CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

CLUE
Clustering-Based Load Understanding and Exploration

Summarizing High-Dimensional Electricity Grid Data for Scenario Analysis

Master's thesis in Computer science and engineering

Linus Magnusson & Rasmus Thorsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

CLUE
Clustering-Based Load Understanding and Exploration

Summarizing High-Dimensional Electricity Grid Data for Scenario Analysis

Linus Magnusson & Rasmus Thorsson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

CLUE Clustering-Based Load Understanding and Exploration
Summarizing High-Dimensional Electricity Grid Data for Scenario Analysis
Linus Magnusson & Rasmus Thorsson

Supervisor: Marina Papatriantafilou & Quang Vinh Ngo, Department of Computer Science and Engineering
Advisors: Joris van Rooij & Mihail Chigrichenko, Göteborg Energi AB
Examiner: Marina Papatriantafilou, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2025

CLUE Clustering-Based Load Understanding and Exploration
Summarizing High-Dimensional Electricity Grid Data for Scenario Analysis
Linus Magnusson & Rasmus Thorsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The deployment of Advanced Metering Infrastructure (AMI) in electricity grids generates vast volumes of high-dimensional time series data, presenting significant challenges for practical analysis and pattern discovery. This thesis presents CLUE (Clustering-Based Load Understanding and Exploration), a modular and flexible toolchain designed to process and analyze high-dimensional temporal data efficiently.

The CLUE toolchain addresses fundamental challenges in time series clustering, including computational complexity, difficulties in parameter selection, and the gap between algorithmic capabilities and domain expertise. The toolchain integrates multiple clustering algorithms, with a focus on IP.LSH.DBSCAN (Integrated Parallel Locality-Sensitive Hashing DBSCAN), which achieves speedups magnitudes higher compared to traditional DBSCAN while maintaining adequate clustering quality. The toolchain features automated parameter optimization through k-distance analysis and multi-metric evaluation, eliminating the need for extensive manual tuning. Additionally, CLUE supports multiple distance metrics (Euclidean, Angular, and Dynamic Time Warping) and provides flexible data representation options through both raw time series and feature-based approaches.

The toolchain's effectiveness is demonstrated through a case study with Göteborg Energi, analyzing electricity consumption patterns from approximately 7,500 customers. The evaluation reveals CLUE's ability to identify meaningful consumption profiles, detect anomalies, and enable interactive exploration of complex temporal consumer patterns.

This work contributes a practical solution for organizations facing the challenge of extracting meaningful insights from large-scale time series data, with applications extending beyond electricity grids to any domain requiring efficient analysis of high-dimensional temporal patterns.

Keywords: CLUE, Time series clustering, High-dimensional data, DBSCAN, IP.LSH.DBSCAN, Locality-Sensitive Hashing, K-Means, Electricity consumption analysis, Parameter optimization,

Acknowledgements

We want to express our sincere gratitude to our supervisors, Marina Papatriantafilou and Quang Vinh Ngo, for their invaluable guidance and insightful feedback throughout this project. We extend special thanks to our industry advisors at Göteborg Energi AB, Joris van Rooij, and Mihail Chigrichenko, for providing us with this exciting opportunity and their practical expertise in the energy sector. We also thank Göteborg Energi AB for granting access to their data and infrastructure. Finally, we would like to thank the group of students and supervisors who have read, listened to, and given valuable feedback during this thesis.

The grammar and spelling of this document were reviewed using Grammarly.

