Measuring and Resource-Efficient Optimization of Clustering Quality

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r I n g e n i e u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Lars Filipp Lenssen

Dortmund

2025

Tag der mündlichen Prüfung:	18. März 2025
Dekan:	Prof. Dr. Jens Teubner
Gutachter:	Prof. Dr. Erich Schubert
	Prof. Dr. Arthur Zimek

Acknowledgments

This thesis reflects a challenging and transformative journey. This would not have been possible without the support and understanding of many people, to whom I am deeply grateful. First and foremost, I am immensely grateful to my supervisor, Prof. Dr. Erich Schubert. His expertise and incredible support have been essential to the completion of this work. Balancing my responsibilities as a father of two children has always been a priority for me, and his patience and flexibility allowed me to fulfill this role with the seriousness it deserves. I am also indebted to my committee members, Prof. Dr. Arthur Zimek, Prof. Dr. Christian Janiesch, and Prof. Dr. Mario Botsch for their valuable advice and constructive critiques.

I am also grateful to my colleagues from the Data Mining Group at TU Dortmund for the many inspiring discussions and valuable insights. I would especially like to thank Erik Thordsen and Andreas Lang. Even during challenging COVID times, I was able to engage in a productive and rewarding exchange. I am also very thankful to Philip Stahmann for our collaboration, especially during the final year of my work, which significantly broadened the applicability of my research. I also extend my thanks to Niklas Strahmann for our collaboration on his bachelor thesis and the resulting publications. Furthermore, I would like to thank the Collaborative Research Center 876 at TU Dortmund and the German Research Foundation for partially funding my research.

My early career was shaped by my work as an IT consultant under Dr. Uwe Strahmann, whose mentorship gave me valuable experience in the German energy market and inspired my academic path. Alongside my fulfilling research, I have continued my consulting work. I thank Gernot Conrad for his understanding of my commitments as a father and researcher, never placing undue demands on my availability. I also thank my friends Marvin, Tim, and Robin for their constant support, especially in challenging times. Special thanks to Max, who shared the journey from fifth grade through our Master's and assisted with proofreading this work.

Finally, my deepest gratitude goes to my family. I am especially grateful to my brother, Jan Eric, who has not only supported me by proofreading but has also been a continual source of inspiration, motivating me to pursue my goals. Enormous gratitude goes to my parents, Brigitte and Roland, whose support and love have been the emotional foundation of this entire journey. Their belief in me has been a steady source of strength, making every challenge beatable. Finally, to Wanda, who has given me our wonderful children, Lino and Leif, and who has always shown understanding and support for my professional and academic ambitions—thank you.

—This thesis is dedicated to Lino and Leif, who bring joy and inspiration to my life.—

Abstract

Cluster analysis is a fundamental task in exploratory data mining, widely used to uncover hidden structures within datasets across various fields. It has broad applications, from identifying subgroups in gene expression data for disease research to segmenting customer bases in industry. Over time, a diverse range of clustering methods has been developed to handle the complex structure of different data domains. Despite these, key challenges remain, particularly in evaluating the quality of clustering results and optimizing the performance of clustering algorithms.

This thesis presents significant advancements in clustering evaluation measures and the development of resource-efficient clustering algorithms, with a focus on both theoretical contributions and practical applications. Cluster analysis focuses on identifying groups of similar samples within a dataset, with three central challenges: determining the number of clusters, finding the optimal clustering solution, and ensuring resource efficiency. The research presents the Average Medoid Silhouette (AMS), a robust approximating variant of the Average Silhouette Width (ASW), which has been demonstrated to improve clustering quality in real-world scenarios such as gene expression analysis. A novel algorithm was designed to optimize AMS, achieving computational improvements, thus enabling medoid-based clustering for larger datasets. Additionally, the DynMSC algorithm was developed to simplify the process of identifying the optimal number of clusters.

For categorical data, two new measures—Average Relative Entropy Score (ARES) and Minimum Relative Entropy Contrast (MREC)—were introduced. These measures formed the basis for the CatRED algorithm, an agglomerative hierarchical clustering algorithm. CatRED has been successfully applied in domains like smart services and business analytics, providing interpretable and efficient results.

Improvements to existing clustering algorithms are also presented, such as enhancing the k-means-CP algorithm by reducing redundant distance computations through the parallel axis theorem. Additionally, practical experiments in energy grid simulations demonstrate the effectiveness of optimizing the FasterPAM approach for sparse problems.

The thesis emphasizes the intrinsic link between clustering algorithms and evaluation methods, clearly demonstrating how optimization strategies can significantly enhance both resource efficiency and clustering quality. Contributions include the development of scalable algorithms like FasterMSC and NCK-means, which are specifically designed to effectively handle much larger datasets. On the other hand, the thesis also presents problem-specific algorithms, such as CatRED, which focuses on applications involving categorical data.

Contents Overview

Contents

Chapter 1

Introduction

Cluster analysis is a critical technique in data science and machine learning, providing invaluable insights across various disciplines by identifying patterns and grouping similar data points. This unsupervised learning method has been instrumental in fields ranging from biology to business analytics, allowing researchers to make sense of complex data sets and uncover hidden structures. In cluster analysis, the user is interested in discovering previously unknown structure in the data, as opposed to classification, where one tries to predict the known structure (i.e., labels) for new data points.

In single-cell transcriptomics, it helps in grouping cells with similar transcriptomic profiles to identify distinct cell subpopulations and states [KKT+15; KMA+15]. In the realm of smart services, cluster analysis is essential for creating structured classifications [FHH+20]. Cluster analysis is also used to classify business analytics solutions, aiding in optimizing manufacturing processes by highlighting the strengths and weaknesses of different approaches [WWW+23]. Moreover, in the generation of distribution grid models on the basis of publicly available data [KSS+17], cluster analysis was employed to generate accurate distribution grid models from public data. This application is crucial for planning and optimizing electrical distribution networks, particularly for integrating renewable energy sources and improving grid resilience.

Sometimes, clustering can also be interpreted as data quantization and approximation, for example, in k-means, where the objective is to minimize the sum of squared errors when approximating the data with k average vectors, spherical k-means, where we maximize the cosine similarities to the k centers, and k-medoids, where we minimize the sum of distances when approximating the data by k data points. Other clustering approaches such as DBSCAN [EKS+96; SSE+17] cannot easily be interpreted this way, but discover structure related to connected components and density-based minimal spanning trees [SHM18; BDH+23].

The evaluation of clusterings is a challenge, as there are no labels available. While many internal ("unsupervised", not relying on external labels) evaluation measures were proposed such as the Silhouette [Rou87], the Davies-Bouldin index [DB79], the Variance-Ratio criterion [CH74], the Dunn index [Dun74], and many more, using these indexes for evaluation suffers from inherent challenges. Vendramin, Campello, and Hruschka [VCH10] survey 40 such measures and variants, and find the Silhouette to be one of the most robust. Jaskowiak et al. [JMF+16] combine different internal validation measures in an ensemble to improve performance. Bonner [Bon64] noted that "none of the many specific definitions [...] seems best in any general sense", and results are subjective "in the eye of the beholder" as noted by Estivill-Castro [Est02]. While these claims refer to clustering methods, not evaluation methods, we argue that these statements are valid for evaluation in a similar way; each internal cluster evaluation method implies a clustering algorithm obtained by enumeration of all candidate clusterings, keeping the best. The main difference between clustering algorithms and internal evaluation then is whether or not we know an efficient optimization strategy. k-means is an optimization strategy for the sum of squares evaluation measure, while the k-medoids algorithms Partitioning Around Medoids (PAM), and alternating optimization [Mar63] are two different strategies for optimizing the sum of distances from a set of k representatives chosen from the data, a variant of the facility location problem. One central problem in existing work is that often distances, evaluation measures, and clustering algorithms are developed independently of one another, instead of considering them to be different sides of the same problem. We will develop an overarching view of cluster quality that directly relates to a cluster similarity measures.

This thesis critically evaluates different internal cluster quality measures, providing a comprehensive analysis of their effectiveness and applicability in various contexts. Second, the thesis focuses on the optimization of clustering algorithms, specifically aimed at directly optimizing these internal clustering evaluation measures.

1.1 Contribution of this Work

The thesis contributions are structured into three different parts: internal evaluation measures, clustering algorithms, and the relationship between them. The structure of contributions in the chapters of this thesis is summarized in Figure 1.1. Chapter 4 introduces a variant of the PAM algorithm that includes the enhancements from FasterPAM, but is also specifically optimized for handling sparse input data. One central contribution of this thesis focuses on the development and optimization of clustering algorithms that directly optimize internal clustering evaluation measures while ensuring resource-efficiency. In Chapter 5, we introduce the Medoid Silhouette, along with its corresponding clustering algorithm, FasterMSC. In Chapter 6, we discuss kNN Consistency and optimization of the clustering algorithm k-means-CP. Chapter 7 presents the Average Relative Entropy Score (ARES), accompanied by a new hierarchical clustering method, CatRED.

In addition, this thesis provides a methodological analysis of the underlying clustering evaluation measures and clustering algorithms. We provide run time analyses for PAM in Chapter 3.1, to substantiate the efficiency and correctness of the algorithm. In Section 5.1, we present an axiomatic characterization of the Medoid Silhouette, affirming its validity as a Clustering-Quality Measure (CQM) according to the axiomatic approach of Ben-David and Ackerman [BA08]. Theoretical

Internal Evaluation Measures

Relative Entropy Score

$$\begin{aligned}\mathrm{RES}(X,D) &= \sum_A \sum_{i\in A} \mathrm{p}_{A_i}(X) \log \frac{\mathrm{p}_{A_i}(X)}{\mathrm{p}_{A_i}(D)} \\ &= \sum\nolimits_A \mathrm{D}_{\mathrm{KL}}(\mathrm{P}_A(X) \parallel \mathrm{P}_A(D)) \\ &= \sum\nolimits_A \left(H(\mathrm{P}_A(X), \mathrm{P}_A(D)) - H(\mathrm{P}_A(X))\right) \\ &= \sum\nolimits_A H(\mathrm{P}_A(X), \mathrm{P}_A(D)) - \sum\nolimits_A H(\mathrm{P}_A(X))\end{aligned}$$

$$\mathrm{ARES}(C,D) = \sum\nolimits_{i=1}^k \tfrac{|C_i|}{N} \, \mathrm{RES}(C_i, D)$$

Section 7.1

Medoid Silhouette

$$s'_i(X,d,M) = \frac{b'_i - a'_i}{\max(a'_i, b'_i)}\ , \text{ where}$$

$$a'_i = d(x_i, M_{l_i})$$

$$b'_i = \min\nolimits_{k \neq l_i} d(x_i, M_k)$$

$$\tilde{s}_i(X,d,M) = \frac{d_2(i) - d_1(i)}{d_2(i)} = 1 - \frac{d_1(i)}{d_2(i)}$$

$$\tilde{S}(X,d,M) = \tfrac{1}{n} \sum\nolimits_{i=1}^n \tilde{s}_i(X,d,M)$$

Section 3.2.1, Section 5.1

k-Nearest-Neighbor Consistency

$$kc_i(X,L) = \begin{cases} 1 & \text{if } \forall x_j \in k\,\mathrm{NN}(x_i) : l_j = l_i \\ 0 & \text{otherwise} \end{cases}\ , \text{ where}$$

$X = \{x_1, \dots, x_n\}$, the given samples

$L = \{l_1, \dots, l_n\}$, the cluster labels

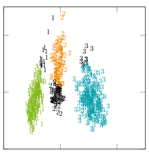
$$KC(X,L) = \frac{1}{n} \sum\nolimits_{i=1}^n kc_i(X,L)$$

Section 3.2.3, Section 6.1

Clustering Algorithms

FasterMSC

Optimize Medoid Silhouette

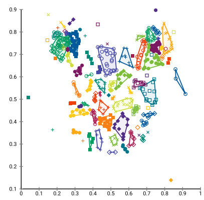


- Optimizing the Medoid Silhouette with PAM
- Integrating run time improvements from FasterPAM

Section 5.2

NCK-means

Fast kNN-Consistent Clustering

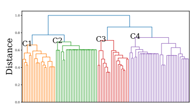


- Closed neighbor sets get assigned to the same cluster
- Performance optimized variant of k-means-CP

Section 6.2

CatRED

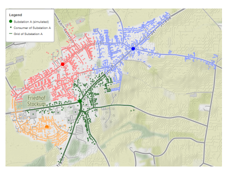
Clustering for Categorical Data



- Hierarchical Agglomerative Clustering for categorical data
- Merging smallest relative entropy-based distance

Section 7.2

Sparse Partitioning Around Medoids



- Optimizing the FasterPAM approach for sparse problems
- Automatically finding the parameter k as part of the optimization problem

Section 4.1

Figure 1.1: Section 3.2 describes the internal clustering evaluation measures discussed in this thesis. Section 5.1 details the axiomatic characterization of the Medoid Silhouette, while Section 7.1 introduces a novel measure based on relative entropy. The primary contributions of this thesis, including various clustering algorithms that optimize these evaluation measures, are presented in Sections 5.2 and 6.2, as well as in Section 7.2.

proofs and runtime analyses for Neighbor-Consistent K-means (NCK-means) are provided in Section 6.2.2, demonstrating the efficiency and correctness of the proposed algorithm.

Finally, we demonstrate the practical applicability of all the methods presented through comparative experiments. For the Medoid Silhouette and k-Nearest-Neighbor (kNN) Consistency, we conduct experiments using two Single-Cell RNA Sequencing (scRNA-seq) data sets, exploring their performance under various parameters [KKT+15; KMA+15]. We evaluate the ARES on taxonomy construction, which deals with the structuring and organization of knowledge [BMI11]. The development of the sparse variant of PAM was driven by a specific need in the simulation of electrical grids [KSS+17], and its performance is also evaluated in this context.

1.2 Organization of the Thesis

This thesis begins with a general introduction and motivation for the study of clustering in Chapter 1, highlighting the importance of evaluating clustering quality in the absence of labeled data. Chapter 2 presents fundamentals of cluster analysis, covering different heuristic optimization methods, an essential background of clustering evaluation, theoretic clustering axioms, and information-theoretic fundamentals. A comprehensive review of related work follows in Chapter 3. It begins with a detailed description of PAM. Next, it presents existing clustering evaluation measures and algorithms that directly optimize these measures, which are subsequently analyzed and further optimized later in the thesis. Additionally, relevant literature in the domain of clustering categorical data is critically examined. The first of the four main chapters follows. Chapter 4 presents an optimization of PAM, enabling efficient processing of sparse data. The subsequent Chapter 5 contains the characterization of the Medoid Silhouette and a new clustering algorithm to optimize it directly. Chapter 6 then presents kNN Consistency, along with an algorithm designed for its direct optimization. As the last central chapter, Chapter 7 addresses clustering categorical data, introducing a novel quality measure and a corresponding clustering algorithm that directly optimizes this measure.

1.3 Author's Publications and Contributions

The research described in this thesis was first published through a series of peer-reviewed publications. This section presents the specific methodologies presented in each publication and outlines the author's contributions to these respective works.

The sparse variant of PAM, discussed in Chapter 4, was published as part of the foundational book Machine Learning under Resource Constraints – Fundamentals 1 in 2023 [LS23]. The author of this thesis contributed the majority of the work, including the development of the ideas, implementation, evaluation, and writing. The analyses and clustering algorithms on the Medoid Silhouette, described in Chapter 5, were published at Similarity Search and Applications (SISAP) in 2022 [LS22b] and a journal version in Information Systems (IS) in 2024 [LS24] by the author of this thesis. The author contributed the majority of the work with respect to idea, proofs, realization, evaluation and writing. The analysis of the clustering evaluation measure, kNN Consistency, together with the optimized clustering algorithm NCK-means, as presented in Chapter 6, was published at the "Learning, Knowledge, Data, Analysis" (LWDA) conference in 2023 [LSS23]. The author of this thesis contributed the majority of the work, including the development of the idea, formal proofs, implementation, evaluation, and writing. The clustering evaluation measure MREC for categorical data, along with the associated clustering algorithm CatRED presented in Chapter 7, was published at the 58th Hawaii International Conference on System Sciences (HICSS) in 2025 [LSJ+25]. The author of this thesis contributed the majority of the work, including the development of the core idea, the implementation, the evaluation, and the writing.

Below is a chronological list of publications produced by the author during his PhD studies, which culminated in this thesis. For each publication, the author's contributions are detailed, along with its relevance and inclusion in this work.

[SL22] Erich Schubert and Lars Lenssen: *Fast k-medoids Clustering in Rust and Python*, Journal of Open Source Software 2022. **Contribution:** The Author contributed to realization, evaluation and writing. **Mentioned:** Section 3.1.1 and Section 5.3.1.

[LS22b] Lars Lenssen and Erich Schubert: *Clustering by Direct Optimization of the Medoid Silhouette*, Similarity Search and Applications 2022. **Contribution:** The author contributed the majority of the work with respect to idea, proofs, realization, evaluation, and writing. **Subject of:** Chapter 5.

[LS23] Lars Lenssen and Erich Schubert: *Sparse Partitioning Around Medoids*, Machine Learning under Resource Constraints – Fundamentals 1, 182-196, 2023: **Contribution:** The author contributed the majority of the work with respect to the ideas, realization, evaluation and writing. **Subject of:** Chapter 4.

[LSS23] Lars Lenssen, Niklas Strahmann and Erich Schubert: *Fast k-Nearest-Neighbor-Consistent Clustering*, Proceedings of the Conference "Learning, Knowledge, Data, Analysis" 2023: **Contribution:** The author contributed the majority of the work with respect to the realization, evaluation and writing. **Subject of:** Chapter 6.

[LS24] Lars Lenssen and Erich Schubert: *Medoid Silhouette clustering with automatic cluster number selection*, Information Systems 2024. **Contribution:** The author contributed the majority of the work with respect to idea, proofs, realization, evaluation and writing. **Subject of:** Chapter 5.

[LSJ+25] Lars Lenssen, Philip Stahmann, Christian Janiesch and Erich Schubert: *Archetype Discovery from Taxonomies: A Method to Cluster Small Datasets of Categorical Data*, Hawaii International Conference on System Sciences 2025, to appear. **Contribution:** The author contributed the majority of the work with respect to idea, realization, evaluation and writing. **Subject of:** Chapter 7.

Chapter 2

Cluster Analysis

Cluster analysis is an essential technique in data science and machine learning, offering significant insights across a wide range of disciplines by uncovering patterns and grouping similar data points. Unlike classification, which focuses on predicting known labels for new data points, cluster analysis aims to discover previously unknown structures within the data itself. Data objects within a single cluster share similarities based on specific criteria. Numerous criteria can be employed to form these clusters, which means that the same data set may yield different classifications depending on the chosen criteria. There are several techniques for cluster analysis, each categorized based on the fundamental methods used in the clustering process. Clustering algorithms that assign each object to a single, distinct cluster are known as partition-based clustering methods. Finding the optimal partition for a given loss function and set of objects is typically NP-hard. Thus, approximative algorithms that find a local optimum are often the best choice.

Another important approach is Hierarchical Agglomerative Clustering (HAC). Unlike partition-based methods, which divide the data into non-overlapping clusters in a single step, agglomerative clustering builds a hierarchy of nested clusters using a bottom-up approach. Initially, each data point is treated as its own cluster, and clusters are gradually merged based on their similarity, forming progressively larger clusters until only one cluster remains or a specified number of clusters is achieved.

This chapter provides an essential foundation in cluster analysis, which is critical for the contributions of this thesis. It begins with Section 2.1, an overview of various heuristic clustering optimization methods, including key algorithms relevant to the thesis. Following this, in Section 2.2 a detailed description of hierarchical agglomerative clustering is presented. The Section 2.3 then presents a general overview of the evaluation of clusterings. Next, theoretical axioms that offer the possibility to prove the suitability of clustering evaluation measures are introduced in Section 2.4. Finally, we summarize information-theoretic fundamentals in Section 2.5.

2.1 Optimization Methods

There are different heuristic optimization methods for clustering to optimize clustering objectives. Expectation-Maximization (EM) is a probabilistic algorithm that iteratively improves parameter estimates by maximizing the likelihood of the observed data, making it particularly effective in models with incomplete or latent data. In contrast, Local Search is an iterative technique that improves a solution by exploring its neighboring configurations to find a local optimum. Both approaches are integral to the clustering techniques explored in this thesis and are described in detail below.

Expectation-Maximization

In many machine learning and statistical models, estimating parameters using Maximimum Likelihood (ML) or Maximum a Posteriori (MAP) methods is straightforward when all relevant data is observed. However, this task becomes substantially more difficult when data is missing or certain variables are latent. One common approach is to use a gradient-based optimizer to minimize the Negative Log-Likelihood (NLL).

While this approach works, it often requires enforcing complex constraints (e.g. ensuring covariance matrices are positive definite or handling the normalization in von Mises-Fisher distributions), which can complicate optimization. A more convenient, though not always faster, alternative is the EM algorithm, a simple iterative procedure with closed-form updates that automatically enforces necessary constraints.

The key idea of EM [Mur12, Ch 11.4] is to exploit the fact that if all data were fully observed, the ML or MAP estimate would be easy to compute. The algorithm alternates between two steps:

1. **Expectation (E-step):** Expect the missing or latent values given the model parameters θ and the data.

2. **Maximization (M-step):** Update the model parameters θ based on the observed data, optimizing the likelihood function.

EM iteratively refines the parameters by alternating between these steps, making it particularly effective for models with incomplete data. Each iteration of EM guarantees either an increase in the log-likelihood (or posterior) or stays the current value, which ensures convergence. The EM algorithm is frequently regarded as synonymous with Gaussian Mixture Modeling (GMM) [DLR77]. However, it is important to note that EM is an underlying principle that can be applied to various statistical models beyond just GMM. Therefore, it is important to make this distinction in the context of this thesis.

k-means

The k-means algorithm is a widely used partition-based clustering method. The underlying principle of the standard algorithm for k-means is the EM algorithm [Mur12, Ch 11.4]. It can be described as follows:

1. **Initialization:** Choose k cluster centers randomly to initialize the model parameters θ.

2. **E-step:** Expect latent variables (cluster labels) based on the current cluster centers.

3. **M-step:** Update the cluster centers by calculating the maximum-likelihood estimation of centrality.

4. **Iteration:** Repeat steps 2 and 3 until a stopping condition is met, such as no changes in cluster assignments or the centroids.

The objective of the k-means algorithm is to minimize the sum of squares:

$$SSQ = \sum_C \sum_d \sum_{x_i \in C} (x_{i,d} - \mu_{C,d})^2$$

with clusters C, the dimension d, and the cluster centers $\mu_{C,d}$. The sum of squares is equivalent to the squared Euclidean distance:

$$\sum_d (x_{i,d} - \mu_{C,d})^2 = \|x_i - \mu_C\|^2 \ .$$

This relationship indicates that each data point is assigned to the "closest" cluster center. However, it is important to note that the k-means algorithm is limited to using the Euclidean distance. This restriction arises from the fact that the arithmetic mean only minimizes the sum of squares.

An important theorem is the parallel axis theorem of König, Huygens and Steiner:

$$\sum_X |x - a|^2 = \sum_X |x - \mu|^2 + N|a - \mu|^2 \ .$$

In k-means, the centroids (μ_C) can be viewed as the "center of mass" of the points within each cluster. Since the arithmetic mean is the optimal estimator for minimizing variance, updating the centroids corresponds to recalculating the center of mass based on the current assignments of points to clusters. This is analogous to adjusting the axis in the Parallel Axis Theorem, where the mass distribution around the centroid dictates how the inertia (or variance, in the case of clustering) is minimized.

Local Search

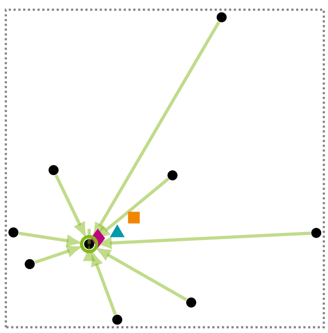
Local search is a heuristic optimization technique that begins with an initial solution and iteratively explores neighboring solutions to improve an objective function. Unlike EM, which relies on probabilistic frameworks and works in two distinct steps (Expectation and Maximization), local search directly seeks to refine solutions through local modifications, making it more straightforward in its approach. The local search process can be envisioned as a descent through a graph representing the search space, where each vertex corresponds to a feasible solution and edges connect neighboring solutions [Tov97]. The algorithm evaluates the neighbors of the current solution and selects the one that provides the best improvement according to the specified objective function. This exploration continues until no neighboring solution yields a better outcome, at which point the search concludes, having reached a local optimum.

k-medoids is a popular example of a problem where local search algorithms are useful. The name was introduced by Leonard Kaufman and Peter J. Rousseeuw in connection with their PAM algorithm [KR90; KR87]. Similar to k-means, it aims at minimizing the sum of distances from a cluster center, but the cluster center in k-medoids is one of the data points and called a medoid, and the distance function here may be arbitrary. This increases the flexibility over k-means, which uses the arithmetic mean as cluster center: the mean minimizes squared errors, and because of this k-means only minimizes Bregman divergences such as the squared Euclidean distance. Even on one-dimensional data it does not minimize the linear error, as easily seen from the difference between the arithmetic mean and the median. While k-means minimizes the *sum-of-squared errors*, k-medoids, with k representative medoids m_i minimizes the absolute error criterion:

$$\mathrm{TD} := \sum\nolimits_{i=1}^{k} \sum\nolimits_{x_c \in C_i} d(x_c, m_i) \tag{2.1}$$

where $d(x_c, m_i)$ is the distance between data point x_c of cluster C_i and medoid m_i; not necessarily the Euclidean distance, and not necessarily a metric. The difference between the arithmetic mean, the per-axis median, the geometric median and the medoid of a data set is exemplified in Figure 2.1. It can be seen that the medoid is less sensitive to outliers than the arithmetic mean, and also that k-means does not minimize Euclidean distances (but the squared distances).

It is important to note that, like k-means, the k-medoids problem is NP-hard [KH79]. Consequently, efficient algorithms are necessary to derive approximate solutions. One widely recognized algorithm for k-medoids clustering is PAM. PAM will be discussed in detail later in this thesis.



Euclidean (L_2), Manhattan (L_1) and squared Euclidean (L_2^2) distance sums to the different central points.

	L_2	L_2^2	L_1
■ arithmetic mean	6.909	**6.330**	8.674
▲ per-axis median	6.761	6.530	**8.267**
♦ geometric median	**6.712**	7.008	8.431
○ Euclidean medoid	6.726	7.391	8.526

Figure 2.1: Four different central points: the arithmetic mean ■, per-axis median ▲, geometric median ♦, and the Euclidean medoid ○.

2.2 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is one of the earliest clustering methods [Sne57; Sib73]. It presents a distinct approach to clustering compared to methods such as k-means and k-medoids. Unlike these partition-based clustering algorithms, which require the number of clusters to be predetermined, HAC operates without the need for this specification. Instead, it begins with the premise that each object is its own cluster. The algorithm incrementally merges the most similar clusters until only a single cluster remains. This characteristic allows for the generation of a hierarchical structure that can reveal relationships among data points at various levels of granularity, which is often visualized using a dendrogram. The fundamental steps of HAC are as follows:

1. **Initialization:** Start with each data point as a separate cluster.
2. **Cluster Merging:** Identify and merge the two most similar clusters.
3. **Iteration:** Repeat the merging process until only one cluster remains.
4. **Dendrogram Visualization:** Construct a dendrogram to illustrate the hierarchical structure of clusters, enabling the selection of meaningful subclusters.

HAC includes various methodologies that differ based on the distance or similarity measures used for objects, as well as the metrics employed to assess distances between clusters after merging. Common distance measures include single-linkage, which focuses on the minimum distance between points in different clusters; complete-linkage, which looks at the maximum distance; average-linkage, which considers the average distance between all pairs of points from each cluster; and centroid-linkage, which calculates the distance between the centroids of the clusters.

2.3 Evaluation of Clusterings

After applying clustering algorithms to a data set, evaluating the quality of the resulting clusters is essential. This evaluation can be approached in several ways, each serving a different purpose.

Internal evaluation emphasizes the intrinsic characteristics of clusters, typically utilizing metrics based on distance or deviation. In contrast, external evaluation compares clustering outcomes with known labels, enabling an assessment of clustering accuracy. Additionally, indirect evaluation considers the impact of clustering on subsequent tasks, especially in supervised learning, to determine how clustering can improve classification performance. Lastly, expert evaluation involves subjective assessments by domain specialists, offering contextually relevant insights.

Internal Evaluation

Internal evaluation assesses the quality of clustering by analyzing the structure of the clusters themselves, without reference to external labels. The basic idea behind internal evaluation is that data points within the same cluster should be more similar to each other than to points in different clusters, typically measured by distances between points and cluster centers.

Several approaches exist to measure this internal cohesion and separation. For example for N data points, the Mean Squared Distance (MSD)

$$MSD(C) = \frac{1}{N} \sum_{C_i} \sum_{x=C_i} \|(x - \mu_{C_i})\|^2$$

calculates the average distance between data points and their cluster centroids. To combine this with the distance to other clusters one can calculate the Total Sum of Squares (TSS) for a data set X with N data points:

$$TSS = \sum_{x \in X} (x - \overline{X})^2 = \frac{1}{2N} \sum_{x,y \in X} (x - y)^2.$$

TSS can be devided into two components: the Within-Cluster Sum of Squares (WCSS) and the Between-Cluster Sum of Squares (BCSS). WCSS evaluates the compactness of clusters by measuring how closely data points are grouped within each cluster, while BCSS assesses the separation between clusters, indicating how distinct one cluster is from another:

$$\begin{aligned} WCSS &= \frac{1}{2N} \sum_{C_i} \sum_{x,y \in C_i} (x - y)^2. \\ BCSS &= \frac{1}{2N} \sum_{C_i \neq C_j} \sum_{x \in C_i, y \in C_j} (x - y)^2. \\ TSS &= WCSS + BCSS. \end{aligned}$$

TSS is a constant value for a given data set.

The goal is to minimize WCSS (or, equivalently, maximize BCSS), leading to higher overall cohesion within clusters and better separation between clusters. One common metric for this is the Explained Variance (R^2), which measures the proportion of variance explained by the clustering:

$$R^2 = BCSS/TSS \in [0, 1].$$

Internal evaluation is commonly used to determine the optimal number of clusters in unlabeled data sets. Later in this thesis, we will explore alternative approaches that are better suited for identifying the optimal number of clusters. Additionally, the following chapter on related work will detail the internal evaluation measures relevant to this thesis.

External Evaluation

External evaluation in clustering refers to the process of assessing the quality of a clustering result by comparing it to a known ground truth, typically in the form of pre-existing labels. Unlike internal evaluation, which assesses clusters based on their internal characteristics (e.g., compactness or separation), external evaluation measures the extent to which the clustering output matches the known class labels.

One of the main challenges in external evaluation is the matching problem, where clusters are often assigned arbitrary labels, while true classes have meaningful, predefined labels. Determining which cluster corresponds to which class is non-trivial because clustering is not a classification task, and the number of clusters may not match the number of classes. Additionally, clusters can contain instances from multiple classes, or classes may be split across several clusters.

Despite these challenges, external evaluation is widely used to compare the performance of clustering algorithms, especially when labeled data is available. It provides a way to objectively measure how well the algorithm identifies meaningful patterns by aligning the clusters with known classes. This type of evaluation is essential when there are labeled subsets of the data or when specific properties can be treated as labels, as it helps in determining which clustering solution offers the best agreement with the reference classification.

Two popular measures for external evaluation are Normalized Mutual Information (NMI) [NEB10] and Adjusted Rand Index (ARI) [HA85], both of which quantify the agreement between the clustering result and the true class labels.

NMI is based on the concept of mutual information from information theory. It measures the amount of shared information between the clustering and the true labels. NMI is normalized, ensuring that its values range between 0 and 1, where 1 indicates perfect agreement between the clusters and the true classes, and 0 means that the clustering result is no better than random. NMI accounts for both the quality of the clusters and the distribution of the clusters across the true classes, making it a robust measure when comparing different clusterings.

