. It consists of the elements *synonyms*, *generalisations*, *specialisations*, and *equivalence rules*. Synonyms are literals in different languages which are referring to the possible search terms in the logs for the concept. Every *Search_Terms* concept has at least one *synonyms* element and every element describes the concept equally in contrast to e.g., SKOS [32] with property *skos:prefLabel*. Generalizations and specializations refer to the concept itself for defining relations between search terms and are optional. Also equivalence rules for defining relations, e.g., for combinations of search terms or

¹ <https://jena.apache.org/index.html>

² http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php

relations which cannot be defined in the outlined elements before, or for generating suggestions for search terms, e.g., based on combinations of search terms entered by a user before, are provided and also optional. The other concept, *Not_Defined* does not contain any knowledge. It acts as helper for post processing the event logs. Non-matching terms between the ontology and the logs are flagged with this concept.

Online Usage With Ontology: The T-Box model in Fig. 2 contains the ontology and thus a knowledge base for search terms in a specific domain. For every performed user search query, which can include multiple search terms, online suggestions are created during the search process for user support. Through rules in the ontology, search term specific combinations in a search query determine suggestions by using logical rule reasoning (e.g., a query containing "mountain" and "sports" results in a suggestion for "hiking"). Further, search term combinations like "x, y" in the same search query are separated into two search terms and for every search term based on its matching synonyms, specializations and generalizations are determined by using hierarchical reasoning (e.g., a specialization of "bike" is "mountainbike") for suggestions. As result, the user gets online suggestions based on the last performed search query with the given ontology.

Post Processing With Ontology: The A-Box [24] model in Fig. 2 contains the event logs which are individuals and defined with the concept *Event Log*. For every log query, which can include multiple search terms, a separate individual per search term is created. Hence, an event log with the search query "x, y" will be separated into two individuals, one for the term *x* and one for *y*. Each individual consists of the elements *log_term*, *log_data*, *log_query* and in the initial phase belongs to the concept *Not_Defined*. *log_term* refers to the search term, which will be processed by using logical rules with the T-Box model (ontology). *log_query* contains the original search query from the user and *log_data* acts as symbolic placeholder for further data from the origin log entry. We applied a logical rule which searches for matching synonyms between a given *log_term* and all *Search_Terms* from the ontology. If there is a match, the type "Not_Defined" from the individual is removed, if not already happened before, and the found *Search_Term* is added as type to the individual. After the rule was applied to all individuals, there is a lookup for individuals which do not contain the type "Not_Defined". For these individuals, the initial search strings of their corresponding logs are replaced with the class names from their *Search_Term* types. Therefore the logs contain pooled search queries for reducing the complexity of mined process models.

In this paper, we employ a controlled ontology (as defined by the analyst). Hence employing post processing does not lead to loss of information in the resulting models when compared to online usage. Contrary to online usage, in case of using an ontology that has not been defined by the analyst, post processing might not reflect the user's intention during search. In general, a user interface for the online ontology support through the search process could positively impact the frequency of its usage as well the quantity of observed logs.

4 Experiment

This section presents the design and execution of the experiment to evaluate the proposed artifacts, i.e., the metrics and the ontology support algorithms. The experiment bases on the implementation described in Section 3 and is conducted with subjects in a real world scenario.

4.1 Experimental Setup

The experiment was conducted with students of one course of the Bachelor Computer Science at the University of Vienna. This leads to a relatively homogeneous group of participants and can be regarded as sufficient with respect to knowledge on working with a tourism app. Overall, 93 students participated in the experiment. From an experimental point of view, 2 independent groups are required, i.e., one group working with ontology support and one without. Due to organizational reasons (the course is held in 4 groups), 4 independent groups were built where 2 worked with and 2 without ontology support.

Each of the 4 groups has the same scenario and task to accomplish: Every participant plans touristic activities on an imaginary three day stay from Friday to Monday in a hotel as tourist in the tourism region Mondsee in Austria. The subject writes down titles of activities on an empty schedule which was handed out [35]. The titles of the activities are searched in a search application which contains touristic activities. The experiments started with an introduction of 5 minutes explaining the task of the experiment. Then the subjects had a 10 minutes time frame to use their own mobile device (smartphone, tablet, notebook) to search for activities in the provided search application. Two of the four groups received ontology support in their search function. The search application and the ontology were provided in German. The search logs were recorded in the respective time frames and for every group. Different cases were recorded in the event log entries for distinguishing them. In Table 1, the groups of the subjects are depicted.

Table 1. Groups of subjects in experiment

| group number | online usage with ontology | number of different devices (case device) | number of subjects |
|--------------|----------------------------|---|--------------------|
| 1 | no | 19 | 20 |
| 2 | yes | 24 | 24 |
| 3 | no | 27 | 24 |
| 4 | yes | 24 | 25 |

Note that there is a difference between subjects and devices. The reason for that is, that two subjects can share the same device and one subject can use multiple devices. In the following, only the used devices are further addressed because the number corresponds to the recorded event logs during the experiment. As explained before, the groups with and without ontology support are to be compared. For this purpose the 4 groups from Table 1 are merged into 2 logs, i.e., group number 1 and 3 and group number 2 and 4 (cf. Table 2)

Table 2. Merged groups of subjects in experiment

| log name | group numbers | online usage with ontology | number of different devices (case device) | number of activities | number of events |
|----------|---------------|----------------------------|---|----------------------|------------------|
| log 1 | 1+3 | no | 46 | 117 | 246 |
| log 3 | 2+4 | yes | 48 | 116 | 331 |

We also implemented a web service, according to Section 3, for post processing the obtained event logs, which uses the same ontology as in the experiment for online supporting the subjects on their search functionality in groups 2 and 4. Therefore *log 2* contains post processed event logs from *log 1* which means that both logs originate from the same log recording. The scheme is analogous between *log 4* and *log 3*. Overall, this results in the 4 logs shown in Figure 3. These logs with their designated log names (*log 1* - *log 4*) build the basis for further analysis.

| Post processing \Rightarrow Online usage \Downarrow | Without ontology | With ontology |
|--|------------------|---------------|
| Without ontology | log 1 | log 2 |
| With ontology | log 3 | log 4 |

Fig. 3. Experimental log creation

4.2 Statistical Comparison of Logs

t-tests [21] are applied to compare the logs from Fig. 3 with respect to the mean of occurred events per *case device*. *Hypothesis I (HI): A higher mean in the logs results when providing online usage with ontology support.* As the subjects tend to perform more search queries because based on the recommendation for further search terms, the user does not have to think about formulating queries. Formulating *H1* is justified by the number of total events as in Table 2. *Hypothesis II (HII): A lower mean results when post processing the logs with ontology support because of pooling search queries.*

First we compared *log 1* with *log 3* to prove a statistical effect on online usage with ontology support. With a one tailed t-test and a 90% confidence interval, we obtained significance for *HI*. There was no evidence on a 95% confidence interval and as well on a two tailed test. Second, we compared *log 1* with *log 2* to prove a statistical effect on post processing logs with ontology support. *HII* cannot be accepted on a 90% confidence interval. Further log comparisons were not suitable because, as mentioned before, online usage with ontology support tends to increase and post processing with ontology support tends to decrease the mean. The complete t-test is shown in the supplemental material [35] (cf. folder "T-Test").

4.3 Process Model Quality Metrics

For assessing the process model quality of each log from Figure 3, we first applied selected process model quality metrics, i.e., *size* and *diameter* (cf. Sect. 2) as they provide an overview on the complexity of the models. Here, we also applied a filter, which counts only the 20% most frequent activities from each log (abstraction). We chose that percentage because of the *pareto principle*, which shows that in many cases, ranging from the economy to the nature behavior, 80% of causes are produced by 20% of activities [22]. Thus, depending on the given log with its distribution of activities and number of events, only activities with a specific frequency are included in the calculations. Table 3 shows the results which are explained in the following. For *log 1* and *log 2* the filtering has no effect, i.e., the least occurrence of activities remains 1. *log 1* had a *size* of 117 and the *size* of *log 2* was 110. The *diameter* was 19 for both logs. This shows that applying post processing has only a slight effect on the number of activities and no effect on the diameter. For *logs 3* and 4, filtering had an effect, i.e., *log 3* filtered on an activity occurrence ≥ 2 and *log 4* an activity occurrence ≥ 3 . With respect to the metrics, this results in a notably reduced size, i.e., for *log 3* a reduction of the size from 116 to 41 and of the diameter from 19 to 15 and for *log 4* a reduction of the size from 109 to 26 and of the diameter from 19 to 13. Hence, it can be interpreted that online ontology support has a considerable effect on reducing the process model metrics size and diameter when using filtering. There is also to mention, that each of *log 1* and *log 2* contained 33 variants of paths where only 3 variants contained more than 1 *case device*. Nearly the same picture was discovered on *log 3* and as well *log 4*. Each of them had 39 variants and only 2 variants contained more than 1 *case device*. This is a good indicator, which shows how highly individual a user performed search process can be.

Table 3. Results of applying regular process model quality metrics

| log name | size unfiltered | diameter unfiltered | size filtered | diameter filtered |
|-------------|--------------------|------------------------|------------------|----------------------|
| log 1 | 117 | 19 | 117 | 19 |
| log 2 | 110 | 19 | 110 | 19 |
| log 3 | 116 | 19 | 41 | 15 |
| log 4 | 109 | 19 | 26 | 13 |

As next step for assessing the process model quality of each log, we applied our defined *Search Process Quality Metric* from Section 2. Table 4 summarizes the results. In regard to the values of the path it is to be noted that there was as well a filter applied which includes only the 20% most frequent activities. But we also excluded paths which contain solely the search terms "*" or *FIRST_RUN-** in any combination. We did the latter because: the search term *FIRST_RUN-** signals, that after loading the application the first time on a device, an automatic "*" -search is performed. A "*" -search in the logs signals that the user just hit the search button without considering a search term as query input. Thus, such a path, where no search term was entered by a subject

has no meaning in our case and was filtered out. As we can see, from *log 1* to *log 4* there was an increase in the quality of the logs. We can also see that the online usage of an ontology had a bigger impact than post processing with ontology. The difference between *log 1* and *log 2* is 0,058 on *spm(SP) unfiltered* where the difference between *log 1* and *log 3* is 0,228. The values of *spm(SP) filtered* have a greater impact through the online usage with ontology. In comparison to the regular process model quality metrics (size and diameter), we can see an improvement in terms of meaningfulness by using our *Search Process Quality Metric* for identifying clusters and quality comparisons of logs. All values in Table 4 increased when using ontology support. We can conclude, that the ontology usage improved the process model quality in general and on specific paths.

Table 4. Results of applying *Search Process Quality Metrics*

| log name | spm(SP) unfiltered entire search process | spm(SP) filtered path highest value | spm(SP) filtered path lowest value |
|----------|---|--|---------------------------------------|
| log 1 | 0,145 | 0,714 | 0,107 |
| log 2 | 0,203 | 0,731 | 0,166 |
| log 3 | 0,373 | 0,781 | 0,412 |
| log 4 | 0,415 | 0,794 | 0,481 |

For visual inspection, process models were discovered for each of the 4 logs from Figure 3 using *Disco* (cf. Figure 4) which uses an adapted version of the Fuzzy Miner, called *Disco miner*, which is geared towards discovering clusters in process model³. The process models show only the most frequent activities and the most dominant paths in their process map. For this the *paths slider* was set to 0% to show only dominant connections between activities that have occurred and the *activities slider* was set to 20% to show only the 20% most frequent activities in the mined process map. Furthermore the absolute frequency of an activity is visualized using color strength. Only from visual inspection the resulting model for *log 4* seems to be less complex and to contain more clusters when compared to logs 1 – 3.

Finally, activities are selected through visual inspection, i.e., those showing high clustering based on color strength, and analyzed using the *Search Process Quality Metric*. We started with the models for logs before and after post processing. Search term "*Kino*" (cinema), for example, is compared for *log 1* with value 0,049 and *log 2* with value 0,285 as well search term "*essen*" (eat) with value 0,239 from *log 1* which was pooled to the term "*Gastronomie*" (gastronomy) with the value 0,292. We also compared the model from *log 3* with the search terms "*Restaurant*" (restaurant) with value 0,299 and "*Berg*" (mountain) with value 0,509 with *log 4* and their scorings for "*Restaurant*" with value 0,506 and "*Berg*" with value 0,539. These results confirm that post processing with ontology usage has a positive impact on search process quality. Then selected activities are compared for the same process model. For the model of *log 4*, we chose two frequent search terms. The first is "*Restaurant*" with term frequency of

³ <https://fluxicon.com/blog/2012/05/say-hello-to-disco/>

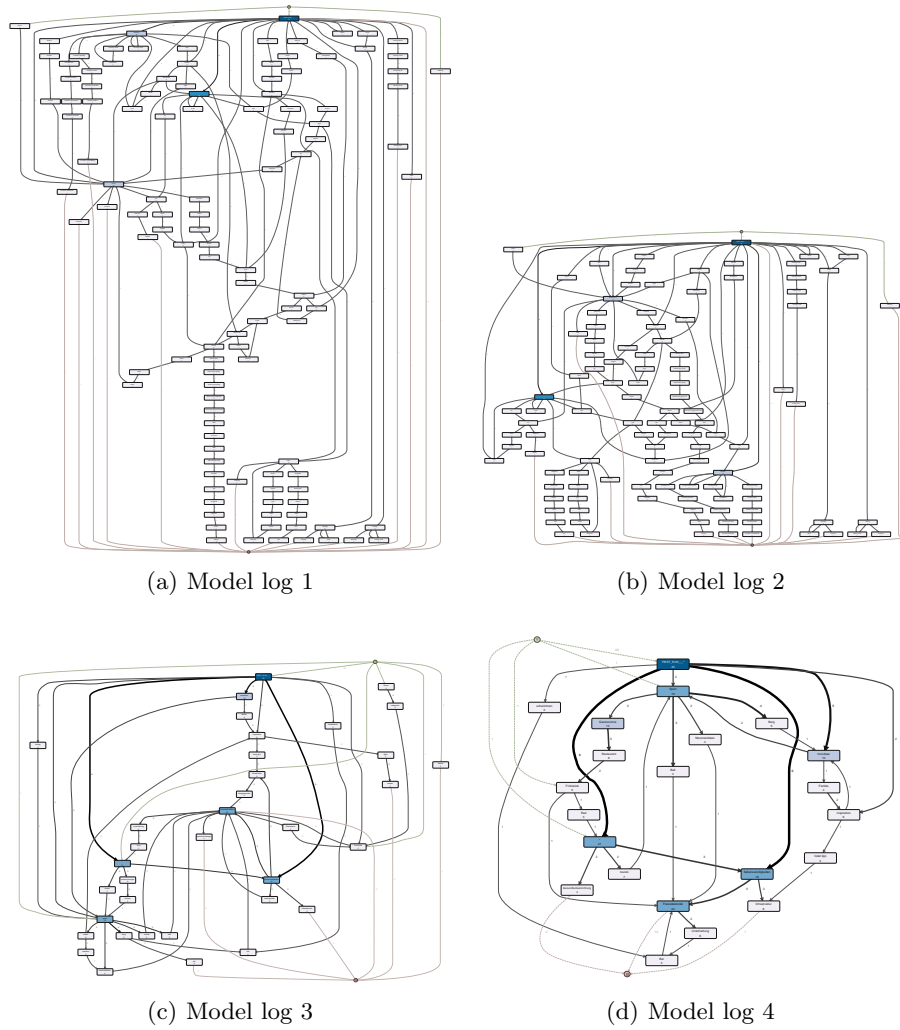


Fig. 4. Visual inspection of process discovery results based on logs 1 to 4 with enabled filter for showing 20% of most frequent activities and only most dominant paths

19 and a $spm(n)$ of 0,567. The second one is "*Freizeitaktivität*" (leisure activity) with term frequency of 24 and a $spm(n)$ of 0,575. Both terms had a lower value than the term "*Ball*" (ball) with value of 0,78 despite the lower frequency of only 3 but with a clearer path (lower degree). For the terms "*Familie*" (family) with a term frequency of 4 and "*Restaurant*" with a term frequency of 8 and with the same value of $spm(n)$, which was 0,506, nearly the same value was indicated than for the two very frequent terms "*Restaurant*" and "*Freizeitaktivität*". The reason for that is, that on "*Familie*" and "*Restaurant*" the incoming and

outgoing nodes were better clustered. We can conclude that the *Search Process Quality Metric* supports to discover and rank important paths and fragments in terms of clustering in search processes. The results, figures and calculations from the experiment can be found in the supplemental material [35] (cf. folder "Experiment").

5 Limitations and Threats to Validity

The results could be improved if the ontology increases in terms of its size and relevance to its domain where the search process is executed for. Moreover, the sample size of the experiment could have been too small for filtering and multiple languages. We will investigate the relation between filtering and sample size in future work. Because of the small sample size and therefore the small number of available subjects, we decided to run the experiment in one language, i.e., German, because the ontology contains a different number of classes and labels for describing them per language. Also rules for suggestions of combined search terms are not the same per language. This could be an explanation why post processing does not show a comparable impact to another study in the tourism domain on a live system that was conducted using German and English at the same time. Find the discovered models in Figure 5. Though there is not enough space to discuss this study in detail, the models give an impression that post processing using an ontology resolves ambiguities with respect to language. Another limitation of the proposed approach is the missing handling of compound nouns in search queries with mixed search terms that contain both, compound nouns and single nouns. In the tourism domain, we have had to deal with this problem and added hyphens between the search terms. The corresponding tourism ontology contains also hyphens for compound nouns, defined as synonyms. For example, if user enters search term "*nature sights*" it is modified to "*nature-sights*" and the label in the ontology is exact the same. But in the particular case of the tourism platform, we have to mention, that search queries with multiple nouns are very seldom. Nevertheless we are planning future research activities, to deal with compound nouns in search queries. A starting point would be the work presented in [28].

6 Related Work

Process mining algorithms have become mature and efficient [1]. There are various ways for simplifying discovered processes [6]. For improving the quality of process mining results, apart from implementing constantly improving algorithms [4], it seems obvious to improve also the event log quality. This can be performed in different kind of approaches [17,11]. One of them is the promising research field of semantic technologies [8], which is also quite proven very well. Process mining, combined with semantic technologies can improve the meaningfulness and therefore the quality of mined processes reasonably well [19,1]. Most approaches, for enhancing process mining with semantic technologies, are using

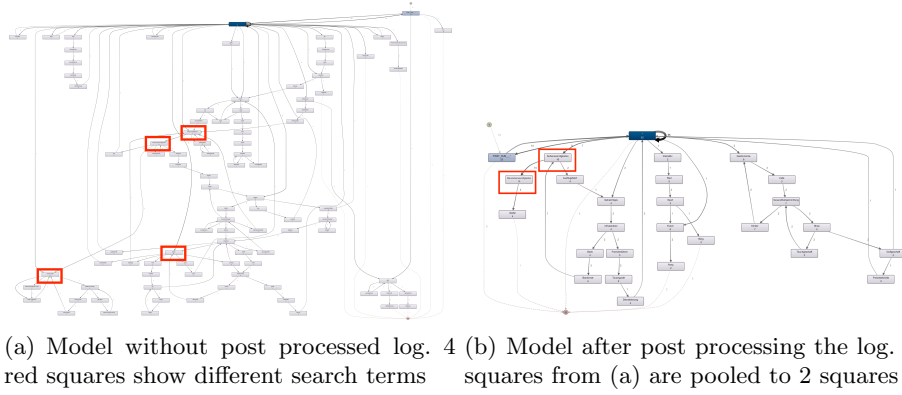


Fig. 5. Mined search processes from logs which contain different languages. Without post processing the log (*left*) and after post processing the log (*right*) which reduces the complexity of the model by pooling the search terms from different languages (*English and German*). The models can be found in the supplemental material [35] (cf. folder "*Figure5Detail*")

ontologies behind the scene for log preparation, e.g., by mapping process labels from event logs to hierarchical links in ontologies [5]. Ontologies are also used, to reduce the complexity of discovered process models by dealing with synonyms, hierarchies, reasoning or constraints [13]. But most ontologies, which are used for creating or enhancing semantic logs are either complex and thus burdensome [23] or too domain specific [7] for using them in different domains in industrial software solutions [29]. As opposed to all the aforementioned approaches this work addresses search processes where the search terms are defined by the users in an arbitrary manner, and not by the application in form of predefined labels as addressed mostly in literature and research [11]. Semantics, together with keyword search is also already covered in literature [9], but without log preparation and meta ontologies, that are easy to adapt with a minimum of effort in the widest possible range of domains and industries. Quality metrics from literature have been discussed in Section 2.

7 Conclusion

Case studies from different domains emphasize the potential of process mining for customer journey understanding and improvement. This work assesses and improves the quality of mined search processes as important brick in customer journeys. Regarding *RQ 1*, a newly proposed quality metric for search processes rates the complexity of the output combined with an assessment of the existence of clusters. The experiment evaluates the feasibility of the metric in comparison with results from visual inspection and existing metrics. Moreover, the experiment evaluates quality improvement when using ontologies for online user

support (*RQ 2*) as well as for post processing the resulting logs (*RQ 3*). The experiment is demonstrated by a case study in the tourism domain. In summary, *RQ 2* has been positively demonstrated in Sect. 4.2 by analyzing the mean of occurred events per *case device* and in Sect. 4.3 – where the results have been significantly reinforced by using filtering – by showing through existing metrics, visual inspection of mined process models, and the proposed quality metric, that the complexity of process models decreases and clusters improve. The same improvements could also be shown in Sect. 4.3 for *RQ 3*, but with slightly less evidence for the experiment because of the limitations as pointed out in Sect. 5. Overall, by answering *RQ 1–3*, it can be seen that complexity of and clustering in search processes improve, in particular, in combination with filtering. In future work, we will also consider time with respect to new metrics and experiment designs and measure the impact of an ontology on how accurate search results are for the users “*to get their jobs done*”. Moreover, the mined search processes will be further analyzed with respect to their differences in relation to context variables such as location or weather.

Acknowledgment: This work has been partly conducted within the CustPro project funded by the Vienna Business Agency.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Bailey, P., White, R.W., Liu, H., Kumaran, G.: Mining historic query trails to label long and rare search engine queries. *TWEB* 4(4), 15:1–15:27 (2010)
3. Celonis: Process mining story postfinance: Optimizing the customer journey in banking. <https://youtu.be/qJ2NcdZSx44> (2018)
4. Dixit, P., Buijs, J.C., van der Aalst, W.M., Hompes, B., Buurman, H.: Enhancing process mining results using domain knowledge. In: *SIMPDA*. pp. 79–94 (2015)
5. Dunkl, R.: Data improvement to enable process mining on integrated non-log data sources. In: *Computer Aided Systems Theory*. pp. 491–498 (2013)
6. Fahland, D., van der Aalst, W.M.P.: Simplifying discovered process models in a controlled manner. *Inf. Syst.* 38(4), 585–605 (2013)
7. Fernandez, F.M.H., Ponnusamy, R.: Data preprocessing and cleansing in web log on ontology for enhanced decision making. *Indian Journal of Science and Technology* 9(10) (2016)
8. Fürber, C.: Data quality management with semantic technologies. Springer (2015)
9. Gulla, J.A.: Applied semantic web technologies. Auerbach Publications (2011)
10. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: *Business Process Management*. pp. 328–343 (2007)
11. Ingvaldsen, J.E., Gulla, J.A.: Industrial application of semantic process mining. *Enterprise Information Systems* 6(2), 139–163 (2012)
12. Kaes, G., Rinderle-Ma, S.: Generating data from highly flexible and individual process settings through a game-based experimentation service. In: *Datenbanksysteme für Business, Technologie und Web*. pp. 331–350 (2017)
13. Koschmider, A., Oberweis, A.: Ontology based business process description. In: *EMOI-INTEROP*. pp. 321–333 (2005)

14. Kuhlthau, C.C.: Inside the search process: Information seeking from the user's perspective. *American Society for Information Science* 42(5), 361–371 (1991)
15. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inf. Syst.* 56, 235–257 (2016)
16. Lux, M., Rinderle-Ma, S.: Problems and challenges when implementing a best practice approach for process mining in a tourist information system. In: *BPM 2017 Industry Track*. *eur*, vol. Vol-1985, pp. 1–12 (2017)
17. Ly, L.T., Indiono, C., Mangler, J., Rinderle-Ma, S.: Data transformation and semantic log purging for process mining. In: *Advanced Information Systems Engineering*. pp. 238–253 (2012)
18. Maechler, N., Neher, K., Park, R.: From touchpoints to journeys: Seeing the world as customers do (March 2016), <http://bit.ly/2AAzjcJ>
19. Alves de Medeiros, A.K., Pedrinaci, C., van der Aalst, W., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L.: An outlook on semantic business process mining and monitoring. In: *On the Move to Meaningful Internet Systems*. pp. 1244–1255 (2007)
20. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: *Business Information Systems*. pp. 142–153 (2008)
21. Mertens, W., Pugliese, A., Recker, J.: *Quantitative Data Analysis: A Companion for Accounting and Information Systems Research*. Springer (2016)
22. Moore, H.: *Cours d'économie politique*. The ANNALS of the American Academy of Political and Social Science 9(3), 128–131 (1897), <http://bit.ly/2FGyANa>
23. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.: A core ontology for business process analysis. *The Semantic Web: Research and Applications* pp. 49–64 (2008)
24. Petnga, L., Austin, M.: An ontological framework for knowledge modeling and decision support in cyber-physical systems. *Advanced Engineering Informatics* 30(1), 77–94 (2016)
25. Pmrig, Y., Yongil, L.: *Customer journey mining* (2018)
26. Richardson, A.: Using customer journey maps to improve customer experience. *Harvard Business Review* 15(1), 2–5 (2010)
27. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* 33(1), 64–95 (2008)
28. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: Analysis of a very large web search engine query log. In: *ACM SIGIR Forum*. vol. 33, pp. 6–12. ACM (1999)
29. Thomas, O., Fellmann, M.: Semantic process modeling - design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering* 1(6), 438–451 (2009)
30. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Evaluating workflow process designs using cohesion and coupling metrics. *Computers in Industry* 59(5), 420–437 (2008)
31. Vanderfeesten, I.T.P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.S.: On a quest for good process models: The cross-connectivity metric. In: *Advanced Information Systems Engineering*. pp. 480–494 (2008)
32. W3C: Skos simple knowledge organization system reference (08 2009), <https://www.w3.org/TR/2009/REC-skos-reference-20090818/>
33. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: *Computational Intelligence and Data Mining*. pp. 310–317 (2011)
34. Wieringa, R.: *Design Science Methodology for Information Systems and Software Engineering*. Springer (2014)
35. WST: Supp. mat. (2018), <http://gruppe.wst.univie.ac.at/projects/CustPro/>

3.2.3 Analyzing User Behavior in Search Process Models

Authors and Contributions: The authors are Marian Lux and Stefanie Rinderle-Ma. The concept was developed and evaluated by Marian Lux and the main writing of the paper was performed by him as well. Stefanie Rinderle-Ma contributed with supervision during concept development, refining and reviewing the paper.