Linus Magnusson, Gothenburg, 2025-06-17
Rasmus Thorsson, Gothenburg, 2025-06-17

Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
ε	Epsilon parameter for neighborhood radius
$MinPts$	Minimum points required for core point designation
L	Number of hash tables in LSH
M	Number of hyperplanes per hash table
k	Number of clusters (K-means) or nearest neighbors
d	Dimensionality of data vectors
n	Number of data points
ARI	Adjusted Rand Index value
DB	Davies-Bouldin Index
SC	Silhouette Coefficient
CH	Calinski-Harabasz Index
DI	Dunn Index
$DBCV$	Density-Based Clustering Validation score
AMI	Advanced Metering Infrastructure
$DBSCAN$	Density-Based Spatial Clustering of Applications with Noise
DTW	Dynamic Time Warping
$IP.LSH.DBSCAN$	Integrated Parallel Locality-Sensitive Hashing DBSCAN
LSH	Locality-Sensitive Hashing
PCA	Principal Component Analysis
$t-SNE$	t-Distributed Stochastic Neighbor Embedding
$UMAP$	Uniform Manifold Approximation and Projection

1

Introduction

Data analytics has become fundamental across domains, transforming how organizations understand patterns and make decisions. As analytics capabilities have evolved, they have become increasingly dependent on processing large volumes of complex data to extract actionable insights. The deployment of Advanced Metering Infrastructure (AMI) in electricity grids has generated unprecedented volumes of time series data, presenting both opportunities and challenges for data analytics.

Clustering, a powerful unsupervised learning technique, offers significant potential for making sense of this complex data. By grouping similar temporal data patterns, clustering can identify consumption patterns, detect anomalies, and reveal underlying structures without requiring predefined classifications. In the context of electricity consumption, clustering can transform millions of individual electricity meter readings into comprehensible behavioral profiles, providing both system-wide understanding and customer-specific insights.

1.1 Motivation

The growing deployment of sensor networks, intelligent monitoring systems, and AMI has created a pressing need for sophisticated analytical frameworks to process high-dimensional data efficiently. Traditional analytical approaches face significant limitations when applied to these complex datasets [1], including computational constraints, inherent problems associated with high-dimensional datasets, and difficulties in effectively capturing temporal dependencies [2].

In the utility sector, high-frequency monitoring generates rich temporal data that, when analyzed appropriately, can reveal consumption behaviors, operational inefficiencies, and opportunities for optimization. However, extracting these insights requires a flexible analytical framework that addresses domain-specific requirements while maintaining computational efficiency.

This thesis was done in collaboration with Göteborg Energi AB. The Swedish energy provider collects electricity consumption measurements from thousands of electricity meters through their AMI, measuring consumption every 15 minutes, resulting in 96 measurements per day for each consumer. These detailed consumption profiles reveal valuable patterns that reflect user behavior, grid performance, and opportunities for optimization. By efficiently processing these multifaceted temporal patterns, the

collaboration aims to enable Göteborg Energi to transform complex measurement data into actionable intelligence, thereby enhancing customer segmentation, improving demand forecasting, and identifying opportunities for grid optimization. This demonstrates how effective time series clustering can address practical challenges in the energy sector, providing a foundation for data-driven decision-making.

1.2 Clustering as Exploratory Analysis

Unlike deterministic queries that extract predefined data relationships, clustering represents a fundamentally exploratory analytical approach. This exploratory nature provides unique value for time series analysis while creating distinct challenges.

Traditional database queries (e.g., SQL) answer specific, predetermined questions based on known data relationships: "Which customers consumed more than X kilowatts during these specified hours?" In contrast, clustering reveals relationships that were not predefined or even suspected, identifying natural groupings that might challenge existing assumptions about consumption patterns [3]. For electricity utilities, this can reveal customer segments that reach across traditional classification schemes based on industry codes or contract types.

1.3 Contributions

This thesis presents CLUE (Clustering-based Load Understanding and Exploration), a flexible and integrated framework with interchangeable components for clustering algorithms, distance metrics, parameter optimization, and feature engineering. This modular architecture enables configuration for diverse analytical requirements across multiple domains, allowing users to select components based on their specific needs.

An advanced density-based clustering algorithm is implemented and extended to provide performance improvements over traditional clustering approaches. This algorithm achieves significant speedups compared to conventional algorithms, while maintaining high clustering quality. This enables interactive analysis of complex data that is difficult with traditional methods.

Our work develops methods for identifying suitable clustering parameters through automated exploration techniques, enabling effective parameter selection without the need for extensive manual tuning. This includes k-distance analysis, elbow detection, and multi-metric evaluation to identify robust parameter sets.