ARI, on the other hand, is based on counting the pairs of data points that are either correctly clustered together or correctly placed in different clusters with respect to the true class labels. Unlike the standard Rand Index, ARI adjusts for chance by taking into account the likelihood of obtaining the same result by random chance. ARI ranges from -1 to 1, with 1 indicating perfect agreement, 0 corresponding to random clustering, and negative values suggesting worse than random results.

Both NMI and ARI are widely adopted in practice for their ability to capture different aspects of clustering quality, and are especially useful in scenarios where direct comparison with labeled data is necessary. These measures offer a clear way to assess how well a clustering algorithm uncovers the underlying structure of a data set when a ground truth is available.

2.4 Axioms for Clustering

We present the well-known Axioms for Clustering-Quality Measures (CQMs) by Ackermann and Ben-David [BA08]. They offer the possibility to prove the suitability of a clustering evaluation measure on the basis of theoretical axioms. Ackerman and Ben-David build upon Kleinberg's work [Kle02], which demonstrated the impossibility of axiomatizing clustering. Kleinberg [Kle02] underscores the importance of developing a clustering theory that is "independent of any particular algorithm, objective function, or generative data model". As a crucial step in this direction, Kleinberg formulates a series of axioms aimed at defining the essence of a clustering function. Despite the intuitive plausibility of these axioms, he demonstrates that they lead to a contradiction – no single function can satisfy all three requirements simultaneously.

Kleinberg's Axioms

Kleinberg [Kle02] introduces three axioms for clustering functions. These axioms aim to delineate which functions, operating on a domain set equipped with a distance function, qualify as clustering functions. Kleinberg demonstrates the inconsistency of this set – there are no functions that satisfy all three axioms. The first two axioms necessitate the clustering's invariance under certain alterations of the input distance function.

Axiom 1 (Function Scale Invariance) *A clustering function f satisfies* scale invariance *if for every distance function d, and every positive λ, $f(d) = f(\lambda d)$.*

Meaning that the output of a clustering function must be invariant to uniform scaling of the input to be scale-invariant.

Definition 1 (Relation $\sim_C$) *For given data points $X = \{x_1, \dots, x_n\}$ with a clustering C and a distance function d, we write $x_i \sim_C x_{i'}$ whenever x_i and $x_{i'}$ are in the same cluster of clustering C, otherwise $x_i \not\sim_C x_{i'}$.*

Definition 2 (C-consistent) *Clustering function d' is a* C-consistent *variant of d, if $d'(x_i, x_{i'}) \leq d(x_i, x_{i'})$ for $x_i \sim_C x_{i'}$, and $d'(x_i, x_{i'}) \geq d(x_i, x_{i'})$ for $x_i \not\sim_C x_{i'}$.*

Axiom 2 (Function Consistency) *A clustering function f satisfies* consistency *if for given a clustering C over (X, d), whenever d' is an C-consistent variant of d, $f(d') = f(d)$.*

Meaning that if we reduce the distances within the same cluster and increase the distances between different clusters, the quality does not decrease.

Axiom 3 (Function Richness) *A clustering function f satisfies* richness *if for given a clustering C over (X, d), there exists a distance function d over X such that $C = f(d)$.*

Meaning that for every possible clustering, there exists a distance for which this solution is optimal.

The axioms are grounded in intuitive and easily understandable principles. Take, for example, the Consistency requirement. This principle suggests that if we bring similar points in a data set closer together while pushing dissimilar points farther apart, the quality of the clustering should improve. In other words, clusters should become more coherent, and our confidence in the clustering results should increase.

While this idea is straightforward when thinking about clustering intuitively, expressing it through clustering functions brings an additional layer of rigor and depth to the discussion. From this perspective, Consistency asserts that if we modify the distance function used to calculate similarities between data points—such as by making similar points closer or dissimilar points more distant—these changes should not lead to a new and different clustering solution. Instead, the best clustering for the data set should remain stable, ensuring that the evaluation of clustering quality is robust to such adjustments. This formalization helps ensure that the clustering process behaves logically and predictably in response to changes in the underlying distance or dissimilarity measures.

Clustering-Quality Measure

Ackermann and Ben-David [BA08] change the primitive defined by the axioms from Kleinberg, this revised formulation is not only consistent but also met by several natural clustering evaluation measures. Additionally, they extend the set of axioms by introducing another axiom necessary to exclude certain measures that should not be classified as Clustering-Quality Measure (CQM). Translating the Scale Invariance axiom into the terminology of CQMs is straightforward.

Axiom 4 (Scale Invariance) *A quality measure f satisfies* scale invariance *if for every clustering C over (X, d), and every positive λ, $f(X, d, C) = f(X, \lambda d, C)$.*

There is a change in the definition of consistency. Although it effectively encapsulates the intuition that consistent alterations to d should not degrade the quality of a given partition, it leaves room for the possibility that, due to such changes, some partitions may improve more than others.

Axiom 5 (Consistency) *A quality measure f satisfies* consistency *if for a for every clustering C over (X, d), whenever d' is an C-consistent variant of d, then $f(X, d', C) \geq f(X, d, C)$.*

Axiom 6 (Richness) *A quality measure f satisfies* richness *if for every clustering C over (X, d), there exists a distance function d over X such that $C = \arg\max_{C'} f(X, d, C')$.*

In the following axiom, Ackermann and Ben-David parallel Puzicha et al. [PHB00] permutation invariance objective function axiom, capturing the need for clustering to remain impartial to the individual identities of the clustered elements.

Definition 3 (Clustering Isomorphism) *Two clusterings C, C' over (X, d), are* isomorphic*, if there exists a distance-preserving isomorphism $\phi : X \to X$, such that for all $x_i, x_{i'} \in X$, $x_i \sim_C x_{i'}$ if and only if $\phi(x_i) \sim_{C'} \phi(x_{i'})$.*

Axiom 7 (Isomorphism Invariance) *A quality measure f is* isomorphism-invariant *if for every clustering C, C' over (X, d) with distance d over X where C and C' are isomorphic, $f(X, d, C) = f(X, d, C')$.*

Meaning that transformations (including data permutations) which preserve distances do not affect clustering quality.

Ackermann and Ben-David propose a shift from clustering functions to CQM as a means to address Kleinberg's impossibility result in clustering. They introduce a set of axioms for CQMs that encompass Kleinberg's principles while ensuring consistency.

2.5 Information-Theoretic Fundamentals

This section summarizes some concepts from information theory as found in many textbooks, which provides a robust framework for understanding and quantifying the relationships between categorical variables. These principles are essential in many domains, including machine learning, data science, and clustering, as they allow us to measure uncertainty, divergence, and dependency between variables in a precise and interpretable manner.

In the following subsections, we explore Entropy, Relative Entropy, and Mutual Information. These concepts form the theoretical foundation for evaluating and comparing clustering algorithms, especially when dealing with categorical data.

Entropy

Entropy [Mur12, Ch 2.8.1] is a fundamental concept in information theory that quantifies the uncertainty or randomness of information. It measures the expected amount of information contained in a message, providing insight into how much uncertainty is present in a probability distribution. In simpler terms, entropy reflects how unpredictable or disordered the outcomes of a random variable are. A higher entropy value indicates more randomness or uncertainty, while lower entropy suggests a more predictable system with less variation in possible outcomes. Mathematically, for a probability distribution p over discrete indexes i, the entropy $H(p)$ is defined as

$$H(p) := -\sum_i p_i \log p_i$$

and the cross-entropy of two distributions is

$$H(p,q) := -\sum\nolimits_i p_i \log q_i \ .$$

Note that $H(p,p) = H(p)$. By the rule of L'Hôpital, we use $p_i \cdot \log p_i = 0$ for $p_i = 0$. Since $p_i \leq 1$, $\log p_i \leq 0$ and the entropy is always non-negative, but the cross-entropy can become infinite if $q_i = 0$ and $p_i \neq 0$. If we use the logarithm with base 2, the entropy can be interpreted as average "bits" of information of p in a hypothetical encoding optimal for q.

Relative Entropy

The Kullback-Leibler divergence [Mur12, Ch 2.8.2], also known as Relative Entropy, is a fundamental concept in information theory and statistics. It is an asymmetric measure that quantifies how one probability distribution p diverges from a second, expected probability distribution q. Unlike standard distance metrics, KL divergence is not symmetric and does not satisfy the triangle inequality, which means it is not a true distance measure in the traditional mathematical sense. The KL divergence for discrete probability distributions p and q is formally defined as:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(p \parallel q) &:= \sum_{x \in X} p(x) \log \tfrac{p(x)}{q(x)} \\ &= -\sum_{x \in X} p(x) \log \tfrac{q(x)}{p(x)} \ . \end{aligned}$$

The Kullback-Leibler divergence is equivalent to the difference of the cross entropy and the entropy:

$$\mathrm{D}_{\mathrm{KL}}(p \parallel q) = H(p,q) - H(p).$$

Intuitively it can be interpreted as the relative entropy of p given q compared to not being conditioned on q.

Mutual Information

Mutual Information [Mur12, Ch 2.8.3] is a measure of the amount of information that "knowing" one variable provides about another. It is given by

$$I(X,Y) := \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \tfrac{p(x,y)}{p(x)p(y)} \ ,$$

where $p(x,y)$ is the joint probability function of X and Y, and $p(x)$ and $p(y)$ are the probabilities of X and Y respectively. Mutual information is the amount by which the joint distribution of X and Y diverges from what it would be if X and Y were independent, and mutual information-based measures are a popular choice in supervised evaluation of clusterings, as they can be used to compare the partitions induced by clusters and by labels.

Chapter 3

Related Work

In this chapter, we review relevant literature and foundational research related to the key topics of this thesis. Section 3.1 focuses on PAM, a method that approximates a solution to the k-medoids problem. Section 3.2 discusses clustering evaluation measures that are central to this thesis, including techniques for their direct optimization. Finally, Section 3.3 explores methodologies for clustering categorical data, addressing the challenges posed by non-numeric attributes and adaptations developed to effectively manage this data type.

3.1 Partitioning Around Medoids

The k-medoids problem is NP-hard [KH79], hence we have to resort to approximate solutions, using greedy and local optimization techniques. The PAM algorithm is such an approach: its initialization (known as BUILD) is a greedy approximation to the k-medoids problem, which afterward is refined using a local search (called SWAP). Greedy initialization chooses k times the point that reduces the error the most; local search then optimizes this solution by searching for the best way to swap one of the cluster centers with a non-center. While the name k-medoids resembles k-means, the standard PAM algorithm works differently from the standard k-means algorithm. A k-means like strategy of alternating optimization for k-medoids has been proposed several times [Mar63; HTF01; RRI+06; PJ09], but was shown to produce worse solutions than a swap-based approach such as PAM [TB68; RHR79; SR21].

To start with a strong initial clustering—rather than employing the random sampling strategy often used in k-means—BUILD selects k points that minimize the total deviation. Initially, it picks the point with the smallest distance to all other points and subsequently adds points that most reduce TD. BUILD is computationally intensive. The aim is to find a solid starting point to minimize the number of swaps. Future experiments will explore the potential of randomized initialization methods as alternatives. The second part of PAM, SWAP, evaluates all potential swaps between the current k medoids and non-medoids, resulting in $k(n-k)$ candidate swaps. If a swap reduces TD, it is executed, and the process continues until no further improvements can be made, achieving a local (but not necessarily global) optimum. The search space for medoids guarantees convergence through the steepest-descent method within a limited number of iterations. The pseudocode for this original SWAP is outlined in Algorithm 1.

Algorithm 1: PAM SWAP: Iterative Improvement

$\Delta TD \leftarrow$ TD of the initial solution M
repeat
  $(\Delta TD_*, M_*) \leftarrow$ (0,null)
  foreach $m_i \in M = \{m_1, \dots, m_k\}$ **do** `// medoids`
    foreach $x_j \notin \{m_1, \dots, m_k\}$ **do** `// non-meds`
      $(\Delta TD, M') \leftarrow (0, M \setminus \{m_i\} \cup \{x_j\})$
      foreach $x_o \in X = \{x_1, \dots, x_n\}$ **do**
        $\Delta TD \leftarrow \Delta TD + \Delta(X, d, M')$ `// Total Deviation`
      if $\Delta TD > \Delta TD_*$ **then** $(\Delta TD_*, M_*) \leftarrow (\Delta TD, M')$
  if $\Delta TD_* \geq \Delta TD$ **then break**
  $(\Delta TD, M) \leftarrow (\Delta TD_*, M_*)$ `// perform swap`
return $(\Delta TD, M)$

Both the greedy initialization as well as the local search require all pairwise distances stored in a distance matrix. Greedy initialization performs k iterations, each of cost $O(n^2)$ to find the best medoid to add. PAM swap evaluates $O(k(n-k))$ potential swaps, each with a reduced effort of $O(n-k)$ operations by computing only the change in the loss function. Hence each swap takes $O(k(n-k)^2)$ time to find, which already was an improvement over the naive approach in $O(k^2(n-k)^2)$. The resulting run time complexity of PAM is $O(kn^2 i)$, where i is the number of iterations until convergence for which little is known except that it usually is reasonably small, and likely has an unfavorably high worst case just as with k-means.

3.1.1 Optimized Variants

Schubert and Rousseeuw have recently proposed improved versions of PAM named FastPAM [SR19a] and FasterPAM [SR21], which provide a substantial speedup over PAM by eliminating the nested loop over the k medoids. By furthermore greedily performing the first swap that improves the loss (instead of the best swap) and random initialization, the run time complexity decreased to $O(n^2 i)$ with an empirically much smaller i (but supposedly a similar theoretical worst case).

Because both methods use each pairwise distance several times—and the method is in particular interesting to use with more complex and hence expensive distance function—it is prohibitive to not use it with a pairwise distance matrix. Hence both methods also require $O(n^2)$ memory. Another recently introduced optimized variant of PAM is BanditPAM [TZM+20]. It employs a bandit-based approach to make medoid selection more efficient. Rather than evaluating all possible medoid combinations, BanditPAM utilizes a Multi-Armed Bandit model to strategically select medoids that are likely to yield the best results. While BanditPAM claims "almost linear run time" [TZM+20], whereas FasterPAM has quadratic run time, BanditPAM appears to have substantial overhead, and a break-even point likely is beyond 500000 samples for MNIST (a size where the memory consumption of the distance matrix makes a stored-distances approach prohibitive to use) [SL22].

3.2 Internal Evaluation Measures

Internal clustering evaluation measures allow the evaluation of clustering quality in scenarios where labeled data is unavailable. Widely recognized internal evaluation measures are the Silhouette [Rou87], the Davies-Bouldin index [DB79], the Variance-Ratio criterion [CH74], and the Dunn index [Dun74]. These internal measures offer complementary perspectives on clustering quality, each emphasizing different aspects of cluster structure. While some focus on the cohesion and separation of individual clusters, others provide a holistic view of the overall clustering configuration. Selecting the appropriate evaluation measure often depends on the specific characteristics of the data and the clustering objectives. Additionally, combining multiple evaluation metrics can provide a more comprehensive assessment, mitigating the limitations inherent in any single measure.

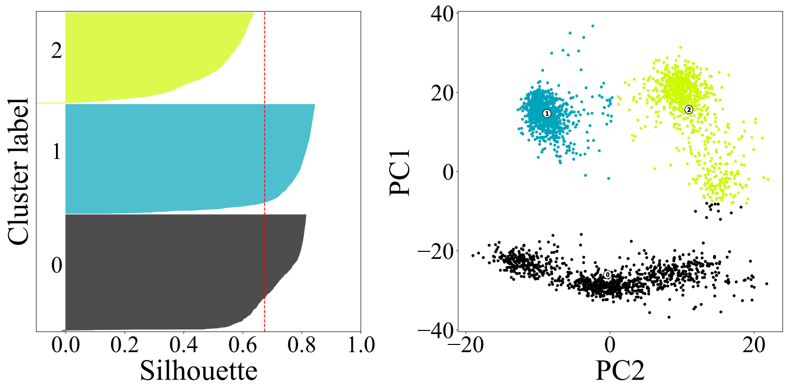
In the following sections, we examine specific internal evaluation measures critical to this thesis, including the Silhouette and Medoid Silhouette, Category Utility, and k-Nearest-Neighbor Consistency.

3.2.1 Silhouette and Medoid Silhouette

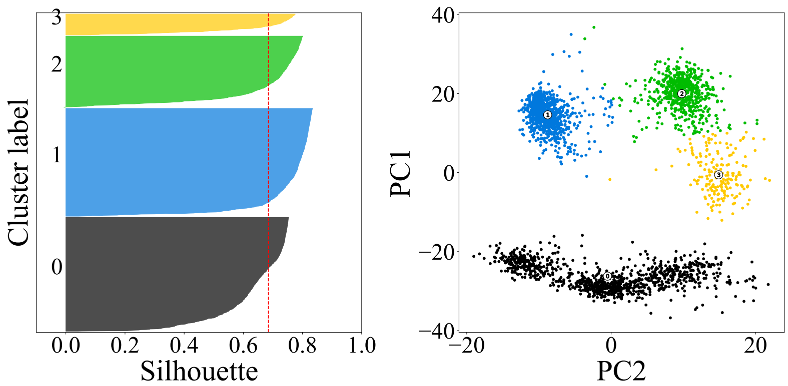
The Silhouette [Rou87] is a popular measure to evaluate clustering validity, and performs very well in empirical studies [AGM+13; BSH+07]. For the given samples $X = \{x_1, \ldots, x_n\}$, a dissimilarity measure $d : X \times X \to \mathbb{R}$, and the cluster labels $L = \{l_1, \ldots, l_n\}$ for each sample in X, the Silhouette of a single element i is calculated based on the average distance to its own cluster a_i and the smallest average distance to another cluster b_i as:

$$
\begin{aligned}
s_i(X, d, L) &= \tfrac{b_i - a_i}{\max(a_i, b_i)} \text{ , where} \\
a_i &= \phantom{\min_{k \neq l_i}} \operatorname{mean}\{d(x_i, x_j) \mid l_j = l_i, i \neq j\} \\
b_i &= \min\nolimits_{k \neq l_i} \operatorname{mean}\{d(x_i, x_j) \mid l_j = k\} \ .
\end{aligned}
$$

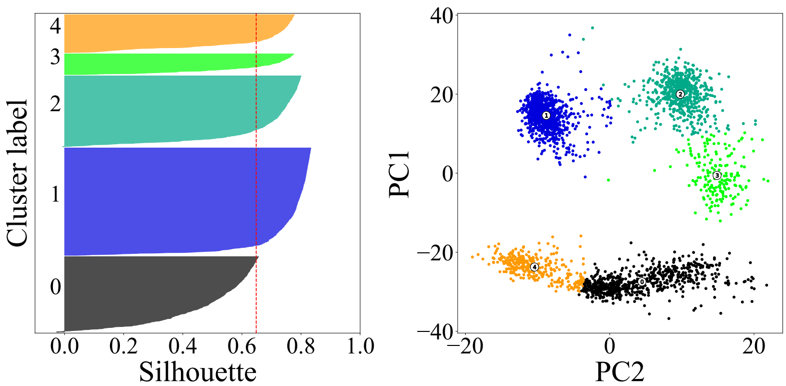
The motivation is that ideally, each point is much closer to the cluster it is assigned to, than to another “second closest” cluster. For $b_i \gg a_i$, the Silhouette approaches 1, while for points with $a_i = b_i$ we obtain a Silhouette of 0, and negative values can arise if there is another closer cluster and hence $b_i < a_i$. If the cluster contains only a single element, a_i is undefined in this equation, and Rousseeuw [Rou87] uses $s_i = 0$ then. The Silhouette values s_i can then be used to visualize the cluster quality by sorting objects by label l_i first, and then descending by s_i, to obtain the Silhouette plot. Figure 3.1 shows an example of the visualization for the data set of Klein et al. [KMA+15]. However, visually inspecting the Silhouette plot is only feasible for small data sets, and hence it is also common to aggregate



(a) $k = 3$ with $ASW = 0.67$



(b) $k = 4$ with $ASW = 0.68$



(c) $k = 5$ with $ASW = 0.65$

Figure 3.1: Average Silhouette Width for clustering results of k-means with $k = 3$ to $k = 5$ on the data set of Klein et al. [KMA+15]. Klein's data set contains embryonic stem cells measured at **four** different time points.

the values into a single statistic, often referred to as the Average Silhouette Width (ASW), but also just as "Silhouette score" of a clustering:

$$S(X, d, L) = \frac{1}{n} \sum_{i=1}^{n} s_i(X, d, L) \ .$$

Hence, this is a function that maps a data set, dissimilarity, and cluster labeling to a real number, and this measure has been shown to satisfy desirable properties for Clustering-Quality Measure (CQM) by Ben-David and Ackerman [BA08].

A key limitation of the Silhouette is its computational cost. It is easy to see that it requires all pairwise dissimilarities, and hence takes $O(N^2)$ time to compute – much more than popular clustering algorithms such as k-means.

For center-based clustering algorithms such as k-means and k-medoids, a simple approximation to the Silhouette is possible by using the distance to the cluster centers respectively medoids $M = \{M_1, \ldots, M_k\}$ instead of the average distance. For this "simplified Silhouette" (which can be computed in $O(Nk)$ time, and which Van der Laan, Pollard, and Bryan [VPB03] called medoid-based Silhouette) we use the distance to the object cluster center a_i' and the distance to the closest other cluster center b_i', to compute the score s_i' of each sample i:

$$\begin{aligned} s_i'(X, d, M) &= \tfrac{b_i' - a_i'}{\max(a_i', b_i')} \text{ , where} \\ a_i' &= \qquad\quad d(x_i, M_{l_i}) \\ b_i' &= \min\nolimits_{k \neq l_i} d(x_i, M_k) \ . \end{aligned}$$

Vendramin, Campello, and Hruschka [VCH10] found the simplified Silhouette to perform comparable to the regular Silhouette, and recommend it for large data sets because of the lower computational requirements.

If each point is assigned to the closest cluster center (as in the standard algorithm for k-means, and also optimal for k-medoids and the Silhouette), we further know that $a_i' \leq b_i'$ and $s_i' \geq 0$, and hence this can further be simplified to the *Medoid Silhouette* $\tilde{s}_i$ of sample i:

$$\tilde{s}_i(X, d, M) = \tfrac{d_2(i) - d_1(i)}{d_2(i)} = 1 - \tfrac{d_1(i)}{d_2(i)} \ .$$

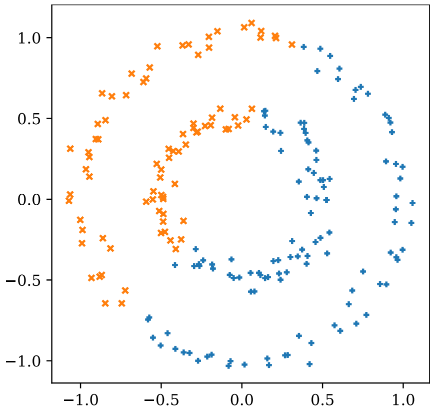
where d_1 is the distance to the closest and d_2 to the second closest center in M. For $d_1(i) = d_2(i) = 0$, we define $\tilde{s} = 1$, corresponding to adding a negligible small value to $d_2(i)$. The Average Medoid Silhouette (AMS) then is defined as

$$\tilde{S}(X, d, M) = \frac{1}{n} \sum_{i=1}^{n} \tilde{s}_i(X, d, M) \ .$$

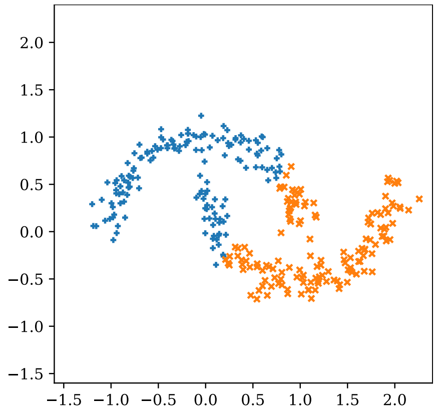
It can easily be seen that the optimum clustering is the (assignment of points to the) optimal set of medoids M such that we minimize an "average relative loss":

$$\arg\max\nolimits_M \tilde{S}(X, d, M) = \arg\min\nolimits_M \operatorname{mean}_i \tfrac{d_1(i)}{d_2(i)}$$

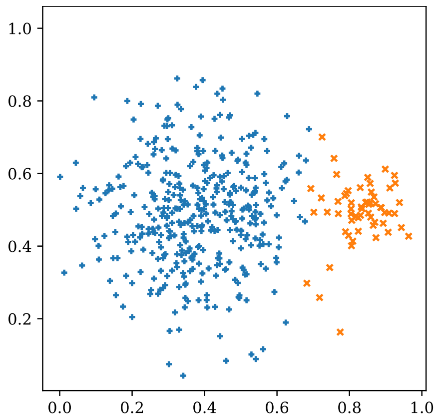
i.e., to optimize the relative contrast of the distance to the nearest and the second nearest cluster center. For clustering around medoids, we impose the restriction



(a) Non-convex: nested circles



(b) Non-convex: two moons



(c) Different diameter

Figure 3.2: Toy examples that violate key Silhouette assumptions, and where Silhouette hence prefers undesired solutions.

$M \subseteq X$; which has the benefit of not restricting the input data to be numerical, and allowing non-metric dissimilarity functions d. This is a key benefit of medoids clustering over, e.g., k-means which restricts the input to $X \subseteq \mathbb{R}^d$ in order to be able to compute the cluster means. On the other hand, this disallows some optimizations particular to the least-squares optimization of k-means. We argue that for k-means, there exist more meaningful and inexpensive evaluation measures than the Silhouette, such as the Variance-Ratio criterion of Caliński and Harabasz [CH74], as discussed by Schubert [Sch23].

From the formulation of both Silhouette and the Medoid Silhouette it should be obvious that the model assumes that each point is best assigned to its “closest” cluster, resembling a generalized Voronoi partitioning of the data. Such approaches work best when clusters are spherical and have the same diameter. When clusters are non-convex, but also when clusters have very different size, this assumption may

Algorithm 2: PAMSIL SWAP: Optimize Silhouette

```
S ← Silhouette sum of the initial solution M
repeat
    (S_*, M_*) ← (0,null)
    foreach m_i ∈ M = {m_1, ..., m_k} do                    // medoids
        foreach x_j ∉ {m_1, ..., m_k} do                    // non-meds
            (S, M') ← (0, M \ {m_i} ∪ {x_j})
            foreach x_o ∈ X = {x_1, ..., x_n} do
                S ← S + s_o(X, d, M')                       // Silhouette
            if S > S_* then (S_*, M_*) ← (S, M')
    if S_* ≥ S then break
    (S, M) ← (S_*, M_*)                                     // perform swap
return (S/N, M)
```

not hold. In such cases, Silhouette may prefer suboptimal solutions. Figure 3.2 shows some toy data sets that violate these assumptions, and where the optimum Silhouette does not yield the desired solution. Note that similar limitations exist in many standard clustering algorithms, in particular in k-means and k-medoids. With k-medoids as well as Silhouette clustering it may be possible to use the density connectivity distance of Beer et al. [BDH+23] that is underlying density-based clustering to improve the results.

PAMSIL

Van der Laan, Pollard, and Bryan [VPB03] proposed to optimize the Silhouette by substituting the Silhouette evaluation measure into a simplified variant of the PAM SWAP procedure (calling this PAMSIL). In each iteration, for each of the $k(N-k)$ possible swaps (exchanging the roles of a medoid and a non-medoid), they compute the full Silhouette in $O(N^2)$, instead of computing the *change* in the loss used by PAM SWAP. Algorithm 2 gives a pseudocode of this procedure. The complexity of PAMSIL hence increases to $O(k(N-k)N^2)$ per iteration, substantially worse than PAM, which is in $O(k(N-k)^2)$. Because this yields a very slow clustering method, they also considered the Medoid Silhouette instead (PAMMEDSIL), which reduced the time complexity to $O(k^2(N-k)N)$ per iteration, still considerably more than the original PAM SWAP iterations.

3.2.2 Category Utility

Most of the internal ("unsupervised", not based on external labels) quality measures such as the Silhouette are designed for continuous numerical data and therefore only work to a limited extent for categorical or binary data. Gluck and Corter [GC85] defined the Category Utility (CU), a measure that quantifies the difference between the number of attribute values within clusters and the expected number of attribute values based on the whole data set. It is solely reliant on the partitioning of the attributes. Given a clustering C with k clusters on the data set D and attributes A, CU is:

$$\mathrm{CU} = \sum_{i=1}^{k} \frac{|C_i|}{N} \sum_{A} \mathrm{P}_{\underline{\underline{A}}}(C_i) - \mathrm{P}_{\underline{\underline{A}}}(D) \text{ , with}$$
$$\mathrm{P}_{\underline{\underline{A}}}(X) = P(x_1 =_A x_2 | x_1 \in D, x_2 \in D)$$

as the probability for the simultaneous occurrence of values of an attribute A when drawing two samples from a set X. CU is an absolute value, allowing a comparison with CU values obtained from other clustering results for the same number of clusters.

3.2.3 k-Nearest-Neighbor Consistency

The kNN Consistency is motivated by kNN Classification, but it can also be used to evaluate clustering validity, as proposed by Ding and He [DH04]. In the following, we only consider clustering that forms a total partitioning of the data set, encoded as a cluster label l_i for each sample i. Overlapping, hierarchical, or incomplete clustering (e.g., noise in DBSCAN) solutions are not considered. For the given samples $X = \{x_1, \ldots, x_n\}$, and the cluster labels $L = \{l_1, \ldots, l_n\}$, the $k\mathrm{NN}(x_i)$ are the k nearest neighbors of x_i. One single sample i is considered to be k-nearest-neighbor consistent if all neighbors $p \in k\mathrm{NN}(x_i)$ have the same cluster label as the sample i:

$$kc_i(X, L) = \begin{cases} 1 & \text{if } \forall x_j \in k\mathrm{NN}(x_i) : l_j = l_i \\ 0 & \text{otherwise} \end{cases}$$

To measure the kNN Consistency of a clustering where not all points are consistent, Ding and He [DH04] also use a fractional kNN Consistency

$$KC(X, L) = \frac{1}{n} \sum\nolimits_{i=1}^{n} kc_i(X, L).$$

To use the consistency as a quality measure, Handl and Knowles [HK07] developed a connectivity-based measure out of the kNN Consistency, which gives more emphasis to the nearest neighbors. The definition of the kNN Consistency can also be applied to other neighborhood concepts, for example, k-Mutual-Nearest-Neighbors (kMNs) [DH04], which symmetrizes k-Nearest-Neighbors and is denoted by $k\mathrm{MN}(x)$:

$$x_j \in k\mathrm{MN}(x_i) \iff x_j \in k\mathrm{NN}(x_i) \wedge x_i \in k\mathrm{NN}(x_j).$$

Mutual nearest neighbors have the benefit of being a symmetric relation, but also being less sensitive to outliers and more likely to contain multiple components.

Intuitively, we remove all unidirectional edges from the k-nearest-neighbor graph. Consider a data set where we have one dense cluster and a far away outlier. The outlier's neighbors will be points of the cluster, but it will not be in the nearest neighbors of the cluster points. Both the kNN and the k-MUTUAL-NEAREST-NEIGHBOR (kMN) are often used in the context of spectral clustering, which in turn is related to DBSCAN [BDH+23].

Ding and He [DH04] aim at enforcing kNN Consistency in their algorithms. We can interpret this as a form of constrained clustering, where we add must-link constraints for neighbors. Such constraints have been previously used, e.g., for model selection [PMC+14] and for integrating external constraints into k-means [WCR+01; BBM04].