Publication Status: Published (Lux & Rinderle-Ma, 2019): Lux, M., & Rinderle-Ma, S. (2019). Analyzing user behavior in search process models. In C. Cappiello & M. Ruiz (Eds.), Information systems engineering in responsible information systems - caise forum 2019, rome, italy, june 3-7, 2019, proceedings (Vol. 350, pp. 182–193). Springer.

The final publication is available at https://link.springer.com/chapter/10.1007/978-3-030-21297-1_16

Analyzing User Behavior in Search Process Models

Marian Lux^{1,3}, Stefanie Rinderle-Ma^{1,2}

¹University of Vienna, Faculty of Computer Science, ²ds@univie, Vienna, Austria
marian.lux@univie.ac.at, stefanie.rinderle-ma@univie.ac.at

³LuxActive KG, Vienna, Austria, Vienna, Austria

Abstract. Search processes constitute one type of Customer Journey Processes (CJP) as they reflect search (interaction) of customers with an information system or web platform. Understanding the search behavior of customers can yield invaluable insights for, e.g., providing a better search service offer. This work takes a first step towards the analysis of search behavior along paths in the search process models. The paths are identified based on an existing structural process model metric. A novel data-oriented metric based on the number of retrieved search results per search activity is proposed. This metric enables the identification of search patterns along the paths. The metric-based search behavior analysis is prototypically implemented and evaluated based on a real-world data set from the tourism domain.

1 Introduction

Mapping and understanding Customer Journey Processes (CJP) has become a new trend recently. Signavio, for example, names customer journeys a “*strategic imperative*”¹. This is underpinned by case studies in several domains including tourism [6] and entertainment [8]. In a nutshell, customer journey describes the customer touchpoints/interactions with a company’s information system [9]. Search processes constitute one type of CJP as they reflect search (interaction) of customers with an information system or web platform. According to literature, “*a better understanding of user search behavior*” [2] is essential, however, has been restricted to the analysis of single events and sequences so far. We aim to bring together process-oriented analysis with the full range of patterns in a process model (cf. <http://www.workflowpatterns.com/>) and search behavior analysis.

Search process models can become complex as typically customer behavior tends to be diverse [7]. Hence, we proposed a structural metric for assessing the complexity of search process models in [7]. It was shown that together with semantic pre- and post-processing it is possible to derive search paths in the models at a structural level.

¹ <https://www.signavio.com/post/customer-journeys-as-a-strategic-imperative/>

In order to tackle the challenge of complexity, in this work, we focus on search paths in search process models and try to find out what the customer was searching for when following a certain path in the model. This is reflected by the key research question of this work:

RQ: How to assess search behavior along search paths?

Being able to answer this question yields a competitive edge for companies, for example, by providing specific offers along the search paths. Also the customers are empowered to inspect and improve their search experience.

In this work, we tackle **RQ** in a quantitative way, i.e., based on a search behavior metric. This metric requires an extension of the search process model definition provided in [7], i.e., considering the total number of search results of a search activity as data element. Search path patterns are suggested based on literature and the search behavior metric. With these patterns, search paths can be assessed with respect to the search behavior along these paths. One example, is a decreasing of search results in the search behavior which might hint at using more and more specialized search terms in this path. Also “jumping” as search behavior can yield interesting insights for the analysts.

The search path metric is prototypically implemented and the approach is applied to a real-world data set from the tourism domain. Several search paths can be identified and suggestions for the search offering of the company can be derived.

The paper is structured as follows: Section 2 repeats the structural metric and introduces the new search metric for search process models. Section 3 introduces and discusses search path patterns. Section 4 describes the evaluation and the application to a real-world data set. Section 5 discusses related work and Sect. 6 the presented approach.

2 Search Path Metrics

The goal of this work is to analyze (structural paths) in search process models with respect to the search behavior of the users along these paths. One parameter reflecting and influencing the search behavior is the number of search results [12]. If a user, for example, receives a too large number of results for a certain query she/her might decide to narrow down the search in order to obtain a lower number and hence a more targeted search result. Definition 1, hence, extends the definition of search process models from [7] by adding the number of search results obtained per search activity:

Definition 1 (Search Process Model with Search Results). *Let S be the set of all search terms. A search process model with search results is defined as directed graph $SP := (N, E, l, nsr)$ where*

- N is a set of nodes
- $E \subseteq N \times N$ denotes the set of control edges
- $l : N \mapsto S$ denotes a function that maps each node to its label, i.e., $\forall n \in N$ n is a search term.

- $nsr : N \mapsto \mathbb{N}_0$ maps each node to the total number of results achieved by the search.

Note that nsr refers to the total number of search results and not to the number of results that are possibly shown to the user (cf. paging). Figure 1 shows an example search process model consisting of four search activities (plus explicit start and end node). The number on the edges reflect the number of instances for which a the path containing the edge has been executed. Also shown is the corresponding process execution log L that contains the execution events for five instances $I1$ to $I5$. Each log entry reflects the execution of one activity together with the number of retrieved search results, e.g., the execution of **fitness** with 50 results for instances $I1$, $I2$, $I3$, and $I4$.

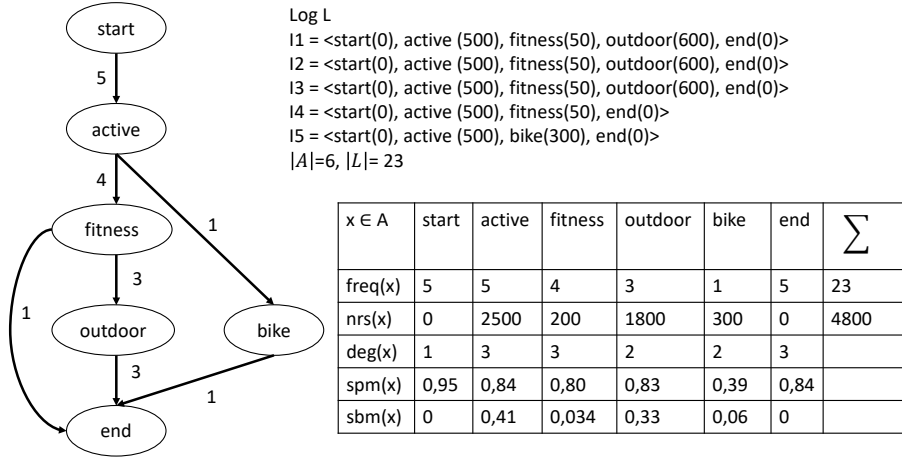


Fig. 1. Example Search Process with sbm and spb metrics

In our previous work [7], the search process quality metrics (spm metric) was found useful to assess the complexity of search process models and to find relevant search paths. The spm metric relates the degree and frequency of a node to the overall number of activities and number of entries in the underlying log (the one the search process model was mined from). The table in Fig. 1 contains the spm metric for each of the activities. For activity **fitness**, for example, spm turns out as $1 - \frac{3 \cdot 6}{4 \cdot 23} = 0.8$. One path of interest can be detected based on spm as **active** → **fitness** → **outdoor**.

To assess the user behavior along a search path in the search process model we introduce the *search behavior metric sbm* that is based on the number of search results. According to [12] search includes an iterative execution of “query formulation + reformulation” and “evaluation of the results”, following certain strategies that depend on the satisfaction with the number of retrieved search

results [11]. The number of search results is also considered in web search analysis [10]. The sbm metrics hence takes the number of search results into consideration and for an activity of interest puts it into relation with the overall success of the search in the search process model. Moreover, it weighs the search results by the relative number of executions of the activity that has produced the search results. Doing so enables to differentiate whether, for example, a high number of search results has been produced by a single activity execution or by several activity executions. The latter shall result in a higher value of sbm as more users have conducted the same search.

Definition 2 (Search Behavior Metric). *Search process model, L the corresponding log, and A the corresponding set of distinct activities. Let further $\text{freq}: A \mapsto \mathbb{N}_0$ count the occurrence of an activity $x \in A$ in L . Then the search behavior metric for x $\text{sbm}(x)$ is defined as*

$$\text{sbm}(x) := \left(1 - \frac{\text{freq}(x)}{\sum_{n \in N} \text{freq}(n)}\right) * \frac{\text{nsr}(x)}{\sum_{n \in N} \text{nsr}(n)}$$

with $\sum_{n \in N} \text{freq}(n) > 0 \wedge \sum_{n \in N} \text{nsr}(n) > 0$.

If activity x does not produce any search results (i.e., $\text{nsr}(x) = 0$), the metric yields a value of 0. As $|L| > 0$ and $|A| \leq |L|$, $|A| > 0$, $\text{sbm}(x) \in [0; 1]$ holds.

Consider again Fig. 1 where the sbm metrics is shown for all activities. Along the search path **active** \rightarrow **fitness** \rightarrow **outdoor** first $\text{sbm}(\text{active}) = 0,41$ is achieved, followed by an obvious narrowing down of the search to $\text{sbm}(\text{fitness}) = 0,034$. Then interestingly, the search is again widened to $\text{sbm}(\text{outdoor}) = 0,33$. Such “jumps” in the search behavior in one path might indicate a shift in the search strategy. How this search behavior can be analyzed and interpreted will be discussed in the Sect. 3.

3 Revealing Search Behavior on Paths in Search Process Models

In the following, we suggest search path patterns suggested in literature and investigate whether and how these patterns can be applied to analyzing search behavior along paths in search process models. The authors in [12] identify two iteratively executed search steps “query formulation + reformulation” and “evaluation of the results” in user search. They further name two basic search strategies applied during these phases, i.e., narrowing and broadening. Narrowing is applied if the number of search results is perceived as too high. It uses more keywords, conjunction (AND), or negation (NOT). Broadening is employed if the number of search results is perceived too low; it uses less keywords, disjunctions, or text processing techniques such as stemming. Also the use of ontologies, resolving synonyms/homonyms, and adding/removing constraints can support both of the strategies.

We “unroll” the steps “query formulation + reformulation” and “evaluation of the results” and their strategies according to [12] into search paths in the

search process model reflecting one line of search that was possibly shared by multiple users. The following patterns are based on a full combination of the strategies “narrowing/no narrowing” and “broadening/no broadening”². Aside the description of the pattern it is discussed how the strategy can be revealed based on the development of the sbm metric of the nodes in the path. The interrelation between the number of search results and the frequency of a node is further elaborated after the pattern descriptions.

Search Path Pattern 1 – Decreasing corresponds to the strategy of narrowing and no broadening, i.e., applying different strategies on the search terms in order to decrease the number of search results along the activities in a path. The manifestation of this pattern in a search process model is a path for which the contained activities unfold a decreasing sbm metric (cf. Fig. 2a).

Search Path Pattern 2 – Increasing corresponds to the strategy of broadening and no narrowing, i.e., increasing the number of search results along the path of interest. This pattern can be deduced from the search process model based on decreasing sbm metric values along a path (cf. Fig. 2b).

Search Path Pattern 3 – Jumping reflects search behavior that jumps between narrowing and broadening along a path, i.e., search results increase and decrease reflected by an increasing and decreasing sbm along a path. This can be caused by using different search terms within one path. We denote the points in the path where the sbm changes from decreasing to increasing and vice versa as jumping points (cf. Fig. 2c).

Search Path Pattern 4 – Constant Search reflects search behavior that produces a similar number of search results along a path, i.e., the variability in the sbm metric is low (cf. Fig. 2d). This pattern can result from using different search terms resulting in a similar amount of search results: *a)* By accident or if narrowing / broadening strategies are not effective, e.g., a specification of the search term does not result in any narrowing. *b)* If entered terms hardly limit search results because the provided search functionality follows the strategy not to limit search results but sorting them by relevance, e.g., during the search, ontology support helps to broaden and sort the results by considering the original entered search terms first, followed by its synonyms, specializations and finally its generalizations. *c)* The underlying overall quantity of possible search results in an information system is small, e.g., an information systems contains merely 10 substantially different documents to search for and most search terms return generally 1 search result. Therefore, most frequent search results contain 1 or 0 search results.

Let us dig a bit deeper into the behavior of the sbm metric for nodes in a path. Let $x_1, x_2 \in N$ be two nodes in a path of a search process model $SP=(N, D, l, nsr)$ where x_2 is a direct successor of x_1 (see Fig. 2). Then:

$$sbm(x_2) > sbm(x_1) \text{ if } nsr(x_2) > \frac{\sum_{n \in N} freq(n) - freq(x_1)}{\sum_{n \in N} freq(n) - freq(x_2)} * nsr(x_1) \quad (1)$$

² Note that we only use conjunction in this work.

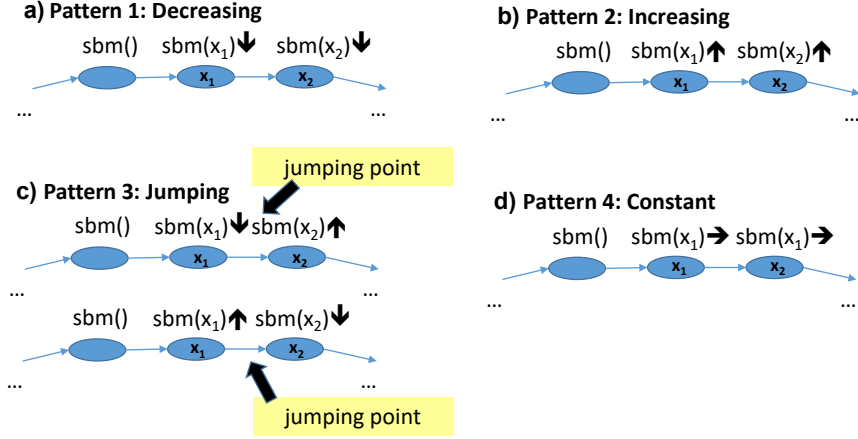


Fig. 2. Overview Search Patterns and Search Metric sbm

Equation 1 holds accordingly for sbm decrease by replacing $>$ by $<$. We fathom Equation 1 by analyzing corner cases.

$$sbm(x_2) > sbm(x_1) \text{ if } \begin{cases} nsr(x_2) > nsr(x_1) & freq(x_1) \approx freq(x_2) \\ nsr(x_2) > \frac{nsr(x_1)}{\sum_{n \in N} freq(n) - 1} & freq(x_1) \gg freq(x_2) \\ nsr(x_2) > nsr(x_1) * (\sum_{n \in N} freq(n) - 1) & freq(x_1) \ll freq(x_2) \end{cases}$$

In the first case, if the frequencies of x_1 and x_2 are roughly the same, the development of sbm follows the development of the search results, i.e., we can directly interpret the results on the strategies narrowing and broadening as described for the search patterns. The second case illustrates the case where $freq(x_1) \gg freq(x_2)$. Hence we put $freq(x_1) = \sum_{n \in N} freq(n) - 1 \wedge freq(x_2) = 1$ as extreme values. Then the number of search results for x_2 has to exceed the number of search results for x_1 divided by the overall number of node frequencies. For high overall frequency, this can mean a drop in search results for x_2 when compared to x_1 , i.e., x_2 has to produce less results than x_1 . If in the third case $freq(x_1) \ll freq(x_2)$, i.e., we set extreme values $freq(x_1) = 1 \wedge freq(x_2) = \sum_{n \in N} freq(n) - 1$, the number of search results for x_2 has to be higher than the number of search results for x_1 multiplied by the overall frequency. For corner cases two and three this can mean quite a difference in how the number of search results evolves for x_2 , however, typically it can be assumed that the number of search results will exceed the number of node frequencies which will cushion the effect. Altogether the sbm metric provides a balanced tool of evaluating the effects of higher or lower search results, but considering the node frequencies that contributed to the search results.

Note that for the abstract example in Sect. 2 the evaluation of the sbm metric does not allow any conclusion as the search results have been synthesized

in an arbitrary manner. The conclusiveness of the proposed search behavior path patterns will be evaluated on a real-world data set in Sect. 4.

The above set of patterns covers all combinations of “narrowing”, “no narrowing”, “broadening”, “no broadening”. However, further patterns are conceivable. **Search Behavior Pattern 5 – Homogeneous Search**, for example, is not tied to a path in a search process model, but results from finding cluster of activities with similar (homogeneous) search behavior. However, due to space restrictions, Pattern 5 will not be investigated in this work.

4 Evaluation

We evaluate the assessment of search behavior in search process models based on a log from a real-world application called *oHA*. This log comprises the data of four instances executed over the period of 1.5 months. The application is a commercial tourism platform which provides touristic information, called *activities*, from Austria (e.g., points of interest, events, tours, etc.) to tourists and as well locals³. The platform is accessible for users as progressive web app⁴ and contains about 294,000 activities. The keyword based search functionality for activities has a location based filter method with the option to define an individual search radius around a current or a selected position. An online ontology support guides users through the search functionality by showing suggestions for search terms based on their previous entered search terms and the same ontology is used to broaden, and as well to sort, the results by taking into account, synonyms, generalizations, and specializations.

The search logs are accessible through a *PostgreSQL* database and used as input to calculate sbm results on a mined search process model from these logs. One log record contains the following fields: The field *case device* contains a unique id which is automatically generated per user device. It is used to trace a users’ search path. *action time* contains the time stamp when a particular log entry was generated for representing the order of occurred activities inside a search path and is therefore used to calculate the edges in a process model. The *search string* represents the activity which results as node in a process model. Finally, the *search result count* shows the number of returned results and is used to calculate the number of search results *nsr*.

The search functionality has some characteristics, which may influence the resulting process model and are described in the following: There exists a special activity * which signals that the user just hit the search button without considering a search term as query input. The activity * is also automatically executed by the system, if a user enters the search functionality the first time, after manually selecting a new the search position or after deleting the whole search query by pressing a clean button. This results in many *-activities in the logs which are naturally reflected in the mined process models. As mentioned before, the system also shows recommendations for search terms which

³ <https://austria.myoha.at>

⁴ <https://developers.google.com/web/progressive-web-apps/>

can be selected by users. After such a term is selected, a new search will be automatically performed by the system with the added search term (e.g., the first query had *tours* and a user selects *mountain* as recommendation, an automatic search query *tours mountain* will be performed). Therefore the search results will be predominantly refined through the before described behaviour. In addition to the search query, the number of returned results is influenced by the user-selected position and distance radius. Therefore, the same search query – and thus activity – can return a different number of results.

Figure 3 depicts the process model mined in Disco (<https://fluxicon.com/disco/>).

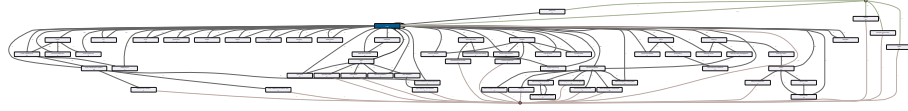


Fig. 3. *oHA* search process model with 10% of the most frequent activities

With a prototypical implementation in Java the spm and sbm metrics were calculated and annotated to the nodes in the search process model. Nodes with high spm results (which are closer to 1) indicate a clear path. Hence, we filter the model depicted in Fig. 3 for nodes with a spm metric, for example, of at least 0.96 (result see Fig. 4).

Then we visually inspect the search behavior based on sbm results of the contained nodes. Since the activity *** appears predominantly in every search path, at least at the beginning as described before, we ignore this activity for the pattern recognition and treat it as “outlier”. We can identify all four search behavior path patterns introduced in Sect. 3. Note that the logs are produced by the platform with a German and an English user interface. Most users used the German user interface. Therefore, the vast majority of the search paths contain German search terms, which are translated into English in the following for a better comprehensibility: “*familie*” = “*family*”, “*Spielplatz*” = “*playing area*”, “*touren*” = “*tours*”, “*wandertouren*” = “*hiking tours*”, “*tipp*” = “*advice*”, “*sehenswuerdigkeiten*” = “*sights*”, “*kulinarik*” = “*cuisine*”, “*kinder*” = “*children*”. The following paths are selected for further discussion regarding the identified pattern for the path, e.g., set P1 contains selected paths for which Search Behavior Path Pattern 1 – Decreasing can be revealed:

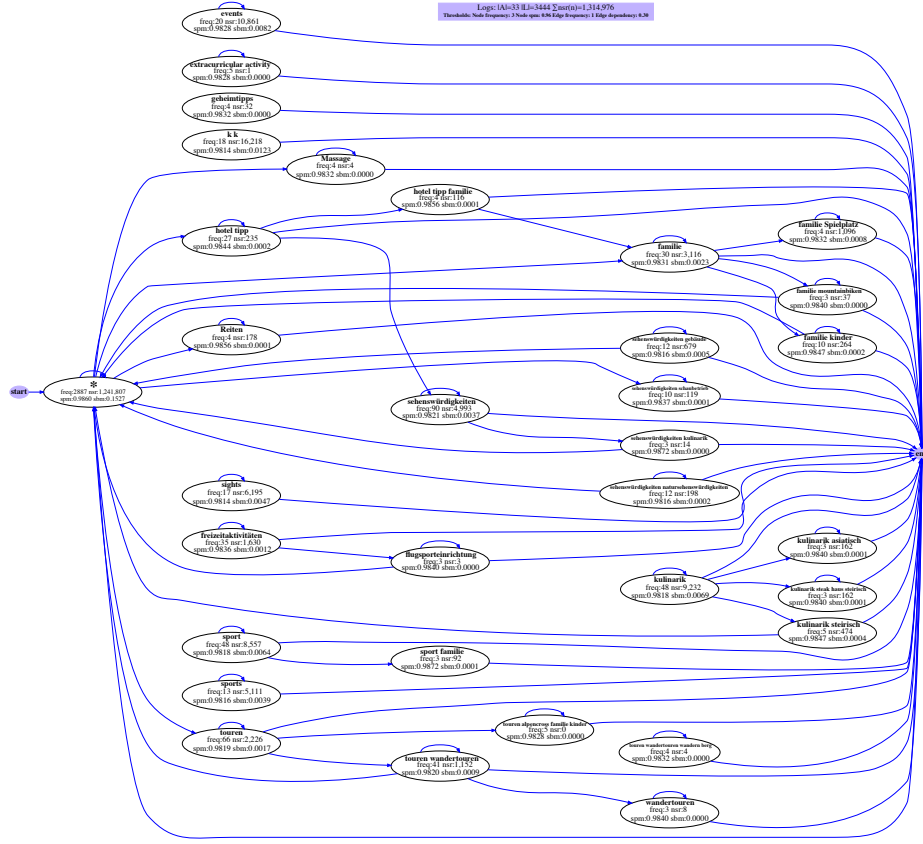


Fig. 4. oHA search process model filtered with $spm \geq 0.96$, produced by implemented prototype

$P1 = \{ \langle start, *(sbm = 0.1527), familie(sbm = 0.0023), familie Spielplatz(sbm = 0.0008), end \rangle, \langle start, *(sbm = 0.1527), sport(spm = 0.0064), sport familie(sbm = 0.0001), end \rangle, \langle start, *(sbm = 0.1527), touren(spm = 0.0017), touren wandertouren(sbm = 0.0009), wandertouren(sbm = 0.0000), end \rangle \}$

$P2 = \{ \langle start, *(sbm = 0.1527), hotel tipp(sbm = 0.0002), sehenswürdigkeiten(sbm = 0.0037), end \rangle \}$

$P3 = \{ \langle start, *(sbm = 0.1527), hotel tipp(sbm = 0.0002), sehenswürdigkeiten(sbm = 0.0037) sehenswürdigkeiten kulinarik(sbm = 0.0000), end \rangle, \langle start, *(sbm = 0.1527), hotel tipp(sbm = 0.0002), hotel tipp familie(sbm = 0.0001), familie(sbm = 0.0023), familie kinder(spm = 0.0002), end \rangle \}$

$P4 = \{ \langle start, *(sbm = 0.1527), hotel tipp(sbm = 0.0002), hotel tipp familie(sbm = 0.0001), end \rangle \}$

$P1$ refers to Pattern 1 – Decreasing and is most frequently revealed for the given use case. The sbm results are decreasing along the search path because

users refined their search results to find useful results. We illustrate one of the paths as model snippet in Fig. 5.

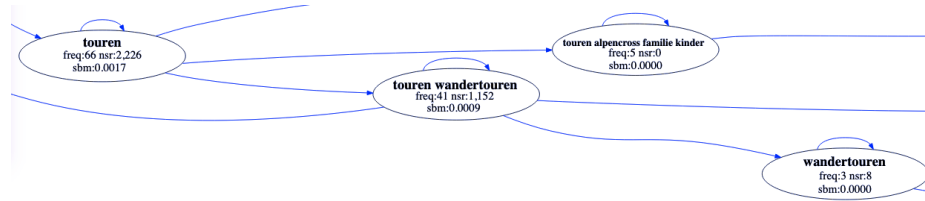


Fig. 5. Search path P1, snippet of search process model, produced by implemented prototype

P2 reflects an increase in the sbm results. Apparently, users tried to broaden their search results because not enough results appeared at the beginning of their search. This pattern appears rarely in the search process model. *P3* shows jumping points (cf. sehenswuerdigkeiten, familie) where users changed their search strategy during the search path. These jumping points can be of interest, e.g., in marketing, by investigating correlations between these jumping points and their previous search terms in their particular paths, for identifying e.g., new user recommendations. *P4* reflects a constant search path. In this example we assume that the users performed the broadening strategy and it was not successful. This pattern appeared only once in the discovered search process model.

Interpretation: The analysis of the search behavior shows that in several search paths users, who are tourists, started with “hotel tipp” which means some advise by the hotel they are staying in. From there, the search continued either with more specialized advises by the hotel (“hotel tipp familie”) or by choosing new, probably connected topic such as sights (“sehenswuerdigkeiten”). We can also see that from the more specialized search, users go back to a connected, but more general search term, e.g., from “hotel tipp familie” to “family” (in English “hotel advise family” to “family”). The interpretation could be that users in general try the advises offered by the hotel and from there apply different search strategies (narrowing and/or broadening) along topics they are specifically interested in, e.g., family or sights. Hence, we could suggest to a hotel owner with such paths to further invest in the hotel advises, particularly targeting certain topics such as family. Also, if the system provides an artificial intelligence function for recommendations of search terms, like the present tourism platform does as described before, the system could replace the suggestion “hotel tipp” with “sehenswuerdigkeiten” and “family”.

5 Related Work

In web search analysis, e.g., [10], search terms and the number of search results are considered, but not the underlying search process models and the search

behavior along paths. This observation is underpinned by current work in the Information Retrieval community stating that “[e]xisting work focuses on modeling and predicting single interaction events, such as clicks” [2] where the very same paper proposes the prediction of click sequences. The work at hand, however, is not restricted to click sequences, but considers process models with all kinds of structural patterns (cf. <http://www.workflowpatterns.com/>).