The toolchain features a configurable feature engineering system that transforms raw temporal data into meaningful attributes. This captures relevant behavioral patterns while reducing dimensionality, allowing users to focus their analysis on domain-specific characteristics. Furthermore, the toolchain's effectiveness is demonstrated by applying it to electricity consumption data from Göteborg Energi, illustrating how it enables enhanced customer segmentation, peak demand analysis, and informed decision-making. While we showcase the toolchain using electricity grid

data, the underlying architecture supports various analytical tasks across multiple domains with similar data challenges.

1.4 Overview

The remainder of this thesis is organized as follows: Chapter 2 provides relevant background on clustering algorithms, time series data analysis, and evaluation methodologies. Chapter 3 defines the problem of efficiently clustering high-dimensional time series vectors while maintaining interpretability, using electricity consumption data as a concrete example. Chapter 4 details the CLUE toolchain architecture and its core components. Chapter 5 evaluates the performance, accuracy, and practical applicability of our approach through both synthetic and real-world electricity consumption data. Chapter 6 discusses the implications of our findings and their practical significance. Chapter 7 outlines directions for future work, and Chapter 8 concludes with a summary of contributions and impact.

2

Background

The analysis of time series data presents unique challenges due to its temporal nature, high dimensionality of vectors, and often noisy characteristics [4]. This chapter provides the necessary background on time series clustering, establishing the theoretical foundation for the rest of this thesis.

2.1 Clustering in Data Mining

Clustering is a data mining technique that seeks to organize objects into groups [5], or clusters, where objects within a cluster exhibit high similarity to each other and low similarity to objects in different clusters. As defined by Han et al. [6], the objective of clustering is to maximize intra-cluster similarity while minimizing inter-cluster similarity. Unlike supervised learning methods, which require labeled data, clustering is an unsupervised approach, making it particularly valuable when exploring patterns in large datasets without prior knowledge of class definitions. The process of clustering reveals inherent structures in the data, facilitating knowledge discovery and supporting decision-making across numerous domains [7].

2.1.1 Overview of Clustering Approaches

Clustering methods encompass various algorithms that can be categorized in multiple ways depending on their underlying principles. According to Aghabozorgi et al. [8], clustering algorithms can be classified based on various criteria, including partitioning approach, hierarchy formation, density consideration, and model assumptions. For this thesis, we focus on several prominent clustering paradigms, particularly relevant to time series data:

Partitioning Methods: These algorithms divide data objects into non-overlapping subsets (clusters) such that each object belongs to exactly one cluster [9]. K-means and k-medoids (PAM) are classic examples of partitioning methods that iteratively refine clusters by minimizing a distance-based objective function. These approaches require pre-specification of the number of clusters (k) and typically produce spherical clusters.

Density-Based Methods: These algorithms define clusters as dense regions of objects separated by areas of lower density. DBSCAN [10] (Density-Based Spatial Clustering of Applications with Noise) is a prominent example that discovers clusters

of arbitrary shapes and effectively handles noise. These methods typically do not require pre-specification of the number of clusters and can identify outliers.

2.1.2 Challenges in Modern Clustering Applications

As data volumes grow exponentially and complex data types become more prevalent, clustering faces several significant challenges:

High-Dimensional Data: The dimensionality of a data point in this context refers to how many elements are contained in the vector. A data point $p = \{p_1, p_2, ..., p_n\}$ is said to be n-dimensional. The "curse of dimensionality" affects clustering algorithms when working with high-dimensional data [11]. As dimensionality increases, distance measures become less discriminative, making it difficult to identify meaningful clusters. This challenge is particularly acute for time series vectors, where each measurement point can be considered a dimension, and the use of improved measurement systems leads to an increase in granularity of measurement points [12].

Time Series Data: Time series vectors introduce unique complexities due to their temporal ordering, variable lengths, presence of noise, and potential shifts in time and amplitude [13]. Traditional clustering algorithms often fail to capture the sequential nature of time series data effectively. The high feature correlation in time series further complicates the application of standard clustering approaches [14].