ENFORCE

The ENFORCE algorithm of Ding and He [DH04] modifies an existing clustering, which can be created by any cluster analysis method, to improve its consistency by reassigning neighborhoods to the most common cluster. To achieve this, ENFORCE uses closed neighborhoods. A set $S \subset X$ is a closed neighborhood if for all $x \in S$ holds $k\text{NN}(x) \subset S$ and for every two elements $x_i, x_j \in S$ there is a path of elements from kNN (x_i) or $k\text{NN}(x_j)$. The definition can be easily extended to any neighborhood function such as kMN. It is easy to see that all points in a closed neighborhood must be in the same cluster for the clustering to be consistent and that every point can only belong to one closed neighborhood. Figure 3.3 shows the closed neighbor sets on an example data set using mutual nearest neighbors with $k = 3$. To enforce a kNN-consistent clustering, ENFORCE iterates through all sets

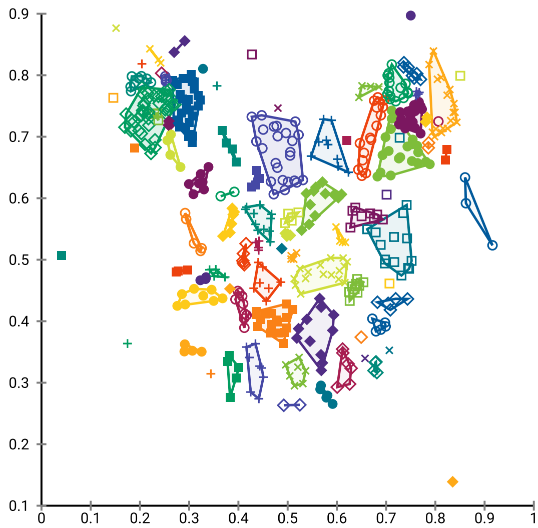


Figure 3.3: Closed neighbor sets for mutual nearest neighbors with $k = 3$.
For neighbor-consistent clustering, these sets must be assigned to the same cluster.

Algorithm 3: ENFORCE

$l \leftarrow$ cluster assignments by input clustering algorithm
$S \leftarrow$ calculate sets of closed neighborhoods
$N \leftarrow$null
foreach $S_i \in S = \{S_1, \ldots, S_h\}$ **do** `// closed neighborhoods`
  $(N) \leftarrow$null
  foreach $x_j \in S_i$ **do** `// count cluster assignments`
    $N_{l_j} = N_{l_j} + 1$
  $o \leftarrow \arg\max N$
  foreach $x_j \in S_i$ **do** $l_j \leftarrow o$ `// assign closed neighborhoods`
return L `// final cluster assignments`

of closed neighborhoods and counts the respective cluster labels. Then it assigns the entire set to the label that occurs most often. There is also an interesting parallel to DBSCAN clustering here, which computes a similar transitive closure, but with a density-based notion of the neighborhood. DBSCAN clusters are closed neighborhoods with respect to density reachability.

3.3 Clustering Categorical Data

Within the domain of categorical data analysis, numerous distance functions, quality measures, and clustering algorithms have emerged to address the unique challenges posed by discrete data representations. We explore the most similar quality measures and clustering algorithms to our following approach, our aim is to differentiate the subtle variations among them.

3.3.1 Distance Functions

Several distance measures for categorical data exist, in particular measures devised for binary data and sets [DD13, Ch 17.3], such as the classic Hamann similarity [Ham61], the simple matching coefficient of Sokal and Michener [SM58], and the Rand similarity [Ran71], all of which are linear transformations of the Hamming distance, or the classic Jaccard index [Jac02]. As set distances are usually defined only in terms of the size of the union, the intersection, the difference of these sets, and the total data set size, they share the problem of having many tied distances. The arithmetic mean of such binary vectors can be interpreted as a concatenation of probability distributions, the mean is precisely the probability of a random instance having a one in the attribute. For sets of points, we hence can also employ divergence measures for (discrete) probability distributions. For individual points (where all probabilities are either one or zero), these measures typically degenerate to some transformation of Hamming distance, too, and hence produce many ties.

An integral information not used by the above measures is the prevalence of values in the entire data set. For motivation, consider Bernoulli distributed binary variables. Sampling twice from a balanced Bernoulli distribution with probability $p = 0.5$ yields a disagreement in 50% of the cases, while sampling twice from a skewed distribution with $p = 0.9$ will only disagree in 18% of samples. The agreement on the rare value only happens in 1% of cases, and we hence may prefer clustering such objects together. Or as articulated by Goodall [Goo66]: "This rule accords with the intuition of the taxonomist, who regards agreement in possessing an uncommon value for an attribute as more convincing evidence of affinity than agreement in having a common value." Having different weights on attributes and values will also help to reduce the number of ties, and hence help stabilizing agglomerative clustering in particular. Goodall's index [Goo66] assigns a similarity of $1 - \sum_{p_i \leq p_k} p_i^2$ for an observed agreement on the value k (with probability p_k). Unfortunately, this index exhibits an undesirable form of instability and discontinuity due to the use of a kind of "cumulative" probabilities: if the value of p_k changes only slightly, entire terms can appear or drop out of the sum. If we have values that are distributed 49% to 51%, an agreement is weighted 0.76 and ≈ 0.5, but if the distribution is exactly 50% to 50% it becomes 0.5 for both. Boriah et al. [BCK08] discuss two variants that simply use $1 - p_k^2$. The use of squared probabilities relates this measure slightly to the Gini index. Other similar measures include the indexes of Smirnov and Anderberg. Boriah et al. [BCK08] survey similarity measures for categorical data in the context of outlier detection. Sulc and Rezanková [SR19b] explore the performance of such similarity measures for hierarchical clustering on synthetic data, but do not consider the agglomerative procedure of the clustering itself.

3.3.2 Clustering Algorithms

Many clustering algorithms for categorical data have been inspired by density-based clustering, such as CACTUS [GGR99], CLICKS [ZPA+07], and CLOPE [YGY02].

Density-based clustering algorithms often require user-specified input parameters, may produce too many clusters or outliers, struggle to find clusters within clusters or subspace clusters, or are sensitive to the order of object input [KR90].

There are also partition-based clustering algorithms for categorical data, such as the k-means variations k-modes and k-prototypes. Instead of the arithmetic mean, these use the mode in categorical attributes as cluster center. They find applicability in scenarios requiring a predetermined number of clusters. COOLCAT [BLC02] is a partition-based clustering algorithm with an information-theoretic background inspired by k-modes. It uses actual data samples as initial cluster centers, afterwards it uses the distribution of values within each partition. It got its name from the objective of reducing the entropy of all clusters, thereby metaphorically "cooling" them. Entropy is often interpreted as a measure of the level of disorder within a distribution. However, a limitation of partition-based clustering algorithms lies in the need for users to manually specify the desired number of clusters as an input parameter.

An alternative approach to clustering categorical data involves hierarchical clustering methods. Hierarchical clustering algorithms organize objects into a tree-like structure called a dendrogram, where each node represents a potential cluster. For small data sets, users often find it valuable to visually inspect this tree, and phylogenetic trees have been used in taxonomy for a long time. In the context of categorical data, these methods typically employ an agglomerative strategy, gradually combining the most similar objects into clusters at different levels within the resulting tree. Popular hierarchical clustering methods for categorical data are ROCK [GRS99], COBWEB [Fis87], and LIMBO [ATM+04], with LIMBO being the most similar to our approach. LIMBO utilizes entropy, similar to COOLCAT, along with the Information Bottleneck (IB) framework to quantify the relevant information retained during clustering. The algorithm employs a bottom-up agglomerative clustering approach, optimizing the Kullback-Leibler divergence of the data distributions before and after merging the clusters. In contrast to our approach, neither LIMBO nor COOLCAT use the overall data distribution.

One central problem in existing work is that often distances, evaluation measures, and clustering algorithms are developed independently of one another, instead of considering them to be different sides of the same problem. We will develop an overarching view of cluster quality for categorical data that directly relates to a cluster similarity measure, a merging strategy for agglomerative clustering, and a notion of cluster stability.

Chapter 4

Sparse Partitioning Around Medoids

In operations research, the k-medoids problem is also known as the (discrete) facility location problem. Several variants of this problem have been researched there. The different variants differ mainly in the objective function to be minimized. Beside the total deviation in k-medoids, k-center minimizes the maximum distance of all points to their assigned cluster centers. There has been deep research in the area of finding approximation algorithms for all these different problems.

Unfortunately, the algorithms k-medoids commonly used are not very scalable to large problems, as we will discuss next.

4.1 Sparse Partitioning Around Medoids

As already mentioned, both greedy initialization and local search in PAM rely on pairwise distances stored in a distance matrix. It is fairly easy to see that a large part of these pairwise distances may be unnecessary to know exactly. It is easy to see that given some assignment of points to medoids, and the maximum distance τ of this assignment, we could replace all larger values in this distance matrix with τ and the solution would not change. Hence there is some natural "cut-off" to distances. If our distance function satisfies the triangle inequality, we may be able to detect and avoid some of these large distances (e.g., with the algorithm of Newling and Fleuret [NF17]).

In this research, we want to focus on a different scenario, where the cut-off may be given in advance (and may be different for each point), but the distance is not necessarily metric. An example from reality for such as problem will be introduced in Section 4.2. While we can (and, effectively, will) treat such uninteresting distances as infinite (or sufficiently large) values, using a sparse storage or the distances only immediately reduces the memory usage, not the run time. Unfortunately, this also easily breaks the optimization procedure, which relies on first finding a *feasible* initial solution, then performing local changes that *improve* the solution. A greedy strategy such as the one discussed above is usually not able to find a valid initial solution for a small k (and in particular, for a very small k the problem may become unsatisfiable with a finite loss). In such cases, the local optimization will also not help, as neighbor solutions will often still be invalid, and hence make no progress.

This is most easily seen if the data set consists of many components that are not connected with edges of finite length.

Instead of searching directly for a solution with k centers, we can solve a second problem of k-medoids clustering at the same time: how should we choose k? Already with k-means clustering, choosing the "optimal" k has shown elusive to a general solution, and is mostly performed by some crude heuristic such as the infamous Elbow criterion.

If we allow the algorithm to vary k, we can much more easily find a valid initial solution (e.g., by choosing the best unconnected vertex until everything is covered). But of course this will usually yield a much higher number of clusters k than desired. But if we perform a multi-criteria optimization in the refinement phase, we may be able to reduce the number of clusters along with minimizing our main objective.

When varying k, we will obtain a Pareto front of solutions that are all optimal in one way or another. This can be formalized as solutions not "dominated" by any other solution in each criterion at the same time. To reduce the set of remaining candidate solutions, it is best if we have some additional constraints to satisfy based on the particular problem to solve.

4.2 Use Case: Simulation of Electrical Substations

We obtain networks using OSMOGrid, which implements ideas of distribution network generation of Kays et al. [KSS+17] on basis of public data (OpenStreetMap). The electrical grid is modeled to follow the streets, and the buildings are used to model consumers. Power consumption is estimated based on zoning and building size, and used to simulate the load flow in the grid. We have made some graph simplifications in preparation for the problems presented below. We remove dead ends, and move the consumers locations (i.e., buildings) to the next point in the street network. Figure 4.1 shows the simulation based on the township Witten Stockum.

On the basis of this graph structure, there are different computational tasks in which resource-efficient clustering models are necessary. One of these tasks is the simulation of electrical substations within the graph. Optimal positions of substations are searched, where the loss in the network is minimized. We approximate the loss as the distance between substations and their connected consumers, which is weighted by the load of the consumers, based on the motivation that electric loss increases with cable length. We describe this as a facility location problem, which comes from urban and public service planing. The objective function FL for facilities and demand points is

$$\mathrm{FL} = \sum\nolimits_{i \in \mathrm{Demands}} d(i, m(i)) + \sum\nolimits_{j \in \mathrm{Centers}} c(j) \quad , \tag{4.1}$$

with $c(j)$ as the cost of opening a facility, and $d(i, m(i))$ as the distance between consumer i and the assigned center $m(i)$. FL has strong similarities to the objective function of k-medoids. We take the facilities as the electrical substations and the consumers as the demand points. Figure 4.2 shows results of clustering the consumers with Fast and Eager Partitioning Around Medoids (FasterPAM) for $k = 4$ substations for the generated graph for Witten Stockum. We can observe that cluster assignment follows the road network, and consumers are not necessarily assigned to the closest center "as the bird flies".

Even with the FasterPAM improvements, the run time complexity is $O(n^2 i)$ for n nodes in the graph and i iterations of the optimization procedure. The underlying OSM planet file contains about 1.2 TB of data. Even though we are only interested in modeling smaller areas of the world, we need to achieve reduction in complexity for solving the task for whole cities or regions in an acceptable run time, as these will nevertheless contain several thousands of houses. We take advantage of some properties of a typical electrical network. It is provable that for this optimization problem, we only need to consider nodes with at least 3 outgoing edges as possible optimal substation locations (except for disconnected points). Hence, it is beneficial to formulate this as an asymmetric problem, where demand points and facility locations are no longer the same set. The distance matrix then no longer has to be calculated for all node pairs, but only for all demand points and substation location connections. This reduces the complexity to $O(nmi)$ for n consumers, m

Figure 4.1: Simulation of an electrical grid based on OpenStreetMap (OSM) data of Witten Stockum.

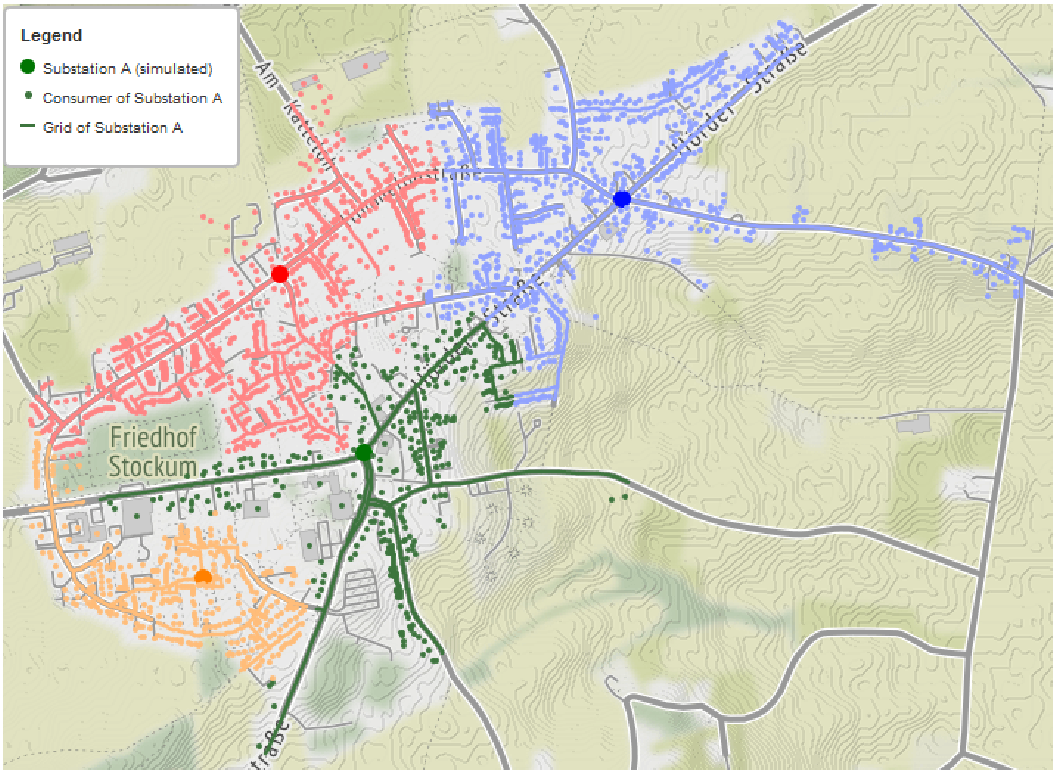


Figure 4.2: Clustering of the demand points of the generated graph structure according to optimal substation locations with $k = 4$ using FasterPAM.

possible substation locations, with $m < n$. If we further limit the maximum distance between a consumer and a substation (which makes sense to limit the power losses), this distance matrix becomes *sparse*, i.e., we now have missing values that we can consider as infinite values. If we do not store these missing values, and iterate using appropriate sparse data structures, we expect to further reduce the run time to $O((e + n + m) \cdot i)$ for e edges. Assuming a similar density of houses and roads everywhere, we can expect the number of edges e to be approximately linear in the *area* of the map we are processing.

4.3 Sparse k-medoids

To use k-medoids clustering for problems with asymmetric and sparse input data, we have to adapt the objective function of k-medoids. We still want to minimize the "total deviation" of all data points $\{x_1, ..., x_n\}$ from the current set of medoids $M \subseteq \{y_1, ..., y_m\}$, but these are now chosen from a different set. Furthermore, for some points, there currently may be no closest reachable medoid $m(x_i)$, and all distances from this x_i to all medoids $m_i \in M$ are undefined. In such cases, we have to incorporate a penalty $\pi(x_i)$ into our loss function L:

$$\mathrm{L} := \sum\nolimits_{i=1}^{n} \begin{cases} \pi(i) & \text{if } m(x_i) = \text{undefined} \\ d(x_i, m(x_i)) & \text{otherwise} \end{cases} \tag{4.2}$$

Note that we allow the set M to change in size below. The penalty $\pi(i)$ can be used to trade the loss of not covering all possible data points against having larger distances. We do not further consider tuning this parameter below, but we instead use $\pi(i) = \pi = \text{const} \to \infty$ to enforce a complete coverage. Because such extreme values can cause numerical problems, our implementation always uses pairs (i, d) to store these losses (and loss changes): an integer i to count the number of unassigned points, and the sum of distances of assigned points d, such that mathematically we have $\mathrm{L} = i \cdot \pi + d$, but do not suffer from numerical problems.

Based on the objective function, we introduce DynBUILD (Dynamic Asymmetric BUILD initialization) as an adaptation of the BUILD algorithm of Kaufman and Rousseeuw [KR87; KR90] to asymmetric sparse input data sets. The greedy BUILD approach is supplemented by a dynamic increase of k, if after choosing k medoids still some objects are not reachable by the current set of medoids. The algorithm hence always choses at least k medoids and covers all consumers. As a baseline, the Random strategy simply uses a given percentage of points as initial cluster centers, and may hence yield an initial solution where constraints are violated, but our improved DynSWAP procedure will repair these while optimizing the assignment. Sparse++ is an adaptation of the well-known k-means++ [AV07] method to sparse data, where cluster centers are chosen proportionally to how many points they cover (again, we continue choosing additional centers until all constraints are satisfied).

We introduce DynSWAP (Dynamic SWAP for asymmetric sparse data) as an dynamic SWAP algorithm based on FasterPAM [SR19a; SR21], adapted to dynamically reduce k, while efficiently processing asymmetric and sparse input data. DynSWAP differs from FasterPAMs SWAP in two ways: To dynamically change k depending on the constraints, we check after each swap whether we can reduce k without violating a constraint (line 30), while if the current object is not suitable for swapping but reduces the number of violated constraints if added as a new medoid (line 34) then we make it an additional medoid. We deliberately chose to only reduce k if we also performed a swap, as to alternate between optimizing the existing medoids and learning the number of clusters k. Both checks are very efficient to implement, as we already know the removal loss change for all medoids ($\Delta\mathrm{L}^{-m_1}, \ldots, \Delta\mathrm{L}^{-m_k}$, also needed by the Fast Partitioning Around Medoids (FastPAM) improvement over PAM) and we also have in $\Delta\mathrm{L}^{+}$ the loss change when adding a new medoid. We can remove the medoid m_i without breaking any constraint, if the π component is zero: $\Delta\mathrm{L}_{\pi}^{-m_i}{=}0$, and making the current candidate y_j a new medoid is beneficial if its $\Delta\mathrm{L}_{\pi}^{+}{<}0$. Whenever adding, removing, or swapping a medoid, we need to update for all data points x_o the nearest medoid nearest(o), the distance to the nearest medoid $d_{\text{nearest}}(o)$, and the distance to the second nearest medoid $d_{\text{second}}(o)$. This can be done more efficiently by updating the previous values, exactly as in FasterPAM. Based on this information, we can also update $\Delta\mathrm{L}^{-m_1}, \ldots, \Delta\mathrm{L}^{-m_k}$, which is the loss change for removing each medoid, efficiently: for each object, removing the nearest medoid incurs a loss change of $(0, d_{\text{second}}(o) - d_{\text{nearest}}(o))$ if there is a second nearest medoid, and $(\pi(o), -d_{\text{nearest}}(o))$ otherwise. Removing another medoid except the nearest medoid does not incur a loss change.

When computing the loss change for adding a new candidate medoid y_c, we initialize an array $\Delta\mathrm{L}$ with the removal loss of each existing medoid, an optimization from FastPAM [SR19a]. To avoid an inner loop over all medoids k, we also incorporate an idea from FasterPAM [SR21], namely to accumulate the loss change that apply to all medoids into the variable $\Delta\mathrm{L}^{+}$. An interesting property of $\Delta\mathrm{L}^{+}$ is that it is the loss change for adding a new medoid, which we use for our logic of dynamically increasing the number of clusters, too. We benefit from sparsity in this approach, because we do not have to consider objects that are not neighbors of the candidate y_c: the loss change by removing existing medoids has already been accounted for, and as they are not reachable from y_c, there is no loss change when adding the replacement medoid. Because of this, our loop only needs to iterate over

Algorithm 4: DynBUILD: Dynamic Asymmetric BUILD initialization

$\mathrm{L}, M \leftarrow (\infty, \infty), \emptyset$
`/* Choose the first medoid: */`
foreach y_j **do** `// compute loss for each` y_j
  $\mathrm{L}_j \leftarrow (\sum_i \pi(i), 0)$ `// everything is unassigned`
  foreach $x_o \in N(y_j)$ **do** `// check neighbors (sparse)`
    $\mathrm{L}_j \leftarrow \mathrm{L}_j + (-\pi(o), d(x_0, y_j))$
  if $\mathrm{L}_j < \mathrm{L})$ **then** $\mathrm{L}, M \leftarrow \mathrm{L}_j, \{y_j\}$ `// current best`
`/* Choose the remaining medoids: */`
for $i = 1 \ldots k-1$ **do**
  $\Delta\mathrm{L}^*, y^* \leftarrow (0,0), \emptyset$ `// storage for best solution`
  foreach $y_j \notin M$ **do**
    $\Delta\mathrm{L}_j \leftarrow (0,0)$ `// loss change accumulator`
    foreach $x_o \in N(y_j)$ **do** `// check neighbors (sparse)`
      $\delta\pi \leftarrow -\pi(o)$ **if** $d_{\text{nearest}}(o) = \infty$ **else** 0
      $\delta d \leftarrow d(x_o, y_j) - d_{\text{nearest}}(o)$
      if $\delta\pi < 0$ **or** $\delta d < 0$ **then** $\Delta\mathrm{L}_j \leftarrow \Delta\mathrm{L}_j + (\delta\pi, \delta d)$
    if $\Delta\mathrm{L}_j < \Delta\mathrm{L}^*$ **then** $\Delta\mathrm{L}^*, y^* \leftarrow \Delta\mathrm{L}_j, y_j$ `// current best`
  $\mathrm{L}, M \leftarrow \mathrm{L} + \Delta\mathrm{L}^*, M \cup \{y^*\}$ `// use best new medoid`
  if $i = k-1$ **and** $\mathrm{L}_\pi > 0$ **then** $k \leftarrow k+1$ `// increase` k
return $\mathrm{L}, \{m_1, ..., m_k\}$

Algorithm 5: DynSWAP: Dynamic SWAP for asymmetric sparse data

$y_{\text{last}} \leftarrow$ invalid
foreach x_o **do** compute nearest$(o), d_{\text{nearest}}(o), d_{\text{second}}(o)$
$\Delta\text{L}^{-m_1}, \ldots, \Delta\text{L}^{-m_k} \leftarrow$ compute loss change removing m_i
while still changing **do**
  foreach $y_c \notin \{m_1, \ldots, m_k\}$ **do**
    break outer loop if $y_c = y_{\text{last}}$ `// no improvements found`
    $\Delta\text{L} \leftarrow (\Delta\text{L}^{-m_1}, \ldots, \Delta\text{L}^{-m_k})$ `// removal loss`
    $\Delta\text{L}^+ \leftarrow 0$ `// accumulator (FasterPAM)`
    foreach $x_o \in N(y_c)$ **do** `// check neighbors (sparse)`
      $d_{oc} \leftarrow d(x_o, y_c)$ `// distance to candidate`
      if $d_{\text{nearest}}(o) = \infty$ **then** `//` x_o `not covered yet`
        $\Delta\text{L}^+ \leftarrow \Delta\text{L}^+ + (-\pi(o), d_{oc})$
      else if $d_{oc} < d_{\text{nearest}}(o)$ **then** `// new nearest`
        $\Delta\text{L}^+ \leftarrow \Delta\text{L}^+ + (0, d_{oc} - d_{\text{nearest}}(o))$
        if $d_{\text{second}}(o) = \infty$ **then** `// no second nearest`
          $\Delta\text{L}_{\text{nearest}(o)} \leftarrow \Delta\text{L}_{\text{nearest}(o)} + (-\pi(o), d_{\text{nearest}}(o))$
        else
          $\Delta\text{L}_{\text{nearest}(o)} \leftarrow \Delta\text{L}_{\text{nearest}(o)} + (0, d_{\text{nearest}}(o) - d_{\text{second}}(o))$
      else if $d_{\text{second}}(o) = \infty$ **then** `// no second nearest`
        $\Delta\text{L}_{\text{nearest}(o)} \leftarrow \Delta\text{L}_{\text{nearest}(o)} + (-\pi(o), d_{oc})$
      else if $d_{oc} < d_{\text{second}}(o)$ **then** `// new second nearest`
        $\Delta\text{L}_{\text{nearest}(o)} \leftarrow \Delta\text{L}_{\text{nearest}(o)} + (0, d_{oc} - d_{\text{second}}(o))$
    $i \leftarrow \arg\min(\{\Delta\text{L}_i\})$ `// best current medoid`
    $\Delta\text{L}_i \leftarrow \Delta\text{L}_i + \Delta\text{L}^+$ `// add accumulator`
    if $\Delta\text{TD}_i < (0, 0)$ **then** `// eager swapping (FasterPAM)`
      swap roles of medoid m^* and non-medoid y_c
      $\text{L} \leftarrow \text{L} + \Delta\text{L}_i$
      update nearest$(o), d_{\text{nearest}}(o), d_{\text{second}}(o)$, $\Delta\text{L}^{-m_1}, \ldots, \Delta\text{L}^{-m_k}$
      $y_{\text{last}} \leftarrow y_c$ `// new stopping position`
      `// After each swap, try to reduce` k`:`
      if $\min(\Delta\text{L}_\pi^{-m_1}, \ldots, \Delta\text{L}_\pi^{-m_k}) = 0$ **then** `//` $Dyn^{\downarrow}$
        $r \leftarrow \arg\min(\{\Delta\text{L}^{-m_i}\})$
        remove medoid m_r
        update nearest$(o), d_{\text{nearest}}(o), d_{\text{second}}(o)$, $\Delta\text{L}^{-m_1}, \ldots, \Delta\text{L}^{-m_k}$
    else if $\Delta\text{L}_\pi^+ < 0$ **then** `//` $Dyn^{\uparrow}$
      add new medoid y_c as it fixes at least one constraint
      update nearest$(o), d_{\text{nearest}}(o), d_{\text{second}}(o)$, $\Delta\text{L}^{-m_1}, \ldots, \Delta\text{L}^{-m_k}$
      $y_{\text{last}} \leftarrow y_c$ `// new stopping position`
return L, M

the neighbors. For each neighbor x_o, we distinguish four cases: (1) the point is currently not yet covered, hence we gain $\pi(o)$ but incur $d(x_o, y_c)$ in line 12, (2) the new medoid is closer than all existing medoids and hence we gain $d_{\text{nearest}}(o) - d(x_o, y_c)$ in line 14. For the case of removing the nearest medoid, we have already included $d_{\text{nearest}}(o)$, and hence we have to cancel this out (either with $-\pi(o)$ or $d_{\text{second}}(o)$). If the new medoid is only second nearest, and there is (3) no previous second nearest, only the loss of removing the nearest medoid needs to be updated in line 20. If (4) a previous, but farther, second nearest exists, we also need to adjust the loss of removing the nearest medoid by the difference between assigning to the new medoid instead of the previous second closest in line 22. Similar case distinctions – except for handling the case of an undefined second closest – can already be found in FasterPAM [SR21].

It is easy to see that the two loops in lines 5 and 9 iterate over all edges, hence the complexity of the procedure is $O((e + n \cdot k) \cdot i)$, where e is the number of edges and i the number of iterations. In the street network example, we can argue that $e \in O(n)$ as we scale the approach to larger networks (as we would keep the maximum distance constant, but increase the area). Hence, this sparse k-medoids version scales linearly for this application. If we have a densely connected graph, then $e \in O(n^2)$ and the run time matches that of standard FasterPAM.

4.4 Experiments

In our experiments, we expect to see a speedup compared to FasterPAM. We also want to check how the dynamic change of k works under consideration of constraints. We have to evaluate how well the Sparse k-Medoids is able to find the smallest possible k still meeting the constraints. Hence, we analyze three initialization methods DynBUILD, Random, and Sparse++. Finally, we perform a qualitative evaluation by comparing our simulations with original substation locations from OSM.

Data Sets To verify the algorithm we need sufficiently large test data sets, and choose constraints such as to obtain sparse distance matrices. In this work, we focus on the processing and evaluation of energy grids generated by OSMOGrid. For quality evaluation, we choose areas where many substations are documented in OSM. Figure 4.3 shows a cutout of the electrical grid generated by OSMOGrid for the city of Witten, using the 127 substation locations from OSM (although this likely is not complete, as seen in Figure 4.3). We can then compare the quality of our calculated models to the model based on the real substations, but we need to remember that there may be additional substations not in OSM, and that real power networks have grown historically, have to satisfy additional constraints, and are hence not optimal either. For the purpose of generating “realistic” networks, it is desirable to achieve a comparable quality, without overfitting to the example data we have. On the data set, we evaluate the dynamic methods for choosing k. The

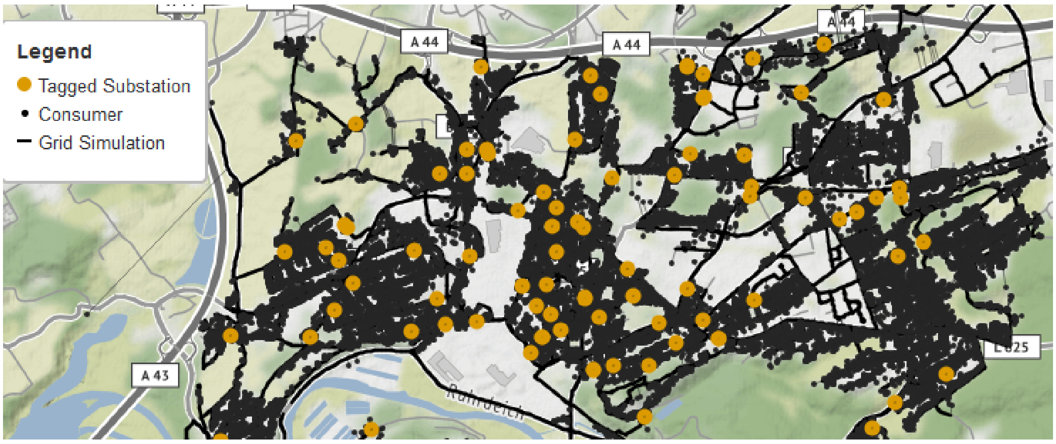


Figure 4.3: Grid simulation and known substation locations in OSM for Witten. The road network contains 37287 edges and 36844 nodes, n=35713 consumer and m=1130 possible places for substations. The location of 127 substations is documented in OSM, but very likely several are missing in particular in the east.

"optimum" k depends on the constraints set, and thus on the sparsity of the distance matrix. Figure 4.4 shows the smallest k to meet the constraint of a maximum cable length in the grid. With increasing cable length, the number of missing distances in the matrix decreases, but the best number of substations decreases much faster.

We evaluate the algorithms in the ELKI open-source toolkit [SZ19] in Java. For comparability, we perform all computations in ELKI and use the original implementations of FasterPAM as reference. This way we avoid side effects caused by different implementations [KSZ17]. We run 100 restarts on an AMD EPYC 7302 processor using a single thread and evaluate the average, maximum, and minimum values.