We can understand search processes as a special type of Customer Journey Processes (CJP). CJP comprise of all interactions and touch points of users – such as searches – with a company’s information system or (web) platform [5, 6]. It seems that currently commercial tools and systems such as Signavio are at the forefront to develop CJP maps and models. CJP mining has been recently discovered as promising in different application domains [8, 6]. [1] suggests clustering and merging CJP maps with process trees in order to detect representative CJP models from event logs. If a merge happens will be decided by the analyst. In [4] an alternative method to detect relevant CJP based on Markov models is introduced. [7] presents pre- and post-processing methods, together with a structural path metric in order to detect paths in CJP models. The proposed metric is employed in the work at hand. None of the mentioned approaches addresses the search behavior.

For mining search process models, metrics like the spm can be used for filtering nodes to show only paths of interest for analysts. For example the process mining framework ProM⁵ offers several metrics as well as edge and node filtering, particularly in the context of the Fuzzy Miner [3].

6 Discussion and Outlook

This work proposes structural and search behavior metrics for discovering and visually inspecting search process models. This enables the detection and analysis of search behavior along paths and facilitates suggestions for improving the search offer in the sequel. For a data set from the tourism domain, for example, it is possible to discover search paths and to derive that the tourism provider (e.g., a hotel) could improve its search suggestions. The novelty of the approach is to employ data and data values into metrics. Using search results is a first step, particularly suited for search process models, but further data can become relevant in explaining customer behavior, e.g., weather or location. Accordingly, new metrics and enhanced mining, filtering, and inspection techniques become necessary to yield the most valuable insights from the data.

References

1. Bernard, G., Andritsos, P.: Cjm-ab: Abstracting customer journey maps using process mining. In: CAiSE Forum 2018. pp. 49–56 (2018)

⁵ <http://www.promtools.org/doku.php>

2. Borisov, A., Wardenaar, M., Markov, I., de Rijke, M.: A click sequence model for web search. In: *Research & Development in Information Retrieval*. pp. 45–54 (2018)
3. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: *Business Process Management*. pp. 328–343 (2007)
4. Harbich, M., Bernard, G., Berkes, P., Garbinato, B., Andritsos, P.: Discovering customer journey maps using a mixture of markov models. In: *Symposium on Data-driven Process Discovery and Analysis*. pp. 3–7 (2017)
5. Lemon, K.N., Verhoef, P.C.: Understanding customer experience throughout the customer journey. *Journal of Marketing* 80(6), 69–96 (2016)
6. Lux, M., Rinderle-Ma, S.: Problems and challenges when implementing a best practice approach for process mining in a tourist information system. In: *Business Process Management, Industry Track*. pp. 1–12 (2017)
7. Lux, M., Rinderle-Ma, S., Preda, A.: Assessing the quality of search process models. In: *Business Process Management*. pp. 445–461 (2018)
8. Pmigg, Y., Yongil, L.: Customer journey mining. Tech. rep., LOEN Entertainment (2018)
9. Richardson, A.: Using customer journey maps to improve customer experience. *Harvard Business Review* 15(1), 2–5 (2010)
10. Silverstein, C., Henzinger, M.R., Marais, H., Moricz, M.: Analysis of a very large web search engine query log. *SIGIR Forum* 33(1), 6–12 (1999)
11. Stelmaszewska, H., Blandford, A.: Patterns of interactions: user behaviour in response to search results. In: *JCDL Workshop on Usability of Digital Libraries*. pp. 29–32. UCL Interaction Centre (UCLIC) (2002)
12. Sutcliffe, A.G., Ennis, M.: Towards a cognitive theory of information retrieval. *Interacting with Computers* 10(3), 321–351 (1998)

3.3 Visualization of Customer Journey Processes to Reduce the Cognitive Load and Gaining Insights

3.3.1 Motivation

It is crucial for business process models, like for discovered *CJPs*, that their visualization is easy to interpret and analyze. Especially, for highly individual process models, like in information systems, this may become a hard task. One promising method that can be applied to make such processes easier to analyze, is to build clusters based on different frequencies – on activities and on connections among them – to visualize these clusters as nodes with different colors or edges with different thicknesses. Based on the color or thickness, a legend next to the process model can be visualized, which indicates the corresponding frequency range. Such a visualization method is demonstrated on coloring nodes with a novel developed heuristic cluster algorithm called *1d distribution cluster algorithm (DDCAL)* in publication *PCL*. It denotes, that such a method may help to reduce the cognitive load when visualizing process models without over emphasizing outliers by the detection and differentiation of smaller clusters. In other words, the algorithm assigns colors to nodes in order to maximize the information gain of the visualization, even in the presence of “dominating” data points. This helps to make the visualization of a process model to become more meaningful. For example, it helps to analyze particular paths on process models, including the *happy flow* of a process, which is the most common path, or to highlight exceptional paths with particularly high or low frequencies of their nodes. Also through heat maps, regions with a high frequency in the process model can be highlighted. *DDCAL* constitutes a heuristic clustering algorithm for evenly distributing data into clusters over a maximum number of low variance clusters. In a nutshell, *DDCAL* uses an iterative approach by utilizing the feature scaling method *min-max normalization*, which is also known as *rescaling*, to normalize a sorted list of one-dimensional data points and by comparing the results against defined boundaries from a set list. Then, for each boundary, outliers from the upper or lower bound are considered as new clusters if the quantity of the elements of the potential clusters is inside a given tolerance factor. Otherwise, the next boundary from the list is considered and tested. If all boundaries were already tested, a tolerance factor for a minimum of elements in a cluster is increased, and the testing of the boundaries starts again. Therefore, in every iteration step, the lower or upper quantity of outliers from a boundary, which is inside a given tolerance factor - that is used to support an even distribution of elements over all clusters - is chosen for building a new cluster. The algorithm was compared to 11 existing clustering algorithms on 5 artificial data sets, which represent common distributions like the *normal*, *exponential* or *uniform distribution* and on 4 real-world data sets, where one of them was used for visualizing a *CJP* from the tourism domain. As shown in the publication, *DDCAL* can be used beyond *PM* as well, like for visualizing choropleth maps, which are in a way similar to business process models when visualizing frequencies in colored classes, but with the difference, that they do use geographic areas.

Thus, the following research questions are tackled with the above introduced publication:

RQ3 How to visualize highly individual customer journey processes in information systems?

[RQ3a] Which methods can be used to simplify highly individual customer journey processes in information systems?

[RQ3b] How to develop new algorithms for reducing the cognitive load of visualized customer journey processes?

To sum up, the publication *PCL* shows that the method of visualizing process model nodes in different colors and edges in different thicknesses, based on their importance, can help to reduce the cognitive load of these models and to show new insights on them, e.g., to reveal heat maps or to highlight important regions, which addresses *RQ3a*. To apply the aforementioned method, a novel cluster algorithm – to classify data based on their importance – is developed, addressing *RQ3b*.

3.3.2 DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling

Authors and Contributions: The authors are Marian Lux and Stefanie Rinderle-Ma. The concept was developed and evaluated by Marian Lux and the main writing of the paper was performed by him as well. Stefanie Rinderle-Ma contributed with supervision during concept development, refining and reviewing the paper.

Publication Status: Published (Lux & Rinderle-Ma, 2023): Lux, M., & Rinderle-Ma, S. (2023). DDCAL: Evenly distributing data into low variance clusters based on iterative feature scaling. *Journal of Classification*, 1–39.

The final publication is available at <https://link.springer.com/article/10.1007/s00357-022-09428-6>



DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling

Marian Lux^{1,2} · Stefanie Rinderle-Ma³ 

Accepted: 7 December 2022
© The Author(s) 2023

Abstract

This work studies the problem of clustering one-dimensional data points such that they are evenly distributed over a given number of low variance clusters. One application is the visualization of data on choropleth maps or on business process models, but without over-emphasizing outliers. This enables the detection and differentiation of smaller clusters. The problem is tackled based on a heuristic algorithm called DDCAL (1d distribution cluster algorithm) that is based on iterative feature scaling which generates stable results of clusters. The effectiveness of the DDCAL algorithm is shown based on 5 artificial data sets with different distributions and 4 real-world data sets reflecting different use cases. Moreover, the results from DDCAL, by using these data sets, are compared to 11 existing clustering algorithms. The application of the DDCAL algorithm is illustrated through the visualization of pandemic and population data on choropleth maps as well as process mining results on process models.

Keywords Heuristic clustering · Classification · Data visualization · Choropleth maps · Process mining

1 Introduction

In 2021, the size of information that was “created, captured, copied, and consumed world-wide” accounted for 79 zettabytes and will grow to 181 zettabytes in 2025.¹ “Information

¹ <https://www.statista.com/statistics/871513/worldwide-data-created/>, accessed 2022-08-08

✉ Stefanie Rinderle-Ma
stefanie.rinderle-ma@tum.de

Marian Lux
marian.lux@univie.ac.at; marian.lux@swisdata.com

¹ Research Group Workflow Systems and Technology, University of Vienna, Vienna, Austria

² SWISDATA GmbH, Vienna, Austria

³ TUM School of Computation, Information and Technology, Technical University of Munich, Munich, Germany

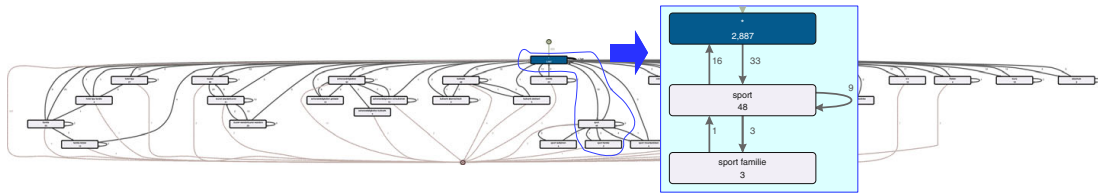


Fig. 1 Mined process model using DISCO on Data Set 1 containing frequencies of activities (search terms)

visualization can accelerate perception, provide insight and control, and harness this flood of valuable data to gain a competitive advantage in making business decisions” (Al-Kassab et al., 2014). One way to visualize information is to enrich an analysis model by assigning colors to different elements of the model where the basic idea originates from cartography (Coulson, 1987). An example for such analysis models is process models that describe the process logic of business processes, e.g., a patient treatment or manufacturing process. Then, for example, the service time or the frequency of an activity representing a node can be mapped to different colors (van der Aalst, 2016). In order to assign the colors, often clustering is employed as pre-processing step, i.e., the data is clustered and colors the area assigned to the resulting clusters (Jiang, 2013).

Process models can be designed by domain experts or discovered from process execution data stored in process event logs, based on process mining techniques (van der Aalst, 2016). Process mining constitutes one of the key technologies for digital transformation (Reinkemeyer, 2022). One example use case is to mine process models from an information system, containing a keyword-based search functionality. Each search term appears several times in the event log and represents a node in the finally mined process model. The resulting process model is depicted in Fig. 1 by using the process mining software DISCO² on Data Set 1 which is further described in Appendix 1.

The search term * is represented by one node in the discovered process model and is marked as the most frequent term in the process model with a frequency of 2.887. The second most frequent search term has a frequency of 90. Both frequencies 2.887 and 90 are unique for the data set. By contrast, 7 different search terms have a frequency of 3 and 6 different search terms have a frequency of 4 where the frequencies of 3 and 4 account for the lowest frequencies in the data set. For all 33 search terms, 17 different frequencies have been observed. Overall, the data set, as for many real-world applications, contains huge gaps between data points, i.e., the frequencies of the search terms. As depicted in Fig. 1, the process mining software DISCO maps the frequencies to colors of the process model nodes which range from dark blue to light gray. As stated in van der Aalst (2016), a process model can become more meaningful by mapping colors to its nodes. However, in this particular process model, the dominating search term * has such a high frequency that as a consequence, less frequent search terms appear all in the same color, i.e., light gray. This prevents differentiation between nodes such as “sport” with a frequency of 48 and “sport familie” with a frequency of 3. Therefore, not much information gain can be achieved through the usage of assigned colors.

Hence, it is of utmost importance to think about how to assign colors to the nodes in order to maximize the information gain of the visualization, even in the presence of “dominating” data points. Of particular interest is also the analysis of paths through the process model, including the happy flow of the process, which denotes the path taken by an average trace

²<https://fluxicon.com/disco/>, accessed 2022-08-08

(Leemans et al., 2014) or more “exceptional” paths with particularly high/low frequencies of their nodes.

For the visualization of analysis models, often clustering is employed as pre-processing step (Jiang, 2013). Colors are then assigned to the clusters. If, for example in the process mining case (cf. Fig. 1), 10 colors are available for assignment to the frequencies and each frequency is assigned to one of these 10 colors, each color can be considered as a cluster. In order to meet these requirements, the clustering should be performed by considering a low variance inside each cluster and a wide distance between nearby clusters and at the same time an even distribution of data over all clusters. Without the latter, especially for real-world data sets, many clusters will be sparse due to outliers, tailed or non-uniform distributions. As a consequence, results can become uniform and colors assigned to clusters do not show much additional information when, e.g., investigating a particular path. However, there is a trade-off between a low variance inside each cluster with a wide distance of nearby clusters, and an even distribution of all clusters. This trade-off is investigated in this paper. In the following, possible approaches for tackling the problem are discussed.

At first, one could argue to classify the frequencies connected to nodes with unclassified colors. Doing so for each frequency a different color is assigned by using a color gradient which has the advantage of a “raw accuracy.” Exactly these discussions between unclassified and classified colors arose already in cartography when generating choropleth maps (Tobler, 1973), which is quite similar to the visualization problem as discussed here on process models. While unclassified colors have the advantage, as mentioned before, of a “raw accuracy,” classified colors are easier to process for humans. This is due to the few number of distinct colors to recognize, which helps to reduce the cognitive load by using a legend that lists the ranges of values corresponding to each color (Dobson, 1973; 1980). A naive pre-processing approach to generate classified colors would be to slice the frequencies into m equal intervals, where m is the targeted number of clusters. This approach is also used when creating histograms, where each class interval has the same width. While being simple and transparent, the approach might result in sparse or even empty classes, if the data set is not uniformly distributed or contains huge gaps, leading to show just two colors in extreme cases.

Another approach would be to use quantiles (n/m , where n is the number of data points) as classification method where the number of data points (aka frequencies) in each class is roughly equal, which overcomes the problem of equal intervals. However, this approach has many disadvantages. While it maximizes an even distribution of data into clusters, it does not consider any clustering by minimizing the variance inside each cluster and maximizing distances between clusters which may lead to problems of interpretation of colors in models. Another problem arises when there are many identical data points in the data set, which can lead to ambiguous classes.³

A more sophisticated approach on pre-processing, which comes from cartography and is well established there, is to use the algorithm called Jenks natural breaks, which produces a classification for a pre-defined number of classes that can be mapped to colors, by minimizing the variation within each class (Jenks, 1967). A result by using this algorithm on Data Set 1 is shown in Fig. 2. Here the blue frame shows the same activities as shown in Fig. 1, but in contrast, it becomes obvious that the three nodes in the process path indicate different frequencies.

The process model depicted in Fig. 2 conveys more information than the process model depicted in Fig. 1. Nevertheless, Jenks natural breaks is not optimized for considering

³<https://geographicdata.science/book>, accessed 2022-08-08

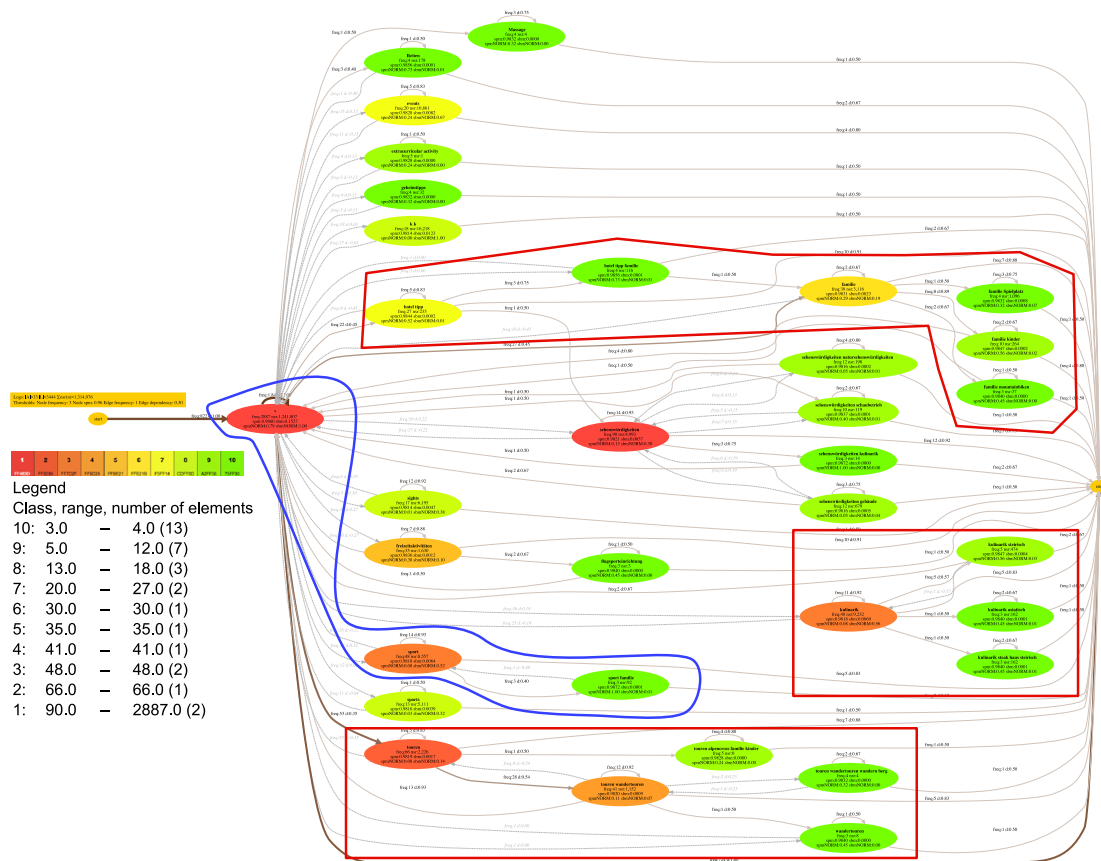


Fig. 2 Mined process model from Data Set 1 containing frequencies of activities (search terms) assigned to colors through pre-processing with Jenks natural breaks

an even distribution of the data points over all clusters, which helps to show differences between frequent elements in the process model, for example between frequencies 3 and 4. When discovering particular paths as shown in the red frames, such a distinction is useful to recognize differences in nodes and in further consequence, subpaths with splits at first glance. Also, a happy flow is hard to discover and “exceptional” paths with particular high/low frequencies of their nodes can be discovered, but not all related paths are highlighted in the process model. Another problem can be discovered from the legend in Fig. 2: Jenks natural breaks produces sparse classes (cf. classes with number 4–6, count only one data point), while putting the main outlier (i.e., the search term with a frequency 2.887) into the same class (cf. class with number 1) as the second most frequent search term (frequency 90) which both occurred just once in the whole process model.

Another use case is to divide students into equal learning groups in relation to an equal number of students and an equal level of knowledge in each group. Consider, for example, a previously executed test with 300 students, where 0–200 points were possible to reach. The test was performed at the beginning of a semester and the goal is to bring all students to a uniform level of knowledge. There are 6 teachers available to support the students during the semester, where each teacher teaches in one of 6 classes. Ideally, an equal number of students and an equal level of knowledge of students in each group is aspired. This problem

can be solved by an approach that favors an even distribution in contrast to an equal level of knowledge.

In the following, the mentioned requirements and desired characteristics of the clustering algorithm for pre-processing to be designed and realized in this work are summarized. The algorithm is supposed to

1. Handle one-dimensional data sets
2. Handle negative and floating point data points
3. Handle non-unique data points in data sets
4. Define a targeted number of clusters (which can also be less), e.g., to map them with predefined colors
5. Avoid overlapping cluster ranges
6. Avoid filtering/omitting of elements by the cluster algorithm because it already performed through pre-processing steps like outlier detection if desired for a particular use case
7. Reproduce the same clusters after each execution on the same data set, i.e., stable results of clusters
8. Work on real-world data sets, e.g., data sets containing gaps between data points
9. Produce fair results with classical clustering metrics such as low sum of variances (SV) from all clusters and a good score of mean silhouette coefficient (MSC)
10. Result in an even distribution of data points into clusters

The requirements and characteristics can be formulated into the following problem:

Problem 1 Given a set $D \subseteq \mathbb{R}$ of (one-dimensional) data points and a targeted number of clusters $M \geq 1$, where m is the number of actually built clusters, $1 \leq m \leq M$.

Find clustering $\text{Cluster}: D \mapsto \mathcal{C} = \langle C_1, \dots, C_m \rangle$ with $C_i \subseteq D$, $\bigcup_i C_i = D$ for $i = 1, \dots, m$ with $\forall C_i \in \mathcal{C}$:

- $|C_i|$ converges to $\frac{|D|}{m}$ (Requirement 1) and ideally M clusters are built
- Minimize variance of C_i (Requirement 2)
- $\forall C_j \in \mathcal{C}, C_i \neq C_j$: maximize distance between C_i and C_j (Requirement 3)
- Requirement 1 is prioritized over Requirement 2 and Requirement 3.

Problem 1 requires a given number of M clusters to be populated with data points under Requirements 1, 2, and 3. M is assumed to be determined based on the application, e.g., number of colors to be used for data visualization. 1 is measured with the metric SED (score even distribution) which is introduced in this paper. Similar to existing algorithms such as Jenks natural breaks (Jenks, 1967), Requirements 2 and 3 aim at finding low variance clusters which are separated from each other. And basically, cluster algorithms which produce low variance clusters do not intend to produce an even distribution of data points into clusters (Requirement 1), see Shapiro (2005). Unlike Jenks natural breaks, we do not consider the maximization of distances between the clusters (Requirement 3), in favor of achieving an equal distribution of the data points over the clusters (Requirement 1). This “forces” every cluster to be populated and avoiding the over-representation of frequent data or outliers at the same time. As discussed before, there might be a trade-off between Requirement 1 on the one side and Requirements 2 and 3 on the other side as filling all clusters might take

a toll on cluster variance. This work addresses Problem 1 based on the following research questions:

- RQ1 How to design a (heuristic) clustering algorithm to evenly distribute 1d data points into a maximum number of low variance clusters?
- RQ2 On which underlying data distributions and real-world data sets does the algorithm perform most effectively?
- RQ3 How does the algorithm support data visualization?

To tackle RQ1–RQ3, this paper presents the heuristic 1d distribution cluster algorithm (DDCAL) that aims at balancing Requirements 1, 2, and 3 for Problem 1. The idea of DDCAL in a nutshell is to use an iterative approach by using the feature scaling method min-max normalization or also known as rescaling, for normalizing a sorted list of one-dimensional data points and to compare the results against defined boundaries from a set list. From each boundary, outliers from the upper or lower bound are considered as new cluster if the quantity of elements of a potential cluster is inside a given tolerance factor. Otherwise, the next boundary from the list is tested or if all boundaries were tested, the tolerance factor is increased and the testing of boundaries starts again. In other words, in every iteration step, the lower or upper quantity of outliers from a boundary, which is inside a given tolerance factor, that is used to support an even distribution of elements over all clusters, is chosen for building a new cluster.

In order to evaluate the effectiveness of the DDCAL algorithms and to compare them to existing algorithms, we use four quality metrics. We first analyze the results of the DDCAL algorithm in comparison to k-means++, Jenks natural breaks, head/tail breaks, DBSCAN, kernel density estimation (KDE), Gaussian mixture model (GMM), mean shift, largest gaps, Gaussian kernel k-means, k-medoids, and trimmed k-means on five artificial data sets, reflecting a selection of common distributions such as the normal and uniform distribution. The DDCAL algorithm is then compared to the before mentioned cluster algorithms based on 4 real-world data sets, i.e., process mining, weather, star distances, and population data, with respect to quality metrics. Finally, the applicability of the DDCAL algorithm for data visualization is demonstrated for process mining, US population, and Corona pandemic data sets. The real-world data sets were chosen because of their different data structures according to different distributions and different number of data points. Based on this, they illustrate the application possibilities of the DDCAL algorithm on different use cases.

Overall, DDCAL achieves better results than existing algorithms with respect to quality metrics and data visualizations on data sets with gaps, i.e., outliers, and tails. Moreover, the DDCAL algorithm yields promising results for data which contain just one peak with a distribution that looks like a bell curve (cf. normal/gumbel distribution). The results are even good if the number of peaks is equal or higher than M . The latter is reflected, for example, by the results for data with a uniform distribution.

The paper is structured as follows: Section 2 presents the DDCAL algorithm as well as the quality metrics. Section 3 comments on the implementation of the different algorithms and evaluates DDCAL on different data sets, which are real-world and synthetic data sets. The evaluation is compared with existing algorithms. Furthermore, different parameter setups on DDCAL are evaluated to demonstrate their implications on, e.g., different distributions. Section 4 shows applications from visualization of process mining results and maps. In Section 5, we discuss related work. Section 6 concludes the paper and discusses ongoing and future work. Appendix 1 details the data sets and Appendix 2 the implementation of the algorithms.

2 1D Distribution Cluster Algorithm (DDCAL)

This section provides the DDCAL algorithm, and metrics for assessing the quality of the clustering with respect to Problem 1.

2.1 DDCAL

Given a set of 1d data points, the idea of DDCAL to tackle Problem 1 is as follows:

In the first step, the data points are sorted in ascending order. In a second step, the list of data points is copied and normalized by using the feature scaling method min-max normalization as described in Section 1. A list of boundaries to test outliers of clusters is defined. Boundaries from this list, starting with the lowest, are tested dynamically in each iteration where outliers from the upper and lower boundary are added to potential clusters. On these potential clusters, neighboring elements are added if the standard deviation inside the potential cluster decreases. If both potential clusters are above a set minimum quantity of elements — which is calculated from the combination of elements to even distribute over all remaining clusters and a set tolerance factor — the cluster with the least difference to the minimum quantity is actually built. If only one potential cluster or even no potential cluster is above the set minimum quantity of elements, the next boundary is tested and if all boundaries were tested without success of a new built cluster, the tolerance factor is increased and the set boundaries from the list, starting first with the lowest, are tested again as described before. Once a cluster is built, only the remaining empty clusters can be filled. The algorithm terminates if no empty cluster is left (i.e., of the maximum number of clusters M) or all data points are already assigned to clusters.