Dynamic k To evaluate the quality with a variable k, we compare the solutions found by the algorithms to the best known solution of all runs. We also compare the different initialization algorithms, and the variants of the dynamic SWAP. We measure the solution's k, the run time of initialization and SWAP, and whether the result satisfies all constraints. Summarized results are shown in Table 4.1. Only Random[5] initialization without dynamic increase of k fails to satisfy constraints. This was to be expected, because it only uses 5% of the possible substations as medoids, but there does not appear to be a solution with just this many clusters. Because DynBUILD is deterministic, it always produced k=93 clusters after initialization. After the SWAP phase, the average k was 76.2, which is 2.2 more than the best known k=74 (we iterate in a randomized order in SWAP to avoid dependence on the input data order). Since the initial solution already satisfied all constraints, and SWAP preserves this property, k can only decrease. Among the random ini-

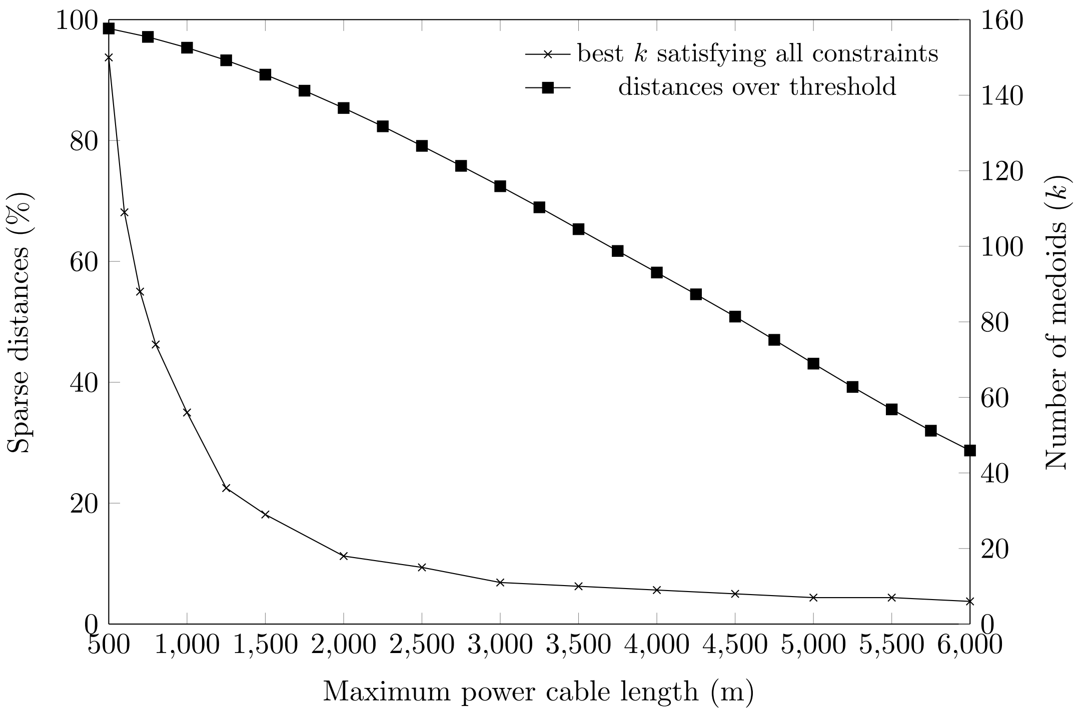


Figure 4.4: Sparsity of the distance matrix depending on a maximum cable length constraint between consumer and substation for the simulation of Witten, and the minimum number of substations k for which no constraint is broken.

tializations Random^5 with DynSWAP$^{\downarrow\uparrow}$ found the best results on average. With k=77.9 after the SWAP phase, the number of stations is on average 3.9 larger than the best k. Finally, the SWAP after DynBUILD needs on average 48.4% of the run time of the SWAP after a random initialization. Due to random initialization, the average number of medoid changes during the SWAP increases substantially from 141 to 310, showing that DynBUILD finds superior starting conditions than both random sampling and Sparse++.

Run Time Speedup In order to evaluate the run time of the different methods, we performed experiments for different constraints and different values of k. Figure 4.5 shows the total run time (initialization and SWAP) for DynBUILD, Random^5, Random^{10}, and Sparse++ initialization. We again use the Witten data set, and choose the distance constraint such that all methods can achieve the desired k. We compare the run time with the FasterPAM implementation with a random initialization (as recommended for FasterPAM). For DynBUILD we evaluate the SWAP with dynamic decrease of k (Dyn$^{\downarrow}$) and for all random initialization the SWAP with both dynamic increase and decrease of k (Dyn$^{\downarrow\uparrow}$). We use a log scale on this plot because of the huge difference: the sparsity optimized DynSWAP over all initializations on average only uses 7% of the run time of the original Faster-

Table 4.1: Comparison of k depending on initialization and swap algorithms for generating a grid for Witten. All results are averaged over 100 restarts.

Samples	Implementation	Init	SWAP	Init	SWAP	**changes**	
Random^{5}	Dyn$^{\downarrow}$	56	56	**0.2**	6919.2	205.3	0%
	Dyn$^{\downarrow\uparrow}$	56	77.9	**0.2**	7862.3	287.9	100%
Random^{10}	Dyn$^{\downarrow}$	113	82.8	**0.1**	11578.8	354.9	100%
	Dyn$^{\downarrow\uparrow}$	113	82.5	**0.1**	11044.4	372.9	100%
Sparse++	Dyn$^{\downarrow}$	182.3	80.3	16.3	8632.2	269.2	100%
	Dyn$^{\downarrow\uparrow}$	182.2	80.6	3.6	8640.3	268.4	100%
DynBUILD	Dyn$^{\downarrow}$	93	**76.3**	389.9	**4296.3**	141.9	100%
	Dyn$^{\downarrow\uparrow}$	93	**76.2**	394.6	4320.3	141.4	100%

Table 4.2: Comparison of the loss of the grid with 127 tagged substations with the calculated loss for $k = 127$. All results are given as average values of 100 restarts. All constraints were satisfied after the SWAP phase.

Algorithm		**loss after SWAP** $\times 10^{7}$				**run time in ms**		**medoid changes**
		$\varnothing$		min				
Init.	SWAP	d	π	d	π	Init	SWAP	
Random^{5}	Dyn$^{\downarrow\uparrow}$	1.3155	**0**	**1.3083**	**0**	**0.1**	12418.6	469.6
Random^{10}	Dyn$^{\downarrow\uparrow}$	**1.3149**	**0**	1.3085	**0**	**0.1**	10395.5	339.8
Sparse++	Dyn$^{\downarrow\uparrow}$	**1.3149**	**0**	**1.3083**	**0**	16.4	10796.5	279.4
DynBUILD	Dyn$^{\downarrow}$	1.3171	**0**	1.3092	**0**	525.6	**9225.4**	294.9
tagged substations		1.86	**0**					

PAM with random initialization, the DynBUILD and DynSWAP$^{\downarrow}$ combination on average even only uses 4%. This was expected as FasterPAM has to process the much larger dense matrix. With increasing k, we can use a more sparse matrix here, which is the reason why the DynSWAP approaches become faster while FasterPAM becomes slower due to the higher number of clusters. The different random initializations differ only slightly in run time, but require on average about twice as long as DynBUILD. In addition to the fast run time, DynBUILD with Dyn$^{\downarrow}$SWAP also produces the lowest number of excess clusters compared to the best known k with an average of 2.2.

Quality In order to evaluate the result quality, we compare the optimized substation locations to the substations tagged in the OSM (however, these likely are incomplete). Table 4.2 shows the results for a target k=127 compared to the loss of the 127 tagged substations shown in Figure 4.3. Sparse++ and Random^{10} initialization with Dyn$^{\downarrow\uparrow}$SWAP results in the lowest loss with 1.3149×10^7 and is 29% lower than the loss of the tagged substations; but the quality difference between the randomized initializations are not significant (the SWAP does a good enough job to always reach a good solution); the main difference here is in the run times, where the strategy to sample more centers than necessary, then decrease, seems to be superior to the others. The minimum loss over 100 restarts is obtained with Sparse++ and Random^5 with 1.3083×10^7. DynBUILD initialization with Dyn$^{\downarrow}$SWAP find a slightly higher loss of 1.3271×10^7, but yields the fastest total run time with 9551.0 ms, despite using the slowest initialization by far.

All methods found results substantially better (1.31 vs. 1.86) than the "gold standard" solution given by the OSM tags. This had to be expected because of supposedly missing tags, but also because the real grids were grown over time and the substations were built one by one, and not automatically optimized. In our

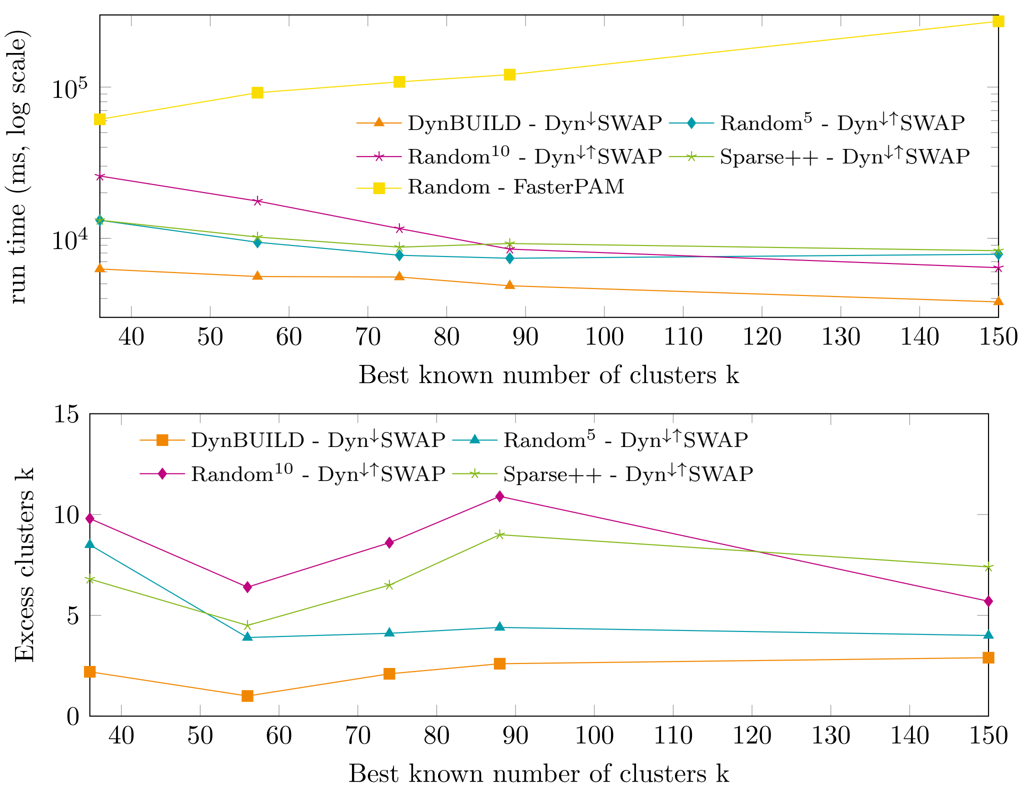


Figure 4.5: Run time of the initialization and SWAP for DynBUILD, Random^5, Random^{10} and Sparse++ initialization depending on the best number of k for the simulation of the grid of Witten. For reference, the SWAP run time of the FasterPAM implementation is included, where the k chosen is the best one we know. The best k is controlled indirectly by the constraints set, see Figure 4.4. In addition to the run time, the deviation from the best known k after SWAP is also shown.

simulation, we have complete information about the grid structure and can thus calculate an optimal substation distribution (green field planning), that cannot be realistically achieved in reality, where the existing power network cannot simply be replaced.

4.5 Discussion

In the experiments, we focused on the specific use case of energy grid simulation. Besides optimizing the FasterPAM approach for sparse problems, we also have begun working on automatically finding the parameter k as part of the optimization problem. For this we combined two losses in our loss function: one corresponding

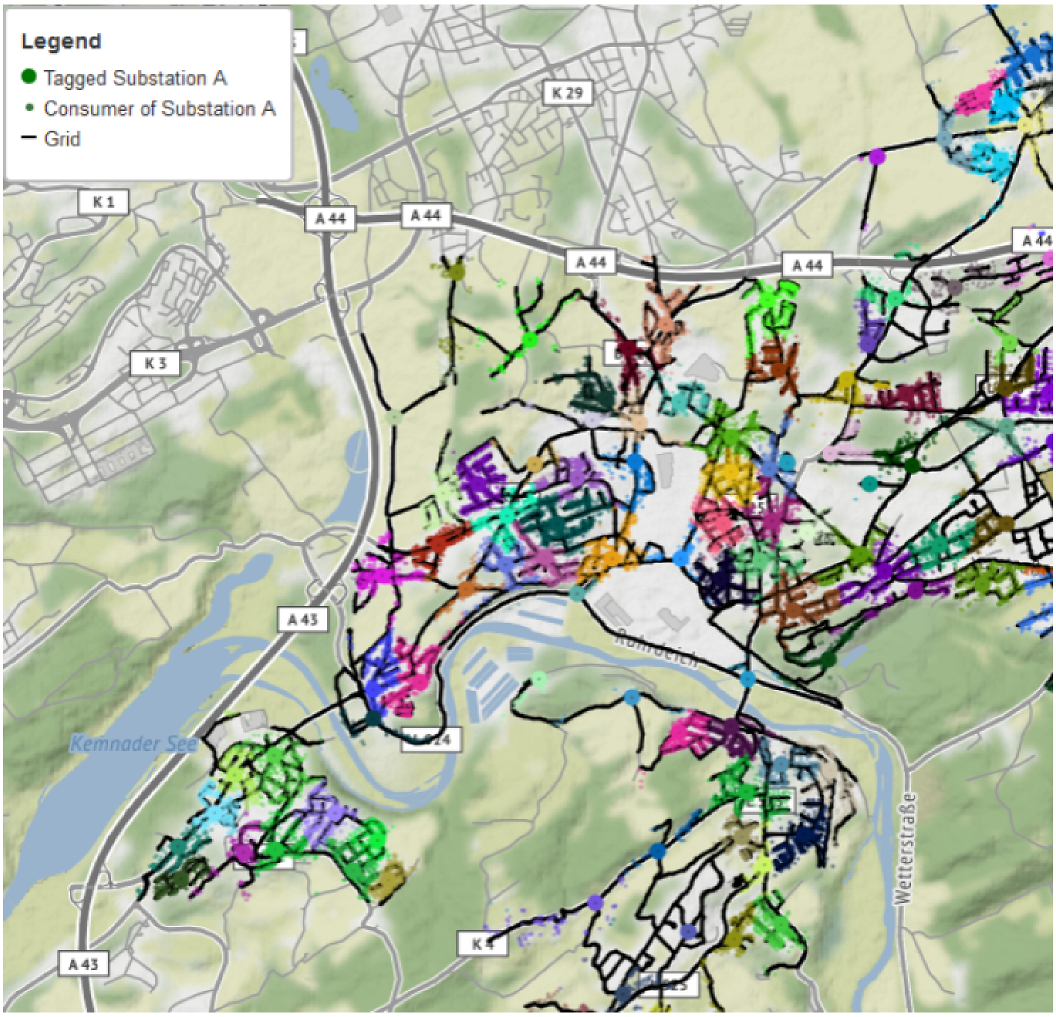


Figure 4.6: Simulation of an electrical grid based on OSM data of Witten. 127 substations calculated with Sparse++ and Dyn$^{\downarrow\uparrow}$SWAP from Table 4.2. All consumers are color-coded to their nearest sub-station.

to the cost of poorly handled locations (which could also be outliers), and the other part being the classic k-medoids problem. It would be easy to incorporate an additional cost term to the opening or closing of locations, and to weight these costs differently. In this experiment, we used a strict requirement to cover all locations (i.e., $\pi \to \infty$), but using a smaller weight may yield interesting approximations.

So far, we have considered the problem of optimal substation positions without a maximum capacity of substations. In reality, there is a maximum load that can be served by a substation. In densely populated areas, we hence may need more substations. This results in a capacitated facility location problem, which contains such an additional capacity constraint, and is worth exploring in future work.

Chapter 5

Clustering by Direct Optimization of the Medoid Silhouette

In this chapter, we focus on the evaluation measure known as the Silhouette [Rou87], more precisely, on the Average Silhouette Width (ASW), and discuss an efficient algorithm to optimize a variant of this measure, inspired by the well-known PAM algorithm [KR87; KR90] and FasterPAM [SR21; SR19a]. Vendramin, Campello, and Hruschka [VCH10] survey 40 such measures and variants, and find the Silhouette to be one of the most robust. Silhouette is also a popular measure to choose the number of clusters in k-medoids or even k-means. However, classic visual inspection of the Silhouette plot is only feasible for small data sets, and users typically rely on the aggregate coefficient. We can choose the number of clusters performantly even on larger data sets with our more efficient integration.

We will first perform a theoretical analysis of the properties of the Medoid Silhouette, to show that it is worth exploring as an alternative to the original Silhouette, then introduce the new algorithm.

5.1 Axiomatic Characterization of Medoid Clustering

To characterize the Medoid Silhouette, we follow the axiomatic approach of Ben-David and Ackerman [BA08] introduced in Section 2.4, i.e., we prove the value of the Average Medoid Silhouette (AMS) as a Clustering-Quality Measure (CQM) by proving that it satisfies some interesting properties. Kleinberg [Kle02] defined three axioms for clustering functions and argued that no clustering algorithm can satisfy these desirable properties at the same time, as they contradict. Because of this, Ben-David and Ackerman [BA08] weaken the original Consistency Axiom and extract four axioms for CQMs: *Scale Invariance* and *Richness* are defined analogously to the Kleinberg Axioms. We first redefine the CQM axioms [BA08] to match the notion of medoid-based clustering before we check them for the Average Medoid Silhouette (AMS).

Definition 4 (Relation $\sim_M$) *For given data points $X = \{x_1, \dots, x_n\}$ with a set of k medoids $M = \{m_1, \dots, m_k\}$ and a dissimilarity d, we write $x_i \sim_M x_{i'}$ whenever x_i and $x_{i'}$ have the same nearest medoid nearest$(i) \subseteq M$, otherwise $x_i \not\sim_M x_{i'}$.*

The $\sim_M$ relation encodes the partitioning of the data set given by the medoids M, and is transitive and symmetric.

Definition 5 (M-consistent) *Dissimilarity d' is an* M-consistent *variant of d, if $d'(x_i, x_{i'}) \leq d(x_i, x_{i'})$ for $x_i \sim_M x_{i'}$, and $d'(x_i, x_{i'}) \geq d(x_i, x_{i'})$ for $x_i \not\sim_M x_{i'}$.*

Meaning that a consistent variant reduces the distances within clusters and increases the distances between clusters.

Definition 6 (Isomorphism of Medoids) *Two sets of medoids $M, M' \subseteq X$ with a distance function d over X, are* isomorphic*, if there exists a distance-preserving isomorphism $\phi : X \to X$, such that for all $x_i, x_{i'} \in X$, $x_i \sim_M x_{i'}$ if and only if $\phi(x_i) \sim_{M'} \phi(x_{i'})$.*

Axiom 8 (Scale Invariance) *A medoid-based quality measure f satisfies* scale invariance *if for every set of medoids $M \subseteq X$ for d, and every positive λ, $f(X, d, M) = f(X, \lambda d, M)$.*

This means that if we scale all distances with the same constant $\lambda > 0$, the quality does not change.

Axiom 9 (Consistency) *A medoid-based quality measure f satisfies* consistency *if for a set of medoids $M \subseteq X$ for d, whenever d' is an M-consistent variant of d, then $f(X, d', M) \geq f(X, d, M)$.*

This means that if we reduce the distances within the same cluster and increase the distances between different clusters, the quality does not decrease.

Axiom 10 (Richness) *A medoid-based quality measure f satisfies* richness *if for each set of medoids $M \subseteq X$, there exists a distance function d over X such that $M = \arg\max_{M'} f(X, d, M')$.*

Meaning that for every possible set of medoids, there exists a distance for which this solution is optimal.

Axiom 11 (Isomorphism Invariance) *A medoid-based quality measure f is* isomorphism-invariant *if for all sets of medoids $M, M' \subseteq X$ with distance d over X where M and M' are isomorphic, $f(X, d, M) = f(X, d, M')$.*

This means that transformations (including data permutations) that preserve distances do not affect clustering quality.

Batool and Hennig [BH21] prove that the Average Silhouette Width (ASW) satisfies the original CQM axioms. We now prove the four adapted axioms for the AMS.

Theorem 1 *The AMS is a* scale invariant *quality measure.*

Proof: If we replace d with λd, both $d_1(i)$ and $d_2(i)$ are multiplied by λ, and the term will cancel out. Hence, $\tilde{s}_i$ does not change for any i:

$$\begin{aligned}\tilde{S}(X, \lambda d, M) &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \tilde{s}_i(X, \lambda d, M) \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} 1 - \tfrac{\lambda d_1(i)}{\lambda d_2(i)} \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} 1 - \tfrac{d_1(i)}{d_2(i)} \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \tilde{s}_i(X, d, M) = \tilde{S}(X, d, M) \ .\end{aligned}$$

Theorem 2 *The AMS is a* consistent *quality measure.*

Proof: Let dissimilarity d' be an M-consistent variant of d. By Definition 5: $d'(x_i, x_{i'}) \leq d(x_i, x_{i'})$ for all $x_i \sim_M x_{i'}$, and $\min_{x_i \not\sim_M x_{i'}} d'(x_i, x_{i'}) \geq \min_{x_i \not\sim_M x_{i'}} d(x_i, x_{i'})$. This implies for all $i \in \mathbb{N}$: $d'_1(i) \leq d_1(i), d'_2(i) \geq d_2(i)$ and it follows:

$$\tfrac{d_1(i)}{d_2(i)} - \tfrac{d'_1(i)}{d'_2(i)} \geq 0 \quad \Leftrightarrow \quad 1 - \tfrac{d'_1(i)}{d'_2(i)} - \left(1 - \tfrac{d_1(i)}{d_2(i)}\right) \geq 0$$

which is equivalent to $\forall_i \ \tilde{s}_i(X, d', M) \geq \tilde{s}_i(X, d, M)$, hence $\tilde{S}(X, d', M) \geq \tilde{S}(X, d, M)$, i.e., AMS is a consistent quality measure.

Theorem 3 *The AMS is a* rich *quality measure.*

Proof: We can simply encode the desired set of medoids M in our dissimilarity d. We define $d(x_i, x_j)$ such that it is 0 if trivially $i = j$, or if x_i or x_j is the first medoid m_1 and the other is not a medoid itself. Otherwise, let the distance be 1.

For M we then obtain $\tilde{S}(X, d, M) = 1$, because $d_1(i) = 0$ for all objects, as either x_i is a medoid itself, or can be assigned to the first medoid m_1. This is the maximum possible AMS. Let $M' \neq M$ be any other set of medoids. Then there exists at least one missing $x_i \in M \setminus M'$. For this object $\tilde{s}_i(X, d, M) = 0$ (as its distance to all other objects is 1, and it is not in M'), and hence $\tilde{S}(X, d, M') < 1 = \tilde{S}(X, d, M)$. That is, for any set of medoids (or similar, for any partitioning of the data set), there exists a dissimilarity function that yields the desired result.

Theorem 4 *The AMS is an* isomorphism-invariant *quality measure.*

Proof: Let $M, M' \subseteq X$ be two sets of medoids with a distance function d over X. If they are isomorphic, there exists a distance-preserving isomorphism $\phi : X \to X$, such that for all $x_i, x_{i'} \in X$, $x_i \sim_M x_{i'}$ if and only if $\phi(x_i) \sim_{M'} \phi(x_{i'})$.

Therefore, every $d_1(i)$ is equal $d_1(i')$ and every $d_2(i)$ is equal $d_2(i')$, and hence $\tilde{S}(X, d, M) = \tilde{S}(X, d, M')$.

Algorithm 6: PAMMEDSIL SWAP: Iterative improvement

$S' \leftarrow$ Simplified Silhouette sum of the initial solution M
repeat
  $(S'_*, M_*) \leftarrow$ (0,null)
  foreach $m_i \in M = \{m_1, \ldots, m_k\}$ **do** // medoids
    foreach $x_j \notin \{m_1, \ldots, m_k\}$ **do** // non-meds
      $(S', M') \leftarrow (0, M \setminus \{m_i\} \cup \{x_j\})$
      foreach $x_o \in X = \{x_1, \ldots, x_n\}$ **do**
        $S' \leftarrow S' + s'_o(X, d, M')$
      if $S' > S'_*$ **then** $(S'_*, M_*) \leftarrow (S', M')$
  if $S'_* \geq S'$ **then break**
  $(S', M) \leftarrow (S'_*, M_*)$ // perform swap
return $(S'/N, M)$

5.2 Direct Optimization of Medoid Silhouette

PAMSIL [VPB03] is a modification of PAM [KR87; KR90] to optimize the Average Silhouette Width (ASW). For PAMSIL, Van der Laan, Pollard, and Bryan [VPB03] adjust the SWAP phase of PAM by always performing the SWAP that provides the best increase in the ASW. When no further improvement is found, the algorithm terminates and a (local) maximum of the ASW has been achieved. In contrast to k-means, this procedure does not alternate between two steps, but every step reduces the loss function, or the algorithm stops. Given that there only exist $\binom{k}{n}$ possible states, only a finite number of improvements is possible. However, where in the original PAM algorithm we efficiently compute only the change in its loss (in $O(N-k)$ time for each of $(N-k)k$ swap candidates), PAMSIL computes the entire ASW in $O(N^2)$ for every candidate, and hence the run time per iteration increases to $O(k(N-k)N^2)$. For a small k, this yields a run time that is cubic in the number of objects N, and the algorithm may need several iterations to converge.

5.2.1 Naive Medoid Silhouette Clustering

PAMMEDSIL [VPB03] uses the Average Medoid Silhouette (AMS) instead, which can be evaluated in only $O(Nk)$ time. This yields a SWAP run time of $O(k^2(N-k)N)$ (now only quadratic in N for small $k \ll N$, but for practical applications the quadratic dependency on k is also noticeable). A pseudocode of PAMMEDSIL shown in Algorithm 6 is almost identical to that of PAMSIL, but using the medoid-based Silhouette scores s'_o instead of the classic Silhouette s_o.

As Schubert and Rousseeuw [SR21; SR19a] were able to reduce the run time of PAM to $O(N^2)$ per iteration, we will now modify the PAMMEDSIL approach accordingly to obtain a similar improvement. We also apply the idea of eager swapping [SR21], i.e., we perform greedy first-descent optimization instead of searching for the steepest-descent.

5.2.2 Finding the Best Swap

We first bring PAMMEDSIL up to par with regular PAM. The trick introduced with PAM is to compute the change in loss instead of recomputing the loss, which can be done in $O(N-k)$ instead of $O(k(N-k))$ time if we store the distance to the nearest and second centers, as the latter allows us to compute the change if the current nearest center is removed efficiently. In the following, we omit the constant parameters X and d for brevity. We denote the previously nearest medoid of i as nearest(i), and $d_1(i)$ is the (cached) distance to it. We similarly define second(i), $d_2(i)$, and $d_3(i)$ with respect to the second and third nearest medoid. We briefly use d_1' and d_2' to denote the new distances for a candidate swap. For the Medoid Silhouette, we can compute the change when swapping medoid $m_i \in \{m_1, \ldots, m_k\}$ with non-medoid $x_j \notin \{m_1, \ldots, m_k\}$ as follows:

$$\begin{aligned}
\Delta\tilde{\mathrm{S}} &= \tfrac{1}{n} \textstyle\sum_{o=1}^{n} \Delta\tilde{\mathrm{s}}_o(M, m_i, x_j) \\
\Delta\tilde{\mathrm{s}}_o(M, m_i, x_j) &= \tilde{\mathrm{s}}_o(M \setminus \{m_i\} \cup \{x_j\}) - \tilde{\mathrm{s}}_o(M) \\
&= 1 - \tfrac{d_1'(i)}{d_2'(i)} - \left(1 - \tfrac{d_1(i)}{d_2(i)}\right) \\
&= \tfrac{d_1(i)}{d_2(i)} - \tfrac{d_1'(i)}{d_2'(i)} \,.
\end{aligned}$$

Clearly, we only need the distances to the closest and second closest center, before *and* after the swap. Instead of recomputing these values by searching, we exploit that only one medoid can change in a swap. We can determine the new values of d_1' and d_2' using a constant set of cached values only, and hence save a factor of $O(k)$ on the run time over the naive approach using a loop to check all medoids.

In the PAM algorithm (where the change would be simply $d_1' - d_1$), the distance to the *second* nearest is cached in order to compute the loss change if the current medoid is removed, without having to consider all $k-1$ other medoids: the point is then either assigned to the new medoid, or its former second closest. To efficiently compute the change in Medoid Silhouette, we have to take this one step further, and we additionally have to cache the identity of the second closest center (denoted second) and the distance to the *third* closest center (denoted d_3). Because we still only change one medoid at a time, both the closest and the second closest must be either the new medoid, or any of the previous three closest (three in case one of them was removed). Depending on which medoid is swapped, we need to distinguish cases.

The change in Medoid Silhouette is then computed roughly as follows: (1) If the new medoid is the new closest, the second closest is either the former nearest, or the second nearest (if the first was replaced). (2) If the new medoid is the new second closest, the closest either remains the former nearest, or the second nearest

Algorithm 7: Change in Medoid Silhouette, $\Delta\tilde{s}_o(M, m_i, x_j)$

if $m_i = \mathit{nearest}(o)$ **then**
    `// Nearest medoid is replaced with xj:`
    if $d(o,j) < d_2(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d(o,j)}{d_2(o)}$
    `// Nearest removed, xj new second nearest:`
    if $d(o,j) < d_3(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d_2(o)}{d(o,j)}$
    `// Nearest removed, xj is farther than third:`
    else return $\frac{d_1(o)}{d_2(o)} - \frac{d_2(o)}{d_3(o)}$
else if $m_i = \mathit{second}(o)$ **then**
    `// Second nearest is replaced, xj closer:`
    if $d(o,j) < d_1(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d(o,j)}{d_1(o)}$
    `// Second nearest is replaced by xj:`
    if $d(o,j) < d_3(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d_1(o)}{d(o,j)}$
    `// Second nearest is replaced, xj far:`
    else return $\frac{d_1(o)}{d_2(o)} - \frac{d_1(o)}{d_3(o)}$
else
    `// xj new closest:`
    if $d(o,j) < d_1(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d(o,j)}{d_1(o)}$
    `// xj new second closest:`
    if $d(o,j) < d_2(o)$ **then return** $\frac{d_1(o)}{d_2(o)} - \frac{d_1(o)}{d(o,j)}$
    `// xj replaced some far medoid:`
    else return 0

(if the first was replaced). (3) If the new medoid is neither, we may still have replaced the closest or second closest; in which case the distance to the third nearest is necessary to compute the new Silhouette. Putting all the cases (and sub-cases) into one equation becomes a bit messy, and hence we opt to use the pseudocode in Algorithm 7 instead of an equivalent mathematical notation. Note that the first term is always the same (the previous loss), except for the last case, where it canceled out via $0 = \frac{d_1(o)}{d_2(o)} - \frac{d_1(o)}{d_2(o)}$. As this is a frequent case, it is beneficial to not have further computations here (and hence, to compute the change instead of computing the loss). Clearly, this algorithm runs in $O(1)$ if $n_1(o)$, $n_2(o)$, $d_1(o)$, $d_2(o)$, and $d_3(o)$ are known. We also only compute $d(o,j)$ once. Modifying PAMMEDSIL (Algorithm 2) to use an incremental computation yields a run time of $O(k(N-k)N)$ to find the best swap, i.e., already $O(k)$ times faster. This integrated a key idea of PAM into this algorithm, but we can further improve this approach with ideas from FastPAM.

5.2.3 Fast Medoid Silhouette Clustering

We now integrate an acceleration added to the PAM algorithm by Schubert and Rousseeuw [SR19a; SR21], that exploits redundancy among the loop over the k medoids to replace. For this, the loss change $\Delta\tilde{\mathrm{S}}(m_i, x_j)$ is split into multiple components: (1) the change by removing medoid m_i (without choosing a replacement, as if we assigned points to their second closest instead), (2) the change by adding x_j as an additional medoid (without replacing any of the currently selected), and (3) a correction term if both operations occur at the same time. The first components can be computed in $O(N)$, the second in $O(N(N-k))$, and the last factor is 0 if the removed medoid is neither of the two closest, and hence is in $O(N)$. After performing a swap it becomes necessary to update the caches, which may involve finding the third nearest medoid and takes $O(k(N-k))$ time in the worst case. This then yields an algorithm that finds the best swap in $O((N-k)(N+k))$ – or less formal in $O(N^2)$, about k times faster for small k than the previous.

First, the changes if we removed medoids $m_i \in M$, and corresponding to lines 4 and 8 in Algorithm 7 are computed for each i as:

$$\Delta\tilde{\mathrm{S}}^{-m_i} = \sum_{\mathrm{nearest}(o)=i} \tfrac{d_1(o)}{d_2(o)} - \tfrac{d_2(o)}{d_3(o)} + \sum_{\mathrm{second}(o)=i} \tfrac{d_1(o)}{d_2(o)} - \tfrac{d_1(o)}{d_3(o)}\,. \tag{5.1}$$

By iterating over all points o and adding to the accumulators for nearest(o) and second(o), we can compute these terms in $O(N)$ for all medoids m_i in one pass.

Secondly, for any non-medoid x_j, we can compute the change when adding this point, corresponding to the lines 10 and 11 in Algorithm 7, as:

$$\Delta\tilde{\mathrm{S}}^{+x_j} = \sum_{o=1}^{N} \begin{cases} \tfrac{d_1(o)}{d_2(o)} - \tfrac{d(o,j)}{d_1(o)} & \text{if } d(o,j) < d_1(o) \\ \tfrac{d_1(o)}{d_2(o)} - \tfrac{d_1(o)}{d(o,j)} & \text{elif } d(o,j) < d_2(o) \\ 0 & \text{otherwise} \end{cases}\,. \tag{5.2}$$

Using these two terms, we can derive the remaining correction term by only considering the cases where nearest(o) $= i$ or second(o) $= i$, including cancel-out terms for summands in $\Delta\tilde{\mathrm{S}}^{-m_i}$ and $\Delta\tilde{\mathrm{S}}^{+x_j}$ where necessary:

$$\begin{aligned} \Delta\tilde{\mathrm{S}}(m_i, x_j) &= \sum_{o=1}^{N} \Delta\tilde{\mathrm{s}}_o(M, m_i, x_j) \\ &= \Delta\tilde{\mathrm{S}}^{-m_i} + \Delta\tilde{\mathrm{S}}^{+x_j} \\ &+ \sum_{\mathrm{nearest}(o)=i} \begin{cases} \tfrac{d(o,j)}{d_1(o)} + \tfrac{d_2(o)}{d_3(o)} - \tfrac{d_1(o)+d(o,j)}{d_2(o)} & \text{if } d(o,j) < d_1(o) \\ \tfrac{d_1(o)}{d(o,j)} + \tfrac{d_2(o)}{d_3(o)} - \tfrac{d_1(o)+d(o,j)}{d_2(o)} & \text{elif } d(o,j) < d_2(o) \\ \tfrac{d_2(o)}{d_3(o)} - \tfrac{d_2(o)}{d(o,j)} & \text{elif } d(o,j) < d_3(o) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Algorithm 8: Fast Medoid Silhouette Clustering (FastMSC)

repeat
  foreach x_o **do** update $\text{nearest}(o), \text{second}(o), d_1(o), d_2(o), d_3(o)$
  $\Delta\tilde{\mathrm{S}}^{-m_1}, \ldots, \Delta\tilde{\mathrm{S}}^{-m_i} \leftarrow$ compute loss change removing m_i using (5.1)
  $(\Delta\tilde{\mathrm{S}}^*, m^*, x^*) \leftarrow (0,\text{null},\text{null})$
  foreach $x_j \notin \{m_1, \ldots, m_k\}$ **do** // each non-medoid
    $\Delta\tilde{\mathrm{S}}_i, \ldots, \Delta\tilde{\mathrm{S}}_k \leftarrow (\Delta\tilde{\mathrm{S}}^{-m_1}, \ldots, \Delta\tilde{\mathrm{S}}^{-m_i})$ // use removal loss
    $\Delta\tilde{\mathrm{S}}^{+x_j} \leftarrow 0$ // initialize shared accumulator
    foreach $x_o \in \{x_1, \ldots, x_n\}$ **do**
      $d_{oj} \leftarrow d(x_o, x_j)$ // distance to new medoid
      if $d_{oj} < d_1(o)$ **then** // new closest
        $\Delta\tilde{\mathrm{S}}^{+x_j} \leftarrow \Delta\tilde{\mathrm{S}}^{+x_j} + d_1(o)/d_2(o) - d_{oj}/d_1(o)$
        $\Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} + d_{oj}/d_1(o) + d_2(o)/d_3(o) - (d_1(o) + d_{oj})/d_2(o)$
        $\Delta\tilde{\mathrm{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_2(o)$
      else if $d_{oj} < d_2(o)$ **then** // new first/second closest
        $\Delta\tilde{\mathrm{S}}^{+x_j} \leftarrow \Delta\tilde{\mathrm{S}}^{+x_j} + d_1(o)/d_2(o) - d_1(o)/d_{oj}$
        $\Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} + d_1(o)/d_{oj} + d_2(o)/d_3(o) - (d_1(o) + d_{oj})/d_2(o)$
        $\Delta\tilde{\mathrm{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_2(o)$
      else if $d_{oj} < d_3(o)$ **then** // new second/third closest
        $\Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{nearest}(o)} + d_2(o)/d_3(o) - d_2(o)/d_{oj}$
        $\Delta\tilde{\mathrm{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\mathrm{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_{oj}$
    $i \leftarrow \arg\max \Delta\tilde{\mathrm{S}}_i$
    $\Delta\tilde{\mathrm{S}}_i \leftarrow \Delta\tilde{\mathrm{S}}_i + \Delta\tilde{\mathrm{S}}^{+x_j}$
    if $\Delta\tilde{\mathrm{S}}_i > \Delta\tilde{\mathrm{S}}^*$ **then** $(\Delta\tilde{\mathrm{S}}^*, m^*, x^*) \leftarrow (\Delta\tilde{\mathrm{S}}, m_i, x_j)$
  break outer loop if $\Delta\tilde{\mathrm{S}}^* \leq 0$
  swap roles of medoid m^* and non-medoid x^* // perform swap
  $\tilde{\mathrm{S}} \leftarrow \tilde{\mathrm{S}} + \Delta\tilde{\mathrm{S}}^*$
return $\tilde{\mathrm{S}}, M$

$$+ \sum_{\text{second}(o)=i} \begin{cases} \frac{d_1(o)}{d_3(o)} - \frac{d_1(o)}{d_2(o)} & \text{if } d(o,j) < d_1(o) \\ \frac{d_1(o)}{d_3(o)} - \frac{d_1(o)}{d_2(o)} & \text{elif } d(o,j) < d_2(o) \\ \frac{d_1(o)}{d_3(o)} - \frac{d_1(o)}{d(o,j)} & \text{elif } d(o,j) < d_3(o) \\ 0 & \text{otherwise} \end{cases} .$$

Computing these additional summands takes $O(N)$ time by iterating over all objects x_o, and adding their contributions to accumulators for $n_1(o)$ and $n_2(o)$.

Once we have identified the best swap, we apply this change and enter the next iteration. In Line 2, the update can also be optimized to use the cached values and only scan for the third closest if one of the closest was removed.

This then gives Algorithm 8, which computes $\Delta\tilde{\mathrm{S}}^{+x_j}$ along with the sum of $\Delta\tilde{\mathrm{S}}^{-m_i}$ and these correction terms in an accumulator array. To help with reimplementing our approach efficiently, we give the final simplified equations in the pseudocode, as the intuition has already been explained above. The algorithm needs $O(k)$ memory for the accumulators in the loop, and $O(N)$ additional memory to store the cached n_1, n_2, d_1, d_2, and d_3 for each object.

This algorithm gives the same result, but FastMSC is $O(k^2)$ faster than the naive PAMMEDSIL, as evidenced by the two main nested loops of FastMSC having $(N-k) \times N$ executions containing only $O(1)$ operations inside.

5.2.4 Eager Swapping and Random Initialization

We can now integrate further improvements by Schubert and Rousseeuw [SR21]. Because doing the best swap (steepest descent) does not appear to commonly find better solutions, but requires a pass over the entire data set for each step, we can converge to local optima much faster if we perform every swap that yields an improvement, even though this means we may repeatedly replace the same medoid and perform "unnecessary" swaps, because the cost for searching is substantially higher than for performing a swap. For PAM this was called eager swapping, and yields the variant FasterPAM. This does not improve theoretical run time (the last iteration will always require a pass over the entire data set to detect convergence), but empirically reduces the number of iterations substantially, while increasing the number of swaps only slightly. It will no longer find the same results, but there is no evidence that a steepest descent is beneficial over choosing the first descent found. The main downside to this is, that it increases the dependency on the data ordering, and hence is best used on shuffled data when run repeatedly. Similarly, we will study a variant that eagerly performs the first swap that improves the AMS as Fast and Eager Medoid Silhouette Clustering (FasterMSC).

Similar to Schubert and Rousseeuw [SR21], where the PAM BUILD initialization had become a bottleneck, we also choose a random initialization. A single pass over the data set with eager swapping tends to find better solutions than the best initialization strategies, and uniform sampling of medoids is very cheap.

5.2.5 Choosing the Number of Clusters

There are many different approaches to determine the optimal number of clusters, yet this remains a challenging task. For k-means, the so-called Elbow method is commonly called, but is not very well suited for this purpose, and some alternatives have been surveyed and evaluated by Schubert [Sch23]. Because the primary objective of k-means (i.e., the sum of squared deviations from the nearest mean) as well as that of k-medoids (the sum of deviations from the nearest medoid) improves

Algorithm 9: Fast and Eager Medoid Silhouette Clustering (FasterMSC): FastMSC with eager swapping

$x_{\text{last}} \leftarrow$invalid
foreach x_o **do** update $\text{nearest}(o), \text{second}(o), d_1(o), d_2(o), d_3(o)$
$\Delta\tilde{\text{S}}^{-m_1}, \dots, \Delta\tilde{\text{S}}^{-m_i} \leftarrow$ compute loss change removing m_i using (5.1)
repeat
  foreach $x_j \notin \{m_1, \dots, m_k\}$ **do** // each non-medoid
    break outer loop if $x_j = x_{last}$
    $\Delta\tilde{\text{S}} \leftarrow (\Delta\tilde{\text{S}}^{-m_1}, \dots, \Delta\tilde{\text{S}}^{-m_i})$ // use removal loss
    $\Delta\tilde{\text{S}}^{+x_j} \leftarrow 0$ // initialize shared accumulator
    foreach $x_o \in \{x_1, \dots, x_n\}$ **do**
      $d_{oj} \leftarrow d(x_o, x_j)$ // distance to new medoid
      if $d_{oj} < d_1(o)$ **then** // new closest
        $\Delta\tilde{\text{S}}^{+x_j} \leftarrow \Delta\tilde{\text{S}}^{+x_j} + d_1(o)/d_2(o) - d_{oj}/d_1(o)$
        $\Delta\tilde{\text{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{nearest}(o)} + d_{oj}/d_1(o) + d_2(o)/d_3(o) - (d_1(o) + d_{oj})/d_2(o)$
        $\Delta\tilde{\text{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_2(o)$
      else if $d_{oj} < d_2(o)$ **then** // new first/second closest
        $\Delta\tilde{\text{S}}^{+x_j} \leftarrow \Delta\tilde{\text{S}}^{+x_j} + d_1(o)/d_2(o) - d_1(o)/d_{oj}$
        $\Delta\tilde{\text{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{nearest}(o)} + d_1(o)/d_{oj} + d_2(o)/d_3(o) - (d_1(o) + d_{oj})/d_2(o)$
        $\Delta\tilde{\text{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_2(o)$
      else if $d_{oj} < d_3(o)$ **then** // new second/third closest
        $\Delta\tilde{\text{S}}_{\text{nearest}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{nearest}(o)} + d_2(o)/d_3(o) - d_2(o)/d_{oj}$
        $\Delta\tilde{\text{S}}_{\text{second}(o)} \leftarrow \Delta\tilde{\text{S}}_{\text{second}(o)} + d_1(o)/d_3(o) - d_1(o)/d_{oj}$
    $i \leftarrow \arg\max \Delta\tilde{\text{S}}_i$ // choose best medoid
    $\Delta\tilde{\text{S}}_i \leftarrow \Delta\tilde{\text{S}}_i + \Delta\tilde{\text{S}}^{+x_j}$ // add accumulator
    if $\Delta\tilde{\text{S}}_i < 0$ **then** // eager swapping
      swap roles of medoid m_i and non-medoid x_o // perform swap
      $\tilde{\text{S}} \leftarrow \tilde{\text{S}} + \Delta\tilde{\text{S}}^*$
      update $\Delta\tilde{\text{S}}^{-m_1}, \dots, \Delta\tilde{\text{S}}^{-m_i}$
      $x_{\text{last}} \leftarrow x_o$
return $\tilde{\text{S}}, M$

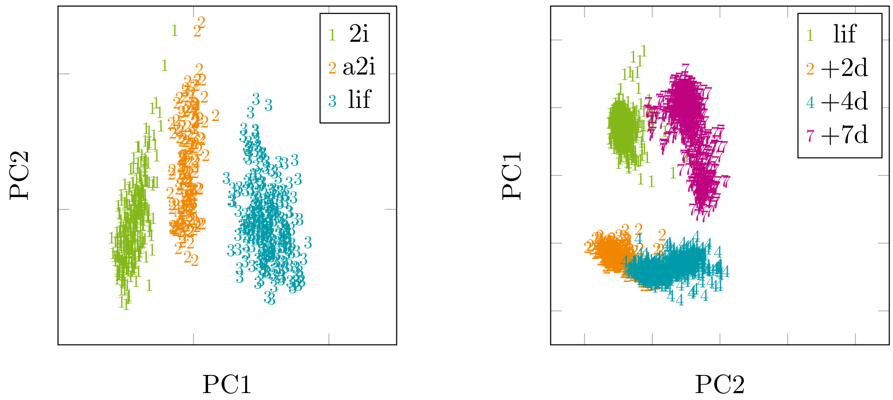
as we increase the number of clusters, it is common to rely on a secondary quality criterion, i.e., an evaluation measure, that does not have this property. One popular such measure is indeed the Silhouette (e.g., [AGM+13; BSH+07]), which tends to drop once clusters get too close to each other, unless they are well separated.

A repeated computation of the Silhouette is expensive on larger data sets. Instead of repeatedly running Fast Medoid Silhouette Clustering (FastMSC)

Algorithm 10: DynMSC: FasterMSC for dynamic k

$k \leftarrow \max k$
while $k \geq 2$ **do**
  $x_{\text{last}} \leftarrow$invalid
  foreach x_o **do**
    update $\text{nearest}(o), \text{second}(o), d_1(o), d_2(o), d_3(o)$
  $\Delta\tilde{\mathrm{S}}^{-m_1}, \ldots, \Delta\tilde{\mathrm{S}}^{-m_i} \leftarrow$ compute loss change (5.1)
  repeat
    foreach $x_j \notin \{m_1, \ldots, m_k\}$ **do**
      break outer loop if $x_j = x_{last}$
      $\Delta\tilde{\mathrm{S}} \leftarrow (\Delta\tilde{\mathrm{S}}^{-m_1}, \ldots, \Delta\tilde{\mathrm{S}}^{-m_i})$
      $\Delta\tilde{\mathrm{S}}^{+x_j} \leftarrow 0$
      foreach $x_o \in \{x_1, \ldots, x_n\}$ **do**
        $\vdots$
        `// see FasterMSC`
      $i \leftarrow \arg\max \Delta\tilde{\mathrm{S}}_i$
      $\Delta\tilde{\mathrm{S}}_i \leftarrow \Delta\tilde{\mathrm{S}}_i + \Delta\tilde{\mathrm{S}}^{+x_j}$
      if $\Delta\tilde{\mathrm{S}}_i < 0$ **then**
        swap roles of m_i and x_o
        $\tilde{\mathrm{S}} \leftarrow \tilde{\mathrm{S}} + \Delta\tilde{\mathrm{S}}^*$
        update $\Delta\tilde{\mathrm{S}}^{-m_1}, \ldots, \Delta\tilde{\mathrm{S}}^{-m_i}$
        $x_{\text{last}} \leftarrow x_o$
  $\hat{\mathrm{S}}_k \leftarrow \tilde{\mathrm{S}}$
  $\hat{\mathrm{M}}_k \leftarrow M$
  $i \leftarrow \arg\max \Delta\tilde{\mathrm{S}}$ `// choose medoid to remove`
  remove medoid m_i `// choose k with highest AMS`
  $k \leftarrow k - 1$
$j \leftarrow \arg\max \hat{\mathrm{S}}$
return $\hat{\mathrm{S}}_j, \hat{\mathrm{M}}_j$

with a different number of clusters and keeping the best result, we present a variant of FastMSC that does not require the number of clusters to be given. In DynMSC, we begin with a maximum number of clusters, optimize the Average Medoid Silhouette (AMS), then decrease the number of clusters by one, and repeat until we have reached a minimum number of clusters. During this process, we store the solution with the highest AMS to return later. However, we integrate this directly with FasterMSC to save redundant computations. At the end of the FasterMSC optimization, we already know the removal loss of each medoid (c.f., Eq. (5.1)), and instead of removing a random medoid, we can remove the one that incurs the least reduction in AMS, which gives us better starting conditions. We can also retain



(a) Kolodziejczyk et al. [KKT+15] (b) Klein et al. [KMA+15]

Figure 5.1: Different kind of Mouse Embryonic Stem Cells (mESCs). For both data sets we have done PCA and plot the first two principal components. (a) shows 704 mESCs grown in three different conditions and (b) 2717 mESCs at the moment of leukemia inhibitory factor (LIF) withdrawal, 2 days after, 4 days after, and 7 days after.

some of the cached data and only need to find the nearest medoids for those, where one of the three nearest medoids has been removed. For others, this metadata remains valid. Removing a medoid is very similar to performing a swap, except that there is no replacement medoid. We then can continue with the reduced k instead of restarting from scratch. Algorithm 10 gives a pseudocode for DynMSC. In Line 12 we omitted code of the inner loop that is identical to FasterMSC.

5.3 Experiments

We next evaluate clustering quality, to show the benefits of optimizing AMS. We report both AMS and ASW, as well as the supervised measures Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) that require labeled data. Afterward, we study the scalability to verify the empirical speedup for our algorithms FastMSC, FasterMSC, and DynMSC.

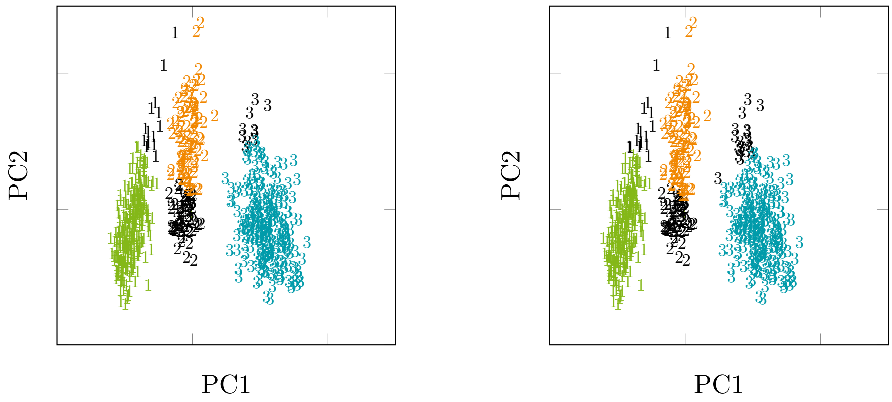
5.3.1 Data Sets

Since it became possible to map gene expression at the single-cell level by RNA sequencing, clustering on these has become a popular task, and Silhouette is a popular evaluation measure there. Hie, Bryson, and Berger [HBB19] use Silhouette coefficient distributions to compare 26 diverse Single-Cell RNA Sequencing (scRNA-seq) data sets under different parameters. scRNA-seq provides high-dimensional data that requires appropriate preprocessing to extract information. After extraction of significant genes, these marker genes are validated by clustering of proper cells. We explore two publicly available data sets with larger sample size (by scRNA-seq standards; the data size is not challenging for cluster analysis) of scRNA-seq of Mouse Embryonic Stem Cells (mESCs). Kolodziejczyk et al. [KKT+15] studied 704 mESCs with 38561 genes grown in three different conditions (2i, a2i, and serum). Klein et al. [KMA+15] worked on the influence LIF withdrawal on mESCs. For this, they studied a total of 2717 mESCs with 24175 genes. The data included 933 cells after LIF withdrawal, 303 cells two days after, 683 cells 4 days after, and 798 cells 7 days after. We normalize each cell by the total counts over all genes, so that every cell has a total count equal to the median of total counts for observations (cells) before normalization, then we perform Principal Component Analysis (PCA) and use the first three principal components for clustering. This preprocessing matches the procedure of Kolodziejczyk et al. [KKT+15] in clustering the mESCs grown in serum, 2i, and a2i media from the original publication.

To test the scalability of our new variants, we need larger data sets. We use the well-known MNIST data set, with 784 features and 60000 samples (PAMSIL will not be able to handle this size in reasonable time). We implemented our algorithms in Rust, extending the `kmedoids` package [SL22], wrapped with Python, and we make our source code available in this package. We perform all computations in the same package, to avoid confounding factors caused by comparing two different implementations [KSZ17]. We run 10 restarts on an AMD EPYC 7302 processor using a single thread and evaluate the average values.

5.3.2 Clustering Quality

We evaluated all methods with PAM BUILD initialization and uniform random initialization. To evaluate the relevancy of the Average Silhouette Width (ASW) and the Average Medoid Silhouette (AMS), we compare to the true labels using the ARI and NMI, two common external measures in clustering.
We first discuss the results for the data set from Kolodziejczyk et al. [KKT+15], shown in Table 5.1. As expected, PAMSIL found the best result with respect to ASW, while the Medoid Silhouette based methods found better results regarding AMS. PAM, which optimizes the total deviation, found slightly worse results in all measures. Regarding the known labels, the highest ARI and NMI scores are achieved by the Medoid Silhouette methods. The different initializations produced the same results for all methods except PAMSIL here. Because of the small $k = 3$, the speedup of FastMSC over PAMMEDSIL is only small, the additional speedup of FasterMSC is due to reducing the number of iterations. FasterMSC was able to find the best solutions, but at a 1255× faster run time than PAMSIL, confirming the expected improvements. Because AMS and ASW are correlated, with rather small differences between the solutions found by the methods, we argue that AMS is a suitable approximation for ASW, at a much reduced run time.



(a) Results for PAMMEDSIL (BUILD) (b) Results for PAMSIL (BUILD)

Figure 5.2: Clustering results for the scRNA-seq data sets of Kolodziejczyk et al. [KKT+15] for PAMMEDSIL and PAMSIL. All correctly predicted labels are colored by the corresponding cluster and all errors are marked as black.

Table 5.1: Clustering results for the scRNA-seq data sets of Kolodziejczyk et al. for PAM, PAMSIL, and all variants of PAMMEDSIL. All methods are evaluated for BUILD and Random initialization, and true known k=3.

Algorithm	Init.	AMS	ASW	ARI	NMI	run time (ms)
PAM	BUILD	0.6566	0.5397	0.6944	0.6549	18.26
PAM	Random	0.6566	0.5397	0.6944	0.6549	22.67
PAMMEDSIL	BUILD	**0.6747**	0.5474	**0.7174**	**0.6953**	62.63
PAMMEDSIL	Random	**0.6747**	0.5474	**0.7174**	**0.6953**	61.91
FastMSC	BUILD	**0.6747**	0.5474	**0.7174**	**0.6953**	25.09
FastMSC	Random	**0.6747**	0.5474	**0.7174**	**0.6953**	24.67
FasterMSC	BUILD	**0.6747**	0.5474	**0.7174**	**0.6953**	**9.95**
FasterMSC	Random	**0.6747**	0.5474	**0.7174**	**0.6953**	10.95
PAMSIL	BUILD	0.6490	**0.5507**	0.6962	0.6677	12493.86
PAMSIL	Random	0.5799	0.5490	0.6652	0.6633	16045.47

Table 5.2: Clustering results for the scRNA-seq data sets of Klein et al. for PAM, PAMSIL and all variants of PAMMEDSIL. All methods are evaluated for BUILD and Random initialization and true known k=4.

Algorithm	Init.	AMS	ASW	ARI	NMI	run time (ms)
PAM	BUILD	0.7673	0.6825	0.8450	0.8726	355.55
PAM	Random	0.7348	0.6292	0.8343	0.8526	476.18
PAMMEDSIL	BUILD	**0.7748**	0.6834	0.8441	0.8685	2076.15
PAMMEDSIL	Random	**0.7748**	0.6834	0.8441	0.8685	3088.77
FastMSC	BUILD	**0.7748**	0.6834	0.8441	0.8685	212.01
FastMSC	Random	**0.7748**	0.6834	0.8441	0.8685	305.00
FasterMSC	BUILD	**0.7748**	0.6834	0.8441	0.8685	163.74
FasterMSC	Random	**0.7748**	0.6834	0.8441	0.8685	**122.63**
PAMSIL	BUILD	0.7649	**0.6838**	**0.8483**	**0.8739**	2026025.10
PAMSIL	Random	0.7220	0.6837	0.8472	0.8724	1490354.10

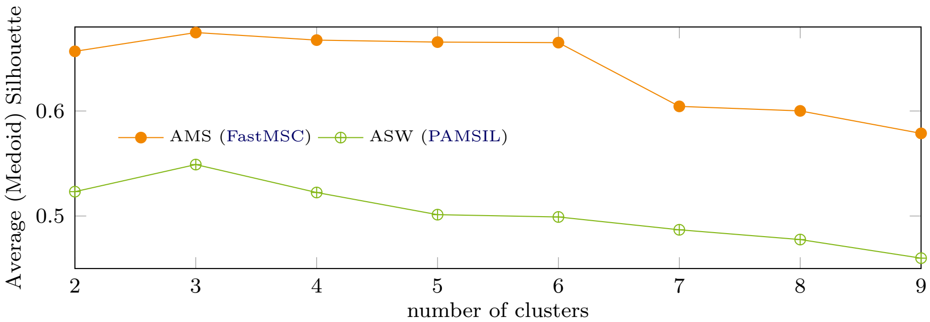
Since there were no variations in the resulting medoids for the different restarts of the experiment, we can easily compare single results visually. Figure 5.2b compares the results of PAMMEDSIL/FastMSC and PAMSIL, showing in black which points are clustered differently than in the given labels. As indicated by the similar evaluation scores, both clusterings are very similar, with class 1 captured slightly better in one, class 3 slightly better in the other result.

Table 5.2 shows the clustering results for the scRNA-seq data sets of Klein et al. [KMA+15]. In contrast to Kolodziejczyk et al. data set, we here obtain a slightly higher ARI and NMI for PAMSIL than for the AMS optimization methods. Again, the results are very similar, with the ASW obtained from the AMS methods being almost identical. The differences in AMS are more pronounced. Because this data set is larger, the gap in run time becomes more pronounced. While FasterMSC still finishes in less than a second, PAMMEDSIL now takes multiple seconds to complete, and PAMSIL already needs half an hour to complete. Because of the larger data set size, FasterMSC has become 16521× faster than PAMSIL and 6× faster than PAMMEDSIL.

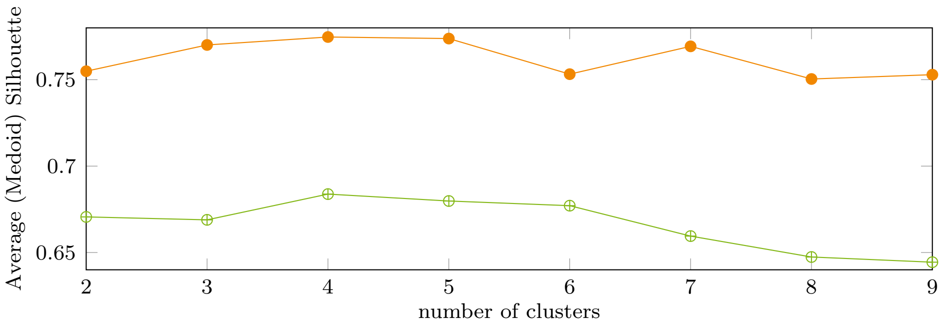
5.3.3 Number of Clusters

The AMS and ASW are based on very similar ideas of cluster quality, and while AMS is *not a numerical* approximation of the ASW, it usually produces a similar ranking of clusterings. Therefore, if the ASW on a data set is a suitable heuristic to determine the optimal number, we can assume that the AMS is also a good heuristic for this purpose.

To evaluate how well the AMS is suited to choose the optimal number of clusters, we perform FastMSC and PAMSIL on the Klein et al. and Kolodziejczyk et al. data



(a) Kolodziejczyk et al. data set contains **three** different groups of Mouse Embryonic Stem Cells.



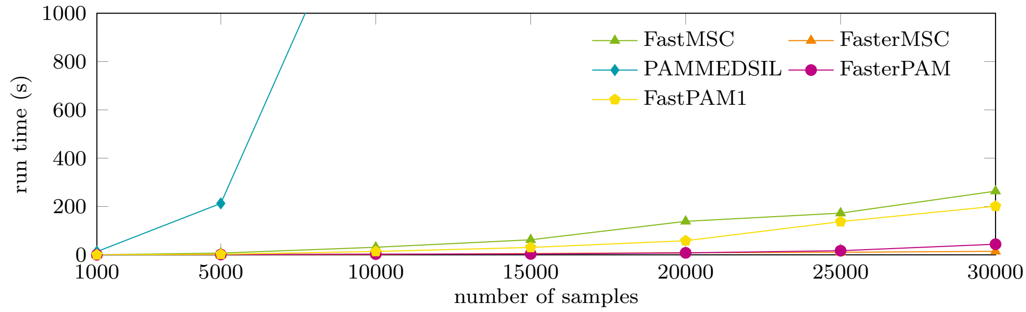
(b) Klein et al. data set contains embryonic stem cells measured at **four** different time points.

Figure 5.3: Average Medoid Silhouette and Average Silhouette Width for different number of Clusters with FastMSC and PAMSIL.

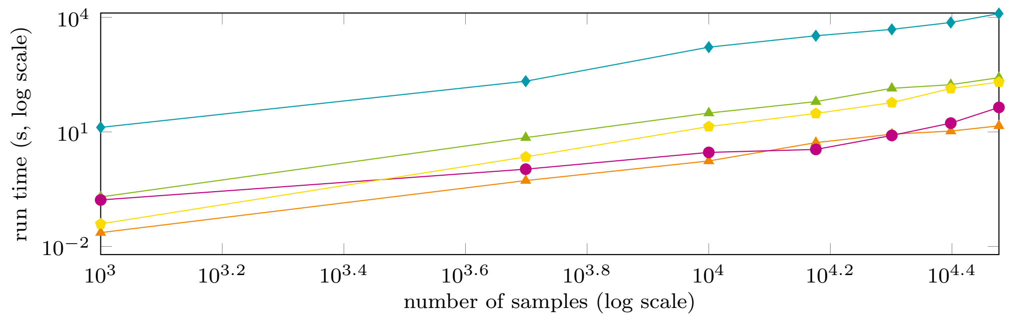
sets for $k = 2$ to 9 clusters. Klein et al. data set contains four different groups of data points and Kolodziejczyk et al. data set contains three different groups. As seen in Figure 5.3, FastMSC finds the largest AMS as well as PAMSIL the largest ASW at the correct number of clusters for both data sets.