The pseudo code of DDCAL is provided in Algorithm 1 and explained in the following: The input data is an unsorted one-dimensional array of data points $d \in D$. The output `clustered_results` is a list containing all data points from the input where each element consists of a data point d with an assigned cluster number. The algorithm has six parameters: (1) M contains the targeted number of clusters, e.g., 10 for 10 clusters to build. (2–3) `boundary_min` and `boundary_max` contain two parameters which define the minimum and maximum boundary for calculating outliers in each iteration to assign the data points from `data` to potential clusters. The parameters contain a value, ranging from >0 to <1 , where, e.g., 0.1 stands for 10%, which means, that for a sorted normalized list of data points, outliers are the data points which are in the first 10% and in the last 10% of the list. (4) `num_simulation` defines the number of boundaries, ranging from `boundary_min` to `boundary_max` which are evenly spaced and stored in a list for finding the best boundary in each iteration step. (5) `q_tolerance` sets the quantity tolerance factor of elements for building a new cluster, which aims to produce only evenly distributed elements over all clusters within the defined factor. (6) `q_tolerance_increase_step` contains a growth factor which is responsible for increasing the `q_tolerance` factor, if all simulation steps for the list of different boundaries (cf. input parameters 2–3) did not satisfy the quantity tolerance factor.

As the first step (line #1), `data` is sorted in ascending order. In line #2, a list of `clusters` is generated from M , which contains M elements with ascending numbers, ranging from 0 to $M-1$. Next in #3, the temporary value `q_aim_lo` is calculated which determines the minimum quantity, a cluster must have to be created. The value is built by dividing the size of `data` which is not yet assigned to a cluster through the current `q_tolerance` factor. In the next step (#4), the `data` to cluster is normalized and stored as `norm_data`. For data points D , the feature scaling method min-max normalization is used with the

Input: data ▷ unsorted 1d array of data points $d \in D$
Output: clustered_results ▷ list, where each element contains d and a cluster number
Parameter 1: M ▷ targeted number of clusters (e.g., =10 for 10 clusters)
Parameter 2-3: boundary_min, boundary_max ▷ simulation boundaries (e.g., 0.1–0.5)
Parameter 4: num_simulation ▷ number of simulation steps for boundaries (e.g., =20)
Parameter 5: q_tolerance ▷ tolerance factor of element in cluster for even distribution
Parameter 6: q_tolerance_increase_step ▷ increase step of tolerance factor in iteration

```

1: sort data ascending ▷ asc, e.g., merge sort  $O(n \log n)$ 
2: create list of clusters, containing numbers from range 0 to M-1
3: calculate q_aim_lo from size of data / M - q_tolerance factor
4: get data_norm by feature scaling of data to cluster ▷ min-max normalization
5: calculate list of num_simulation evenly spaced boundaries from boundary_min to max
6: for each boundary in list of boundaries, starting with lowest boundary do
7:   fetch outlier elements of data in q-up/lo by using data_norm with current boundary
8:   add neighboring elements from data to cluster in q-up/lo if SD decreases
9:   if number of elements from q-up and q_lo are within tolerance of q_aim_lo then
10:    calculate differences between q-up/lo and q_aim_lo in diff_q_aim_up/lo
11:    continue with #20
12:   else
13:    test next boundary from list of boundaries by continuing with #6
14:   end if
15: end for
16: if diff_q_aim_up OR diff_q_aim_lo have no set difference then
17:   increase q_tolerance factor by q_tolerance_increase_step
18:   continue with #3 ▷ start over again with first boundary to test
19: end if
20: if diff_q_aim_lo ≤ diff_q_aim_up then
21:   build cluster from q_lo by using first cluster number from list of clusters
22:   remove first cluster from list of clusters
23:   remove data points of q_lo from data to cluster
24: else
25:   build cluster from q_up by using last cluster number from list of clusters
26:   remove last cluster from list of clusters
27:   remove data points of q_up from data to cluster
28: end if
29: add built cluster to clustered_results
30: if the list of clusters contains only one element which represents a cluster then
31:   add all remaining data to last remaining cluster label from list of clusters
32:   add built cluster to clustered_results
33:   terminate loop and continue with #39
34: end if
35: if no remaining data to cluster exist then
36:   terminate loop and continue with #39
37: end if
38: continue with #3
39: return clustered_results

```

Algorithm 1 DDCAL ($O(n \log n)$).

following formula (Milligan & Cooper, 1988) to calculate relative results $\text{norm}(d)$ of data point $d \in D$ depending on the respective minimum $\min(D)$ and maximum value $\max(D)$:

$$\text{norm}(d) = \frac{d - \min(D)}{\max(D) - \min(D)} \quad (1)$$

Continuing with step #5, a list containing num_simulation evenly spaces boundaries is created which comprises a sorted list of decimal numbers, ranging from boundary_min

to `boundary_max`. From #6 to #15, boundaries from the list of boundaries, which was created in #5 and starting with the lowest boundary (i.e., first element in the list), are treated. On step #7, all outlier elements from data are determined by using the current boundary from the list of boundaries and comparing it with `norm_data` which contains the same data points on the same position as `data`. The lower outliers are stored in `q_lo` and the upper outliers are stored in `q_up`. Step #8 adds neighboring elements from `q_lo/up` from not yet added data, as long as the standard deviation inside `q_lo/up` decreases. From line #9 to #14, there is a check with handling defined, if the number of elements from `q_up` and `q_lo` is above `q_aim_lo`. If yes, the differences from the number of elements from `q_up/lo` and `q_aim_lo` are calculated and stored in `diff_q_aim_up/lo` and the algorithm continues with line #19. If no, a next current boundary will be tested by continuing with line #6. The lines #16–#19 are only reached if no boundary was found (#6–#15) which produced both, `diff_q_aim_lo/up`. Thus, `diff_q_aim_up` or `diff_q_aim_lo` is not set and the `q_tolerance` factor is increased by the factor of `q_tolerance_increase_step` (#17). The algorithm continues with line #3 by calculating the `q_aim_lo` again for all boundaries to test. In steps #20–#28, a cluster is built from the outliers `q_lo` or `q_up`. `q_lo` is chosen, if the associated `diff_q_aim_lo` is smaller or equal `diff_q_aim_up`. Then, on line #21, a cluster is built from the data points of `q_lo` where each data point is assigned with the first element from the list of clusters. Next, the first element is removed from the list of clusters (#22) and the data points from `q_lo` are removed from data (#23). Otherwise, if the `diff_q_aim_up` is smaller than `diff_q_aim_lo`, a cluster is built from the data points of `q_up` where each data point is assigned with the last element from the list of clusters (#25). Then, the last element is removed from the list of clusters (#26) and the data points from `q_up` are removed as well from data (#27). In both cases, the build cluster is added to the list of `clustered_results` (#29). Line #30–#34 handles if the list of clusters contains only one element. Then, a cluster is built with all remaining data points from data which are assigned to the element from the list of clusters (#31). Next on steps #32–#33, the cluster is added to the list of `clustered_results` and the algorithm continues with step #39. If there exist no remaining data to cluster, the algorithm continues as well with step #39 (#35–#37). Otherwise, after a new cluster was built and added to the list of `clustered_results` and there exists data to cluster and as well a list of clusters containing more than one element, the algorithm starts the next iteration for building a cluster and continues with step #3 (#38). Finally, on #39, the built list of `clustered_results` is returned as output of the algorithm.

DDCAL produces stable results, i.e., the output of the algorithm remains the same after each execution if the data set and the parameters do not change. Furthermore, the algorithm does not contain nested loops of the input list data which results basically in $O(n)$ with $n = |D|$ in terms of runtime or space requirements if the number of clusters is considered as constant and the sorting step (#1) is left out, where all general sorting functions are, at best, $O(n \log n)$.

The algorithm uses the feature scaling method min-max normalization with boundaries in an iterative approach. The concept is inspired from statistical tests based on data distributions and their significance threshold which is in the case of this algorithm defined through a boundary. DDCAL compares in every iteration the whole remaining data set for building a new cluster. The term remaining means in this case that data points are assigned to a cluster, which are typically outliers, and then these data points are removed from the data set of data to cluster. By addressing Problem 1 (cf. Section 1), DDCAL aims to increase the

number of elements in each cluster which is reflected in evenly distributed elements over all clusters (cf. Requirement 1). The even distribution can be influenced with the parameter `q_tolerance`, which sets the minimum amount of elements for building a cluster on a given range of boundaries to test. And if these boundaries do not produce the minimum amount of elements as outliers, the parameter `q_tolerance.increase_step` increases the `q_tolerance` factor and thus, influences the even distribution of the results as well.

By using this iterative approach on the remaining data for building clusters with outliers, it becomes more likely to be able to add more data points to clusters in subsequent iteration steps on lower boundaries, because they contain less outliers. To add too many data points into a cluster is hampered through the logic, that on each iteration, clusters are built starting with the lowest boundary to test and a cluster can actually be built, if the minimum amount of elements on both sides, the upper and lower bound, are reached or exceeded where the side with the fewer data points is used for building a cluster. This benefits of course Requirement 1, but through the iterative approach of removing outliers in every iteration, the variance of the whole remaining data set decreases. Therefore, clusters with outliers have lower variances in their clusters in later iterations which benefits Requirement 2 and also indirectly Requirement 3.

2.2 Clustering Quality Metrics

We use four quality metrics, i.e., number of used clusters (NUC), sum even distribution (SED), sum variances (SV), and mean silhouette coefficient (MSC) that capture the coverage of a clustering result with respect to Problem 1.

How evenly the data is distributed over built clusters is measured with the metric SED. The aim of DDCAL, which is described in detail in Algorithm 1, is to produce clusters with high SED values (Requirement 1), as top priority in addressing Problem 1, but also to consider Requirements 2 and 3 which are measured through the quality metrics SV and MSC. Section 3.4 employs the four quality metrics in order to compare DDCAL with other clustering algorithms.

Note, while DDCAL produces high SED values, we performed different simulations of parameters with existing algorithms if available and picked the results with the highest SED values in order to foster a fair comparison.

Definition 1 (Quality Metrics) Consider Problem 1 with $D \subseteq \mathbb{R}$ being a set of (one-dimensional) data points. Then, quality metrics NUC, SED, SV, and MSC measure the quality of a clustering $C = \langle C_1, \dots, C_m \rangle$ ($m \leq M$), where M denotes the targeted number of clusters. With respect to Requirements 1 and 2, the following quality metrics are defined:

- $\text{NUC} := \frac{m}{M}$
- $\text{SED} := \prod_{i=1}^m |C_i|$
- $\text{SV} := \sum_{i=1}^m \sum_{j=1}^{|C_i|} \|x_{ij} - z_i\|$ where m is the number of the actually built clusters, x_{ij} denotes the j th data point in cluster C_i , and z_i the mean value of cluster C_i (see (Faber, 1994)).
- $\text{MSC} := \frac{1}{|D|} * \sum_{d \in D} s(d)$ where $s(d)$ denotes the silhouette coefficient for data point $d \in D$ with $s(d) = \frac{b(d) - a(d)}{\max\{a(d), b(d)\}}$ where $a(d)$ is the average dissimilarity of d to all other objects of C_i , $b(d)$ is the minimum $\text{dis}(d, C_j)$ with $i \neq j$, and $\text{dis}(d, C_j)$ is the average dissimilarity of d to all objects of C_j (see (Rousseeuw, 1987)).

$NUC \in [0; 1]$ assesses the number of used clusters, in particular the implicit requirement to fill M clusters. $NUC=1$ denotes the best result and $NUC=0$ the worst. When comparing the different algorithms as described in Section 3, we compare just results with $NUC=1$ which indicates that all targeted clusters were actually built and a fair comparison with the other clustering quality metrics is therefore valid. SED, the score of even distribution, evaluates how evenly the data is distributed over the clusters (\mapsto Requirement 1); the product of the cluster cardinalities is maximized for clusters of equal size and if $NUC=1$. It is minimized for clusters, where every cluster contains just one element with the exception of one cluster which contains all remaining elements of a data set. Higher results are better, where the best result appears, if all clusters contain the same number of elements and the worst result will be if all clusters contain just one element with the exception of one cluster which contains the remaining elements. SV, the sum of variances, measures the homogeneity of the clusters, which considers intra-cluster distances. Thus, it assesses low variance clusters which are compact (\mapsto Requirement 2) and is defined according to literature (e.g., (Faber, 1994)). A low value is therefore better than a higher one. MSC, the mean silhouette coefficient, considers compact (cf. SV) and clearly separated clusters (\mapsto Requirements 2 and 3). The possible results range from -1 to 1 where 1 signals the optimum and thus clusters are built well apart from each other and are clearly distinguished. 0 shows that clusters are indifferent and -1 is the worst result, where clusters are wrong assigned (e.g., many overlapping cluster ranges).

3 Assessment Based on Quality Metrics

3.1 Implementation of Algorithms

In order to evaluate the quality of Algorithm 1 (DDCAL), we compare it to a set of existing algorithms such as kmeans++ based on the quality metrics defined in Section 2.2. Furthermore, the parameters of DDCAL are evaluated.

In Section 3.2, the comparison is based on artificial data sets with different distributions and on these distributions, the evaluation of the DDCAL parameters is performed in Section 3.3. Finally, Section 3.4 compares the algorithms on real-world data sets. For this, we implemented Algorithm 1.⁴ as well as a selection of existing algorithms for comparison.

We used Python 3.9 for implementing all algorithms. For data operations, the numpy framework (version 1.21.2) was used. Because every algorithm produces different outputs, like clustered data points, centroids, breaks, or extrema on a curve, we converted each output to a uniform list of resulting elements, where each element contains a cluster number and a data point. Through these conversion steps, the performance of an algorithm may differ and a comparison of runtime should be considered with caution. Nevertheless, long runtimes when comparing different algorithms on huge data sets are pointed out in this work. Further details on the implementation of all algorithms can be found in Appendix 2.

3.2 Evaluation Based on Artificial Data with Different Distributions

We first want to understand how the results produced by Algorithm 1 relate to the distribution of the underlying data, also in comparison with k-means++, Jenks natural

⁴<https://github.com/luxmar/DDCAL>

breaks, head/tail breaks, DBSCAN, KDE, GMM, mean shift, largest gaps, Gaussian kernel k-means, k-medoids, and trimmed k-means.

We generate artificial data sets of 1000 data points for different distributions. One thousand data points seem to result in meaningful data distributions to be distinguished from each other. Each data set contains at least one-third non-unique data points (cf. Section 1) to represent roughly a real-world data set. Moreover, when clustering the data sets by using $M = 10$ targeted clusters, each cluster contains about 100 data points if the results are evenly distributed over the clusters. Therefore, built clusters are small enough to be still be observed manually but are big enough to represent meaningful results. For representing different distributions, we have chosen well-known statistical distributions, to cover a broad range of possible data sets and to show how each algorithm, especially DDCAL, performs differently on different distributions. The following data sets are also accessible with details at⁵ normal distribution with 369 unique data points, gumbel distribution with 393 unique data points, uniform distribution with 558 unique data points, exponential distribution with 587 unique data points, and two peaks distribution with 362 unique data points which consists for two randomly generated normal distributions where the first has 300 and the second has 700 data points (cf. Table 1).

We compare existing clustering algorithms with DDCAL by using $M = 10$ on the artificial data sets with different distributions. The results are shown in Tables 1, 2 and 3 by comparing based on metrics SED, SV, and MSC (cf. Section 2.2, and more details of the results can be found here.)⁶ The parameters set for the different algorithms for distributions to produce these results are shown in Table 4. The default parameters, which are not changed regarding a particular distribution, except for DDCAL which is described in the following, are shown in Appendix 2, for each particular algorithm. Also for stochastic algorithms, where the results depend on trial because of random methods in the particular algorithm, the variances from 10 trials are shown. The results show one execution. k-medoids and Gaussian kernel k-means result in the highest variances and thus the worst results. Furthermore, for k-medoids, metric SED showed the highest variance on all distributions, with exception of the exponential distribution, where Gaussian kernel k-means has the highest variance. The trimmed k-means algorithms lead to the lowest variances, with the exception of metric SV on uniform distribution with trimmed k-means O+. Thus, the trimmed k-means algorithms perform best and also k-means++ shows low variances over all data sets.

For DDCAL (cf. Algorithm 1), we use the following input parameter values, i.e., `boundary_min=0.1`, `boundary_max=0.49`, `num_simulation=20`, and `q_tolerance_increase_step=0.5`. The parameter `q_tolerance` is set based on the distribution: for normal, gumbel, and two peaks distribution `q_tolerance=0.45` and for uniform and exponential with `q_tolerance=0.1`. The setting of the input parameters is discussed in Section 3.3.






For other algorithms which need input parameters, we implemented simulation methods which aim to maximize the result of the metric SED if the precondition of $NUC=1$ is fulfilled. Further details on each algorithm are described in Appendix 2.

The algorithm containing the best result is highlighted by a bold value in each table and on each distribution. We omit results and consequently algorithms which produce less than 10 clusters (i.e., $NUC < 1$) due to a fair comparison of the different algorithms. Also,

⁵<https://github.com/luxmar/DDCAL/tree/main/tests/data>

⁶<https://github.com/luxmar/DDCAL/tree/main/supplemental>

Table 1 Algorithm results for SED if NUC=1 from different artificially generated distributions

| Algorithm | Normal  | Gumbel  | Uniform  | Exponential  | Two Peaks  |
|-----------------------|---|---|--|--|--|
| DDCAL | 6.69e+19 | 7.08e+19 | 9.93e+19 | 5.29e+19 | 3.97e+19 |
| k-means++ | 8.50e+18 | 1.16e+19 | 9.47e+19 | 1.29e+18 | 1.61e+19 |
| variance of 10 trials | 2.29e+36 | 1.83e+37 | 2.79e+36 | 1.49e+34 | 3.90e+36 |
| Jenks natural breaks | 6.24e+18 | 2.56e+18 | 9.37e+19 | 1.63e+18 | 1.84e+19 |
| head/tail breaks | — | — | — | — | — |
| DBSCAN | 7.08e+04 | 2.37e+04 | — | — | 3.86e+08 |
| KDE | 1.48e+14 | 7.89e+11 | 5.73e+19 | 4.96e+10 | 3.23e+14 |
| GMM | 8.33e+18 | 9.37e+17 | 7.43e+19 | 1.30e+18 | 1.78e+19 |
| variance of 10 trials | 1.42e+37 | 8.86e+36 | 3.04e+37 | 3.90e+34 | 4.94e+37 |
| mean shift | — | — | — | 5.64e+14 | — |
| largest gaps | 2.35e+05 | 7.90e+03 | 3.25e+06 | 3.51e+05 | 4.74e+08 |
| Gauss. kernel k-means | 2.69e+19 | 1.45e+19 | 8.83e+19 | 4.35e+19 | 5.87e+19 |
| variance of 10 trials | 1.25e+37 | 5.72e+35 | 4.37e+35 | 7.70e+38 | 1.36e+38 |
| k-medoids | 1.43e+19 | 1.99e+18 | 8.31e+19 | 1.61e+18 | 2.92e+19 |
| variance of 10 trials | 2.00e+38 | 6.73e+37 | 8.89e+36 | 1.05e+36 | 1.12e+38 |
| trimmed k-means O- | 2.69e+19 | 2.06e+19 | 3.36e+19 | 1.37e+19 | 1.90e+19 |
| variance of 10 trials | 8.16e+34 | 1.19e+35 | 1.05e+35 | 9.06e+34 | 4.01e+34 |
| trimmed k-means O+ | 7.35e+19 | 6.42e+19 | 9.73e+19 | 2.97e+19 | 6.30e+19 |
| variance of 10 trials | 4.14e+35 | 1.29e+36 | 3.55e+34 | 5.31e+35 | 2.21e+35 |

Number of elements for each distribution =1000 and $M = 10$ where max SED =1.00e+20 (except for trimmed k-means O- (where O- means without outliers), which means that a defined percentage (=10%) of elements to cluster were filtered because they were considered as outliers).

Bold values indicate the best results if NUC=1.0

results from trimmed k-means are excluded in the evaluation of the results because the algorithm filters outliers. The parameter of the trim factor is set to 0.1 which means that for 10% of the data to cluster, a separate cluster is built which contains identified outliers of the data set. This behavior violates 2 of 10 requirements set out in Section 1: (6) to avoid filtering/omitting of elements by the cluster algorithm and (5) to avoid overlapping cluster ranges. The latter is violated, because outliers are thrown in a separate cluster which contains likely elements from all areas of the data set. For this reason, bold values from trimmed k-means are highlighted in color blue if the result is best among the compared algorithms. We executed trimmed k-means twice on each data set, where the postfix O- indicates that only results without outliers are considered for metrics SED, SV, and MSC in order to avoid results with overlapping clusters as described before. Therefore, if 10% of the data points in the data set are trimmed, only the remaining 90% of the data points with their built clusters are evaluated. Hence, when $M = 10$ clusters are targeted, as described before, the actual targeted number on the algorithm is set to $M+1$ (i.e., 11). From the actually built

Table 2 Algorithm results for SV, if NUC =1, from different artificially generated distributions

| Algorithm | Normal | Gumbel | Uniform | Exponential | Two peaks |
|-------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| DDCAL | 3.87e-01 | 7.88e-01 | 5.51e-01 | 1.53e+01 | 3.69e-01 |
| k-means++ | 5.25e-01 | 8.12e-01 | 5.12e-01 | 7.75e+00 | 2.49e-01 |
| Variance of 10 trials | 3.04e-05 | 5.77e-04 | 2.11e-05 | 2.90e-01 | 3.36e-06 |
| Jenks natural breaks | 5.22e-01 | 8.11e-01 | 5.07e-01 | 9.57e+00 | 6.68e-01 |
| Head/tail breaks | — | — | — | — | — |
| DBSCAN | 8.59e-01 | 1.17e+00 | — | — | 4.54e-01 |
| KDE | 3.15e-01 | 5.17e-01 | 5.40e-01 | 5.21e+00 | 2.75e-01 |
| GMM | 5.26e-01 | 7.32e-01 | 5.46e-01 | 6.88e+00 | 2.50e-01 |
| Variance of 10 trials | 5.68e-03 | 6.23e-04 | 1.98e-04 | 7.15e-01 | 2.38e-05 |
| Mean shift | — | — | — | 4.57e+00 | — |
| Largest gaps | 1.21e+00 | 1.42e+00 | 3.39e+00 | 1.03e+01 | 4.68e-01 |
| Gaussian kernel k-means | 4.66e-01 | 8.13e-01 | 5.35e-01 | 1.47e+01 | 2.84e-01 |
| Variance of 10 trials | 9.98e-05 | 8.31e-08 | 2.32e-07 | 5.23e+01 | 1.99e+00 |
| k-medoids | 5.32e-01 | 8.02e-01 | 5.23e-01 | 7.85e+00 | 2.55e-01 |
| Variance of 10 trials | 6.07e-03 | 9.23e-04 | 9.99e-05 | 4.04e+00 | 1.46e-05 |
| Trimmed k-means O- | 7.74e-02 | 9.77e-02 | 3.78e-01 | 8.07e-02 | 6.26e-01 |
| Variance of 10 trials | 2.44e-08 | 4.89e-08 | 9.85e-07 | 7.12e-06 | 4.06e-08 |
| Trimmed k-means O+ | 4.22e+00 | 4.89e+00 | 9.33e+00 | 1.72e+01 | 4.14e+00 |
| Variance of 10 trials | 6.66e-05 | 6.06e-08 | 1.02e-02 | 4.64e-06 | 1.11e-04 |

Number of elements for each distribution =1000 and $M = 10$.
 Bold values indicate the best results if NUC=1.0

clusters, only $m-1$ clusters, without the 1 outlier cluster are considered with the evaluation metrics. With this approach, the trimming of outlier data to cluster is performed, but no overlapping cluster ranges are produced. In contrast, if the postfix O+ is shown for trimmed k-means, also the cluster which contained the filtered outliers is considered based on metrics SED, SV, and MSC. However, using this approach, the appearance of overlapping cluster ranges is likely and thus it might not be useful for certain use cases (cf. Data Set 1 in Appendix 1).

In Table 1, the SED results are shown where the highest values represent the best ones. DDCAL performs best on the normal, gumbel, uniform, and exponential distribution and is slightly behind Gaussian kernel k-means (average rank of 2.2) on the two peaks distribution, resulting in an average rank of 1.2. trimmed k-means O+ performs also good on all distributions and best on normal and two peaks distribution. However, as mentioned before, overlapping clusters are produced on all distributions due to the outlier cluster, which contained 10% of the whole data set to cluster.

Table 2 shows the result for SV, where the lowest results indicate the best performance of an algorithm. DDCAL performs below average, except on the normal (rank 2) and gumbel (rank 3) distribution. In comparison, when calculating the average rank of all distributions, KDE performed best with 2.4, followed by k-means++ with 3.4 and GMM with 3.6. DDCAL had an average of 5.2 and was ranked as “6th” best algorithm, out of 8.

Table 3 Algorithm results for MSC if NUC=1, from different artificially generated distributions

| Algorithm | Normal | Gumbel | Exponential | Two peaks | |
|-------------------------|-------------|-------------|-------------|-------------|-------------|
| DDCAL | 0.5 | 0.52 | 0.53 | 0.51 | 0.5 |
| k-means++ | 0.53 | 0.53 | 0.56 | 0.57 | 0.55 |
| Variance of 10 trials | 5.05e-07 | 2.45e-06 | 1.36e-05 | 1.18e-06 | 1.20e-06 |
| Jenks natural breaks | 0.53 | 0.53 | 0.56 | 0.57 | 0.55 |
| Head/tail breaks | — | — | — | — | — |
| DBSCAN | 0.25 | 0.28 | — | — | 0.35 |
| KDE | 0.44 | 0.46 | 0.53 | 0.56 | 0.43 |
| GMM | 0.53 | 0.53 | 0.52 | 0.57 | 0.55 |
| Variance of 10 trials | 1.93e-05 | 7.06e-06 | 1.13e-04 | 1.01e-04 | 4.21e-05 |
| Mean shift | — | — | — | 0.54 | — |
| Largest gaps | 0.26 | 0.56 | 0.17 | 0.62 | 0.36 |
| Gaussian kernel k-means | 0.52 | 0.53 | 0.54 | 0.49 | 0.52 |
| Variance of 10 trials | 1.48e-05 | 3.98e-06 | 1.32e-08 | 2.62e-03 | 3.76e-04 |
| k-medoids | 0.52 | 0.52 | 0.55 | 0.57 | 0.52 |
| Variance of 10 trials | 2.39e-05 | 6.79e-05 | 3.53e-05 | 4.97e-05 | 1.36e-04 |
| Trimmed k-means O– | 0.55 | 0.56 | 0.61 | 0.56 | 0.57 |
| Variance of 10 trials | 1.09e-07 | 1.39e-08 | 7.87e-07 | 1.11e-06 | 1.99e-08 |
| Trimmed k-means O+ | 0.42 | 0.44 | 0.46 | 0.51 | 0.44 |
| Variance of 10 trials | 1.84e-07 | 2.12e-07 | 2.42e-08 | 2.50e-08 | 2.42e-08 |

Number of elements for each distribution =1000 and $M = 10$.

Bold values indicate the best results if NUC=1.0

Table 3 shows the results of metric MSC, where the highest value shows the best result. k-means++, Jenks natural breaks, and GMM perform best on all distributions. These algorithms achieve always the first or second place, with the exception of GMM on the uniform distribution with only rank 7. Note that equal results have the same rank. DDCAL ranks 5 on uniform distribution, 6 on normal, gumbel, and two peaks distribution and 7 on the exponential distribution. By calculating the average rank of all distributions, k-means++ and Jenks natural breaks share the first place as best algorithms with 1.4, followed by GMM with 2.6 and k-medoids with 4. DDCAL scores below the average with an average rank of 6. Only KDE achieves a lower result with 6.6. MSC ranges from -1 to $+1$ and in most cases, except for the exponential distribution, the results for DDCAL differ from the results of the other algorithms on the second decimal place. Hence, we can still say that DDCAL performs close to the best performing algorithms.

When putting the ranks of all metrics on each algorithm together, by calculating the average rank on each distribution, DDCAL and Gaussian kernel k-means perform best on the normal distribution with an average order value of 3.0 (e.g., for DDCAL: $\frac{1+2+6}{3}$) from SED, SV, and MSC. On the gumbel distribution, DDCAL and GMM perform best with 3.3. For the other distributions, DDCAL performs worse than on the normal and gumbel distribution, where, for example, on the uniform distribution, k-means++ and Jenks natural breaks have the best average rank with 1.7 and DDCAL is on “5th” place. When looking at the exponential distribution, GMM scores best with 3.0 and DDCAL performs poorly with the “7th” place and an average rank of 5.3. Finally, on the two peaks distribution, k-means++ and GMM perform best with 2.7. DDCAL ranks 6 with an average score of 4.7.

Table 4 Algorithm parameters which were set for the different distributions as shown in Tables 1, 2, and 3

| Algorithm | Normal | Gumbel | Uniform | Exponential | Two peaks |
|-------------------------------|------------|------------|------------|-------------|------------|
| DDCAL q_tolerance= | 0.45 | 0.45 | 0.1 | 0.1 | 0.45 |
| DBSCAN min_pts= eps= | 1, 0.09 | 1, 0.11 | 1, 0.04 | 1, 0.47 | 1, 0.07 |
| KDE h= | 0.07 | 0.08 | 0.14 | 0.30 | 0.02 |
| Mean shift q= | 0.002 | 0.002 | 0.002 | 0.214 | 0.005 |
| Gauss. kernel k-means var= | 13.39 | 15.81 | 26.38 | 0.40 | 0.31 |
| Trimmed k-means O-/+ trim= | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

Additionally to the ranking techniques as shown above and to get an even better picture of the effectiveness of the algorithms, we consider the differences between the scores for metrics SED, SV, and MSC by each algorithm. First, we normalize the results for each of the three metrics for each distribution by using the feature scaling method min-max normalization with the equation from Section 2 to obtain scores, ranging between 0 and 1, where 1 shows the best result, e.g., on SED for normal distribution DDCAL=1.0 (indicates the best rank), KDE =0.59 (indicates that the result is about less the half than the best ranking result), and largest gaps =0 (indicates worst rank). On SV, we have to do an extra step after normalization, because the best result has the lowest score. Thus, we subtract each normalized result from 1, e.g., for KDE: $1 - 0 = 1$. For metrics SED on the normal distribution, we discover that, for example, DDCAL has a normalized result of 1 and the second best is Gaussian kernel k-means with only 0.40. Nearly the same picture is shown when we look at the gumbel distribution, where the DDCAL has a normalized result of 1 and the second is Gaussian kernel k-means with 0.20. With the exception of the two peaks distribution, where DDCAL ranks second best 0.68 behind Gaussian kernel k-means, DDCAL performs best on all tested distributions. When DDCAL is compared based on metric SV, the results are about average, i.e., 0.91 on normal distribution, 0.69 on gumbel distribution, 0.98 on uniform distribution, and 0.71 on two peaks distribution, except for the exponential distribution with a result of 0. The normalized results for MSC show nearly the same picture as shown for the previous metric SV: 0.89 on normal distribution, 0.6 on gumbel distribution, 0.92 on uniform distribution, and 0.74 on two peaks distribution. Similarly to SV, DDCAL shows an below-average performance with 0.15 for MSC with the exponential distribution. By averaging the normalized results from all metrics, where each metric is considered with equal weight, on each distribution, DDCAL performs best on normal distribution with $0.94 \left(\frac{1+0.92+0.89}{3} \right)$, followed by Gaussian kernel k-means with 0.73. On gumbel distribution, DDCAL performs best with an average of 0.77, followed by Gaussian kernel k-means with 0.53 which was close to k-means++ with 0.51. For the uniform distribution, the first three algorithms lead to similar results, i.e., 0.984 for k-means++, 0.981 for Jenks natural breaks, and 0.97 for DDCAL. On the exponential distribution, KDE performs best with an

average of 0.51 and DDCAL ranks 7 with a score of 0.38. Finally, on the two peaks distribution, Gaussian kernel k-means has the highest average score with 0.92 and DDCAL ranks 4th place with a score of 0.71. Based on the sum of the normalized results over all distributions from all metrics the algorithms can be ranked as follows: DDCAL with 3.77 ($0.94+0.77+0.97+0.38+0.71$), Gaussian kernel k-means with 3.41, k-means++ with 3.35, GMM with 3.26, k-medoids with 3.25, Jenks natural breaks with 2.91, KDE with 2.67, and largest gaps with 0.99.

Through the analysis of the different metrics, it seems that DDCAL performs well on normal and gumbel distributions. By considering the mechanics of DDCAL, outliers above or below a threshold are the first generated clusters. After every iteration, these outliers are removed continuously from the input list and assigned to clusters. Through this procedure, the input list becomes more evenly distributed. DDCAL builds only a cluster in an iteration if on both sides, above and below a threshold, outliers are identified as potential clusters, containing a minimum amount of data points. Hence, the algorithm performs better, if just one peak with two tails exists in the distribution of a given data set, and also because outliers are considered as local in DDCAL instead of global. DDCAL can be improved on data sets with two or more peaks by a pre-processing step which cuts the data set into two or more slices where every data set contains one distribution.

3.3 Evaluation of DDCAL Parameters on Different Distributions

In the previous sections, we proposed particular values for parameters on DDCAL which we also used for evaluation. These default parameters (`boundary_min=0.1`, `boundary_max=0.49`, `num_simulation=20`, `q_tolerance_increase_step=0.5`, `q_tolerance=0.45` resp. 0.1) are chosen because of the results from simulations on different distributions, as shown in Figs. 3, 4, and 5 for the DDCAL algorithm (more details of the results can be found here.)⁷ We left out the gumbel distribution because the impact on the results, by setting different input parameters, is nearly the same as on the normal distribution.

At first, we evaluate different settings for parameter `q_tolerance`, which defines the minimum quantity of elements for building a cluster. As shown in Fig. 3, we tested different values between 0.005 and 0.5 with the observation that DDCAL performs well on normal (also gumbel) and two peaks distribution, when the parameter is set to 0.45. On the uniform and exponential distribution, 0.1 performs well. On the two peaks and exponential distribution, there exist many jumps in the results of the metrics. Therefore, a high `q_tolerance` factor of, for example, 0.4 on the exponential distribution can perform well, resulting in a lower SED value and the best value for MSC.

Overall, we recommend for real-world data sets, which are similar to the tested distributions, to set `q_tolerance` to a higher value like 0.45 on normal and gumbel distribution and to a lower value, like 0.1 on a uniform distribution. On exponential and two peaks distribution, we recommend to test both high and low `q_tolerance` factors, such as 0.45 and 0.1.

The `q_tolerance` has huge impact on the results on different distributions and can be used to adjust the trade-off between the results of SED and the classical clustering metrics SV and MSC in small steps, e.g., by setting the parameter from 0.45 to 0.5 on the normal

⁷<https://github.com/luxmar/DDCAL/tree/main/supplemental>



Fig. 3 Visualization of metrics SED, SV, and MSC for DDCAL on different distributions with different $q_tolerance$ parameters and $M = 10$

distribution, where a higher $q_tolerance$ factor decreases SED (which is still high) but produces better results for SV (which gets lower) and MSC (which increases).

For further tests on different parameters, we use the best performing $q_tolerance$ setting on each distribution as discovered before. Note, we also performed tests on the uniform and exponential distribution by setting $q_tolerance$ to 0.45, even though our observation showed that a lower $q_tolerance$ factor seems to perform better. In most cases, except for testing the parameter `boundary_min` on exponential distribution, similar curves in the diagram are produced and the results on all three metrics decrease.

For DDCAL, we consider boundaries up to 0.49 as meaningful with respect to achieving fair results and because clustering outliers above 49%, these results cannot be considered as “outliers,” as they make up more than half of the remaining data points on each iteration step. The results depicted in Fig. 4, for which different `boundary_max` parameters (ranging from 0.15 to 0.9) are tested, show that the overall performance is good with a maximum boundary of 0.49. Thus, we recommend to use the parameter `boundary_max=0.49` on all distributions. The exponential distribution with a higher value performs also good and even better on SED.

When investigating the parameter `boundary_min`, where the results are shown in Fig. 5, it turns out that to start with a 10% boundary (minimum boundary) for clustering



Fig. 4 Visualization of metrics SED, SV, and MSC for DDCAL on different distributions with different boundary_max parameters and $M = 10$

outliers, seems to be a good balance between a high SED value and good results on SV, and MSC. Starting with 15%, leads to DDCAL performing even better in some cases. For example, on the exponential distribution boundary_min=0.15 yields better results than =0.1 on SV and MSC on slight costs of SED, but on normal distribution this observation is reversed, where SED is slightly higher on costs of SV and MSC. Therefore, we recommend to use boundary_min=0.1 on all distributions and to test boundary_min=0.15 for fine tuning.

With the parameter num_simulations, as described in Algorithm 1, we defined the number of simulation steps for testing different boundaries on each iteration in DDCAL. These boundaries are evenly spaced values from the defined input parameters boundary_min to boundary_max with the defined number of simulation steps, where we consider that 20 elements are sufficient to test all necessary boundaries.

This number is confirmed through testing different numbers of simulation steps (above and below) which show that less than 20 simulation steps decrease the results from the metrics and more than 20 simulation steps do not show significant improvements, but increase the runtime of the algorithm.

Also, the parameter q_tolerance_increase_step is tested with different values. A value of 0.5 seems to perform best, but slight changes have hardly any impact on the results. When setting q_tolerance_increase_step too low, more iteration steps are performed, which increases the runtime of the algorithm. A too high value such as 3 decreases

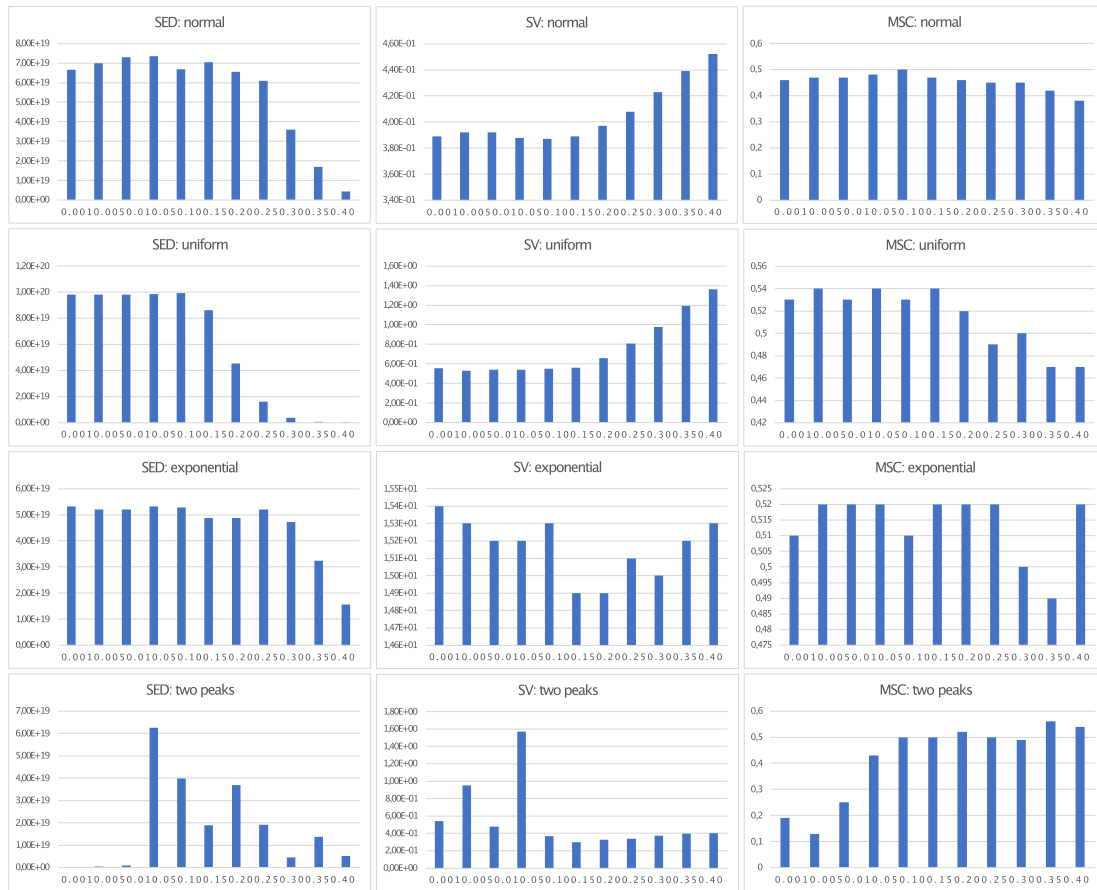


Fig. 5 Visualization of metrics SED, SV, and MSC or DDCAL Advanced on different distributions with different boundary_min parameters and $M = 10$

the performance of the metrics, especially for SED, because clusters are built too early when the minimum quantity for building a cluster is reached immediately.

We tested DDCAL by changing more than one parameter at the same time, but the results from the metrics always decrease. Thus, we recommend to change only one parameter value and to use the recommendations as described before. In detail, parameter `q.tolerance` can be used as pivot for better results. For slight improvements, the parameter `boundary_min` can be changed, as well. An exception, where the `boundary_min` changes the results completely, is shown on the two peaks distribution by setting `boundary_min` to the value 0.35 instead of 0.1, which shows, when comparing DDCAL with 7 other algorithms (cf. Tables 1, 2, and 3) the following results: the performance of SED declines from ranks 2 to 6. The rank of SV remains 6 and MSC improves from 6 to the best performing algorithm (rank 1) with an MSC value of 0.56.

Discussion: Throughout the paper, we assume $M = 10$, but other M values were tested on the data sets described in Section 3.2. On data set normal distribution, for example, we compared DDCAL with Jenks natural breaks on different values of M (more details of the results can be found here.)⁸ We have chosen Jenks natural breaks for comparison, because it was the best performing algorithm in Section 3.2 (cf. Tables 1, 2, and 3) with stable results on $M = 10$. Jenks natural breaks ranks third behind DDCAL and k-means++ when

⁸<https://github.com/luxmar/DDCAL/tree/main/supplemental>

calculating the average rank for each distribution (cf. Section 3.2) and ranks sixth behind DDCAL and the stochastic algorithms Gaussian kernel k-means, k-means++, GMM, and k-medoids when calculating the normalized results.

For DDCAL, we used the default parameters (`boundary_min=0.1`, `boundary_max=0.49`, `num_simulation=20`, `q_tolerance_increase_step=0.5`, `q_tolerance=0.45`) as stated above. First we observe that for $M \in \{3, 4\}$, Jenks natural breaks shows better results for SED, SV, and MSC. For $M \in \{5, 6, 8, 12, 15, 30\}$, SED and SV perform better for DDCAL, but MSC turns out slightly worse than Jenks natural breaks, which shows the same observation as with $M = 10$ (cf. Tables 1, 2, and 3). As shown before, in some cases, MSC can be improved without costs of other parameter performance, when setting the parameter `boundary_min` to `=0.15` instead of `0.1`. For $M = 12$, for example, SED improves from $7.78e+22$ to $9.01e+22$, SV remains the same value of $3.69e-01$, and MSC improves from 0.48 to 0.5 where Jenks natural breaks has values of $SED=8.11e+21$, $SV=4.90e-01$ and $MSC=0.54$. The same observation of the improvement of DDCAL by changing this parameter is discovered for $M = 6$, where all metrics improve. By changing the parameter for other values of M , the performance of the metrics decreases.

When setting M to a high value of, e.g., 100 (with just 1000 data points in the data set), Jenks natural breaks produces 100 clusters where DDCAL produced only 44 clusters. The lower number of produced clusters in the output results from the termination of the algorithm, if no data is left for clustering (cf. Algorithm 1 step #36) and hence it makes no sense to produce more clusters. Because of the fewer clusters, the results of DDCAL rank behind Jenks natural breaks for metrics SED ($1.51e+52$ vs. $3.50e+90$) and SV ($3.97e-01$ vs. $3.86e-01$) and better for metric MSC (0.53 vs. 0.5). However, a fair comparison can only be made if both algorithms produce the same number of clusters. When setting the parameter `boundary_min` to `0.01` instead of `0.1`, DDCAL produces 100 clusters, where all metrics perform better than Jenks natural breaks ($SED=1.17e+97$, $SV=3.40e-01$, and $MSC=0.54$). By lowering `boundary_min`, it is more likely that the envisaged number of data points per cluster is found, because even low deviations of the data points are counted as outliers (cf. Algorithm 1 step #7). In other words, if not a sufficient number of clusters ($m < M$) are built after executing DDCAL, `boundary_min` can be lowered, to fill the target number of clusters (M) with data points.

3.4 Evaluation Based on Real-world Data Sets

To compare DDCAL with existing algorithms as described in Sections 3.1 and 3.2, we use four real-world data sets from different domains in order to show the wide range of possible applications, i.e., Data Set 1 on search processes, Data Set 2 on US population, Data Set 3 on star distances, and Data Set 4 on weather (all data sets are described in Appendix 1).

The results of comparing selected algorithms (cf. Appendix 2) are shown in Tables 5, 6, 7, and 8 (more details of the results can be found here).⁹ Like shown on Tables 1, 2, and 3 (cf. Section 3.2), each table contains variances from 10 trials for stochastic algorithms, where the yielded results depend on the trial because of random methods in the particular algorithm. The results of each of these algorithms represent one execution for the actual comparison among the other algorithms.

In the description of each table, the maximum SED value is listed for the data set to indicate how close each algorithm performs to this value. Bold values indicate the best results

⁹<https://github.com/luxmar/DDCAL/tree/main/supplemental>

Table 5 Results for Data Set 1 (process mining) by using $M = 10$ where max SED = 1.40e+05 (except for trimmed k-means O- (where O- stands for without outliers), which means that a defined percentage (=10%) of elements to cluster were filtered because they were considered as outliers)

| Algorithm name | NUC (max) | SED (max) | SV (min) | MSC (max) |
|-----------------------------------|------------|-----------------|-----------------|-------------|
| DDCAL q_tol=0.45 boundary_min=0.1 | 1.0 | 4.08e+04 | 1.68e+02 | 0.78 |
| k-means++ | 1.0 | 9.60e+02 | 5.81e+00 | 0.7 |
| Variance of 10 trials | 0 | 0 | 0 | 0 |
| Jenks natural breaks | 1.0 | 2.18e+03 | 1.96e+06 | 0.35 |
| Head/tail breaks | 0.2 | 3.20e+01 | 4.31e+02 | 0.96 |
| DBSCAN min_pts=1 eps=3 | 1.0 | 9.60e+02 | 5.81e+00 | 0.7 |
| KDE h=0.1 | 0.2 | 3.20e+01 | 4.31e+02 | 0.96 |
| GMM | 1.0 | 3.51e+03 | 1.41e+01 | 0.59 |
| Variance of 10 trials | 0 | 3.91e+06 | 1.09e+01 | 1.26e-04 |
| Mean shift q=0.1 | 0.2 | 3.20e+01 | 4.31e+02 | 0.96 |
| Largest gaps | 1.0 | 9.60e+02 | 1.96e+06 | 0.63 |
| Gaussian kernel k-means var=20.39 | 1.0 | 6.32e+03 | 2.38e+0 | 0.61 |
| Variance of 10 trials | 0 | 1.18e+08 | 1 2.40e+03 | 1.08e-02 |
| k-medoids | 1.0 | 3.51e+03 | 1.41e+01 | 0.59 |
| Variance of 10 trials | 0 | 5.31e+04 | 5.80e+00 | 1.43e-04 |
| Trimmed k-means O- trim=0.1 | 1.0 | 2.27e+03 | 2.00e+00 | 0.69 |
| Variance of 10 trials | 0 | 0 | 1.18e-32 | 4.44e-33 |
| Trimmed k-means O+ trim=0.1 | 1.0 | 1.36e+04 | 1.77e+06 | 0.6 |
| Variance of 10 trials | 0 | 0 | 6.62e+11 | 7.84e-05 |

Bold values indicate the best results if NUC=1.0

among the compared algorithms in a table if NUC=1.0 and therefore M is reached ($m=M$). As described in Section 3.2, in the following, we discuss only those results with NUC=1.0 and without trimmed k-means where bold values from this algorithm are highlighted in color blue if the result was best among the compared algorithms.

The default parameters for the comparison algorithms which are not changed regarding a particular data set are shown in Appendix 2. For the DDCAL algorithm, we used the default parameters (boundary_min=0.1 resp. 0.15, boundary_max=0.49, num_simulation=20, q_tolerance_increase_step=0.5, and q_tolerance=0.45 resp. 0.1) as described in Sections 3.2 and 3.3. For Data Set 4, q_tolerance is set to 0.1 because of the uniform distribution and on all other data sets, q_tolerance is set to 0.45. For Data Set 2, boundary_min is set to 0.15 because the metrics SED, SV, and MSC perform better. For Data Set 4, boundary_min is set to 0.15, where SED performs a little weaker than 0.1, but the performance increases for metrics SV and MSC.

DDCAL achieves outstanding results given an equal distribution of the data points over the clusters (SED) for all real-world data sets, where DDCAL hit always rank 1 in comparison to all other algorithms. On SV and MSC, the performance of DDCAL is in the midfield or below.

On Data Set 1 (process mining), DCAL ranks 6 for SV 1 for MSC. k-means++ and DBSCAN yield the same output and thus the metrics have the same results. They both rank 1 for metric SV, 2 for MSC, and second last for metric SED which demonstrates a trade-off between SED and the classical clustering metrics SV and MSC. For the stochastic

Table 6 Results for Data Set 2 (U.S. population 2018) by using $M = 10$ where max SED = 6.25e+06 (except for trimmed k-means O- (where O- stands for without outliers), which means that a defined percentage (=10%) of elements to cluster were filtered because they were considered as outliers)

| Algorithm name | NUC (max) | SED (max) | SV (min) | MSC (max) |
|---------------------------------------|------------|-----------------|-----------------|-------------|
| DDCAL q_tol=0.45 boundary_min=0.15 | 1.0 | 3.70e+06 | 6.69e+13 | 0.49 |
| k-means++ | 1.0 | 2.02e+05 | 1.96e+12 | 0.61 |
| Variance of 10 trials | 0 | 5.28e+06 | 7.46e+18 | 2.53e-06 |
| Jenks natural breaks | 1.0 | 3.81e+05 | 4.31e+13 | 0.4 |
| Head/tail breaks | 0.4 | 1.45e+03 | 3.82e+13 | 0.56 |
| DBSCAN min_pts=1 eps=565973.5 | 1.0 | 3.59e+03 | 2.67e+12 | 0.42 |
| KDE h=428947 | 1.0 | 5.24e+04 | 1.84e+12 | 0.52 |
| GMM | 1.0 | 3.14e+05 | 2.50e+12 | 0.56 |
| Variance of 10 trials | 0 | 7.96e+09 | 9.85e+22 | 9.79e-04 |
| Mean shift q=0.074 | 1.0 | 8.74e+04 | 1.79e+12 | 0.55 |
| Largest gaps | 1.0 | 3.18e+04 | 3.11e+13 | 0.51 |
| Gaussian kernel k-means var=559009.01 | 1.0 | 3.63e+06 | 1.21e+14 | 0.41 |
| Variance of 10 trials | 0 | 7.90e+11 | 1.24e+28 | 9.21e-03 |
| k-medoids | 1.0 | 4.03e+05 | 2.38e+12 | 0.56 |
| Variance of 10 trials | 0 | 2.12e+10 | 9.63e+22 | 4.35e-04 |
| Trimmed k-means O- trim=0.1 | 1.0 | 4.35e+05 | 4.96e+11 | 0.66 |
| Variance of 10 trials | 0 | 1.11e+09 | 1.36e+20 | 3.50e-05 |
| Trimmed k-means O+ trim=0.1 | 1.0 | 3.05e+06 | 6.29e+13 | 0.6 |
| Variance of 10 trials | 0 | 3.98e+10 | 1.36e+20 | 2.44e-05 |

Bold values indicate the best results if NUC=1.0

algorithms, k-means shows the same results for all trials and the variance for all metrics thus turns out as 0. Also, the trimmed k-means algorithms performs with no variance for SED. The Gaussian kernel k-means performs worst regarding variances from the 10 trials.

For Data Set 2 (U.S. population 2018), mean shift followed by KDE performed best on metric SV and DDCAL had the “9th” rank out of 10. Regarding metric MSC, DDCAL ranks on place number 7 and k-means++ ranks the first place, followed by GMM. Similar to the previous discussed data set, the variance of k-means++ is the lowest and thus the best and for Gaussian kernel k-means the variance is the highest and thus the worst on stochastic algorithms on this data set.

For Data Set 3 (Stars), with respect to metric SV, DDCAL ranks 4 out of 7 algorithms. k-means++ and GMM perform best. For MSC, DDCAL hits the last place where DBSCAN hits the first, and largest gaps the second place. When comparing the variances of the stochastic algorithms, the trimmed k-means algorithms perform best with low variances where GMM shows the highest variances after 10 trials.

Finally, on Data Set 4 (Weather), for metrics SV and MSC, DDCAL scores between the “6th” and the “7th” rank out of 8. GMM performs best for SV and MSC. k-medoids has the second best results for SV and the best ones for SV. Gaussian kernel k-means performs second best for MSC. For the stochastic algorithms, the variances of trimmed k-means O+ are the highest and thus worst among the compared algorithms. The variances for Gaussian kernel k-means are the lowest and thus the algorithm performs best when comparing the deviations on the metrics from 10 trials.