5.3.4 Scalability

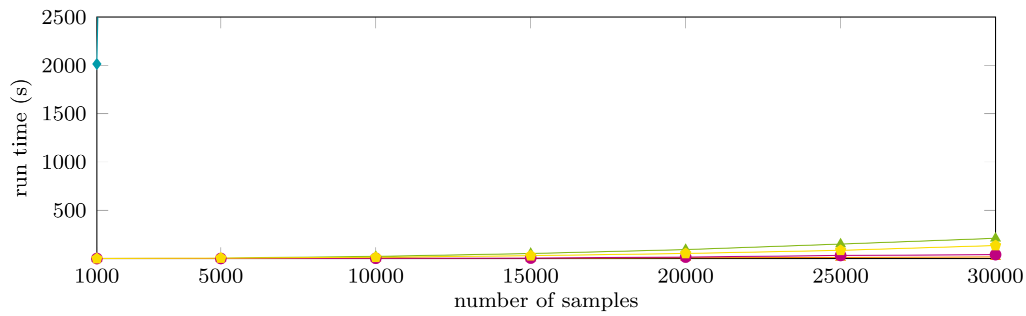
To evaluate the scalability of our methods, we use the well-known MNIST data, which has 784 variables (28×28 pixels) and 60000 samples. We use the first $N = 1000$ to 30000 samples and compare $k = 10$ and $k = 100$. Due to its high run time, PAMSIL is not able to handle this size in a reasonable time. In addition to the methods for direct AMS optimization, we include the FastPAM1 and FasterPAM algorithms [SR21; SR19a]. For all methods we use random initialization. As expected, all methods scale approximately quadratic in the sample size N. FastMSC is on average $50.66\times$ faster than PAMMEDSIL for $k = 10$ and $10464.23\times$ faster for $k = 100$, supporting the expected $O(k^2)$ improvement by removing the nested loop and caching the distances to the nearest centers. For FasterMSC we achieve even $639.34\times$ faster run time than for PAMMEDSIL for $k = 10$ and $78035.01\times$ faster



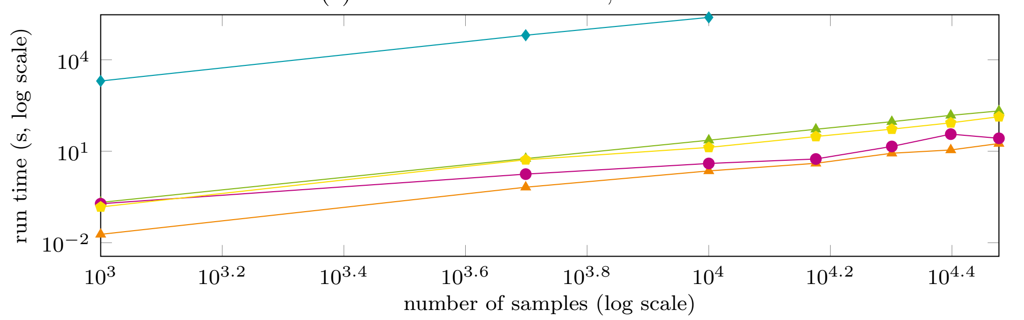
(a) run time with k=10, linear scale



(b) run time with k=10, log-log plot

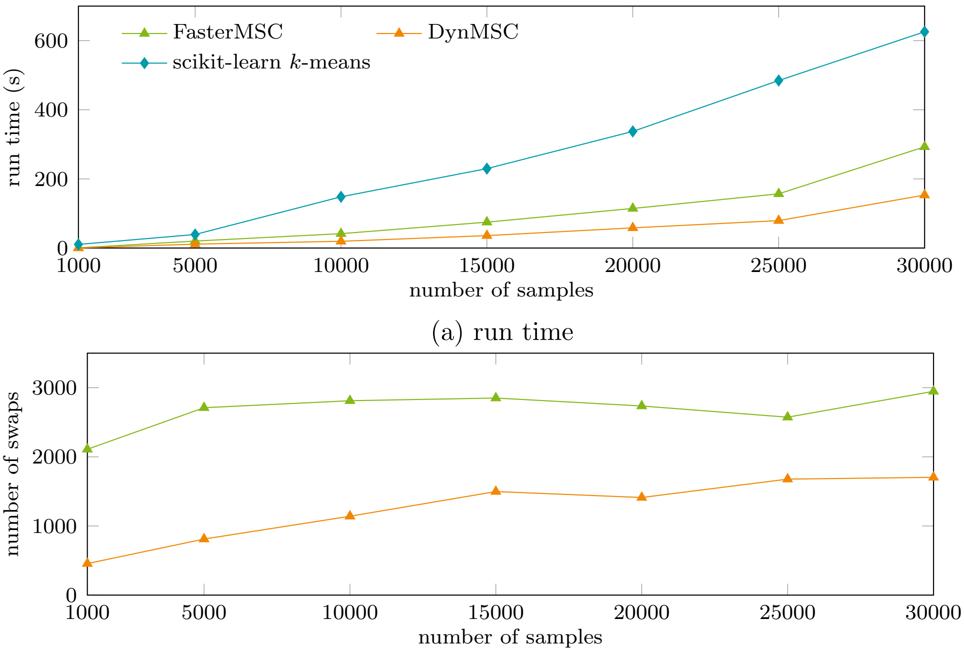


(c) run time with k=100, linear scale



(d) run time with k=100, log-log plot

Figure 5.4: Run time on MNIST data (time out 24 hours)



(b) number of swaps for DynMSC and FasterMSC

Figure 5.5: Run time and number of swaps on MNIST data for 1000 to 30000 samples. Comparing DynMSC, FasterMSC, and k-means in scikit-learn including Silhouette score calculation. We evaluate $k = 2$ to $k = 50$, for FasterMSC single calls with random initialization, for scikit-learn single calls with *kmeans++* initialization with calling the Silhouette score, and DynMSC with maximum $k = 50$ and random initialization.

run time for $k = 100$. We expect FastPAM1 and FastMSC and also FasterPAM and FasterMSC to have similar scalability; but since MSC also needs the third nearest neighbor, it needs to maintain more data and access more memory. We observe that FastPAM1 is 2.50× faster than FastMSC for $k = 10$ and 1.57× faster for $k = 100$, which is larger than expected and due to more iterations necessary for convergence in the MSC methods: FastPAM1 needs on average 14.86 iterations while FastMSC needs 33.48. In contrast, FasterMSC is even 1.65× faster than FasterPAM for $k = 10$ and 1.96× faster for $k = 100$.

To evaluate the scalability of DynMSC, we test it on MNIST for 1000 to 30000 samples for a maximum $k = 50$. We compare it to FasterMSC with random initialization run once for each $k = 2$ to $k = 50$ (keeping the best). As additional baseline, we include a naive approach using scikit-learn (version 1.2.2) k-means for clustering, and Silhouette only for choosing the number of clusters. We observe that DynMSC is 1.97× faster than repeatedly running FasterMSC. This speedup is primarily due to requiring 2.15× fewer swaps. The difference in the factor between swaps and run time is explained by additional remove operations in DynMSC, which are similarly expensive as the swaps and have to be performed 48× for the range of $k = 2 \ldots 50$. DynMSC is on average 9.01× faster than the popular sklearn routine because of the cost to repeatedly compute the Silhouette.

5.4 Discussion

We showed that the Average Medoid Silhouette satisfies desirable theoretical properties for Clustering-Quality Measures, and as an approximation of the Average Silhouette Width yields desirable results on real problems from gene expression analysis. We propose a new algorithm for optimizing the Average Medoid Silhouette, which provides a run time speedup of $O(k^2)$ compared to the earlier PAMMEDSIL algorithm by caching the nearest centers and of partial results based on FasterPAM. This makes clustering by optimizing the Medoid Silhouette possible on much larger data sets than before. The Medoid Silhouette can also be used to determine the number of clusters in a data set, and the DynMSC algorithm introduced in this article optimizes this process by avoiding redundant computations. The ability to optimize a variant of the popular Silhouette measure directly demonstrates the underlying property that any internal cluster evaluation measure specifies a clustering itself, and that the proper unsupervised evaluation of clusterings remains an unsolved problem. But since Vendramin, Campello, and Hruschka [VCH10] found the Silhouette (and its variants) to be among the best and most robust clustering quality criteria, direct optimization of the Medoid Silhouette may be desirable.

Users of cluster analysis are advised to carefully choose the right evaluation measure and clustering method for their problem rather than following a "default" recommendation. Even though the Silhouette appears to be popular and scores high in benchmarks [VCH10], it may not the best choice for every problem. In particular Silhouette may perform poorly when (1) distance is not measured appropriately, (2) data preprocessing is poor, (3) clusters have non-convex shape, (4) clusters vary substantially in diameter, (5) clusters exist only in subspaces, or (6) a hierarchy of clustering structures exists.

Chapter 6

Fast k-Nearest-Neighbor-Consistent Clustering

In this chapter, we are interested in a relatively unknown quality criterion called the kNN Consistency [DH04], which measures if the nearest neighbors of each point belong to the same cluster. The kNN Consistency was introduced in Chapter 3.2.3. This is closely related to the central idea of clustering that similar objects should be in the same cluster, but also to nearest-neighbor classification, as we would then predict the correct class. Ding and He [DH04] also proposed two clustering algorithms to optimize for this criterion. In this article, we propose a fast variant of the consistency-preserving k-means algorithm, that provably yields the same results. We first introduce clustering with consistency in Section 6.1. We then propose a faster algorithm in Section 6.2. We show experimental evidence of the performance benefits in Section 6.3 and conclude the paper in Section 6.4.

6.1 Clustering with Nearest Neighbor Consistency

As already mentioned, the ENFORCE algorithm modifies an existing clustering, which can be created by any cluster analysis method, to improve its consistency by reassigning neighborhoods to the most common cluster. In contrast to ENFORCE, k-means-CP does not work on an existing clustering but integrates consistency into the standard k-means procedure. k-means uses the quadratic Euclidean distance, although there are other approaches, such as spherical k-means [DM01; SLF21], they will not be considered here. First, initial cluster centers $\mathrm{M} = \{\mu_1, ..., \mu_k\}$ are chosen with any of the standard heuristics. Then an alternating optimization as in the standard algorithm is performed. But in order to obtain a kNN-consistent result, k-means-CP, like ENFORCE, always assigns entire closed neighborhoods to the same cluster. Thus, Ding and He [DH04] define the nearest cluster center nearest(S_i) of a closed neighborhood S_i by the sum of squared euclidean distances

$$\operatorname{nearest}(S_i) = \arg\min_j \sum_{x \in S_i} \|x - \mu_j\|^2 \ .$$

All points in the closed neighborhood S_i are then assigned to this cluster. The assignment step is alternated with a recalculation of the cluster centers. A new

Algorithm 11: Consistency-Preserving k-means (k-means-CP)

$C \leftarrow$ initialize k cluster
$S \leftarrow$ calculate sets of closed neighborhoods
$l \leftarrow$ dummy assignments to non-existent cluster
repeat
  foreach $S_i \in S = \{S_1, \ldots, S_h\}$ **do** `// closed neighborhoods`
    $N \leftarrow$null
    foreach $\mu_j \in M = \{\mu_1, \ldots, \mu_k\}$ **do** `// determine nearest cluster`
      foreach $x_u \in S_i$ **do**
        $N_j \leftarrow N_j + \|x_u - \mu_j\|^2$
    $o \leftarrow \arg\min N$
    foreach $x_j \in S_i$ **do** $l_j \leftarrow o$ `// assign closed neighborhoods`
  if l *is unchanged* **then break**
  $M \leftarrow$ calculate cluster centers for C
return L `// final cluster assignments`

cluster center μ_j of the cluster C_j is determined as usual in k-means using the arithmetic average of the assigned data

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x \ .$$

These steps are repeated until no point is reassigned (and hence the cluster centers do not change anymore). Convergence guarantees of the standard algorithm still apply. This algorithm computes exactly $N \cdot k$ distances in each step to determine the nearest clusters, and these distance computations make up a major part of the algorithm's run time. We can also optimize the recomputation of the cluster centers by using an incremental computation, but this has much less effect. In the following, we discuss why many of the distance computations performed above are unnecessary, and we can hence improve the run time of this algorithm.

6.2 Fast k-Nearest-Neighbor-Consistent Clustering

The Consistency-Preserving k-means (k-means-CP) algorithm determines the nearest cluster of a closed neighborhood set by the sum of the squared Euclidean distances of all elements of a closed neighborhood set to the different cluster centers. Using the parallel axis theorem of König, Huygens and Steiner, we can prove that instead of considering the distances of all samples in a closed neighborhood set S_i, it is sufficient to consider only the mean vector $\bar{s_i} = \frac{1}{|S_i|} \sum_{x \in S_i} x$. We can also trivially use the mean vector for updating the cluster centers, if we weight it by the number of points in the closed neighborhood set. The same property has been previously used by Lee, Kao, and Cheng [LKC07] to reduce uncertain UK-means clustering

Algorithm 12: Neighbor-Consistent K-means (NCK-means)

$C \leftarrow$ initialize k cluster
$\bar{S} \leftarrow$ calculate representatives of closed neighborhood sets
repeat
  foreach $\bar{s}_i \in \bar{S} = \{\bar{s}_1, \ldots, \bar{s}_h\}$ **do** `// representatives of closed neighborhoods`
    $N \leftarrow$null
    foreach $\mu_j \in M = \{\mu_1, \ldots, \mu_k\}$ **do** `// determine nearest cluster`
      $N_j \leftarrow N_j + \|\bar{s}_i - \mu_j\|^2$
    $o \leftarrow \arg\min N$
    foreach $x_j \in S_i$ **do** $l_j \leftarrow o$ `// assign closed neighborhoods`
  if l *is unchanged* **then break**
  $M \leftarrow$ calculate cluster centers for C
return L `// final cluster assignments`

to regular k-means clustering. Our improved Neighbor-Consistent K-means (NCK-means) hence uses the mean vectors of the closed neighborhood sets. This yields a faster variant of consistency-preserving k-means with a substantially lower run time and the guaranteed same result.

We will reference the set which contains all closed neighborhood sets by S. Because all elements of a closed neighborhood set will be assigned to the same cluster, we will reference the set that contains all closed neighborhood sets whose elements have cluster label i by C_i.

6.2.1 Proof of correctness

To prove the equivalence of the results of NCK-means and k-means-CP, we show that the substeps of the algorithm always produce the same results. We first show that for given cluster centers, NCK-means and k-means-CP assign the same cluster labels to a closed neighborhood set. We start by considering how the original k-means-CP assigns cluster labels:

$$S_j \in C_i \iff i = \arg\min_i \sum_{x \in S_j} \|x - \mu_i\|^2 \ .$$

NCK-means assigns its cluster labels by the equivalent optimization:

$$S_j \in C_i \iff i = \arg\min_i \|\bar{s_j} - \mu_i\|^2 \ .$$

This equivalence easily follows from the following version of the parallel axis theorem:

$$\sum_{x \in X} |x - a|^2 = \sum_{x \in X} \left(|x - \bar{X}|^2\right) + N|a - \bar{X}|^2$$

by using the vector form (a sum over all components) and then $a = \mu_i$ and $X = S_j$:

$$\sum_{x \in S_j} \|x - \mu_i\|^2 = \sum_{x \in S_j} \left(\|x - \bar{s_j}\|^2\right) + |S_j| \cdot \|\mu_i - \bar{s_j}\|^2 \ .$$

Because only the last term depends on i, we can omit the others for the minimization over i:

$$\arg\min_i \sum_{x \in S_j} \|x - \mu_i\|^2 = \arg\min_i \|\bar{s_j} - \mu_i\|^2 \ .$$

We can prove the above version of the parallel axis theorem as follows:

$$\begin{aligned}
\sum_{x \in X} |x - a|^2 &= \sum_{x \in X} \left|(x - \bar{X}) - (a - \bar{X})\right|^2 \\
&= \sum_{x \in X} \left((x - \bar{X})^2 - 2(x - \bar{X})(a - \bar{X}) + (a - \bar{X})^2\right) \\
&= \sum_{x \in X} (x - \bar{X})^2 - 2(a - \bar{X}) \underbrace{\sum_{x \in X} (x - \bar{X})}_{=0} + N|a - \bar{X}|^2 \\
&= \sum_{x \in X} |x - \bar{X}|^2 + N|a - \bar{X}|^2
\end{aligned}$$

The classic parallel axis theorem is obtained for $a = 0$.

We now consider the second step, updating the cluster centers. We start with the original calculation of the cluster centers μ_i. k-means-CP calculates this via the mean of the elements.

$$\mu_i = \frac{1}{|\{x_j \mid l_j = i\}|} \sum_{x_j \in X | l_j = i} x_j$$

Because the labels of each closed set are consistent, we can rewrite this to:

$$\mu_i = \frac{1}{\sum_{S_j \in C_i} |S_j|} \sum_{S_j \in C_i} \sum_{x_k \in S_j} x_k$$

and by using $\bar{s_j} = \frac{1}{|S_j|} \sum_{x_k \in S_j} x_k$, we obtain

$$\mu_i = \frac{1}{\sum_{S_j \in C_i} |S_j|} \sum_{S_j \in C_i} |S_j| \bar{s_j} \ .$$

This shows that for closed neighborhood sets with given cluster labels, NCK-means creates the same cluster centers as k-means-CP. Because both steps of the algorithms produce the same results, we have proven that the final result of the algorithms is the same.

6.2.2 Expected run time improvement

The run time improvement depends on the data reduction using closed neighbor sets. The standard k-means algorithm has a run time of $O(NKdi)$ where N is the number of points, k is the number of clusters, d is the dimensionality, and i is the number of iterations. Finding the closed neighbor sets additionally takes $O(N^2)$ time (although indexes for similarity search may yield a considerable speedup), and k-means-CP of Ding and He [DH04] hence has a complexity of $O(N^2 + Nkdi)$. NCK-means reduces this to $O(N^2 + |S|Kdi)$ where $|S|$ is the number of closed neighbor sets, and we must assume $|S| \in O(N)$. In a worst-case asymptotic analysis, the improvements hence are likely negligible because the cost of finding the closed neighbor sets dominates, but in practice, it usually offers a decent run time improvement on the order of $|S|/N$ in the clustering step, at only the cost of little additional memory to store the means of each closed neighbor set. Hence, there is no reason not to use it. As it is a best practice to run k-means several times and keep the best outcome [Sch23], the cost to find the closed neighbor sets can also be amortized over multiple restarts.

The choice of the neighborhood has an important effect on the run time. Finding the nearest neighbors becomes more expensive with a larger neighborhood size, but at the same time increasing the number of edges decreases the number of closed neighbor sets. For NCK-means, a lower number is beneficial, and both algorithms also benefit from an often lower number of iterations because reassignments are less likely to happen with the additional consistency constraints.

6.2.3 Further run-time improvements

Because our proof shows that it is sufficient to use the mean of each closed neighbor set, it is trivially possible to combine this with acceleration techniques of k-means such as the algorithms of Elkan [Elk03], Hamerly [Ham10], or the more recent Exponion [NF16] and Shallot [Bor20] algorithms. The weight can also be used to build a BETULA tree [LS22a]. We have not experimentally verified these additional speedups, as this would distract from the main objective of this paper. Furthermore, the differences between these algorithms will often be small compared to the cost of finding the nearest neighbors of all points to construct the closed neighborhoods in the beginning.

6.3 Experiments

To empirically verify the run time improvements, we implemented both versions with Java in the ELKI 0.8.0 data mining framework [Sch22]. All implementations are within the same codebase to reduce confounding factors and improve comparability [KSZ17]. We perform 10 restarts on an Intel i4690K processor using a single thread and evaluate the average values. We analyze the run time for the *MNIST 784* data set. *MNIST 784* contains grayscale pictures of handwritten digits with a picture size of 28×28 pixels, which is vectorized to a vector of length 784. We compared the algorithms with subsets of sizes $n = 1000, 2000, \ldots, 10000$ for the kNN and kMN neighborhood for $k = 1, 2, 3$ neighbors. The initial cluster centers were

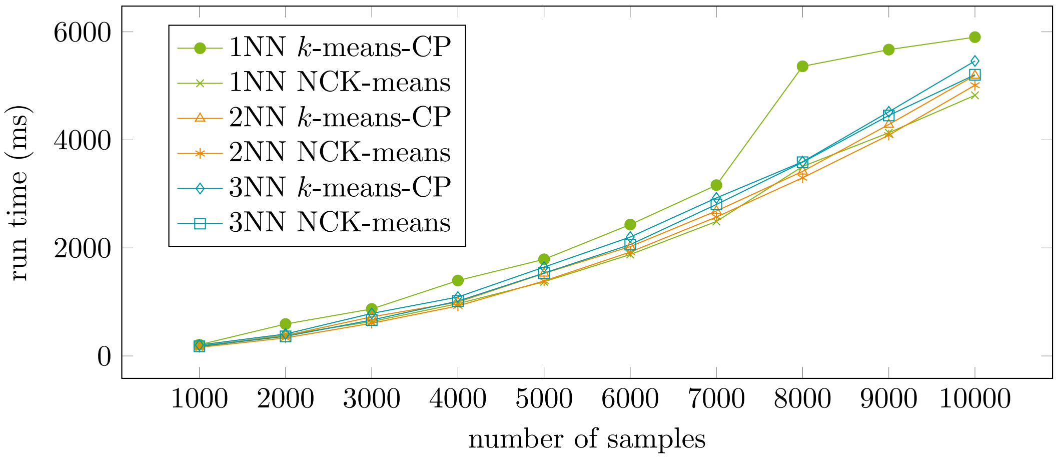


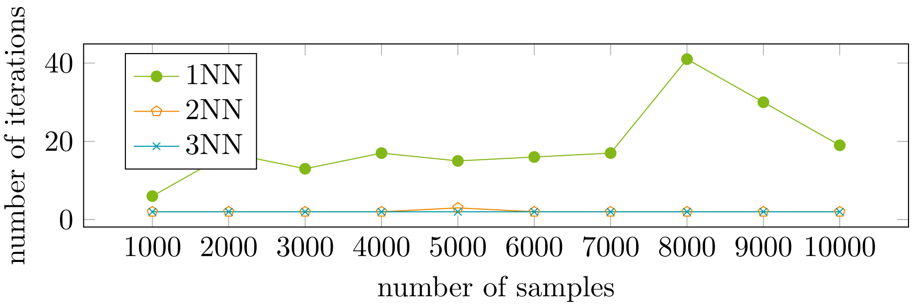
Figure 6.1: Run time of k-means-CP and NCK-means for different number of samples of MNIST, for kNN neighborhood with $k = 1, 2, 3$.

chosen with k-means++[AV07]. The number of clusters is set to $k = 10$ because there are ten digits in the data set.

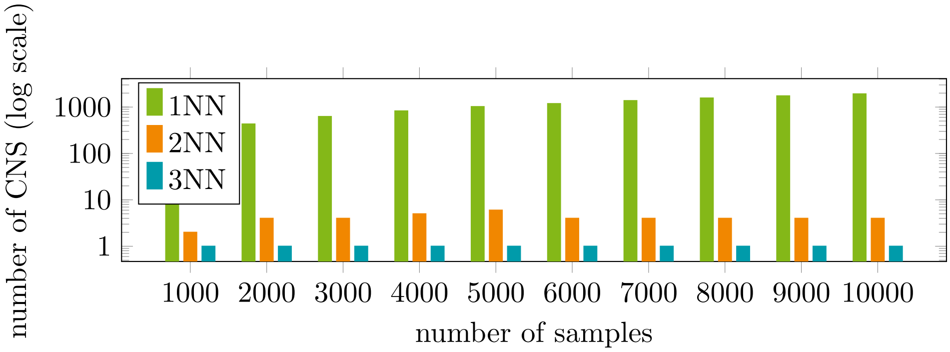
The time measured is the time required for calculating the closed neighborhood sets until the algorithm converges. The time to calculate the kNN of the data set is omitted because it is required for both algorithms and is the most time-consuming part. Therefore time variances in the calculation of the kNNs would overshadow the actual time difference of the algorithms. The ELKI framework automatically uses a k-d-tree or a vantage-point tree to accelerate nearest neighbor search [Sch22], depending on data set characteristics and the distance function used. Because of the dimensionality, ELKI chose a VP-tree automatically for this data set. In Figure 6.5, we plot the run time of 1-Mutual-Nearest-Neighbor (1MN) computation and NCK-means for different number of samples of MNIST data. For MNIST with 784 dimensions, the kNN/kMN computation takes a large part compared to the clustering computation. For MNIST with 784 dimensions, the kNN/kMN computation takes a large share. We expect the kNN/kMN computation to be much more lightly weighted for lower dimensions.

In Figure 6.1, we plot the results for asymmetric kNN. While the run time improved in all cases, there are only noticeable differences for $k = 1$, which get larger at $n = 8000$. The little improvements for $k = 2, 3$ are caused by kNN inducing only very few closed neighborhood sets. For $k = 2$, the maximum amount of six closed neighborhood sets is reached with $n = 5000$. For $k = 3$ all data sets are covered by a single closed neighborhood set. This leads to algorithms only needing one or two iterations until convergence. The spike for $k = 1$ can also be explained by the iterations needed for the algorithms to converge, which reaches its clear maximum for $n = 8000$. In this case, the run time improves by over 34%.

In Figure 6.3, we plot the results for symmetric kMN. There are significant differences for all values of k used. This is because the amount of induced closed



(a) Number of iterations required by the k-means-CP for different number of samples for kNN with $k = 1, 2, 3$.



(b) Number of *closed neighborhood sets* (CNS) for different number of samples for kNN neighborhood with $k = 1, 2, 3$.

Figure 6.2: Number of iterations and closed neighborhood sets on MNIST data.

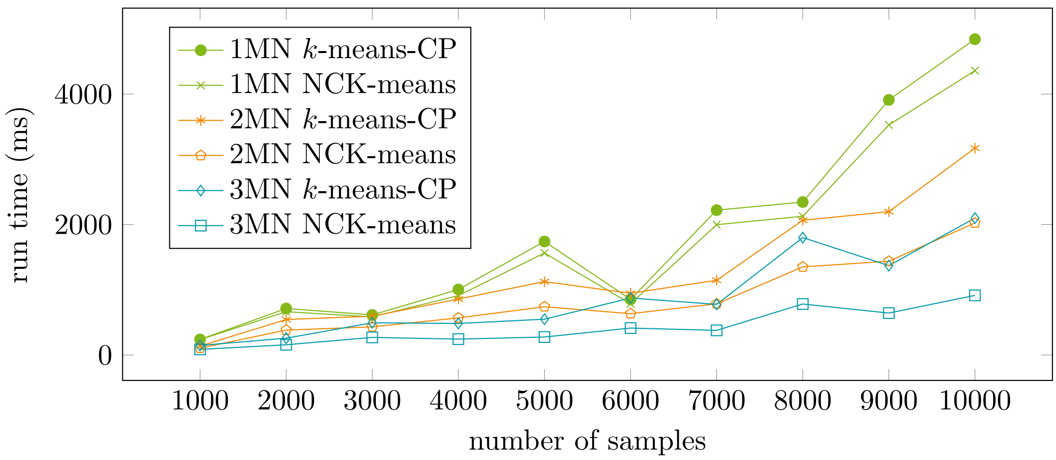
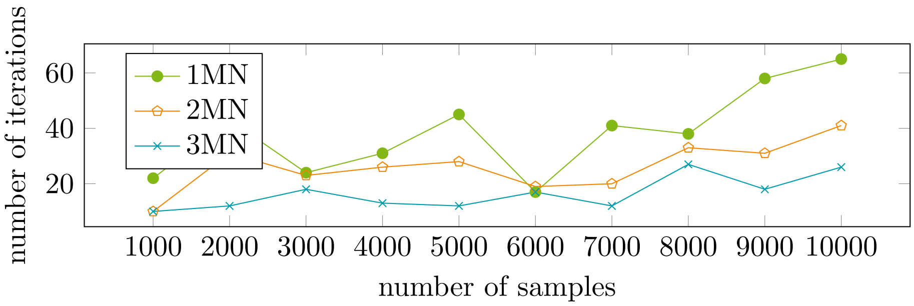
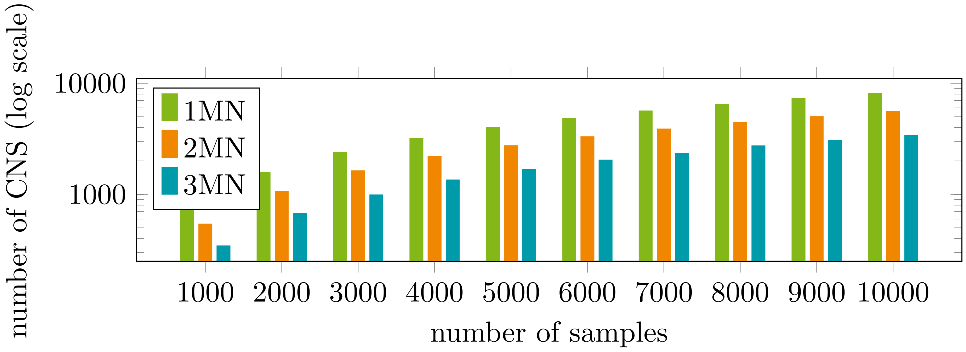


Figure 6.3: Run time of k-means-CP and NCK-means for different number of samples for kMN with $k = 1, 2, 3$.

neighborhood sets seems to scale approximately linearly for kMN. The spikes in the run time differences can again be explained by the varying number of iterations



(a) Number of iterations required by the k-means-CP for different number of samples for kMN with $k = 1, 2, 3$.



(b) Number of *closed neighborhood sets* for different number of samples for kMN with $k = 1, 2, 3$..

Figure 6.4: Number of iterations and closed neighborhood sets on MNIST data.

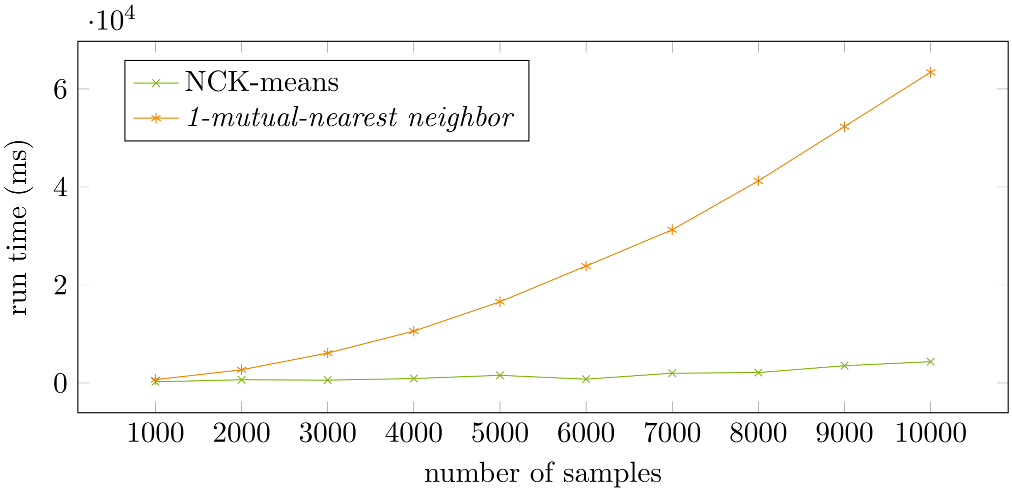


Figure 6.5: Run time of 1-Mutual-Nearest-Neighbor (1MN) computation and NCK-means for different number of samples of MNIST data.

required for convergence. The greatest relative improvement is reached by $k = 3$ for $n = 8000$, reducing the run time by over 56%. Across all k and n, we observe an average run time reduction of 30%.

The run time improvement per iteration would be the largest if there were very few closed neighborhood sets because in this case, we save a lot of time by using the representative. But if there are too few closed neighborhood sets, the algorithm converges very fast, e.g., in the case of 3NN, where the initial clustering is already converged. The clustering quality for very few closed neighborhood sets is also questionable but sometimes surprisingly good. This is probably because of the relationship to DBSCAN, which also builds closed neighborhoods, and spectral clustering, which partitions the nearest-neighbor graph.