Table 7 Results for Data Set 3 (distances to stars) by using $M = 10$ where $\max \text{SED} = 6.00\text{e}+40$ (except for trimmed k-means O- (where O- stands for without outliers), which means that a defined percentage (=10%) of elements to cluster were filtered because they were considered as outliers)

| Algorithm name | NUC (max) | SED (max) | SV (min) | MSC (max) |
|-----------------------------------|------------|-----------------|-----------------|-------------|
| DDCAL q_tol=0.45 boundary_min=0.1 | 1.0 | 3.22e+40 | 1.59e+04 | 0.55 |
| k-means++ | 1.0 | 5.71e+39 | 1.04e+04 | 0.57 |
| Variance of 10 trials | 0 | 3.13e+78 | 1.49e+05 | 6.99e-06 |
| Jenks natural breaks | 1.0 | 6.08e+39 | 8.58e+07 | 0.57 |
| Head/tail breaks | 0.2 | 1.12e+09 | 3.63e+04 | 1.0 |
| DBSCAN min_pts=1 eps=8.34 | 1.0 | 4.25e+24 | 3.32e+04 | 0.69 |
| KDE h=0.1 | 0.2 | 1.12e+09 | 3.63e+04 | 1.0 |
| GMM | 1.0 | 8.92e+39 | 1.06e+04 | 0.57 |
| Variance of 10 trials | 0 | 1.17e+79 | 4.51e+05 | 2.26e-05 |
| Mean shift q=0.068 | 1.0 | 2.24e+34 | 1.40e+04 | 0.58 |
| Largest gaps | 1.0 | 4.66e+24 | 8.58e+07 | 0.68 |
| Gaussian kernel k-means | — | — | — | — |
| k-medoids | — | — | — | — |
| Trimmed k-means O- trim=0.1 | 1.0 | 1.29e+40 | 2.24e+03 | 0.58 |
| Variance of 10 trials | 0 | 5.58e+76 | 2.38e+01 | — |
| Trimmed k-means O+ trim=0.1 | 1.0 | 3.82e+40 | 2.05e+04 | 0.53 |
| Variance of 10 trials | 0 | 2.39e+77 | 2.34e+01 | — |

Bold values indicate the best results if $\text{NUC}=1.0$

As for the synthetic data sets (cf. Section 3.2), we sum up all results for all metrics for each distribution and algorithm by using the feature scaling method min-max normalization using the equation provided in Section 2. The scores range from 0 to 1 where 1 indicates the best result. For Data Set 1 (process mining), DDCAL hit the first place, followed by k-means++ and DBSCAN. For Data Set 2 (U.S. population 2018), k-means++ performs best, followed by DDCAL and k-medoids. For Data Set 3 (Stars), DDCAL shows the best overall performance, followed by DBSCAN and GMM. Finally, for Data Set 4 (Weather), DDCAL performs best and is followed by k-means++ and Jenks natural breaks.

In summary, for metrics SV and MSC, none of the algorithms achieves a good ranking on all real-world data sets. In contrast, the SED results on DDCAL come close to the maximum SED value for each of the real-world data sets. Furthermore, the results show that there exists a trade-off between metric SED and the classical clustering metrics SV and MSC, as discussed earlier. Section 4 will illustrate why a high SED value from clustered results as pre-processing of 1d data for visualization is favorable based on different use cases, as also already mentioned in Section 1.

The comparison of the characteristics of the algorithms leads to the following observations: k-means++ shows a good overall performance on the real-world data sets. One reason is that the real-world data sets are similar to an exponential and uniform distribution, where the algorithm also achieves good results (cf. Section 3.2).

k-means++, GMM, Gaussian kernel k-means, k-medoids, and trimmed k-means (which was left out in the comparison) are struggling to be reproducible, because different results are yielded depending on the trial (Thrun, 2021). Thus, after each run, the clustered results differ and thus, do not produce stable results like DDCAL. This behavior produces a weak

Table 8 Results for Data Set 4 (weather) — min temperatures by using $M = 10$ where max SED = 4.31e+15 (except for trimmed k-means O- (where O- stands for without outliers), which means that a defined percentage (=10%) of elements to cluster were filtered because they were considered as outliers)

| Algorithm name | NUC (max) | SED (max) | SV (min) | MSC (max) |
|-----------------------------------|------------|-----------------|-----------------|-------------|
| DDCAL q_tol=0.1 boundary_min=0.15 | 1.0 | 3.91e+15 | 5.13e+00 | 0.51 |
| k-means++ | 1.0 | 3.19e+15 | 4.82e+00 | 0.55 |
| Variance of 10 trials | 0 | 5.07e+28 | 3.09e-03 | 2.15e-05 |
| Jenks natural breaks | 1.0 | 3.52e+15 | 4.84e+00 | 0.54 |
| Head/tail breaks | 0.2 | 3.35e+04 | 1.83e+01 | 0.63 |
| DBSCAN min_pts=1 eps=0.3 | 0.9 | 1.94e+08 | 1.67e+01 | 0.24 |
| KDE h=0.62 | 0.7 | 7.42e+10 | 6.60e+00 | 0.52 |
| GMM | 1.0 | 4.14e+14 | 4.48e+00 | 0.56 |
| Variance of 10 trials | 0 | 6.99e+29 | 1.95e-02 | 1.77e-04 |
| Mean shift q=0.15 | 1.0 | 3.99e+13 | 5.91e+00 | 0.53 |
| Largest gaps | 1.0 | 5.15e+09 | 1.67e+01 | 0.19 |
| Gaussian kernel k-means var=1.61 | 1.0 | 3.40e+15 | 4.75e+00 | 0.56 |
| Variance of 10 trials | 0 | 0 | 1.97e-31 | 1.97e-33 |
| k-medoids | 1.0 | 4.39e+14 | 4.48e+00 | 0.56 |
| Variance of 10 trials | 0 | 7.63e+29 | 1.86e-02 | 3.35e-05 |
| Trimmed k-means O- trim=0.1 | 1.0 | 1.38e+15 | 2.52e+00 | 0.57 |
| Variance of 10 trials | 0 | 1.60e+27 | 6.64e-05 | 1.60e-05 |
| Trimmed k-means O+ trim=0.1 | 1.0 | 3.59e+15 | 7.72e+01 | 0.46 |
| Variance of 10 trials | 0 | 1.29e+28 | 6.08e+01 | 5.21e-05 |

Bold values indicate the best results if NUC=1.0

violation of one requirement set out in Section 1. We speak of a weak violation, because trial sensitive random methods can be set to a fixed number (e.g., the `random_state` parameter for k-means++ on the Python framework sklearn)¹⁰, with the effect that the random method will be deterministic and returns always the same results. However, this approach may not lead to the best results of an algorithm.

The variance of the before mentioned algorithms is shown for the metrics in Tables 1, 2, 3, 5, 6, 7, and 8 because we use random methods without restrictions. Head/tail breaks, DBSCAN, KDE, and mean shift do not entail any parameter to define the number of aimed clusters, M . Except for head/tail breaks, the aforementioned algorithms have input parameters for which we use simulation methods to maximize metric SED which implies to optimize m . We use the simulation method also in order to not exceed M for the particular algorithms. In terms of runtime, mean shift, trimmed k-means, and Gaussian kernel k-means are the slowest algorithms on the biggest Data Set 3. For 119,614 data points, the Gaussian kernel k-means is not able to terminate within 48 h, the trimmed k-means algorithm takes more than 15 h, and the mean shift algorithm was quite slow on this data set with an execution time of about 4 min without the simulation method. DDCAL has a runtime of about 3 s with its built-in simulation method (the algorithm has a complexity of $O(n \log n)$, by including the sorting step at the beginning. Regarding the observed runtime of the algorithms, it should be noted that their implementation in Python (cf. Section 3.1) may use different

¹⁰<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>, accessed 2022-08-08

frameworks and data structures under the hood. This might cause different results in terms of performance. Thus, the observations regarding runtime should be considered with caution and interpreted as experimental run and are not further discussed. Also, k-medoids is not able to produce results on the biggest Data Set 3, because the computer, where the algorithm was executed, ran out of memory.

4 Application: Visualization of Data on Maps and Process Models

Getting the most insights out of the data can be supported by the “[t]ight integration of visual and automatic data analysis methods” (Keim et al., 2008); particularly, the model building can be seen as the iterative application of data mining methods and visual exploration.

Hence, we can identify pre-processing of 1d data for visualization on business process models and (cloropleth) maps as applications for DDCAL (Algorithm 1).

The visualization of business process models represented by directed graphs, as introduced in Section 1, can be enriched by data connected with the nodes of the graph. Data Set 1, for example, stores the number of search results for nodes in a process model that reflects the search behavior of customers. Figure 6 depicts the process model that is used as underlying structure for visualizing Data Set 1 using different clustering algorithms, i.e., k-means++ and DBSCAN in Fig. 6(a) and DDCAL in Fig. 6(b), for coloring each node based on their frequency, labeled as freq in each node. The results for all clustering algorithms are shown in Table 5, where k-means++ and DBSCAN had the same output. We have chosen k-means++ and DBSCAN to compare with the DDCAL because its overall normalized performance among algorithms from the metrics SED, SV, and MSC together was the second best performing with NUC=1, where DDCAL performed best (cf. Section 3.4).

At first sight, both visualization results differ. In Fig. 6(a) which used k-means++ and DBSCAN for coloring the nodes, there is no difference between the most frequently occurred frequencies in the process model, namely between frequencies 3 and 4 which were assigned to the the same “green” color gradient. By contrast, in Fig. 6(b), there is a distinction between frequency 3 and 4 which is highlighted as a blue frame in the process model and for readability, and these nodes are magnified as well in a blue box on the left side. Such a distinction is important as the whole process model shows 33 nodes in total and nodes with frequency 3 appear 7 times (i.e., 21%) and nodes with frequency 4 appear 6 times (i.e., 18%). Furthermore, if a visualization tool for process models supports a filter option to filter out nodes with low frequencies, for example to show just frequencies >4, the distinction between often occurred nodes with colors is important because an observer can see visually, which elements are affected from such a filtering step. Generally, a higher variety of colors apart from just “green” tones is shown in the process model which helps to recognize heat maps, for highlighting regions with a high frequency. Such an example is demonstrated through the violet highlighted region in the process model. This also enables a stronger distinction of execution frequencies and in the sequel execution behavior of users between these activities. Furthermore, a happy flow of process paths can be easily discovered at first glance through the coloring of nodes based on their frequencies in process models. A happy flow from process paths which contain more than 4 nodes is highlighted with a red frame.

Note that the clusters do not have to be necessarily mapped onto colors for visualization. For the process mining use case, a mapping onto edges in terms of stroke width is also conceivable.

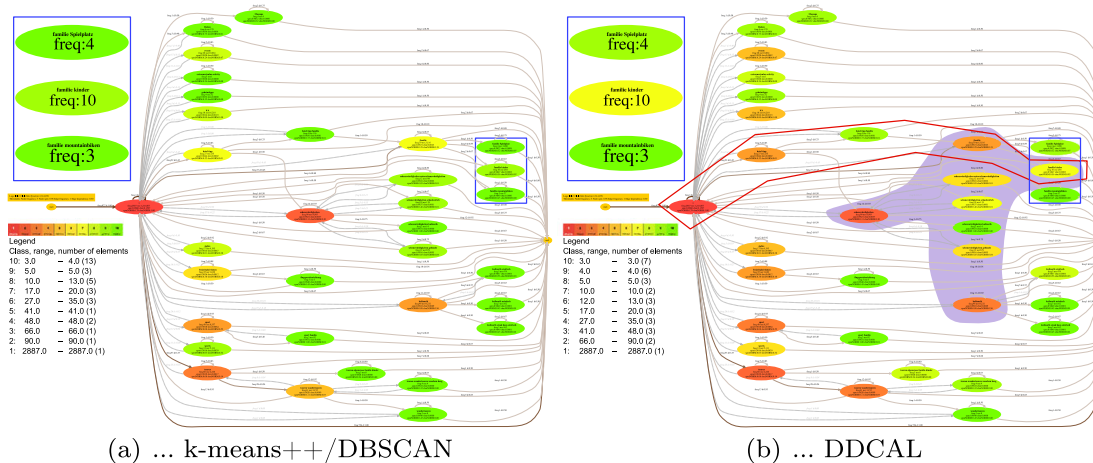


Fig. 6 Process mining data set pre-processed by (a) k-means++/DBSCAN and (b) DDCAL

A different visualization is shown on Data Set 2. The goal on this data set is to visualize the data on a US map in order to distinguish the population sizes of different states. For visualization of a choropleth map, we downloaded the shape files of the US states from the year 2018 from [census.gov](https://www.census.gov).¹¹ Then, the data is pre-processed by clustering using (a) Jenks natural breaks and (b) DDCAL. The results of all clustering algorithms were already introduced in Table 6 and discussed as well in Section 3.4. We used the algorithm Jenks natural breaks to compare with DDCAL because it was designed to visualize choropleth maps (Jenks, 1967). Figure 7 depicts the results for both algorithms. In Fig. 7(a), it is shown that the clustering by using Jenks natural breaks does not result in well distinguishable colors for states with low population sizes due to the “dominance” of a few states with high population size in the south and east. Thus, the majority of the states with lower populations (in green colors) can hardly be distinguished. Compared to Jenks natural breaks, the visualization in Fig. 7(b), by using the clustering algorithm DDCAL, the assigned colors to the states lead to a more fine-grained exploitation, enabling the distinction of the population sizes of more states, particularly of those with lower population sizes.

Additionally, Data Set 5 features data on the Corona pandemic that has posed highest priority to monitor closely the infection development in order to enable quick reactions to (local) outbreaks and subsequent mitigation actions (Thomas et al., 2020). Several visualizations have been provided, for example, the dashboard of the World Health Organization (WHO).¹² For the visualization with cluster pre-processing, we take the subset of absolute confirmed cases from 2020-06-05. That day, the US accounted for around 28% of all confirmed cases world-wide while the 52 African states captured by the data together account for around 2.6%. Hence, we can speak of outliers in the data. Figure 8 displays two visualizations of the data set that have been pre-processed by clustering and are displayed using the app.¹³ In (a), k-means++ is used for clustering, in (b) the DDCAL (cf. Section 2) algorithm to be proposed in this work. At first glance, it can be seen that in (a) there is almost no visual differentiation for African countries, whereas in (b) different clusters can be differentiated. For example, the cluster containing Egypt (0.5%) and South Africa (0.6%) can be distinguished from a cluster containing — among other countries —

¹¹<https://tinyurl.com/tgw4vty>, accessed 2022-08-08

¹²<https://covid19.who.int>, accessed 2022-08-08

¹³<https://corona.swis.io/>

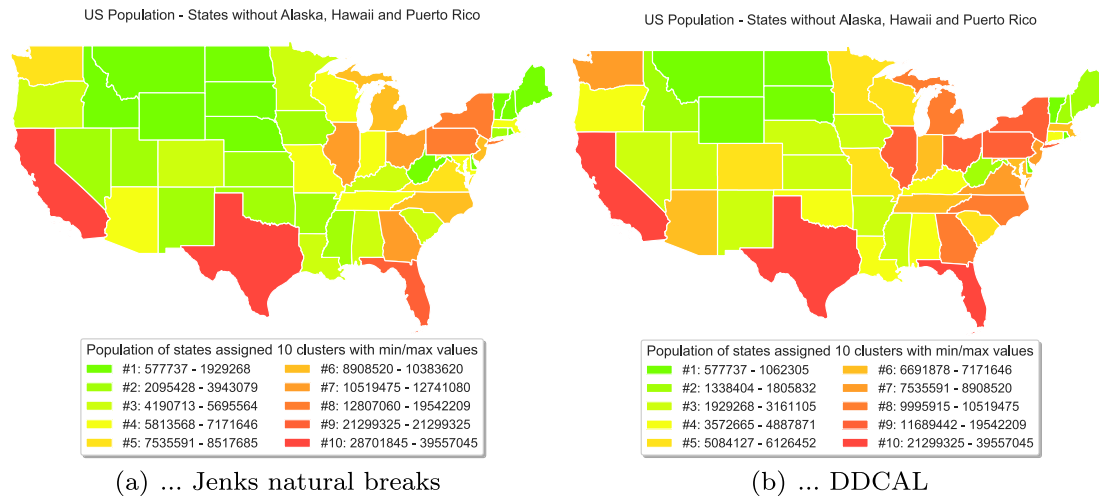


Fig. 7 Process mining data set pre-processed by (a) Jenks natural breaks and (b) DDCAL

Nigeria (0.18%) and Algeria (0.15%) and furthermore, Mali (0.002%) can be distinguished from Niger (0.001%). The numbers in brackets denote the percentages of confirmed cases in the respective countries in relation to the number of confirmed cases world-wide.

5 Related Work

The definition of clustering is still an open discussion (Estivill-Castro, 2002). In some ways, it is described to group similar data into a cluster for permitting a significant generalization (Bonner, 1964) and it helps in data mining to identify interesting structures in data (Thrun et al., 2020). A different clustering concept, which is not treated in this work, is conceptual clustering, which accepts a set of object descriptions to produce a classification scheme over observations (Fisher, 1987). Section 3 provides a detailed comparison of DDCAL with a set of related clustering algorithms, i.e., k-means++ (Arthur & Vassilvitskii, 2007) (Arthur & Vassilvitskii, 2006), Jenks natural breaks (Jenks, 1967), head/tail breaks (Jiang, 2013), DBSCAN (Ester et al., 1996), KDE (Scott, 2015), GMM (Reynolds, 2009), mean shift (Comaniciu & Meer, 2002), largest gaps, Gaussian kernel k-means (Dhillon et al., 2004), k-medoids (Park & Jun, 2009), and

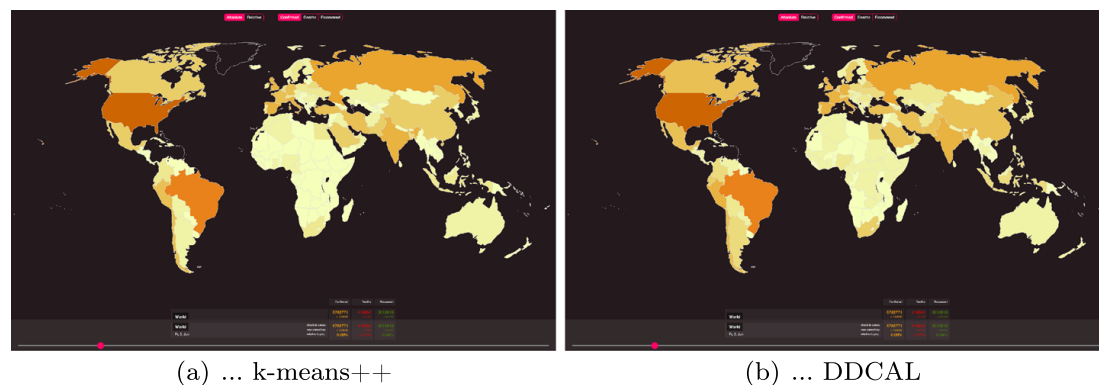


Fig. 8 Corona visualization of absolute confirmed cases for the date 2020-06-05 from the Johns Hopkins CSSE data set pre-processed by (a) k-means++ and (b) DDCAL

trimmed k-means (Cuesta-Albertos et al., 1997). Further algorithms which can be used for clustering of (1d) data points are, for example, hierarchical cluster methods like pairwise agglomerative clustering as described in Faber (1994). This method has a major drawback in terms of performance on large data sets because each iterative step requires multiple distance calculations. Many popular algorithms such as k-means have various modifications of their basic concept, e.g., the k-medoids algorithms (Thrun, 2018) which are designed to be less sensitive to outliers than the original algorithm. One implementation here is the PAM algorithm (Kaufman & Rousseeuw, 1990) which uses medoids instead of centroids, i.e., data points in the data set itself and the Minkowski distance. One-dimensional clustering can be used to generate choropleth maps (Wright, 1938) for visualizing how a measurement varies across a geographic area. Another use case for one-dimensional clustering is to generate heat maps, e.g., for visualizing the frequency of nodes or edges in directed graphs from mined processes by using process mining algorithms (van der Aalst, 2016), which are similar to choropleth maps, but with the difference that they do not use geographic areas. Fair clustering techniques such as Chierichetti et al. (2017) aim at minimizing distances between the data points within the clusters and maximizing the distances of the points between different clusters. On top of that, they try to respect fairness constraints with respect to possibly sensitive features such as age. The DDCAL algorithm can be interpreted as fair clustering technique with respect to the even distribution of the data points over the clusters, but do not assign any sensitivity to features.

6 Summary and Outlook

Summary Regarding Research Questions: This work addresses research questions RQ1–RQ3 as set out in Section 1 as follows: DDCAL constitutes a heuristic clustering algorithm for evenly distributing data into clusters over a maximum number of low variance clusters based on the iterative feature scaling method min-max normalization which is also known as rescaling (RQ1). Regarding RQ2, we studied DDCAL on several synthetic and real-world data sets and compared the results to 11 existing clustering algorithms. From the synthetic data sets it can be observed that DDCAL performs well for data with outliers and data following (tailed) distributions with one peak, which show a bell curve such as normal and gumbel distribution. Additionally, DDCAL has a good performance on uniformly distributed data, or the number of peaks in the data set is equal or higher than the number of targeted clusters (M). If the number of peaks is lower than the number of targeted clusters or the data set to cluster is exponentially distributed, the DDCAL shows weaknesses. For DDCAL, outliers are particularly treated in a way such that they do not “dominate” the resulting clusters. Three use cases from process model/mining to map visualization indicate that DDCAL results in a more differentiated color grading and hence might lead to a more effective visualization of the data (RQ3).

Discussion: The assessment of the algorithms is based on four quality metrics applied to synthetic as well as real-world data sets. We do not employ supervised quality measures because no gold standard for the clusters on the used data sets is available which perfectly suits the trade-off between the metrics SED, an even distribution of data points into clusters and the classical clustering metrics SV and MSC. As shown in Section 5, even the definition of clustering is still an open discussion. The usage of unsupervised quality measures is always biased (Handl et al., 2005; Thrun, 2021) and therefore we evaluate the results based on use cases where we show what additional information could be observed by using

DDCAL in comparison to other algorithms which had a good performance on the employed quality metrics.

We observed that DDCAL basically outperforms the other algorithms when evenly distributing data over all clusters and shows average results on building low variance clusters. Thus, DDCAL can be seen as good “all-rounder” for use cases demanding for evenly distributing data elements into a given number of clusters. k-means++ yields good results for all quality metrics. Gaussian kernel k-means performs well regarding an even distribution of data over all clusters. Both algorithms have one drawback, which is the missing “reproducibility” of the results because the initial cluster centers are set differently after each initiation run, which leads to different cluster results after every execution. Gaussian kernel k-means has further problems: The algorithm produces overlapping clusters in some cases and has a high time and space complexity. Moreover, for Gaussian kernel k-means and other existing algorithms analyzed in this work requiring input parameters, there is no research for setting these parameters for 1d data to produce results with an even distribution of data over all clusters and at the same time, having a low variance in clusters. We tackled this problem by simulating different input parameters, but possibly better approaches may be used with further research.

Higher-Dimensional Data: Currently, we are working on extending DDCAL for clustering multidimensional data sets. Starting with 2d data, we follow two directions using (a) several distance measures and (b) pre-processing the data. For (a), we can use, for example, the Euclidean distance, Manhattan distance, cosine coefficient, Jaccard coefficient, dice coefficient, Minkowski distance, root mean squared error coefficient¹⁴, and TS-SS (Heidarian & Dinneen, 2016) by comparing them with extreme data points, like (max-x-value/min-y-value), (min-x-value/max-y-value), (max-x-value/max-y-value), and (min-x-value/min-y-value). Option (b) includes an additional pre-processing step which is responsible for converting the multidimensional data points into one dimension and then using the core concepts of DDCAL. Subsequently, we aim at developing an algorithm for merging the produced clusters from each dimension for building multidimensional clusters.

Future Work: Beyond data visualization, we will evaluate how DDCAL can be used for problems like clustering test scores of students for grading (Faber, 1994) or to build evenly distributed learning groups based on previous test scores. Additionally, we will investigate whether and how DDCAL can be used for indexing of data to achieve, for example, a faster information retrieval.

Appendix 1. Data set descriptions

Data Set 1 (Process Mining, Search Process Models) The data set¹⁵ contains the search frequencies (column freq) of 33 nodes labeled with the corresponding search term from a customer search process (i.e., a directed graph) that was discovered using process mining techniques. For an overview on process mining techniques such as the Heuristic Miner, we refer to van der Aalst (2016). For detailed insights into customer process mining, we refer to Lux et al. (2018) and Bernard and Andritsos (2019).

¹⁴<https://developers.google.com/machine-learning/clustering/similarity/manual-similarity>, accessed 2022-08-08

¹⁵<https://github.com/luxmar/DDCAL>

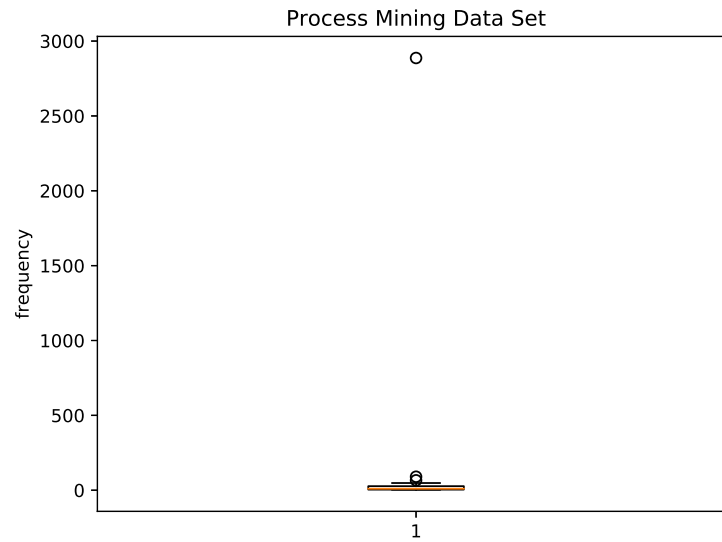


Fig. 9 Box plot of Data Set 1

In general, the representation of process mining results poses a challenge, and representations of process models are denoted as “maps” for process execution behavior for users (van der Aalst & et al, 2011). Specifically, for the analysis of customer behavior, the process model reflects search processes which were mined from event logs collected through the information system.¹⁶ These logs were generated by tourists through keyword-based entered search terms to find touristic activities in this information system over a period of

¹⁶Available at <https://github.com/luxmar/DDCAL/tree/main/supplemental>

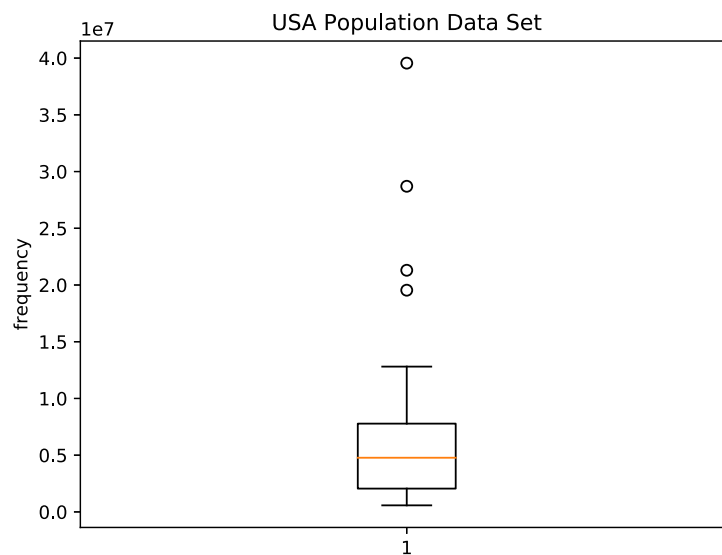


Fig. 10 Box plot of Data Set 2

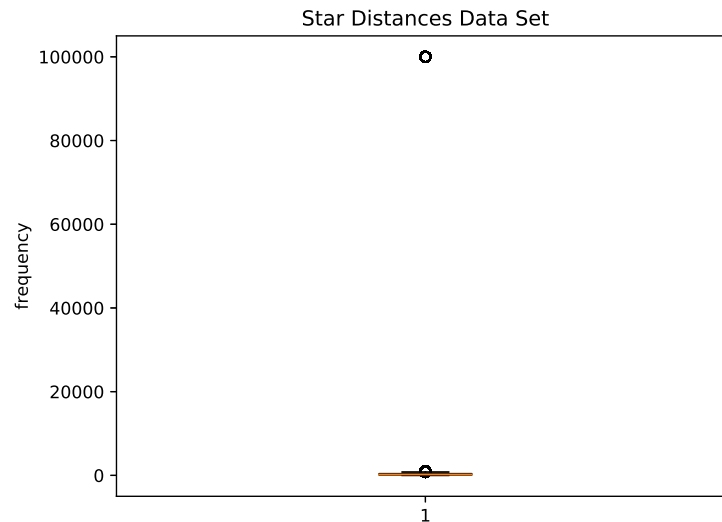


Fig. 11 Box plot of Data Set 3

time. The data contains one outlier reflecting the * search¹⁷ which accounts for 2887 out of 3444 searches (i.e., 84%). Moreover, the data is skewed, i.e., contains more nodes with small frequencies than nodes with large ones: 16 out of the 33 nodes (i.e., 48%) have a search frequency ≤ 5 , reflecting search terms such as hiking tours or sports family. One could argue to remove the * search term from the analysis. However, the * search is part of different search paths in the customer processes, and hence conveys meaningful insights into the search behavior (Lux et al., 2018). Without the outlier, which is responsible for a huge gap, the data is comparable to an exponential distribution. The box plot of the data set is shown in Fig. 9.

Data Set 2 (United States Population) The estimated population of the United States (US) per state in 2018 (1 year) is provided from the census data base [census.gov](https://www.census.gov) of the US government.¹⁸ Column S0101_C01_001E contains the number of total population per state. We exclude Puerto Rico, Hawaii, and Alaska by deleting the associated rows, as these states are not displayed in our final visualization on the map. We added the population from District of Columbia to the state Maryland and removed the corresponding row, because it is part of the latter state. Also, the row United States, which contains the whole estimated population of the United States, was removed. Thus, the final data set of US population per state covers to 48 states. The data is skewed and comparable to an exponential distribution with many gaps: 25 (52%) of the 48 states, for example, have less than 5 Mio inhabitants where 4 (8%) states have more than 15 Mio inhabitants. The box plot of the data set is shown in Fig. 10.

Data Set 3 (Stars) Distances from earth to observed stars are stored in a star database.¹⁹ Column dist represents the distance information in 119,614 rows. Distances are stored in the unit parsecs, which is the most common unit in astrometry. For conversion of parsecs to light years, they are multiplied by 3.262. A distance $\geq 100,000$ indicates that the data is missing or dubious (e.g., negative). Ten thousand two hundred fifteen elements have a distance of

¹⁷<https://www.luxactive.com/en/>

¹⁸* means that a keyword based search was performed without entering a search term

¹⁹<https://tinyurl.com/wnpo23y>, accessed 2022-08-08

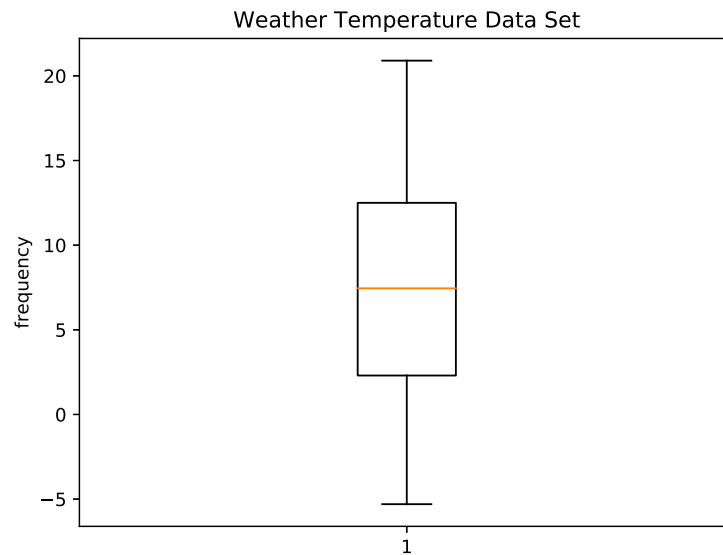


Fig. 12 Box plot of Data Set 4

100,000. The data set has therefore two peaks, where the second peak has the same value. There is also a large gap between both peaks. Without these high distance elements, the data set can be seen as heavy tailed and be comparable to a mixture of exponential and Gumbel distribution. Through cluster algorithms like DDCAL, the distances from this data set can be clustered and the number of stars in each cluster can be visualized, for example, to show infographics for the closest, over distant up to the farthest stars away from earth. A corresponding infographics can be found here.²⁰ The box plot of the data set is shown in Fig. 11.

Data Set 4 (Weather) From the source, there is no description where the data set²¹ originates from and if it is artificially generated or not. However, it seems to be a real-world data set. The data set is stored in the file `weather.csv` with the column `MinTemp` and contains 366 elements. We used the lowest temperatures in Celsius over a period of time as input for comparing the algorithms summarized in Section 3.1. The data set follows a uniform distribution. One use case could be to cluster the data into low, medium, and high temperatures and to show the occurrences in an infographics. Alternatively, if more points exist, we could plot a heat map on a geographical map, respectively, a choropleth map. The box plot of the data set is shown in Fig. 12.

Data Set 5 (Corona Pandemic) The data set relies on the 2019 Novel Coronavirus COVID-19 (2019-nCoV) Data Repository by Johns Hopkins CSSE.²² Particularly, we are interested in the number of confirmed infections, deaths, and recoveries for all countries starting from 22 January 2020.

²⁰<https://tinyurl.com/sp3pdgx>, accessed 2022-08-08

²¹<https://github.com/luxmar/DDCAL/tree/main/supplemental>

²²<https://www.kaggle.com/zaraavagyan/weathercsv>, accessed 2022-08-08

Appendix 2. Implementation of algorithms

DDCAL: Following the pseudo code descriptions of Algorithm 1, the implementation was made in the programming language Python without using additional frameworks, with exception of numpy (version 1.21.2). DDCAL has 6 parameters to set where recommendations on specific data sets are described in Section 3.3. The algorithm produces stable results. The Python implementation of DDCAL is accessible on GitHub.²³ and usable on PyPI.²⁴

kmeans++: The kmeans++ implementation is based on Arthur and Vassilvitskii (2006). We used the Python framework `scikit-learn` (version 0.24.1) to create the cluster centers. An input parameter is given for the number of aimed cluster centers. Then for each data point, the nearest cluster center is calculated and assigned to this cluster by using the Euclidean distance. The initialization method (k-means++) which selects the initial cluster centers for k-means clustering is performed according to Arthur and Vassilvitskii (2007) and thus is a stochastic algorithm, which may produces different results after each run. We used this method, because it is the most common method for k-means clustering which is used as well as default in the popular Python framework `scikit-learn` and has two advantages: (a), to avoid k-means for converging to a local minimum and (b), to avoid iterations which saves computing time. However, we also tried “random” as initialization method and the results were very close to the results from the “k-means++” initialization method. E.g., after execution of 10 runs on normal distribution (cf. Section 3.2), we observed, that the SED value was most of the times slightly higher, SV was most of the times nearly the same MSC was always the same (cf. metrics from Section 2.2). Details about further default parameters (e.g., maximum iterations = 300) which were used as parameters in the algorithm are shown in.²⁵

Jenks natural breaks: The core idea of the algorithm is based on Jenks natural breaks (Jenks, 1967) and improved in terms of time complexity with the Fisher-Jenks algorithm as described in.²⁶ We used the Python framework `jenkspy` (version 0.2.0) for creating the breaks. Based on the calculated breaks, we created classes and assigned every data point to one of these classes. The algorithm contains one input parameter for defining the number of aimed clusters and aims to reduce the variance within classes and maximizes the variance between classes. It is popular in cartography to generate choropleth maps. The algorithm contains basically three steps²⁷, where in the first step, for each data point, the “sum of squared deviations for array means” (SDAM) is calculated. In the second step, for each range combination, the “sum of squared deviations for class means (SDCM_ALL) is calculated. Next, smallest variation within classes is chosen which leads to the third step that assesses the “goodness of variance fit” ($G VF := \frac{SDAM - SDCM}{SDAM}$) which ranges from 1 (perfect fit) to 0 (worst fit). Thus, the best combination has the highest value for GVF and is finally chosen as result. The algorithm produces stable results.

Head/tail breaks: This algorithm is mainly inspired by the Jenks natural breaks algorithm (Jiang, 2013) and creates breaks as well as output. We used an existing Python implementation.²⁸ and assigned the data points to classes like explained in Jenks natural breaks. The

²³<https://github.com/CSSEGISandData/COVID-19>, accessed 2022-08-08

²⁴<https://github.com/luxmar/DDCAL>

²⁵<https://pypi.org/project/ddcal/>

²⁶<https://scikit-learn.org/1.0/modules/generated/sklearn.cluster.KMeans.html>, accessed 2022- 08-08

²⁷<https://tinyurl.com/2dpfr2sr>, accessed 2022-08-08

²⁸<https://www.ehdp.com/methods/jenks-natural-breaks-explain.htm>, accessed 2022-08-08

results of the algorithm are stable. The algorithm contains no input parameter. Thus, the number of envisaged clusters cannot be specified. Basically, the algorithm performs best for visualization of choropleth maps with data sets where far more small data points than large ones exist.

DBSCAN: The core idea of the algorithm is based on Ester et al. (1996). We used the Python framework `scikit-learn` (version 0.24.1) for creating the clusters. The algorithm produces stable results and has two input parameters. The parameter which is responsible for the minimum number of points was set to 1 which implies that no data points were discarded. The parameter ϵ was calculated through simulations for suitable maximum allowed gaps. There exist many methods to calculate an optimum value for ϵ , like plotting distances between data points and selecting ϵ as the point of the maximum curvature. These approaches did not work in our case because we have a fixed number of envisaged clusters which is not considered in such approaches. Furthermore, the algorithm has no parameter for a number of aimed clusters as upper boundary. Therefore, we implemented a simulation method with the input of different ϵ values which identified the maximum score of even distributed data points over all clusters (SED) within the given number maximum clusters (M), which were in our case the number of aimed clusters (for further details on SED and M , cf. Section 2.2). The range of ϵ values used for simulation was determined by the minimum and maximum gap between data points in a given data set. We used 1000 simulation steps to test evenly spaced ϵ values based on the determined range.

KDE: Kernel density estimation is a statistical method to estimate the probability density function of a random variable (Scott, 2015). KDE has many application possibilities, for example on visualization of data, to plot a density curve in place of plotting a histogram. We used the KDE method for clustering in a Python implementation which is described as follows.²⁹ First we calculated an evenly spaced interval by using the KDE with the Gaussian kernel. Then, we calculated its relative minima. We sorted the minima descending and took the first (largest) minimum according to the desired number of clusters. Then, we created a list of classes, which represented our clusters, with the minima as splitting points. Finally, we assigned the data points to the list of classes. For the KDE algorithm we used the framework `scikit-learn` (version 0.24.1) and for calculating the minima we used the framework `scipy` (version 1.6.2). The KDE has only one input parameter called bandwidth h . Like on DBSCAN, we implemented a simulation method with different input parameters of h which had the goal to identify the maximum score of evenly distributed data points over all clusters (SED) for a given number of maximum clusters (M) which is the number of aimed clusters (cf. Section 2.2). Thus, there exists no parameter to define aimed clusters and the number of clusters was identified through simulation of different h values. The range of different h values for simulation was determined, by testing different ranges of h values which produced m clusters. After a matching range was found, we executed on this range 1000 simulation steps and the highest result for SED was kept. The results of the algorithm are stable.

GMM: Gaussian mixture model attempts to find clusters from a mixture of a finite number of Gaussian distributions with unknown parameters (Reynolds, 2009), (VanderPlas, 2016). It is a density estimator like the previously introduced KDE. We used the framework `scikit-learn` (version 0.24.1) for our Python implementation where the default parameter for EM iterations to perform, was set from 100 to 500, otherwise the default

²⁹https://github.com/chad-m/head_tail_breaks_algorithm, accessed 2022-08-08

parameters were used.³⁰ The framework requires as input parameter the number of Gaussian distributions M , and the clusters are built based on the calculated probabilities from the Gaussian mixture model, where data points are assigned to the most probable distribution. The algorithm is based on expectation maximization (EM), where yielded results depend on a trial (Thrun, 2021) and thus, the algorithm is stochastic.

mean shift: The algorithm is described in Comaniciu and Meer (2002) and for the Python implementation we used the framework `scikit-learn` (version 0.24.1). The framework is designed to assign each data point to its associated cluster and uses the Flat kernel. The algorithm requires as input parameter a defined quantile to estimate the bandwidth h . Following the same logic on simulating input values, as described on DBSCAN and KDE, we implemented a simulation method for different values of quantiles with the goal to identify the best score of even distributed data points over all clusters (SED) within the given number of maximum clusters (M) where further details are on SED and M , are discussed in Section 2.2. Thus, the algorithm has basically no parameter for aimed clusters or maximum clusters, but the number of aimed clusters was determined through simulation by testing quantiles ranging 0.005 to 1. The exact range on each data set of different quantiles for simulation was determined, as described on KDE, by simulating different quantiles which produced about m clusters. On each simulation, we executed 100 simulation steps to test a range of quantiles. The algorithm produces stable results.

largest gaps: This algorithm is not based on literature and implemented in a Python framework. It is an ordinary method based on the largest gaps in a sorted data set between adjacent data points for clustering which was implemented as follows: First we sorted the list of data points in ascending order. Then, we calculated and stored for every data point (a) the gap between the current and the previous data point and (b) the average value between the current and the previous data point. After that, we sorted the list of data points based on their calculated gaps (a) in ascending order. The algorithm had only one input parameter which was the number of aimed clusters (M). Based on the set number of aimed clusters, we took the number first elements from the list of gaps (a) and stored for every data point the average values (b) in a separate list. This list was finally our classification list for assigning each data points to clusters. For example the first cluster, which was the cluster containing the lowest data points, ranges from the minimum data point to the lowest element of the classification list. The second cluster ranges from the lowest element of the classification list plus one, up to the second lowest element of the classification list, and so on. This algorithm is very intuitive and simple to implement which produces stable results.

Gaussian Kernel k-means: This algorithm is a variant of k-means which was introduced before, but uses the the Gaussian kernel method as non-linear distance function. Therefore, it is a stochastic algorithm, which may produces different results after each run. Because there exists no implementation in the Python Package Index, we used an implementation.³¹ which is based on Dhillon et al. (2004). The implementation was extended by us for handling empty clusters on an iteration by terminating the algorithm if an empty cluster was found. The algorithm has three input parameters. The first parameter defines the number of aimed clusters (M), the second parameter defines the initial positions of the centroids, which was set to “random” and the third parameter defines the variance of the kernel. The latter parameter was optimized by using a simulation method, as described in DBSCAN, KDE,

³⁰<https://stackoverflow.com/a/35151947>, accessed 2022-08-08

³¹<https://scikit-learn.org/1.0/modules/generated/sklearn.mixture.GaussianMixture.html>, accessed 2022-08-08

and mean shift. As mentioned on KDE and following this approach, the range of different variance values for simulation was determined by testing variances, on which the results produced m clusters. With exception of Data Set 1, where we tested variances ranging from 10,000 to 1,000,000, the range for simulation of the variances was 0.1 to 30 by using 100 simulation steps. Because the initial positions of the centroids were chosen randomly (second parameter as mentioned before), the simulation method was extended to simulate each variance value 10 times which keeps the best result with the maximum score for even distributed data points over all clusters (cf. metric SED from Section 2.2). Thus, on each range, 1000 simulation steps were executed in total and the result containing the highest SED value was kept. The algorithm has two major drawbacks: (a) it was the only one which produced overlapping cluster ranges. Such results were filtered out through a python implementation which extended the simulation method. (b) The time and space complexity of $O(n^2)$ (Sarma et al., 2013) resulted in long execution times on “huge” data sets (containing >1000 data points).

k-medoids: The algorithm is another variant of k-means which was introduced before. Instead of calculating centroids like in k-means which may not be actual data points in a data set, where they are the average between the points in the cluster, k-medoids chooses always actual data points as centers, which are called “medoids” (Park & Jun, 2009). Through the minimizing of the sum of pairwise dissimilarities instead of minimizing the sum of squared Euclidean distances, the algorithm is more robust to noise and outliers than k-means. In contrast to k-means which requires generally the Euclidean distance as dissimilarity measure, k-medoids can be used with arbitrary dissimilarity measures for producing effective solutions, where we used for example the Manhattan distance. For the Python implementation, we used the framework `scikit-learn-extra` (version 0.2.0) with the dissimilarity metric Manhattan and the initialization method `k-medoids++`. Because of the random initialization method, the algorithm is, like `k-means++`, a stochastic algorithm, which may produce different results after each run.

Trimmed k-means: It is a variant of k-means which optimizes the algorithm under trimming a portion of the data points in a data set. The algorithm has the aim of robustifying k-means (Cuesta-Albertos et al., 1997). For implementation of the algorithm used, the `trimcluster` packages from R, which was called from Python where the analysis method of the results was performed. We used this particular R package, and not a python framework, because there was no implementation of trimmed k-means in the Python Package Index. The trim factor was set to 0.1 which means that 10% of outliers are detected by the algorithm from the data to cluster and put in an separate “outlier” cluster. The number of algorithm runs from initial means, which are randomly chosen from the data points, was set to 500. The rest of the parameters were set to their default parameters, as described in.³²

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability The generated and analyzed data during assessment of the algorithms are (partly) available in a GitHub repository.³³

Declarations

Competing Interests The authors declare no competing interests.

³²<https://github.com/ardianumam/Machine-Learning-From-The-Scratch>, accessed 2022-08-08

³³<https://rdrr.io/cran/trimcluster/man/trimkmeans.html>, accessed 2022-08-08

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Al-Kassab, J., Ouertani, Z. M., Schiuma, G., & Neely, A. (2014). Information visualization to support management decisions. *International Journal of Information Technology & Decision Making*, 13(02), 407–428.
- Arthur, D., & Vassilvitskii, S. (2006). *K-means++: The advantages of careful seeding*. Stanford: (Tech. Rep.)
- Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Symposium on discrete algorithms symposium on discrete algorithms* (pp. 1027–1035).
- Bernard, G., & Andritsos, P. (2019). Discovering customer journeys from evidence: A genetic approach inspired by process mining. In *CAiSE forum caise forum* (pp. 36–47).
- Bonner, R. E. (1964). On some clustering techniques on some clustering techniques. *IBM Journal of Research and Development*, 81(1), 22–32.
- Chierichetti, F., Kumar, R., Lattanzi, S., & Vassilvitskii, S. (2017). Fair clustering through fairlets. In *Advances in neural information processing systems* (pp. 5029–5037).
- Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603–619.
- Coulson, M. R. (1987). In the matter of class intervals for choropleth maps: With particular reference to the work of George F Jenks. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 24(2), 16–39.
- Cuesta-Albertos, J. A., Gordaliza, A., & Matrán, C. (1997). Trimmed *k*-means: An attempt to robustify quantizers. *The Annals of Statistics*, 25(2), 553–576.
- Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel *k*-means: Spectral clustering and normalized cuts. In *Knowledge discovery and data mining knowledge discovery and data mining* (551–556).
- Dobson, M. W. (1973). Choropleth maps without class intervals?: A comment. *Geographical Analysis*, 5(4), 358–360.
- Dobson, M. W. (1980). Unclassed choropleth maps: A comment. *The American Cartographer*, 7(1), 78–80.
- Ester, M., Kriegel, H.P., Sander, J., & Xu, X (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining Knowledge discovery and data mining*, vol. 96, pp. 226–231.
- Estivill-Castro, V. (2002). Why so many clustering algorithms: A position paper. *ACM SIGKDD Explorations Newsletter*, 4(1), 65–75.
- Faber, V. (1994). Clustering and the continuous *k*-means algorithm. *Los Alamos Science*, 22(138144.21), 67.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2), 139–172.
- Handl, J., Knowles, J., & Kell, D. B. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15), 3201–3212.
- Heidarian, A., & Dinneen, M. J. (2016). A hybrid geometric approach for measuring similarity level among documents and document clustering. In *Big Data Computing Service and Applications Big data computing service and applications* pp. 142–151.
- Jenks, G. F. (1967). The data model concept in statistical mapping. *International Yearbook of Cartography*, 7, 186–190.
- Jiang, B. (2013). Head/tail breaks: A new classification scheme for data with a heavy-tailed distribution. *The Professional Geographer*, 65(3), 482–494.
- Kaufman, L., & Rousseeuw, P. J. (1990). Partitioning around medoids (program PAM). In *Finding groups in data: an introduction to cluster analysis* (pp. 68–125). Wiley Online Library.
- Keim, D. A., Andrienko, G. L., Fekete, J., Görg, C., Kohlhammer, J., & Melançon, G. (2008). Visual analytics: definition, process, and challenges. In *Information visualization – Human-centered issues and perspectives* (pp. 154–175).

- Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2014). Exploring processes and deviations. In *Business process management business process management* (pp. 304–316).
- Lux, M., Rinderle-Ma, S., & Preda, A. (2018). Assessing the quality of search process models assessing the quality of search process models. In *Business process management business process management* (pp. 445–461) https://doi.org/10.1007/978-3-319-98648-7_26.
- Milligan, G. W., & Cooper, M. C. (1988). A study of standardization of variables in cluster analysis. *Journal of Classification*, 5(2), 181–204.
- Park, H. S., & Jun, C. H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2), 3336–3341.
- Reinkemeyer, L. (2022). Status and future of process mining: From process discovery to process execution. In *Process mining handbook process mining handbook* vol. 448, pp. 405–415, https://doi.org/10.1007/978-3-031-08848-3_13.
- Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of Biometrics*, pp. 659–663.
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
- Sarma, T. H., Viswanath, P., & Reddy, B. E. (2013). Single pass kernel k-means clustering method. *Sadhana*, 38(3), 407–419.
- Scott, D. W. (2015). *Multivariate density estimation: Theory, practice, and visualization*, Wiley.
- Shapiro, H. M. (2005). *Practical flow cytometry*, Wiley.
- Thomas, O., Hagen, S., Frank, U., Recker, J., Wessel, L., Kammler, F., & Timm, I. J. (2020). Global crises and the role of BISE. *Business Information Systems Engineering*, 62(4), 385–396.
- Thrun, M. C. (2018). Approaches to cluster analysis. In *Projection-based clustering through self-organization and swarm intelligence projection-based clustering through self-organization and swarm intelligence* pp. 21–31.
- Thrun, M. C. (2021). Distance-based clustering challenges for unbiased benchmarking studies. *Scientific Reports*, 11(1), 1–12.
- Thrun, M. C., Gehlert, T., & Ultsch, A. (2020). Analyzing the fine structure of distributions. *PloS one*, 15(10), e0238835.
- Tobler, W. R. (1973). Choropleth maps without class intervals. *Geographical Analysis*, 5(3), 262–265.
- van der Aalst, W., & et al (2011). Process mining manifesto. In *Business process management workshops business process management workshops* (pp. 169–194).
- van der Aalst, W. M. P. (2016). *Process mining – Data science in action*, second edition. Springer.
- VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. Inc: O'Reilly Media.
- Wright, J. K. (1938). Problems in population mapping in notes on statistical mapping, with special reference to the mapping of population phenomena. *American geographical society*.

4 Discussion

4.1 Research Questions Answered

From the introduced publications of this cumulative dissertation, the research questions *RQ1-RQ3*, as set out in Sect. 2.1, are addressed in the following sub chapters as follows:

4.1.1 Questions Answered on *RQ1*

Regarding *RQ1*, both publications – *PPM* and *PM1* – show questions on *CJs* which can be answered by applying *PM* techniques on logs in information systems (cf., *RQ1a*) based on different stages of a *CJ* and on different granularities, like from a session level of a user to a whole region. Answering those questions by revealing *CJPs* helps companies to gain an advantage over their competitors and to win in the market by improving their *CJ*. According to *RQ1b*, a data warehouse architecture is built in publication *PPM* for processing data to apply *PM* on logs. This is demonstrated on an existing information system in the tourism domain. For data enrichment, micro services are proposed, where in *PM1*, a particular micro service was implemented for pooling search terms as post processing step of event logs by using ontology support.

4.1.2 Questions Answered on *RQ2*

RQ2 handles how to assess *CJPs* in information systems, where in *PM1*, a novel quality metric for search processes is developed and evaluated (cf., *RQ2a*). For the evaluation of the metric, an experiment was conducted on a real world setting to measure the quality of discovered search process models by using ontology support, *a*) during the search and *b*) after the logs were recorded to improve them by applying pooling of search terms. The metric can be used to assess the quality in terms of complexity and clustering of a whole process model for a particular path or even a node in the process model. This metric has the characteristic, that it considers the *complexity* through the number of edges incident to an activity – where an activity is a node in a process model – and also the *clustering* through the number of executions of an activity together with the number of all occurred events in a log, where the latter is the number of log entries containing activities. A different metric is developed in *PM2*, which assesses the search behavior along search paths in search process models on *CJPs*, which answers *RQ2b*. The novelty and characteristic of this metric is, that it employs also data values, which is the number of the returned search results from a particular search. This number of search results is taken into consideration for an activity of interest (aka search term) and put it into relation with the overall success of the search in the whole search process model, but search results are also weighed by the relative number of executions by the activity that produced the search results. This enables the metric to differentiate if a high number of search results has been produced by a single activity execution or by several activity executions. Through the assessment of the user behavior, with this metric, new insights can be found along search paths. For example, if users are first narrowing their search results by retrieving less search results, and then broadening them by retrieving more search results, a pattern called *Jumping* is discovered, which may be an indicator that users changed their search strategy during the search, because, for example, the second last entered search did not deliver the expected outcome. Such search path analysis results can help companies to discover niches and then targeting certain topics to invest in.