6.4 Discussion

We improve the k-means-CP algorithm for clustering with kNN Consistency by avoiding unnecessary distance computations. Using the parallel axis theorem, we prove that we can reduce all closed neighborhoods to a single representative, and use these in clustering instead of considering all samples individually. This allows clustering with nearest-neighbor consistency on larger data sets than before.

The technique could be integrated into further algorithms. In COP-K-means [WCR+01] and PCK-means [BBM04], for example, this allows us to reduce must-link constraints into using weighted means, too. However the speedup is only noticeable if we have many such constraints, and the typical scenario for constrained clustering is with only a few constraints given by the user to guide the clustering in a semi-supervised way.

Chapter 7

Cluster Quality of Categorical Data

In the context of categorical data sets, clustering can be more challenging because it is not at all obvious to quantify reliably how “similar” two samples are. Categorical data is qualitative and attributes represent distinct categories or groups. These categories do not have a natural order or a quantifiable numerical difference. Common distance measures such as Euclidean distance are designed for continuous numerical data and require the feature values to have meaningful differences and comparable scales. Categorical data, even when transformed to numbers for example by “one hot encoding” using only ones and zeros, does not yield useful distances with common metrics. Since every feature then is either identical or has a difference of exactly one, Euclidean distance degenerates to the square root of twice the Hamming distance – the simplest possible distance that simply counts the number of differences. Manhattan distance becomes exactly twice the Hamming distance. This is problematic for distance-based clustering as there will be many “ties”, i.e., objects with exactly the same difference. For example in hierarchical clustering, this leads to an unclear merge order, which may cause a high variability in possible clusterings. Furthermore, many popular clustering methods such as k-means assume that the mean is an optimal summary of the cluster and that it is desirable to minimize the squared errors. Ward, centroid, and medoid linkage are also derived from least squares optimization. On binary data, the mean no longer represents an actual data point, and neither does minimize the squared deviations yield easily interpretable semantics for further analysis.

Several distance measures devised for categorical data, binary variables, and sets [DD13, Ch 17.3] have been proposed, such as the classic Hamann similarity, the simple matching coefficient of Sokal and Michener [SM58], the Rand similarity, and the classic Jaccard index [Jac02]. Because set distances are usually defined in terms of the size of the union, the intersection, and the difference of these sets, they share the problem of having many tied distances and offer little discrimination if we only have few variables. An integral information not used by above measures is the prevalence of values in the entire data set. We will discuss approaches to resolve this, such as Goodall’s index [Goo66], below.

In this chapter, we shift the focus to the analysis of categorical data and its challenges. We begin by examining the quality of clusters formed on categorical data, highlighting inherent complexities. Furthermore, we propose a method to optimize this clustering quality through hierarchical clustering. Our study aims to provide practical insights that reconcile theoretical advancements with pragmatic considerations in categorical data analysis and help practitioners with clustering qualitative data. We exemplify the method with the formation of archetypes from taxonomies, a common theme in contemporary information systems research.

7.1 Relative Entropy

To assess the quality of clusters formed from categorical data, we propose a measure that quantifies the divergence between the probability distribution within a cluster and the overall data set distribution. This measure is motivated by the premise that agreement on rare attribute values is more important than agreement on frequent values. We propose the Relative Entropy Scores (RES) of a subset $X \subseteq D$ of a categorical data set D to simply be the sum of the Kullback-Leibler divergences (i.e., relative entropy) of each attribute.

$$\begin{aligned}
\mathrm{RES}(X, D) &= \sum_{A} \sum_{i \in A} \mathrm{p}_{\mathrm{A_i}}(X) \log \frac{\mathrm{p}_{\mathrm{A_i}}(X)}{\mathrm{p}_{\mathrm{A_i}}(D)} \\
&= \sum\nolimits_{A} \mathrm{D}_{\mathrm{KL}}(\mathrm{P_A}(X) \parallel \mathrm{P_A}(D)) \\
&= \sum\nolimits_{A} \big(H(\mathrm{P_A}(X), \mathrm{P_A}(D)) - H(\mathrm{P_A}(X))\big) \\
&= \sum\nolimits_{A} H(\mathrm{P_A}(X), \mathrm{P_A}(D)) - \sum\nolimits_{A} H(\mathrm{P_A}(X))
\end{aligned}$$

where $\mathrm{P_A}(X)$ denotes the distribution of values in attribute A within subset X and $\mathrm{p}_{\mathrm{A_i}}(X)$ denotes the probability of an individual value A_i in attribute A within the subset X. The last two lines demonstrate that we obtain the same result if we first aggregate the cross-entropy.

If the value distribution in X is similar to the value distribution in the entire data set D, the Relative Entropy Scores (RES) becomes small, and hence a uniformly sampled subset of sufficient size is expected to score low. A high *divergence* from the overall data set indicates a subset whose members hence are more similar to each other than to the remainder of the data set, i.e., more homogeneous. Such subsets must have characteristic values and are of interest for further analysis.

For a clustering C into k disjoint clusters $C_1, ..., C_k$ on a data set D of N samples, we define the Average Relative Entropy Score (ARES) as the average relative entropy, weighted by the cluster size:

$$\mathrm{ARES}(C, D) = \textstyle\sum_{i=1}^{k} \frac{|C_i|}{N} \mathrm{RES}(C_i, D) \,.$$

As the number of clusters increases, it becomes easier to make them homogeneous, and hence the maximum possible ARES increases with k, and we should not directly use this score to compare different k or choose the "optimum" number of clusters. We could look for an inflection point, but we expect this to be as problematic as with the “elbow” criterion for k-means [Sch23]. If we completely partition the data set into one-elemental clusters, we obtain

$$\begin{aligned}\mathrm{ARES}_0 &= \sum\nolimits_{x\in D} \frac{1}{N}\,\mathrm{RES}(\{x\}, D)\\ &= \frac{1}{N}\sum\nolimits_{x\in D}\sum\nolimits_{A} H(\{A(x)\}, P_A(D)) - 0\\ &= \sum\nolimits_{A} -\mathrm{p}_{\mathrm{A_i}}(D)\log \mathrm{p}_{\mathrm{A_i}}(D) = \sum\nolimits_{A} H(\mathrm{P_A}(D))\end{aligned}$$

where $A(x)$ denotes the value of x in attribute A. This follows from $H(\{A(x)\}) = 0$, $H(\{A(x)\}, P_A(D)) = -\log \mathrm{p}_{\mathrm{A_i}}(D)$ and $\mathrm{p}_{\mathrm{A_i}}(D) = P(A(X) = A_i)$. The relative entropy of the split data set given the full data set hence simply equals the full entropy of the data set.

To evaluate clusterings, we instead take inspiration from the Silhouette [Rou87], which compares the distances to the nearest and the second nearest cluster. We define the Relative Entropy Contrast (REC) of a single cluster C_i as the difference between its relative entropy and the relative entropy of the merge with the best other cluster:

$$\mathrm{REC}_i(C, D) = \mathrm{RES}(C_i, D) - \max\nolimits_{i\neq j} \mathrm{RES}(C_i \cup C_j, D)\ .$$

We aggregate these values into a single statistic that can serve as a CQM. Because the measure compares the current clustering with the hypothetical best next merge, it is more suitable as a stopping criterion for choosing the number of clusters. A high Minimum Relative Entropy Contrast (MREC) indicates a “stable” result. We define the Minimum Relative Entropy Contrast (MREC) as

$$\mathrm{MREC}(C, D) = \min\nolimits_i \mathrm{REC}_i(C, D)\ .$$

Last but not least, we also define the Relative Entropy Dissimilarity (RED) for use in hierarchical clustering and visualization, because the notion of “distance” seems to be more intuitive for end users than “similarity”. For two disjoint subsets $X_1, X_2 \subset D$ we calculate the dissimilarity (assuming base 2 logarithms were used before) as

$$\mathrm{RED}(X_1, X_2, D) = 2^{-\mathrm{RES}(X_1\cup X_2, D)}\ .$$

This transformation resembles the inverse of a “relative perplexity”, but we have not further analyzed its properties yet. For visualizing the clustering dendrogram, we found it beneficial to further scale the scores with the baseline of the trivial clustering into N clusters, ARES_0:

$$\mathrm{RED}'(X_1, X_2, D) = 2^{-\mathrm{RES}(X_1\cup X_2, D)/\mathrm{ARES}_0}\ .$$

The clustering into the maximum number of clusters yields the minimum entropy of zero, but this does not necessarily minimize the relative entropy. A result with the same entropy as this trivial clustering scores 0.5 in the dendrogram, suggesting that structure below this threshold is “obvious” (as it improves even over this baseline), but the more interesting patterns are found above.

7.2 Optimization of the Relative Entropy

There are already various clustering methods that directly optimize an intrinsic quality measure and thus deliver a self-fulfilling result on their own. Van der Laan [VPB03] presented with PAMSIL a variant of PAM [KR87], that is directly optimizing the Silhouette. Another notable recent variant is FasterMSC [LS24], a modification of PAM that incorporates enhancements proposed by Schubert and Rousseeuw [SR21], optimizing the Medoid Silhouette directly. To optimize for the Relative Entropy Score, we choose an agglomerative hierarchical clustering approach. The utilization of agglomerative hierarchical clustering aligns with our objective and improves the interpretability of results, addressing a common challenge associated with categorical data analysis, while the run time complexity of $O(N^3)$ is not as relevant for the data set size of usually less than a thousand instances commonly found in our application domain.

Algorithm 13, denoted as Clustering for Categorical Data with Relative Entropy Dissimilarity (CatRED), presents an approach for hierarchical clustering for categorical data. The clusters are built up from the bottom up and merged step by step, as in the standard algorithm for hierarchical clustering. In each iteration, the two clusters with the smallest relative entropy-based distance are merged, and we have to recompute the distances between the one merged cluster and all other clusters. The algorithm returns a merge history in the same format as used by other hierarchical algorithms, enabling the construction of a dendrogram that illustrates the sequence of merges.

In contrast to k-means style algorithms, the user does not have to specify the number of clusters beforehand, and there is no need for restarts as the initialization is deterministic. The dendrogram produced by CatRED provides an easy to understand visualization of the cluster structure, enabling researchers to analyze

Algorithm 13: Clustering for Categorical Data with Relative Entropy Dissimilarity (CatRED)

Initialize C with each data point in D as a separate cluster
Compute a distance matrix M for all pairs (C_i, C_j) with $\text{RED}(C_i, C_j, D)$
Initialize an empty merge history L
while *more than one cluster remains in* C **do**
    Find the pair of clusters $C_i \neq C_j$ in M with the smallest distance
    Merge clusters C_i and C_j into a new cluster C_{new}
    Remove C_i and C_j from C
    Add C_{new} to C
    Compute distances between C_{new} and all remaining clusters, update M
    Append to L the indices of C_i and C_j, the distance, and the size of C_{new}
return L

multiple levels of granularity. Notably, compared to other hierarchical clustering methods, CatRED better incorporates rare attribute values into cluster formation.

The main limitation of CatRED is its cubic run time complexity, which means it can only be used for data sets of up to a few thousand instances, but larger data sets are not common in its primary application domain. For larger data sets, the run time can sometimes be reduced by immediately merging duplicate records. By storing value histograms for each cluster, we can efficiently merge clusters and update distances. In practice, our method performed well, completing in just a few seconds on real-world data sets, so further optimization was not necessary.

The standard algorithm for HAC typically uses predefined linkage criteria, such as single, complete, or average linkage, to define how the distance between clusters is computed during the merging process, based on a chosen distance metric. In CatRED we use our RED measure to evaluate the dissimilarity between clusters, which gives a form of categorical linkage that takes into account how frequent values are within the cluster compared to the entire data set. As in classic HAC, CatRED always merges the clusters with the lowest RED score. Another difference to the standard hierarchical clustering algorithms is that we do cannot employ Lance-Williams equations to aggregate the merged distances from the current stored distances. The asymptotic complexity does not increase substantially: in either case we have $N-1$ iterations, where we need $O(N^2)$ time to find the best merge. If we store value histograms of each cluster, we can merge clusters in $O(V)$ where V denotes the number of values across all attributes, and compute the updated distances in $O(N \cdot V)$. The complexity of the algorithm hence is $O(N^3 + N^2V)$, which is not substantially worse than regular hierarchical clustering's $O(N^3)$, unless the number of attributes V is much larger than the number of data points N. On the real-world datasets, the run times were within a few seconds, hence we did not explore further optimizations. It likely is possible to reduce the complexity to $O(N^2V)$ using the nearest-neighbor chain approach of Murtagh [Mur83], which has successfully been used with other complex linkages such as Medoid linkage [Sch21].

We implemented our algorithm in Python, creating the `categoricalclustering` package, and we make our source code available in this package. The GitHub repository includes a comprehensive guide and a link to a notebook that outlines best practices for clustering categorical data.

7.3 Cluster Analysis in Information Systems Research

Cluster analysis stands out as frequently used in socio-technical research for identifying relationships, encompassing both homogeneous and heterogeneous properties, among diverse IT artifacts within the organizational context [BMI11].

One part of this is research related to the field of taxonomy construction, which is concerned with the structuring and organization of knowledge [GV95]. Taxonomies group objects of interest in a domain based on common characteristics.

They enable to study the relationships between concepts and enable hypothesizing about them. Consequently, they are of explanatory, not descriptive, nature and constitute a means to perform ex-post theory building [NVM13]. Currently, they are a significant driver of theorizing in information systems research.

So far, researchers have relied on combinations of several agglomerative clustering algorithms using different distance measures to come to a reasonable solution [WWW+23; GRR18; FHH+20; SSH+24; MWW+23]. Examples include single linkage, complete linkage, group average linkage, centroid linkage, and Ward's method as well as Euclidean distance, squared Euclidean distance, Jaccard distance, Hamming distance, Dice distance, and Yule distance. The resulting cluster representations (e.g., dendrograms and cross-tabulation analyses) are typically reviewed and discussed qualitatively to derive said archetypes.

In the following, we apply the proposed method for clustering of categorical data to two information systems research taxonomies and archetypes that are somewhat distinct in their formation to understand differences and similarities of the results. Fischer et al. [FHH+20] form five archetypes built on 100 objects with mutually exclusive characteristics, while the work of Wanner et al. [WWW+23] is a more comprehensive procedure based on 854 publications with non-exclusive characteristics resulting in six archetypes. The public availability of the coding and the clustering results enable us to compare our method to their effort.

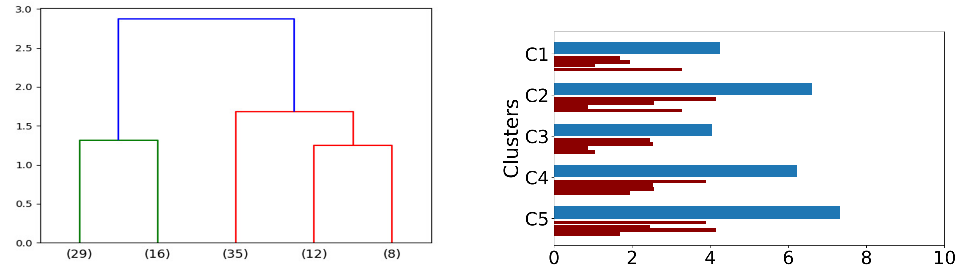
7.3.1 A Taxonomy and Archetypes of Smart Services for Smart Living

Fischer et al. [FHH+20] present a comprehensive framework for structuring and understanding smart services for smart living. They introduce a taxonomy that classifies smart services based on the characteristics of the smart product as well as the digital service. Additionally, they identify and describe archetypes as prototypes of smart services, providing typical examples within each category. The goal of the paper is to contribute to a better understanding of the diverse landscape of smart services and to facilitate the development and deployment of innovative solutions for smart living environments. The data set contains 100 objects that have been coded into a taxonomy consisting of 2 meta-characteristics, 8 dimensions, and 20 mutually exclusive characteristics.

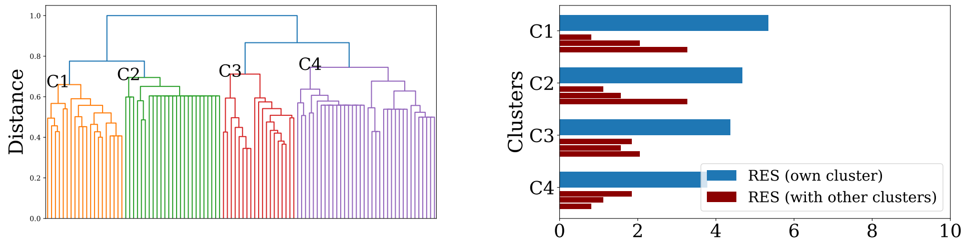
Diverse hierarchical clustering methods are applied in the paper, and subsequently, one of the clustering results is selected manually. Then, they formulate an archetype for each cluster. In our investigation, we systematically compare the clusters from the paper with those generated by our proposed measure, providing a comprehensive assessment of the clustering quality.

Quantitative Analysis

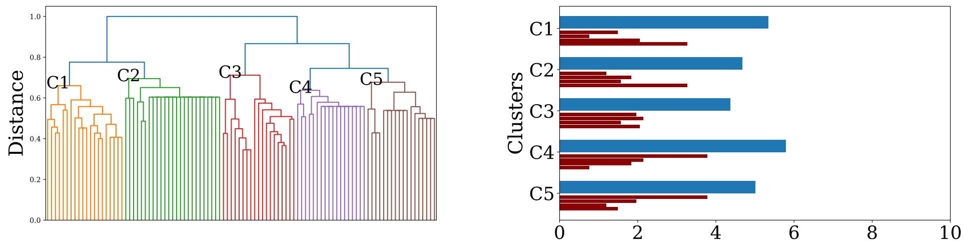
Figure 7.1 illustrates a comparison between the outcomes presented in the paper and those generated by the CatRED algorithm. The relative entropy quality measure is



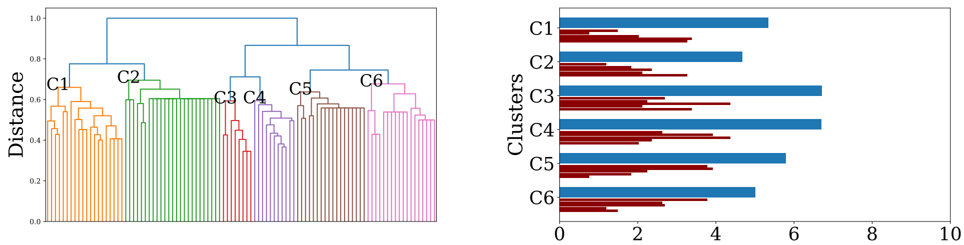
(a) $k = 5$ original clusters with $MREC = 0.99$ and $ARES = 5.04$



(b) $k = 4$ with $MREC = 1.41$ and $ARES = 4.43$



(c) $k = 5$ with $MREC = 1.23$ and $ARES = 5.01$



(d) $k = 6$ with $MREC = 1.23$ and $ARES = 5.47$

Figure 7.1: Comparison of the quality of the original cluster result from Fischer et al. [FHH+20] with the results of CatRED. The plot shows both the relative entropy scores of a cluster itself and of the unions with the other clusters. This makes it possible to assess the usefulness of merging clusters.

computed to provide a purely technical evaluation of cluster quality. It is noticeable that with the rather small data set, small k provide higher MREC values. Overall, the clusters formed correspond very closely to the clusters from the original paper and thus confirm them. Based on the MREC, we would prefer $k = 4$. However, $k = 5$ and $k = 6$ also provide meaningful clusters. With $k = 4$, the old C1 would correspond to the new C2, the old C2 to the new C1, the old C3 to the new C4, and the old C4 to the new C3. Only the old C5 would be almost completely included in the new C4. With this clustering, 91% of the samples would be assigned in the same way as in the original paper.

Qualitative Analysis

With more than three-quarters of the objects having been assigned to the same clusters as proposed by Fischer et al. [FHH+20], this method offers a very similar result. The key difference is the proposal of four archetypes as opposed to five archetypes even though by a slim margin. The most different cluster is thus *personal tracker* (C4), which now also comprises most *trainable assistant* services (formerly C5). To some respect, this shift also makes sense from a qualitative point of view as more than two-thirds of the old cluster C5 – chiefly smart vacuum cleaners – are now rather being considered *command execution* (C3). On the one hand, given the old clusters this would result in a very small prior cluster C5, which could more or less only be justified by its potential and forward-looking nature, but not necessarily through data at the time. On the other hand, the dismissed five-cluster proposal of CatRED shows that C4 would be evenly distributed in two clusters necessitating a slightly different distinction between tracker and assistant services most likely diluting the later concept of an intelligent assistant to a servant or advisor. Other than that, clustering differences are minimal and comprise sensibly moving some configurable services from *monitoring* (C1) to *diagnostics and automation* (C2) and a few enhanced, service-to-user monitoring services from C3 to C4.

7.3.2 A Taxonomy and Archetypes of Business Analytics in Smart Manufacturing

Wanner et al. [WWW+23] present an expansive framework for structuring and understanding business analytics in smart manufacturing. Their quadripartite taxonomy classifies research based on its domain, orientation, data, and technique. In addition to their taxonomy. They derive six archetypes of prototypical business analytics applications in smart manufacturing with the aim of providing an entry point to the field but also serving as a reference work and a guide for one's own instruments. The data set contains 854 objects that have been coded into a taxonomy consisting of 4 meta-characteristics, 7 dimensions, and 52 in parts non-exclusive characteristics.

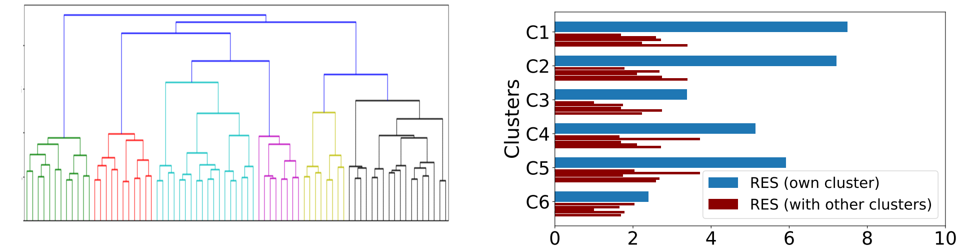
They, too, apply a diverse set of hierarchical clustering methods and according to distance measures in their paper, manually select one of the clustering results after considering multiple combinations, and formulate an archetype per cluster. Again, we systematically compare the clusters from their paper with those generated by our proposed measure, providing a comprehensive assessment of the clustering quality.

Quantitative Analysis

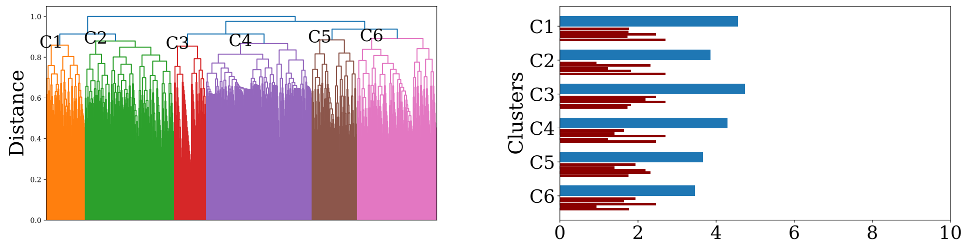
Figure 7.2 presents a comparative analysis between the results presented in the paper and those obtained through the CatRED algorithm. The analysis is based on the calculation of relative entropy, serving as a purely technical evaluation of cluster quality. Upon examination of the original paper results, it becomes apparent that two out of the six clusters exhibit notably lower Relative Entropy Scores (RES). In contrast, two of the remaining four clusters demonstrate substantially high RES, surpassing even the values observed in the clusters produced by the CatRED algorithm. Based on the MREC, we would prefer $k = 7$. However, $k = 6$ and $k = 8$ also provide meaningful clusters. Approximately 60% of the samples retain their assignment to the same cluster. Notably, clusters C4 and C5 from the original assignment, characterized by the notably high relative entropy scores, are predominantly merged with parts of cluster C6 into the new largest cluster, denoted as C4. The newly formed cluster C6 predominantly comprises samples from the former cluster C1, yet encompasses only 70% of the entries originally present in cluster C1. Only the new cluster C7 cannot be clearly assigned to an old cluster; it draws mainly from the old C1, C3, and C4, but also from the others. Nevertheless, it has a high RES, and it should be investigated what characterizes this cluster and whether a suitable archetype can be built for it.

Qualitative Analysis

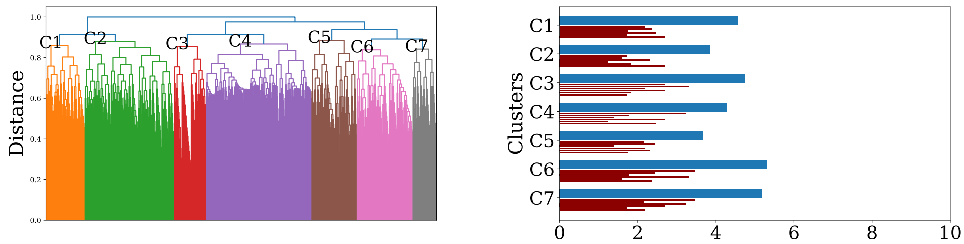
With around 60% of the objects having been assigned to the same clusters as proposed by Wanner et al. [WWW+23], this method provides a reasonably comparable result, but it also highlights the challenges of clustering categorical values of larger data sets. While the formation of the original archetypes was focused very much on the function dimensions of the data set, reinforcing a dimension that could potentially be considered a dependent variable based on the other dimensions, this result offers a more diverse perspective on the archetypes by highlighting different aspects: The novel clusters focus on non-integrated maintenance, repair and operations (MRO) planning and monitoring (C1) and mostly vertically integrated MRO planning and operations (C2) that were previously largely combined. They also highlight smaller groupings of design and sustainability aspects (C3) as well as custom development especially for security (C5) that previously only found its way into a more summative offline predictive maintenance and monitoring clusters. Various aspects of predictive maintenance aspects (C4) form the largest cluster that were previously distributed across multiple clusters also incorporating smaller subjects now found in C3. Quality management (C6) remains largely unchanged and the final cluster C7 represents non-integrated, predictive, machine analytics heavily



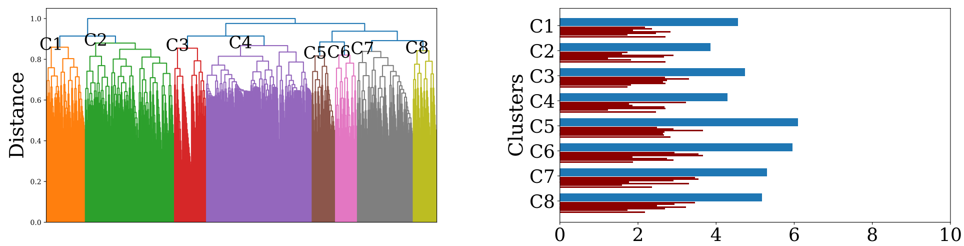
(a) $k = 6$ original clusters with $MREC = 0.36$ and $ARES = 4.87$



(b) $k = 6$ with $MREC = 0.99$ and $ARES = 4.01$



(c) $k = 7$ with $MREC = 1.06$ and $ARES = 4.38$



(d) $k = 8$ with $MREC = 0.95$ and $ARES = 4.65$

Figure 7.2: Comparison of the quality of the original clusters of Wanner et al. [WWW+23] with the results of CatRED. The plot shows both the relative entropy scores of a cluster itself and of the unions with the other clusters. This makes it possible to assess the usefulness of merging clusters.

relying on classification and deep learning, comprising approaches from recent years that can be termed *advanced quality & maintenance analytics*. Together, these clusters offer a slightly orthogonal perspective on the data set with less emphasis on one dimension that would enhance the qualitative discussion of the innate structuring value of the data. Using weights on the dimensions would potentially enable discussion from the perspective of each dimension. An interactive exercise not yet conducted in information systems research.

7.4 Discussion

We have introduced the ARES and the Minimum Relative Entropy Contrast (MREC), novel clustering evaluation measure specifically tailored for categorical data. Furthermore, we have presented CatRED, a hierarchical clustering algorithm using this measure to merge clusters. MREC can also aid in determining the optimal number of clusters. Our methods have been evaluated through archetype formation from taxonomy data in two distinct domains: smart services for smart living, and business analytics in smart manufacturing. Our approach enables the efficient identification of meaningful clusters, eliminating the need for ensembles of clustering algorithms. On the one hand, our approach confirmed their results, on the other hand, it enabled taking a novel perspective that was less dominated by a single dimension in larger data sets.

We provide a measure of cluster quality based on the premise that rare values are more important in cluster formation. The method offers a solution not only for clustering on taxonomy data. Other areas in which categorical data must be clustered are also relevant as applications. Through our qualitative evaluation conducted during the creation of archetypes from taxonomies, we elucidated the significance of attribute weighting in clustering. As categorical data are prevalent across various domains, accurate clustering methodologies are essential for deriving meaningful insights. The obvious addition of a feature weighting to our measure would increase usability for many areas in which expert knowledge of the data is essential. Similarly, it may be relevant to integrate categorical outlier detection to avoid including unusual samples in nearby clusters.

Chapter 8

Conclusion and Future Work

In the following, the contributions of this thesis are summarized in Section 8.1, with the results reviewed and discussed in Section 8.2. Finally, potential directions for future work are outlined in Section 8.3.

8.1 Summary

This thesis has focused on the development of advanced clustering evaluation measures and resource-efficient clustering algorithms for direct optimization of these. The contributions span theoretical proofs, algorithmic improvements, and practical applications, all aimed at improving the quality and efficiency of clustering.

The first major contribution, introduced in Chapter 4, was the optimization of the FasterPAM algorithm for sparse problems, demonstrated through practical experiments on energy grid simulations. This work involved refining the algorithm by introducing a method for automatically determining the optimal number of clusters. To achieve this, two losses were incorporated into the objective function: one addressing the cost of poorly managed locations, including potential outliers, and the other representing the traditional k-medoids problem. This approach aims to enhance the overall clustering performance for sparse data sets.

The second main contribution, introduced in Chapter 5, was the Average Medoid Silhouette (AMS), a robust alternative to the Average Silhouette Width (ASW), which was particularly effective in real-world applications such as gene expression analysis. A novel algorithm was developed to optimize AMS computation by leveraging partial results and cached nearest centers, leading to significant computational speedups by a factor of $O(k^2)$. This improvement enabled medoid-based clustering to be applied to much larger data sets. Additionally, the DynMSC algorithm was introduced to simplify the process of determining the optimal number of clusters, further enhancing the clustering framework.

The third main contribution, introduced in Chapter 6, focused on improving the k-means-CP algorithm by incorporating kNN Consistency and minimizing redundant distance computations. This optimization was achieved through the application of the parallel axis theorem. These enhancements made the algorithm more suitable for large-scale data clustering tasks.

Finally, the thesis introduced two new measures for clustering categorical data—Average Relative Entropy Score (ARES) and Minimum Relative Entropy Contrast (MREC)—in Chapter 7. To exploit these measures, the hierarchical clustering algorithm CatRED was developed, which allowed for efficient merging of clusters and determination of the optimal number of clusters. This approach was successfully applied in domains such as smart services and business analytics, providing more interpretable and efficient clustering results for categorical data. Together, these contributions substantially advanced the capabilities of clustering algorithms.

8.2 Discussion of Results

The algorithms and methods presented in this thesis demonstrate not only improvements in efficiency and scalability but also highlight the intrinsic connection between clustering algorithms and internal evaluation methods. Each internal cluster evaluation method, such as those introduced in this work (e.g., AMS, ARES, MREC, and kNN Consistency), inherently implies a clustering algorithm, where the best solution is selected. The key difference between internal evaluation methods and the clustering algorithms themselves lies in the availability of an efficient optimization strategy. This distinction is crucial in understanding the advancements made by the proposed algorithms.