4.1.3 Questions Answered on *RQ3*

Finally, *RQ3* is about how to visualize highly individual *CJPs* in information systems. According to this questions, the publications *PC1* and *PCL* show how to simplify highly individual customer journey processes in information systems. *PM1* contributes on *RQ3a*, by using ontology support for pooling search terms from event logs and by providing online ontology support for users, to simplify

search process models from highly individual *CJPs*. The publication *PCL* shows how to simplify highly individual *CJPs* in information systems by coloring nodes with predefined colors based on their frequencies in a process model. With this method (cf., *RQ3a*), most common process paths (*happy flows*), regions with a high or low frequency as heat map, and exceptional paths with particular high or low frequencies can be highlighted in *CJPs*, which helps to maximize the information gain of the visualization of such process models. Such a coloring with predefined colors needs a cluster algorithm, but without overemphasizing outliers, which means, that the result should produce an even distribution of data points into clusters, but also with the aim of building low variance clusters. There is a trade-off between both aims, and the developed algorithm *DDCAL* handles this trade-off by prioritizing the even distribution over the low variance, but provided parameters allow some steering of the prioritization. The algorithm uses basically an iterative approach, where the feature scaling method *min-max normalization* is used for normalizing a sorted list of one-dimensional data points. In each iteration, results are compared against defined boundaries from a set list to build clusters from outliers. The boundaries can be increased dynamically by using a tolerance factor if an even distribution of clusters is not reachable. Furthermore, additional optimizations and parameters are included in *DDCAL* and also a Python implementation for demonstration exists. Overall, it is shown that the algorithm helps to reduce the cognitive load of visualized *CJPs* by visualizing colored nodes to highlight particular areas in a process model at first glance (cf., *RQ3b*). This is demonstrated on a real world *CJP*-model from the tourism domain but the clustering algorithm can be used beyond business process model visualization as well.

4.2 Conclusion and Outlook

The study of *CJs* gains insights from customers about their behavior which is already approved across many domains. The promising data driven technique “*PM*” helps to discover *CJPs* from information systems. But such processes are usually highly individual, for example, when considering a keyword based search functionality with arbitrary entered search terms by the users. Therefore, methods have to be found to yield improvements of *CJs* from visualized *CJPs*. This is exactly the scope of this dissertation where research questions were introduced and answered through the put in place publications, from preparing information systems for *PM*, to discover highly individual *CJPs*, to the assessment of those processes, till their visualization to gain insights and in further consequence, improvements on *CJs* to help companies to win in the market over their competitors.

Viewed holistically, with this work, the following end-to-end approach for using *PM* on highly individual *CJPs*, can be proposed:

1. Ask process oriented questions on *CJs* which indicate what should be answered by using *PM* to discover process models and as well what to track with a process (i.e., define *cases*)
2. Develop a data warehouse where different loose coupled systems and instances report to, and where the data warehouse is physically separated from live systems
3. Implement pre-processing methods for data in the data warehouse by using, e.g., micro services to enhance the data quality for increasing the quality of answered questions through *PM* techniques
4. Use different process discovery algorithms (e.g., the heuristic miner) on different cases to receive *CJPs*
5. On the discovered *CJPs*, different visualization methods and metrics can be applied to answer particular questions and to get insights from a *CJ* where several elements should be considered:
 - (a) Simplify too complex process models by applying filter methods which may lead to adapt existing process discovery algorithms or to filter/cluster events (i.e., event log entries) before executing the process discovery algorithms and continue with *step #4*
 - (b) Evaluate the discovered process models in terms of quality by using quality metrics to get evidence of the results
 - (c) Use node and edge visualization (coloring/thickness) to visualize, at first glance, comprehensible process models and to identify particular *CJ* paths or heat maps for optimization
 - (d) Use pattern recognition methods on the discovered *CJPs* to identify the user behavior for further *CJ* optimizations
6. Improve the *CJ* according to the insights and answered questions (see *step #1*) and continue either with *step #1*, if new questions arose during this approach, or continue with *step #2*, if more or different data has to be collected to answer the current defined questions

Additionally, the developed artifact and quality metric *spm* can be used on a broader scope than just assessing the quality of search process models as described in publication *PM1*, being discussed in the following: As a reminder, the metric is used to show clear nodes, paths and even process models in terms of complexity and clustering from search process models. One precondition for the usage of this metric is, that there exist event logs, comprising the following content: user entered search terms which represent events (*activities*), date and time or other ordinal values which indicate the order of happened events (e.g., *timestamps*) and identifiers on a particular level which define traces, like for a user, device or session (*case*). The narrow area of application for just search processes could be extended when dealing with similar event logs and also expecting complex process models. For

example, by considering a complex application or website with many views or elements, where users can navigate to each view without limitations in an arbitrary way, in an associated event log, an *activity* could refer to the name of a visited view or to the name of a performed action like “hitting the order button”. Thus, by switching the *activity* from a user entered search term to a performed action or visited view, the metric *spm* can be used on a variety of highly individual process models – beyond from just search process models – for assessing their quality. Furthermore, the metric *spm* is not just limited to assessment tasks as it can be used as well on process discovery algorithms as filter method for nodes in resulting process models, as explained as follows: For process discovery on *PM*, one of the most basic algorithms is the α -*algorithm* (aka alpha miner), which is used to visualize *Petri Nets* from event logs containing at least the input fields, as described before. It shows all activities as nodes and directly followed activities are either connected to each other or, if they directly followed in both ways, they are shown in parallel and are connected to their common previous activity. The algorithm is not considering the frequency of occurred events for an activity and has no filter options for nodes or edges by providing parameters. Therefore, the resulting *Petri Nets* visualize all occurrences of events. A different algorithm, that contains filter options for edges, is the *heuristic miner*, which is usually utilized to visualize *Workflow Nets*, that are a subclass of *Petri Nets*, where all nodes are on a path from source to sink. With the *heuristic miner*, dependency measures between all occurred activities in the event log are calculated, where the value ranges between $[-1; 1]$, and 1 means there is a high directly follows dependency between both events. Thus, the values can be used for filtering edges of the resulting process model visualizations by setting a threshold. (Van Der Aalst, 2016) When applying the *spm* metric on each activity from the event logs, before applying any of the previously introduced process discovery algorithms, node filtering could be achieved. This was actually done in *PM2* and *PCL* with the *heuristic miner* on the introduced business process models from the tourism domain for evaluation of the concepts. It reduced the complexity of these real world process models. The application of the *spm* metric on event logs for process mining as filter method for nodes, or the incorporation of the metric as filter method in existing process discovery algorithms, could be a promising utilization of the metric, which is worth considering and evaluating in future research.

Based on the research from this cumulative dissertation, a much broader and supplementary research field is to consider environmental conditions, like weather data, on mined *CJPs*. This can be achieved by using machine learning technologies, like neural networks, to visualize decisions made by customers on process models, where those decisions are mapped to features as input of a neural network, which can be achieved by implementing explainable AI methods. In addition to visualization approaches on process models, even findings can be discovered to show companies with what measures a *CJP* – and in further consequence, a *CJ* – can be improved to satisfy more customers along their journey.

Finally, beyond the scope of this work, the algorithm *DDCAL* could be extended to cluster multidimensional data points, where the main problem with the current algorithm arises, with the ordering step of multidimensional data points before processing feature scaling in each iteration, which could not be solved until now.

4.3 Limitations and Threats to Validity

The research with the overriding goal – to improve *CJs* by using *PM* techniques to deal with highly individual process models – is a wide ranging topic, as shown in Sect. 2. Hence, the research space in the work at hand is limited to particular topics.

One of them is, in order to deal with highly individual processes, the investigation of search process models from information systems. But even on this narrow topic on search processes, which were simplified by using ontology support in *PM1*, limitations and threads of validity were pointed out, as described in the following: The used ontology to simplify search processes and even the sample size of the experiment for the evaluation of the tested concepts could have been even greater to show an even stronger evidence, in particular, to use post processing of event logs to simplify process models. Also the language – of user entered search terms – in the experiment was limited to German due to the small sample size. If more languages occurred, the pooling effect of search terms (i.e. post-processing) would have been even greater, as discussed and shown in *PM1*. Another limitation, when dealing with search processes, is the missing handling of compound nouns in search queries with mixed search terms that contain both, compound nouns and single nouns. On evaluation with real world data on the tourism business case, hyphens between search terms were added in the user entered query and also in the ontology (e.g., “nature sights” was modified to “nature-sights”). But as discovered in the tourism domain, where the developed concepts were evaluated, queries containing multiple nouns are very seldom. However, to handle compound nouns in search queries, a starting point could be (Silverstein et al., 1999).

Aside from those discussed limitations on search processes, the behavior, how customers will use search functionalities in information systems, may change in the near future, because of the recently emerged interest on large language models, like *ChatGPT*. (Sun et al., 2023) (Lund & Wang, 2023) (Omar, Mangukiya, Kalnis, & Mansour, 2023) This research topic is not covered in this work but it may change search queries, and in further consequence, event logs completely. For instance, questions are formulated in natural language instead of key words as search queries and the results could be different as well, e.g., by showing only the most likely answer, when users are in a conversation with an artificial intelligence. Hence, the author’s opinion is, that it is very likely, that the search behavior of users on *CJs* may change in the future for some domains, like in the tourism domain, for example, by replacing a search functionality with a chatbot (Sanji, Behzadi, & Gomroki, 2022), running a large language model under the hood. However, logs from such systems (e.g., when considering questions to, and answers from a chatbot), may change, but even core concepts – like process discovery algorithms, metrics, classification algorithms and visualization methods – may still remain. Moreover, an artificial intelligence can still use those aforementioned concepts – also by furthermore selecting the right one for a particular case – as they are transparent and thus, human interpretable. Finally, it should be noted that one of the biggest challenges, when working with generative artificial intelligence is, that it may produce correct-sounding results, which are logically incorrect results and those false results are therefore hard to discover. (Dwivedi et al., 2023)

A different subject is, that the logging format *eXtensible Event Stream (XES)* was not covered in this work, in particular in *PPM*. *XES*, published by the IEEE ⁷, is the standard exchange format for event logs for *PM*. This format is the successor of *Mining eXtensible Markup Language (MXML)* and is therefore basically a *XML* format. *XES* is capable to have any number of attributes, which can be nested as well and there exists 5 core types, which are: *String*, *Date*, *Int*, *Float* and *Boolean*. (Van Der Aalst, 2016) When the data warehouse architecture was designed in *PPM*, *XES* was left out because just 3 attributes for further *PM* were used (i.e., *activity*, *time* and *case*, cf., Sect. 2). Furthermore, the flat file format *CSV*, which was used in this work for data exchange when dealing with *PM* tools, was sufficient and is also easier and faster to implement on existing information systems. However, there exist converters from *CSV* to *XES* and vice versa, like the *pm4py* library ⁸, which

⁷<https://www.xes-standard.org/>, accessed 2023-07-05

⁸<https://pm4py.fit.fraunhofer.de/documentation>, accessed 2023-07-06

enables to use the standard format if needed. Even most prominent *PM* tools, like *Celonis* or *DISCO*, support the flat *CSV* file format for importing event logs. But in *PM2*, a new attribute – the number of search results – was considered as well and if more attributes are used and also a huge amount of data has to be processed, for, e.g. process discovery techniques, a different relational database schema – aligned to the *XES* standard that considers also improved memory requirements – may make more sense, like *DB-XES* (Syamsiyah, van Dongen, & van der Aalst, 2018).

Another assumption made in *PPM* was, that all systems, which report data to a data warehouse are under control of the same provider or company. In other words, collecting data from multiple different systems, in which systems may have different owners, was not covered in this dissertation. For dealing with such situations - this could be the incorporation of external data, like weather data to perform *PM* tasks - standards like *FIWARE* in combination with data spaces (Ahle & Hierro, 2022), or the *GAIA-X* initiative, launched by the *EU* with the goal to create the next generation data infrastructure for Europe that meets the highest standards in terms of digital sovereignty (Braud, Fromentoux, Radier, & Le Grand, 2021), have to be considered as well.

Additionally, as discussed in the outlook (cf., Sect. 4.2), the proposed metric *spm* could be used as well not only to evaluate process models, but also for node filtering on process discovery algorithms, which was actually applied in *PM2* and *PCL* but not evaluated in detail, also apart from search process models. Furthermore, a topic which was not investigated in this dissertation, is to consider environmental data on *CJPs*, like to take weather data into account, e.g., to show how this data influences the customer’s behavior and decisions. And last, based on the given outlook according to environmental data, multidimensional data for visualization in process models were not considered within this dissertation, in particular in *PCL*, where the developed *DDCAL* algorithm supports only one-dimensional data points to cluster.

4.4 Findings and Results

Finally, the main contributions and findings of this work are the following:

- An end-to-end approach on how to improve *CJPs* by using *PM* techniques, even applicable on existing systems, for companies to win in the market
- A data warehouse architecture for information system, which is capable to improve the quality of data and to apply *PM* techniques, demonstrated on a real world business case from the tourism domain, but which is still adaptable to other domains
- An evaluated approach for using ontology support – with an easy to implement- and maintainable ontology – for simplifying search process models by using post-processing of event logs and online ontology support for users during the search
- A quality metric (*sbm*), developed for assessing search process models in terms of quality to evaluate clear paths, but with the capability to be used as filter for activities beyond search processes as well (e.g., for pre-processing of event logs on process discovery algorithms)
- A search process behavior metric (*sbm*), which reveals patterns from the search behavior of users by considering the total number of search results from entered search queries
- A novel, stable and high performing cluster algorithm (*DDCAL*), which uses a completely new approach for clustering one-dimensional data elements to improve the visualization of process models but which can be used beyond this narrow scope, e.g., if an even distribution of elements into clusters is needed, but with a low variance inside each cluster

References

- Aftab, U., & Siddiqui, G. F. (2018). Big Data Augmentation with Data Warehouse: A Survey. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2785–2794).
- Ahle, U., & Hierro, J. J. (2022). FIWARE for data spaces. *Designing Data Spaces*, 395.
- Arthur, D., & Vassilvitskii, S. (2006). *K-means++: The advantages of careful seeding* (Tech. Rep.). Stanford.
- Berisha, B., Mëziu, E., & Shabani, I. (2022). Big data analytics in Cloud computing: an overview. *Journal of Cloud Computing*, 11(1), 24.
- Bernard, G., & Andritsos, P. (2017). CJM-ex: Goal-oriented Exploration of Customer Journey Maps using Event Logs and Data Analytics. In *BPM (demos)*.
- Bernard, G., & Andritsos, P. (2018). CJM-ab: Abstracting Customer Journey Maps using Process Mining. In *Information Systems in the Big Data Era: CAiSE Forum 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings 30* (pp. 49–56).
- Bernard, G., & Andritsos, P. (2019). Discovering Customer Journeys from Evidence: a Genetic Approach Inspired by Process Mining. In *Information Systems Engineering in Responsible Information Systems: CAiSE Forum 2019, Rome, Italy, June 3-7, 2019, Proceedings 31* (pp. 36–47).
- Borisov, A., Wardenaar, M., Markov, I., & De Rijke, M. (2018). A Click Sequence Model for Web Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 45–54).
- Bose, R. J. C., Mans, R. S., & Van Der Aalst, W. M. (2013). Wanna Improve Process Mining Results? In *2013 IEEE symposium on computational intelligence and data mining (CIDM)* (pp. 127–134).
- Braud, A., Fromentoux, G., Radier, B., & Le Grand, O. (2021). The road to European digital sovereignty with Gaia-X and IDSA. *IEEE network*, 35(2), 4–5.
- Buijs, J. C., Bergmans, R. F., & El Hasnaoui, R. (2019). Customer journey analysis at a financial services provider using self service and data hub concepts. In *BPM (Industry Forum)* (pp. 25–36).
- Comaniciu, D., & Meer, P. (2002). Mean Shift: A Robust Approach toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603–619.
- Coulson, M. R. (1987). In the matter of class intervals for choropleth maps: with particular reference to the work of George F Jenks. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 24(2), 16–39.
- De Medeiros, A. A., Pedrinaci, C., Van der Aalst, W. M., Domingue, J., Song, M., Rozinat, A., ... Cabral, L. (2007). An Outlook on Semantic Business Process Mining and Monitoring. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops: OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part II* (pp. 1244–1255).
- Dijkman, R., Gao, J., Syamsiyah, A., Van Dongen, B., Grefen, P., & ter Hofstede, A. (2020). Enabling efficient process mining on large data sets: realizing an in-database process mining operator. *Distributed and Parallel Databases*, 38, 227–253.
- Dixit, P. M., Buijs, J. C., van der Aalst, W. M., Hompes, B., & Buurman, H. (2015). Enhancing Process Mining Results using Domain Knowledge. In *SIMPDA* (pp. 79–94).
- Dobson, M. W. (1973). Choropleth Maps Without Class Intervals?: A Comment. *Geographical Analysis*, 5(4), 358–360.
- Dobson, M. W. (1980). Unclassed Choropleth Maps: A Comment. *The American Cartographer*, 7(1), 78–80.
- Dunkl, R. (2013). Data Improvement to Enable Process Mining on Integrated Non-log Data Sources. In *Computer Aided Systems Theory-EUROCAST 2013: 14th International Conference, Las*

- Palmas de Gran Canaria, Spain, February 10-15, 2013, Revised Selected Papers, Part I 14* (pp. 491–498).
- Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A., Kar, A. K., ... others (2023). “So what if ChatGPT wrote it?” Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management*, 71, 102642.
- Eder, J., Olivotto, G. E., & Gruber, W. (2002). A Data Warehouse for Workflow Logs. *EDCIS*, 2, 1–15.
- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Knowledge Discovery and Data Mining* (Vol. 96, pp. 226–231).
- Fahland, D., & Van Der Aalst, W. M. (2013). Simplifying Discovered Process Models in a Controlled Manner. *Information Systems*, 38(4), 585–605.
- Fernandez, F. M. H., & Ponnusamy, R. (2016). Data Preprocessing and Cleansing in Web Log on Ontology for Enhanced Decision Making. *Indian Journal of Science and Technology*, 9(10), 1–10.
- Gómez-Carmona, O., Buján-Carballal, D., Casado-Mansilla, D., López-de Ipiña, D., Cano-Benito, J., Cimmino, A., ... others (2023). Mind the gap: The AURORAL ecosystem for the digital transformation of smart communities and rural areas. *Technology in Society*, 74, 102304.
- Gupta, G. K. (2014). *Introduction to data mining with case studies*. PHI Learning Pvt. Ltd.
- Hansson, M., Angel, K., Mannhardt, F., & Kvale, K. (2021). How Can a Service Provider Utilize Process Mining on Customer Journeys to Gain Actionable Insights for Service Delivery Improvements? In M. de Leoni, M. Song, & M. Röglinger (Eds.), *Proceedings of the Industry Forum at BPM 2021 co-located with 19th International Conference on Business Process Management (BPM 2021), Rome, Italy, September 6-10, 2021* (Vol. 3112, pp. 15–26). CEUR-WS.org. Retrieved from <https://ceur-ws.org/Vol-3112/paper3.pdf>
- Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems: Theory and Practice*. Springer.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MisQuarterly. MISQ Discovery*, 28(1).
- Ingvaldsen, J. E., & Gulla, J. A. (2012). Industrial application of semantic process mining. *Enterprise Information Systems*, 6(2), 139–163.
- Jareevongpiboon, W., & Janecek, P. (2013). Ontological approach to enhance results of business process mining and analysis. *Business Process Management Journal*.
- Jenks, G. F. (1967). The Data Model Concept in Statistical Mapping. *International Yearbook of Cartography*, 7, 186–190.
- Jiang, B. (2013). Head/tail Breaks: A New Classification Scheme for Data with a Heavy-tailed Distribution. *The Professional Geographer*, 65(3), 482–494.
- Koschmider, A., & Oberweis, A. (2005). Ontology Based Business Process Description. In *EMOI-INTEROP* (pp. 321–333).
- Kuhlthau, C. C. (1991). Inside the Search Process: Information Seeking from the User’s Perspective. *Journal of the American society for information science*, 42(5), 361–371.
- Lemon, K. N., & Verhoef, P. C. (2016). Understanding Customer Experience Throughout the Customer Journey. *Journal of Marketing*, 80(6), 69–96.
- Lund, B. D., & Wang, T. (2023). Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library Hi Tech News*, 40(3), 26–29.
- Lux, M., & Rinderle-Ma, S. (2017). Problems and Challenges When Implementing a Best Practice Approach for Process Mining in a Tourist Information System. In M. Brambilla & T. T. Hildebrandt (Eds.), *Proceedings of the BPM 2017 Industry Track co-located with the 15th International Conference on Business Process Management (BPM 2017), Barcelona, Spain, September*

- 10-15, 2017 (Vol. 1985, pp. 1–12). CEUR-WS.org. Retrieved from <https://ceur-ws.org/Vol-1985/BPM17industry01.pdf>
- Lux, M., & Rinderle-Ma, S. (2019). Analyzing User Behavior in Search Process Models. In C. Capiello & M. Ruiz (Eds.), *Information Systems Engineering in Responsible Information Systems - CAiSE Forum 2019, Rome, Italy, June 3-7, 2019, Proceedings* (Vol. 350, pp. 182–193). Springer. Retrieved from https://doi.org/10.1007/978-3-030-21297-1_16 doi: 10.1007/978-3-030-21297-1_16
- Lux, M., Rinderle-Ma, S., & Preda, A. (2018). Assessing the Quality of Search Process Models. In M. Weske, M. Montali, I. Weber, & J. vom Brocke (Eds.), *Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings* (Vol. 11080, pp. 445–461). Springer. Retrieved from https://doi.org/10.1007/978-3-319-98648-7_26 doi: 10.1007/978-3-319-98648-7_26
- Lux, M., & Rinderle-Ma, S. (2023). DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling. *Journal of Classification*, 1–39.
- Ly, L. T., Indiono, C., Mangler, J., & Rinderle-Ma, S. (2012). Data Transformation and Semantic Log Purging for Process Mining. In *Advanced Information Systems Engineering: 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings 24* (pp. 238–253).
- Maechler, N., Neher, K., & Park, R. (2016). From touchpoints to journeys: Seeing the world as customers do. *McKinsey & Company*, 1–10.
- Mendling, J., & Strembeck, M. (2008). Influence Factors of Understanding Business Process Models. In *Business Information Systems: 11th International Conference, BIS 2008, Innsbruck, Austria, May 5-7, 2008. Proceedings 11* (pp. 142–153).
- Nabli, A., Bouaziz, S., Yangui, R., & Gargouri, F. (2015). Two-ETL phases for data warehouse creation: Design and implementation. In *Advances in Databases and Information Systems: 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings 19* (pp. 138–150).
- Omar, R., Mangukiya, O., Kalnis, P., & Mansour, E. (2023). ChatGPT versus Traditional Question Answering for Knowledge Graphs: Current Status and Future Directions Towards Knowledge Graph Chatbots. *arXiv preprint arXiv:2302.06466*.
- Pedrinaci, C., Domingue, J., & Alves de Medeiros, A. K. (2008). A Core Ontology for Business Process Analysis. In *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings 5* (pp. 49–64).
- Reynolds, D. A. (2009). Gaussian Mixture Models. *Encyclopedia of Biometrics*, 659–663.
- Richardson, A. (2010). Using Customer Journey Maps to Improve Customer Experience. *Harvard business review*, 15(1), 2–5.
- Rozinat, A., & Van der Aalst, W. M. (2008). Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1), 64–95.
- Sanji, M., Behzadi, H., & Gomroki, G. (2022). Chatbot: an intelligent tool for libraries. *Library Hi Tech News*, 39(3), 17–20.
- Scott, D. W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.
- Silverstein, C., Marais, H., Henzinger, M., & Moricz, M. (1999). Analysis of a Very Large Web Search Engine Query Log. In *ACM SIGIR Forum* (Vol. 33, pp. 6–12).
- Stolba, N. (2007). *Towards a sustainable data warehouse approach for evidence-based healthcare* (Ph.d. thesis). TU Wien.
- Sugumaran, V., & Gulla, J. A. (2011). *Applied Semantic Web Technologies*. CRC Press.
- Sun, W., Yan, L., Ma, X., Ren, P., Yin, D., & Ren, Z. (2023). Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv preprint arXiv:2304.09542*.
- Syamsiyah, A., van Dongen, B. F., & van der Aalst, W. M. (2018). DB-XES: Enabling Process Discovery in the Large. In *Data-Driven Process Discovery and Analysis: 6th IFIP WG 2.6 In-*

- ternational Symposium, SIMPDA 2016, Graz, Austria, December 15-16, 2016, Revised Selected Papers 6* (pp. 53–77).
- Syed, R., Leemans, S. J., Eden, R., & Buijs, J. A. (2020). Process Mining Adoption: A Technology Continuity versus Discontinuity Perspective. In *Business Process Management Forum: BPM Forum 2020, Seville, Spain, September 13–18, 2020, Proceedings 18* (pp. 229–245).
- Terragni, A., & Hassani, M. (2019). Optimizing Customer Journey Using Process Mining and Sequence-Aware Recommendation. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (pp. 57–65).
- Thomas, O., & Fellmann MA, M. (2009). Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. *Business & Information Systems Engineering, 1*, 438–451.
- Tobler, W. R. (1973). Choropleth maps without class intervals. *Geographical analysis, 5*(3), 262–265.
- Van Der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (Vol. 2). Springer.
- Van Der Aalst, W. (2016). *Process Mining: Data Science in Action* (Vol. 2). Berlin, Heidelberg: Springer.
- Van Der Aalst, W., Adriansyah, A., De Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., ... others (2012). Process Mining Manifesto. In *Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9* (pp. 169–194).
- Vanderfeesten, I., Reijers, H. A., Mendling, J., van der Aalst, W. M., & Cardoso, J. (2008). On a Quest for Good Process Models: The Cross-Connectivity Metric. In *Advanced Information Systems Engineering: 20th International Conference, CAiSE 2008 Montpellier, France, June 16-20, 2008 Proceedings 20* (pp. 480–494).
- Vanderfeesten, I., Reijers, H. A., & Van der Aalst, W. M. (2008). Evaluating Workflow Process Designs using Cohesion and Coupling Metrics. *Computers in Industry, 59*(5), 420–437.
- Weijls, D., & Caron, E. (2022). Customer Journey Analytics: A Model for Creating Diagnostic Insights with Process Mining. In *International Conference on Software Technologies* (pp. 418–424).
- Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Springer.