Internal Evaluation Measures We have analyzed the existing evaluation measures Average Medoid Silhouette (AMS) and k-Nearest-Neighbor Consistency. We showed that the AMS satisfies desirable theoretical properties for clustering quality measures, and as an approximation of the ASW yields desirable results on real problems from gene expression analysis. We have introduced Minimum Relative Entropy Contrast (MREC) and Average Relative Entropy Score (ARES), two measures for categorical data. MREC and ARES are grounded in the premise that agreement on rare attribute values carries greater significance than agreement on more frequent values. This premise has demonstrated its utility in real-world applications, specifically in taxonomies used for smart services in smart living environments, as well as in business analytics within smart manufacturing contexts.

Clustering Algorithms This thesis contributes to the ongoing challenge of improving both the measurement and optimization of clustering quality, with a strong focus on resource efficiency. We introduced several clustering algorithms that aim to strike a balance between computational efficiency and clustering quality. Three of these algorithms—sparse Partitioning Around Medoids, FasterMSC, and NCK-means—have been shown to scale effectively for larger data sets, providing resource-efficient clustering solutions while maintaining high-quality results. By offering performance-optimized variants of existing algorithms, they enable the application of these methods to much larger data sets. In addition to these performance-optimized algorithms, we introduced CatRED, a new algorithm specifically designed for small categorical data sets, addressing a specialized application area. As Bonner [Bon64] noted "none of the many specific definitions [...] seems best in any general sense", and results are subjective "in the eye of the beholder" as noted by Estivill-Castro [Est02]. Our evaluations confirm this, demonstrating that no single measure is universally optimal. Consequently, the criteria for an evaluation measure must be defined based on the specific requirements of each problem or application, underscoring the need for flexibility in selecting appropriate clustering methods.

These contributions provide a foundation for future research on the theoretical and practical aspects of clustering and its evaluation.

8.3 Future Work

Future work should focus on the selection of clustering evaluation measures and methods that are specifically tailored to the problem at hand, as commonly used metrics like the Silhouette score may underperform in certain contexts. This is particularly true when (1) distance is not measured appropriately, (2) data preprocessing is poor, (3) clusters have non-convex shape, (4) clusters vary substantially in diameter, (5) clusters exist only in subspaces, or (6) a hierarchy of clustering structures exists. The development of alternative evaluation measures remains an important area of research, as different problems demand different types of evaluation.

This is also relevant in the domain of categorical and mixed data (numerical and categorical), where many existing measures or algorithms are insufficient to address the challenges posed by such data types. The cluster evaluation measure MREC introduced in this research, which highlights the significance of rare attribute values, offers a promising step forward but can be further enhanced. Future improvements could include the integration of feature weighting, allowing the algorithm to give more importance to specific attributes based on their relevance. Heumann et al. recently presenting a guideline for clustering of taxonomy-based data [HKW+24]. Building on this, our proposed feature weighting and other preprocessing methods could be integrated into the guideline at various stages. Additionally, implementing and elaborating this extended guideline with a detailed application to the surveyed data from the referenced publication would provide practical insights and validation for our proposed evaluation measure and clustering algorithm.

Finally, numerous techniques for optimizing the performance of existing algorithms were applied and demonstrated in this research. These optimization strategies can also be extended to other algorithms, providing a valuable direction for future research. For example, the parallel axis theorem, successfully utilized in NCK-means, could be integrated into additional clustering algorithms. In COP-K-means [WCR+01] and PCK-means [BBM04], for example, this allows us to reduce must-link constraints into using weighted means, too.

List of Figures

List of Tables

List of Algorithms

Acronyms

1MN	1-Mutual-Nearest-Neighbor 70, 72
3NN	3-Nearest-Neighbor 73
k-means-CP	Consistency-Preserving k-means iii, 2, 65–73, 87
kMN	k-Mutual-Nearest-Neighbor 26, 27, 69–72
kNN	k-Nearest-Neighbor 2, 4, 5, 21, 26, 27, 65, 69–71, 73, 87, 88
AMS	Average Medoid Silhouette iii, 23, 45–48, 55–60, 63, 87, 88
ARES	Average Relative Entropy Score iii, 2, 4, 76, 85, 87, 88
ARI	Adjusted Rand Index 13, 56–59
ASW	Average Silhouette Width iii, 22, 23, 45, 46, 48, 56–60, 63, 87, 88
BCSS	Between-Cluster Sum of Squares 12
CatRED	Clustering for Categorical Data with Relative Entropy Dissimilarity iii, 2, 5, 78–85, 88
COP-K-means	Constrained K-means Clustering with Background Knowledge 73, 89
CQM	Clustering-Quality Measure 2, 14–16, 23, 45, 46, 63, 77
CU	Category Utility 21, 25, 26
DBSCAN	Density-Based Spatial Clustering of Applications with Noise 1, 26–28, 73
DynMSC	FasterMSC with Dynamic k iii, 55, 56, 62, 63, 87
ELKI	Environment for DeveLoping KDD-Applications Supported by Index-Structures 39, 69, 70
EM	Expectation-Maximization 7–9

FasterMSC	Fast and Eager Medoid Silhouette Clustering iii, 2, 53–60, 62, 78, 88
FasterPAM	Fast and Eager Partitioning Around Medoids iii, 2, 20, 33–36, 38–43, 45, 53, 60, 62, 63, 87
FastMSC	Fast Medoid Silhouette Clustering 52–60, 62
FastPAM	Fast Partitioning Around Medoids 20, 35, 36, 50
GMM	Gaussian Mixture Modeling 8
HAC	Hierarchical Agglomerative Clustering 7, 11, 79
LIF	leukemia inhibitory factor 56, 57
MAP	Maximum a Posteriori 8
mESCs	Mouse Embryonic Stem Cells 56, 57, 60
ML	Maximimum Likelihood 8
MREC	Minimum Relative Entropy Contrast iii, 5, 77, 82, 83, 85, 87–89
MSD	Mean Squared Distance 12
NCK-means	Neighbor-Consistent K-means iii, 4, 5, 67–72, 88, 89
NLL	Negative Log-Likelihood 8
NMI	Normalized Mutual Information 13, 56–59
OSM	OpenStreetMap 32, 33, 38, 39, 41–43
PAM	Partitioning Around Medoids 2, 4, 10, 19, 20, 25, 35, 45, 48–51, 53, 57–59, 78, 88
PAMMEDSIL	Partitioning Around Medoids with Medoid Silhouette 25, 48–50, 53, 57–60, 63, 95
PAMSIL	Partitioning Around Medoids with Silhouette 25, 48, 57–60, 78
PCA	Principal Component Analysis 56, 57

PCK-means	Pairwise Constrained K-Means 73, 89
REC	Relative Entropy Contrast 77
RED	Relative Entropy Dissimilarity 77, 79
RES	Relative Entropy Scores 76, 83
scRNA-seq	Single-Cell RNA Sequencing 4, 56–59
TSS	Total Sum of Squares 12
WCSS	Within-Cluster Sum of Squares 12

Bibliography

[AGM+13] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. "An extensive comparative study of cluster validity indices". In: *Pattern Recognit.* 46.1 (2013), pp. 243–256. DOI: 10.1016/J.PATCOG.2012.07.021 (Cited on pages 21, 54).

[ATM+04] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. "LIMBO: Scalable Clustering of Categorical Data". In: *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings*. Ed. by E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, and E. Ferrari. Vol. 2992. Lecture Notes in Computer Science. Springer, 2004, pp. 123–146. DOI: 10.1007/978-3-540-24741-8_9 (Cited on page 30).

[AV07] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245 (Cited on pages 35, 70).

[BA08] S. Ben-David and M. Ackerman. "Measures of Clustering Quality: A Working Set of Axioms for Clustering". In: *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Curran Associates, Inc., 2008, pp. 121–128 (Cited on pages 2, 14 sq., 23, 45).

[BBM04] S. Basu, A. Banerjee, and R. J. Mooney. "Active Semi-Supervision for Pairwise Constrained Clustering". In: *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*. Ed. by M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn. SIAM, 2004, pp. 333–344. DOI: 10.1137/1.9781611972740.31 (Cited on pages 27, 73, 89).

[BCK08] S. Boriah, V. Chandola, and V. Kumar. "Similarity Measures for Categorical Data: A Comparative Evaluation". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*. SIAM, 2008, pp. 243–254. DOI: 10.1137/1.9781611972788.22 (Cited on page 29).

[BDH+23] A. Beer, A. Draganov, E. Hohma, P. Jahn, C. M. Frey, and I. Assent. "Connecting the Dots – Density-Connectivity Distance Unifies DBSCAN, k-Center and Spectral Clustering". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '23. Long Beach, CA, USA: Association for Computing Machinery, 2023, pp. 80–92. DOI: 10.1145/3580305.3599283 (Cited on pages 1, 25, 27).

[BH21] F. Batool and C. Hennig. “Clustering with the Average Silhouette Width”. In: *Comput. Stat. Data Anal.* 158 (2021), p. 107190. DOI: 10.1016/J.CSDA.2021.107190 (Cited on page 46).

[BLC02] D. Barbará, Y. Li, and J. Couto. “COOLCAT: an entropy-based algorithm for categorical clustering”. In: *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*. ACM, 2002, pp. 582–589. DOI: 10.1145/584792.584888 (Cited on page 29).

[BMI11] V. Balijepally, G. Mangalaraj, and K. Iyengar. “Are We Wielding this Hammer Correctly? A Reflective Review of the Application of Cluster Analysis in Information Systems Research”. In: *J. Assoc. Inf. Syst.* 12.5 (2011), p. 1. DOI: 10.17705/1JAIS.00266 (Cited on pages 4, 79).

[Bon64] R. E. Bonner. “On Some Clustering Techniques”. In: *IBM J. Res. Dev.* 8.1 (1964), pp. 22–32. DOI: 10.1147/RD.81.0022 (Cited on pages 2, 89).

[Bor20] C. Borgelt. “Even Faster Exact k-Means Clustering”. In: *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*. Ed. by M. R. Berthold, A. Feelders, and G. Krempl. Vol. 12080. Lecture Notes in Computer Science. Springer, 2020, pp. 93–105. DOI: 10.1007/978-3-030-44584-3_8 (Cited on page 69).

[BSH+07] M. Brun, C. Sima, J. Hua, J. Lowey, B. Carroll, E. Suh, and E. R. Dougherty. “Model-based evaluation of clustering validation measures”. In: *Pattern Recognit.* 40.3 (2007), pp. 807–824. DOI: 10.1016/J.PATCOG.2006.06.026 (Cited on pages 21, 54).

[CH74] T. Caliński and J. Harabasz. “A dendrite method for cluster analysis”. In: *Communications in Statistics* 3.1 (1974), pp. 1–27. DOI: 10.1080/03610927408827101 (Cited on pages 1, 21, 24).

[DB79] D. L. Davies and D. W. Bouldin. “A Cluster Separation Measure”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909 (Cited on pages 1, 21).

[DD13] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2013. ISBN: 9783642309588. DOI: 10.1007/978-3-642-30958-8 (Cited on pages 28, 75).

[DH04] C. H. Q. Ding and X. He. “K-nearest-neighbor consistency in data clustering: incorporating local information into global optimization”. In: *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*. Ed. by H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock. ACM, 2004, pp. 584–589. DOI: 10.1145/967900.968021 (Cited on pages 26 sq., 65, 69).

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data Via the EM Algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. ISSN: 0035-9246. DOI: 10.1111/j.2517-6161.1977.tb01600.x (Cited on page 8).

[DM01] I. S. Dhillon and D. S. Modha. “Concept Decompositions for Large Sparse Text Data Using Clustering”. In: *Mach. Learn.* 42.1/2 (2001), pp. 143–175. DOI: 10.1023/A:1007612920971 (Cited on page 65).

[Dun74] J. C. Dunn. “Well-Separated Clusters and Optimal Fuzzy Partitions”. In: *Journal of Cybernetics* 4.1 (1974), pp. 95–104. DOI: 10.1080/01969727408546059 (Cited on pages 1, 21).

[EKS+96] M. Ester, H. Kriegel, J. Sander, and X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. Ed. by E. Simoudis, J. Han, and U. M. Fayyad. AAAI Press, 1996, pp. 226–231 (Cited on page 1).

[Elk03] C. Elkan. “Using the Triangle Inequality to Accelerate k-Means”. In: *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. Ed. by T. Fawcett and N. Mishra. AAAI Press, 2003, pp. 147–153 (Cited on page 69).

[Est02] V. Estivill-Castro. “Why so many clustering algorithms – A Position Paper”. In: *SIGKDD Explorations* 4.1 (2002), pp. 65–75. DOI: 10.1145/568574.568575 (Cited on pages 2, 89).

[FHH+20] M. Fischer, D. Heim, A. Hofmann, C. Janiesch, C. Klima, and A. Winkelmann. “A taxonomy and archetypes of smart services for smart living”. In: *Electronic Markets* 30 (01/2020). DOI: 10.1007/s12525-019-00384-5 (Cited on pages 1, 80 sqq.).

[Fis87] D. H. Fisher. “Knowledge Acquisition via Incremental Conceptual Clustering”. In: *Mach. Learn.* 2.2 (1987), pp. 139–172. DOI: 10.1007/BF00114265 (Cited on page 30).

[GC85] M. A. Gluck and J. E. Corter. “Information Uncertainty and the Utility of Categories”. In: *Proc. Conf. of Cognitive Science Society*. 1985, pp. 283–287 (Cited on page 25).

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan. “CACTUS - Clustering Categorical Data Using Summaries”. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*. Ed. by U. M. Fayyad, S. Chaudhuri, and D. Madigan. ACM, 1999, pp. 73–83. DOI: 10.1145/312129.312201 (Cited on page 29).

[Goo66] D. W. Goodall. “A New Similarity Index Based on Probability”. In: *Biometrics* 22.4 (1966), pp. 882–907. ISSN: 0006-341X. DOI: 10.2307/2528080 (Cited on pages 29, 75).

[GRR18] H. Gimpel, D. Rau, and M. Roeglinger. “Understanding FinTech start-ups – A taxonomy of consumer-oriented service offerings”. In: *Electronic Markets* 28 (08/2018), pp. 245–264. DOI: 10.1007/s12525-017-0275-0 (Cited on page 80).

[GRS99] S. Guha, R. Rastogi, and K. Shim. "ROCK: A Robust Clustering Algorithm for Categorical Attributes". In: (1999). Ed. by M. Kitsuregawa, M. P. Papazoglou, and C. Pu, pp. 512–521. DOI: 10.1109/ICDE.1999.754967 (Cited on page 30).

[GV95] R. L. Glass and I. Vessey. "Contemporary Application-Domain Taxonomies". In: *IEEE Softw.* 12.4 (1995), pp. 63–76. DOI: 10.1109/52.391837 (Cited on page 79).

[HA85] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (12/1985), pp. 193–218. DOI: 10.1007/BF01908075 (Cited on page 13).

[Ham10] G. Hamerly. "Making k-means Even Faster". In: *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*. SIAM, 2010, pp. 130–140. DOI: 10.1137/1.9781611972801.12 (Cited on page 69).

[Ham61] U. Hamann. "Merkmalbestand und Verwandschaftsbeziehungen der Farinosae. Ein Beitrag zum System der Monokotyledonen". In: *Willdenowia* 2 (1961), pp. 639–768 (Cited on page 28).

[HBB19] B. Hie, B. Bryson, and B. Berger. "Efficient integration of heterogeneous single-cell transcriptomes using Scanorama". In: *Nature Biotechnology* 37 (06/2019). DOI: 10.1038/s41587-019-0113-3 (Cited on page 56).

[HK07] J. Handl and J. D. Knowles. "An Evolutionary Approach to Multiobjective Clustering". In: *IEEE Trans. Evol. Comput.* 11.1 (2007), pp. 56–76. DOI: 10.1109/TEVC.2006.877146 (Cited on page 26).

[HKW+24] M. Heumann, T. Kraschewski, O. Werth, and M. H. Breitner. "Reassessing taxonomy-based data clustering: Unveiling insights and guidelines for application". In: *Decision Support Systems* 187 (2024), p. 114344. ISSN: 0167-9236. DOI: 10.1016/j.dss.2024.114344 (Cited on page 89).

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. New York, USA: Springer, 2001. ISBN: 978-1-4899-0519-2 (Cited on page 19).

[Jac02] P. Jaccard. "Distribution de la florine alpine dans la Bassin de Dranses et dans quelques regiones voisines". In: *Bulletin del la Société Vaudoise des Sciences Naturelles* (1902) (Cited on pages 28, 75).

[JMF+16] P. A. Jaskowiak, D. Moulavi, A. C. S. Furtado, R. J. G. B. Campello, A. Zimek, and J. Sander. "On strategies for building effective ensembles of relative clustering validity criteria". In: *Knowl. Inf. Syst.* 47.2 (2016), pp. 329–354. DOI: 10.1007/S10115-015-0851-6 (Cited on page 1).

[KH79] O. Kariv and S. L. Hakimi. "An Algorithmic Approach to Network Location Problems. I: The p-Centers". In: *SIAM Journal on Applied Mathematics* 37.3 (1979), pp. 513–538. DOI: 10.1137/0137040 (Cited on pages 10, 19).

[KKT+15] A. A. Kolodziejczyk, J. K. Kim, J. C. Tsang, T. Ilicic, J. Henriksson, K. N. Natarajan, A. C. Tuck, X. Gao, M. Bühler, P. Liu, J. C. Marioni, and S. A. Teichmann. "Single Cell RNA-Sequencing of Pluripotent States Unlocks Modular Transcriptional Variation". In: *Cell Stem Cell* 17.4 (2015), pp. 471–485. ISSN: 1934-5909. DOI: 10.1016/j.stem.2015.09.011 (Cited on pages 1, 4, 56 sqq., 91).

[Kle02] J. M. Kleinberg. "An Impossibility Theorem for Clustering". In: *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. Ed. by S. Becker, S. Thrun, and K. Obermayer. MIT Press, 2002, pp. 446–453 (Cited on pages 14, 45).

[KMA+15] A. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. Weitz, and M. Kirschner. "Droplet Barcoding for Single-Cell Transcriptomics Applied to Embryonic Stem Cells". In: *Cell* 161 (05/2015), pp. 1187–1201. DOI: 10.1016/j.cell.2015.04.044 (Cited on pages 1, 4, 21 sq., 56 sq., 59).

[KR87] L. Kaufman and P. J. Rousseeuw. "Clustering by means of medoids". In: *Statistical Data Analysis Based on the L_1 Norm and Related Methods*. Ed. by Y. Dodge. North-Holland, 1987, pp. 405–416 (Cited on pages 10, 35, 45, 48, 78).

[KR90] L. Kaufman and P. J. Rousseeuw. "Partitioning Around Medoids (Program PAM)". In: *Finding Groups in Data*. John Wiley&Sons, 1990. Chap. 2, pp. 68–125. ISBN: 9780471878766. DOI: 10.1002/9780470316801 (Cited on pages 10, 29, 35, 45, 48).

[KSS+17] J. Kays, A. Seack, T. Smirek, F. Westkamp, and C. Rehtanz. "The Generation of Distribution Grid Models on the Basis of Public Available Data". In: *IEEE Transactions on Power Systems* 32.3 (2017), pp. 2346–2353. DOI: 10.1109/TPWRS.2016.2609850 (Cited on pages 1, 4, 32).

[KSZ17] H.-P. Kriegel, E. Schubert, and A. Zimek. "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?" In: *Knowledge and Information Systems* 52.2 (2017), pp. 341–378. DOI: 10.1007/S10115-016-1004-2 (Cited on pages 39, 57, 69).

[LKC07] S. D. Lee, B. Kao, and R. Cheng. "Reducing UK-Means to K-Means". In: *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*. IEEE Computer Society, 2007, pp. 483–488. DOI: 10.1109/ICDMW.2007.40 (Cited on page 66).

[LS22a] A. Lang and E. Schubert. "BETULA: Fast clustering of large data with improved BIRCH CF-Trees". In: *Inf. Syst.* 108 (2022), p. 101918. DOI: 10.1016/J.IS.2021.101918 (Cited on page 69).

[LS22b] L. Lenssen and E. Schubert. "Clustering by Direct Optimization of the Medoid Silhouette". In: *Similarity Search and Applications*. 2022, pp. 190–204. DOI: 10.1007/978-3-031-17849-8_15 (Cited on page 5).

[LS23] L. Lenssen and E. Schubert. "Sparse Partitioning Around Medoids". In: *Machine Learning under Resource Constraints – Fundamentals*. Ed. by K. Morik and P. Marwedel. Vol. 1. De Gruyter, 2023. Chap. 5.1, pp. 182–196. DOI: 10.1515/9783110785944-005 (Cited on pages 4 sq.).

[LS24] L. Lenssen and E. Schubert. "Medoid Silhouette clustering with automatic cluster number selection". In: *Information Systems* 120 (2024), p. 102290. ISSN: 0306-4379. DOI: 10.1016/j.is.2023.102290 (Cited on pages 5, 78).

[LSJ+25] L. Lenssen, P. Stahmann, C. Janiesch, and E. Schubert. "58th Hawaii International Conference on System Sciences, HICSS 2025". In: 2025. DOI: 10.24251/HICSS.2025.145 (Cited on pages 5 sq.).

[LSS23] L. Lenssen, N. Strahmann, and E. Schubert. "Fast k-Nearest-Neighbor-Consistent Clustering". In: *Lernen, Wissen, Daten, Analysen (LWDA) Conference Proceedings, Marburg, Germany, October 9-11, 2023*. Ed. by M. Leyer and J. Wichmann. Vol. 3630. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 387–398 (Cited on page 5).

[Mar63] F. E. Maranzana. "On the Location of Supply Points to Minimize Transportation Costs". In: *IBM Syst. J.* 2.2 (1963), pp. 129–135. DOI: 10.1147/SJ.22.0129 (Cited on pages 2, 19).

[Mur12] K. P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012. ISBN: 0262018020 (Cited on pages 8, 16 sq.).

[Mur83] F. Murtagh. "A Survey of Recent Advances in Hierarchical Clustering Algorithms". In: *Comput. J.* 26.4 (1983), pp. 354–359. DOI: 10.1093/COMJNL/26.4.354 (Cited on page 79).

[MWW+23] M. Muschkiet, T. Wulfert, R. Woroch, G. Strobel, and L. Banh. "Unleashing the digital building bricks A smart service taxonomy for retail". In: *Electronic Markets* 33 (10/2023), p. 51. DOI: 10.1007/s12525-023-00666-z (Cited on page 80).

[NEB10] X. V. Nguyen, J. Epps, and J. Bailey. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance". In: *J. Mach. Learn. Res.* 11 (2010), pp. 2837–2854. DOI: 10.5555/1756006.1953024 (Cited on page 13).

[NF16] J. Newling and F. Fleuret. "Fast k-means with accurate bounds". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by M. Balcan and K. Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 936–944 (Cited on page 69).

[NF17] J. Newling and F. Fleuret. "A Sub-Quadratic Exact Medoid Algorithm". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. Ed. by A. Singh and X. (Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 185–193 (Cited on page 31).

[NVM13] R. Nickerson, U. Varshney, and J. Muntermann. “A Method for Taxonomy Development and its Application in Information Systems”. In: *European Journal of Information Systems* 22 (05/2013). DOI: 10.1057/ejis.2012.26 (Cited on page 80).

[PHB00] J. Puzicha, T. Hofmann, and J. Buhmann. “A theory of proximity based clustering: Structure detection by optimization”. In: *Pattern Recognition* 33 (04/2000), pp. 617–634. DOI: 10.1016/S0031-3203(99)00076-X (Cited on page 16).

[PJ09] H. Park and C. Jun. “A simple and fast algorithm for K-medoids clustering”. In: *Expert Syst. Appl.* 36.2 (2009), pp. 3336–3341. DOI: 10.1016/J.ESWA.2008.01.039 (Cited on page 19).

[PMC+14] M. Pourrajabi, D. Moulavi, R. J. G. B. Campello, A. Zimek, J. Sander, and R. Goebel. “Model Selection for Semi-Supervised Clustering”. In: *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*. Ed. by S. Amer-Yahia, V. Christophides, A. Kementsietsidis, M. N. Garofalakis, S. Idreos, and V. Leroy. OpenProceedings.org, 2014, pp. 331–342. DOI: 10.5441/002/EDBT.2014.31 (Cited on page 27).

[Ran71] W. M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356 (Cited on page 28).

[RHR79] K. E. Rosing, E. L. Hillsman, and H. Rosing-Vogelaar. “A Note Comparing Optimal and Heuristic Solutions To the p-Median Problem”. In: *Geographical Analysis* 11.1 (1979), pp. 86–89. DOI: 10.1111/j.1538-4632.1979.tb00674.x (Cited on page 19).

[Rou87] P. J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7 (Cited on pages 1, 21, 45, 77).

[RRI+06] A. P. Reynolds, G. Richards, B. de la Iglesia, and V. J. Rayward-Smith. “Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms.” In: *J. Math. Model. Algorithms* 5.4 (2006), pp. 475–504. DOI: 10.1007/s10852-005-9022-1 (Cited on page 19).

[Sch21] E. Schubert. “HACAM: Hierarchical Agglomerative Clustering Around Medoids - and its Limitations”. In: *Proc. LWDA*. 2021, pp. 191–204. URL: https://ceur-ws.org/Vol-2993/paper-19.pdf (Cited on page 79).

[Sch22] E. Schubert. “Automatic Indexing for Similarity Search in ELKI”. In: *Similarity Search and Applications - 15th International Conference, SISAP 2022, Bologna, Italy, October 5-7, 2022, Proceedings*. Ed. by T. Skopal, F. Falchi, J. Lokoc, M. L. Sapino, I. Bartolini, and M. Patella. Vol. 13590. Lecture Notes in Computer Science. Springer, 2022, pp. 205–213. DOI: 10.1007/978-3-031-17849-8_16 (Cited on pages 69 sq.).

[Sch23] E. Schubert. “Stop using the elbow criterion for k-means and how to choose the number of clusters instead”. In: *SIGKDD Exploration Newsletter* 25.1 (07/2023), pp. 36–42. ISSN: 1931-0145. DOI: 10.1145/3606274.3606278 (Cited on pages 24, 53, 69, 77).

[SHM18] E. Schubert, S. Hess, and K. Morik. “The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering”. In: *Proceedings of the Conference ”Lernen, Wissen, Daten, Analysen”, LWDA 2018, Mannheim, Germany, August 22-24, 2018*. Ed. by R. Gemulla, S. P. Ponzetto, C. Bizer, M. Keuper, and H. Stuckenschmidt. Vol. 2191. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 330–334 (Cited on page 1).

[Sib73] R. Sibson. “SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method”. In: *Comput. J.* 16.1 (1973), pp. 30–34. DOI: 10.1093/COMJNL/16.1.30 (Cited on page 11).

[SL22] E. Schubert and L. Lenssen. “Fast k-medoids Clustering in Rust and Python”. In: *Journal of Open Source Software* 7.75 (2022), p. 4183. DOI: 10.21105/joss.04183 (Cited on pages 5, 21, 57).

[SLF21] E. Schubert, A. Lang, and G. Feher. “Accelerating Spherical k-Means”. In: *Similarity Search and Applications - 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 - October 1, 2021, Proceedings*. Ed. by N. Reyes, R. Connor, N. M. Kriege, D. Kazempour, I. Bartolini, E. Schubert, and J. Chen. Vol. 13058. Lecture Notes in Computer Science. Springer, 2021, pp. 217–231. DOI: 10.1007/978-3-030-89657-7_17 (Cited on page 65).

[SM58] R. R. Sokal and C. D. Michener. “A statistical method for evaluating systematic relationship”. In: *The University of Kansas Science Bulletin* 38 (1958), pp. 1409–1438 (Cited on pages 28, 75).

[Sne57] P. H. A. Sneath. “The Application of Computers to Taxonomy”. In: *Microbiology* 17.1 (1957), pp. 201–226. ISSN: 1465-2080. DOI: https://doi.org/10.1099/00221287-17-1-201 (Cited on page 11).

[SR19a] E. Schubert and P. J. Rousseeuw. “Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms”. In: *Proceedings of the 12th International Conference on Similarity Search and Applications, SISAP 2019, Newark, NJ, October 2-4*. 2019, pp. 171–187. URL: https://doi.org/10.1007/978-3-030-32047-8_16 (Cited on pages 20, 35 sq., 45, 48, 51, 60).

[SR19b] Z. Sulc and H. Rezanková. “Comparison of Similarity Measures for Categorical Data in Hierarchical Clustering”. In: *J. Classif.* 36.1 (2019), pp. 58–72. DOI: 10.1007/S00357-019-09317-5 (Cited on page 29).

[SR21] E. Schubert and P. J. Rousseeuw. “Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms”. In: *Information Systems* 101 (2021), p. 101804. DOI: 10.1016/j.is.2021.101804 (Cited on pages 19 sq., 35 sq., 38, 45, 48 sq., 51, 53, 60, 78).

[SSE+17] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”. In: *ACM Trans. Database Syst.* 42.3 (2017), 19:1–19:21. DOI: 10.1145/3068335 (Cited on page 1).

[SSH+24] F. Sterk, A. Stocker, D. Heinz, and C. Weinhardt. “Unlocking the value from car data: A taxonomy and archetypes of connected car business models”. In: *Electron. Mark.* 34.1 (2024), p. 13. DOI: 10.1007/S12525-024-00692-5 (Cited on page 80).

[SZ19] E. Schubert and A. Zimek. "ELKI: A large open-source library for data analysis - ELKI Release 0.7.5 "Heidelberg"". In: *CoRR* abs/1902.03616 (2019). arXiv: 1902.03616 (Cited on page 39).

[TB68] M. B. Teitz and P. Bart. "Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph". In: *Oper. Res.* 16.5 (1968), pp. 955–961. DOI: 10.1287/OPRE.16.5.955 (Cited on page 19).

[Tov97] C. A. Tovey. "Local improvement on discrete structures". In: *Local Search in Combinatorial Optimization*. Ed. by E. Aarts and J. K. Lenstra. Princeton: Princeton University Press, 1997, pp. 57–90. ISBN: 9780691187563. DOI: doi:10.1515/9780691187563-006 (Cited on page 10).

[TZM+20] M. Tiwari, M. J. Zhang, J. Mayclin, S. Thrun, C. Piech, and I. Shomorony. "BanditPAM: Almost Linear Time k-Medoids Clustering via Multi-Armed Bandits". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. 2020 (Cited on page 20).

[VCH10] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka. "Relative clustering validity criteria: A comparative overview". In: *Stat. Anal. Data Min.* 3.4 (2010), pp. 209–235. DOI: 10.1002/sam.10080 (Cited on pages 1, 23, 45, 63).

[VPB03] M. Van der Laan, K. Pollard, and J. Bryan. "A new partitioning around medoids algorithm". In: *Journal of Statistical Computation and Simulation* 73.8 (2003), pp. 575–584. DOI: 10.1080/0094965031000136012 (Cited on pages 23, 25, 48, 78).

[WCR+01] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. "Constrained K-means Clustering with Background Knowledge". In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Ed. by C. E. Brodley and A. P. Danyluk. Morgan Kaufmann, 2001, pp. 577–584 (Cited on pages 27, 73, 89).

[WWW+23] J. Wanner, C. Wissuchek, G. Welsch, and C. Janiesch. "A Taxonomy and Archetypes of Business Analytics in Smart Manufacturing". In: *ACM SIGMIS Database* 54.1 (2023), pp. 11–45. DOI: 10.1145/3583581.3583584 (Cited on pages 1, 80, 82 sqq.).

[YGY02] Y. Yang, X. Guan, and J. You. "CLOPE: a fast and effective clustering algorithm for transactional data". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM, 2002, pp. 682–687. DOI: 10.1145/775047.775149 (Cited on page 29).

[ZPA+07] M. J. Zaki, M. Peters, I. Assent, and T. Seidl. "Clicks: An effective algorithm for mining subspace clusters in categorical datasets". In: *Data Knowl. Eng.* 60.1 (2007), pp. 51–70. DOI: 10.1016/J.DATAK.2006.01.005 (Cited on page 29).