Scalability Issues: With the emergence of big data, clustering algorithms must scale to handle massive datasets efficiently. Many traditional approaches have quadratic or higher computational complexity, making them impractical for large-scale applications. This problem is amplified for time series vectors, which tend to be high-dimensional [15].

2.2 Density-Based Clustering Approaches

Density-based clustering methods define clusters based on the density distribution of data points, making them particularly suitable for discovering clusters of arbitrary shapes and handling noise [16]. According to Ester et al. [10], these methods identify dense regions separated by sparse regions, allowing for the detection of clusters without assuming specific shapes or requiring predefined cluster numbers.

2.2.1 Concepts of Density-Based Clustering

Density-based clustering relies on several key concepts initially formalized by Ester et al. [10] in their seminal DBSCAN paper:

Core Points, Border Points, and Noise: A core point contains at least a minimum number of points (MinPts) within a specified radius epsilon (ε). A border point falls within the neighborhood of a core point but lacks sufficient neighbors to be considered a core point in its own right. Points that are neither core nor border points are classified as noise.

Density Connectivity and Reachability: Two points are directly density-reachable if one is a core point and the other lies within its ε-neighborhood. Density-connectivity extends this concept through chains of directly density-reachable points. These concepts form the foundation for defining clusters as maximal sets of density-connected points.

2.2.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a fundamental density-based clustering algorithm introduced by Ester et al. [10]. The algorithm identifies clusters as dense regions separated by areas of lower density, making it particularly well-suited for datasets where clusters may have irregular shapes and sizes.

Two essential parameters control the algorithm's behavior: ε and MinPts. Epsilon (ε) is the radius that defines the neighborhood of a point. MinPts is the minimum number of points required within the neighborhood to mark a point as a core point.

The Parameter selection is critical, as inappropriate values can lead to poor clustering results. Various heuristics and visualization techniques, such as k-distance graphs proposed in the original paper [10], can guide parameter selection.

The algorithm is described in more detail in Pseudocode 2.1. For a dataset D with parameters epsilon and minimum points (MinPts), the DBSCAN algorithm operates as follows:

```
DBSCAN
Input: Dataset D, epsilon ε, minPts
Output: Clustering of the dataset with cluster labels

1. clusterIndex = 0

2. For each point p in D:
If p is not processed:
  Mark p as processed
   Get neighbors N = {points within  distance of p}
    If |N| < minPts:
     Mark p as noise
Else:
  Create a new cluster with label clusterIndex
  Mark p as core point and assign to clusterIndex
  Expand_Cluster(p, N, clusterIndex, ε, minPts)
  Increment clusterIndex

Expand_Cluster(p, N, clusterIndex, ε, minPts):
1. For each point q in N:
If q is noise:
 Assign q to clusterIndex

```

```
If q is not processed:
  Mark q as processed
    Get neighbors N_q = {points within ε distance of q}
      If |N_q|  minPts:
        Mark q as core point
        Add all points in N_q to N
    Assign q to clusterIndex
```

Pseudocode 2.1: DBSCAN Algorithm Implementation

The time complexity of DBSCAN is $O(n^2)$, where n is the number of data points. This quadratic complexity stems from the need to compute distances between all pairs of points to determine neighborhoods.

2.2.3 Integrated Parallel Density-Based Clustering through Locality-Sensitive Hashing (IP.LSH.DBSCAN)

Integrated Parallel Density-Based Clustering through Locality-Sensitive Hashing (IP.LSH.DBSCAN), developed by Keramatian et al. [17], represents a significant advancement in density-based clustering algorithms by fusing the indexing and clustering processes. Unlike traditional approaches (DBSCAN [10]) that treat indexing as a separate pre-processing step, IP.LSH.DBSCAN integrates the creation of a Locality-Sensitive Hashing (LSH) index with the formation of clusters, resulting in an efficient parallel algorithm that scales effectively with both dataset size and dimensionality.

Understanding Locality-Sensitive Hashing

Unlike conventional hash functions that aim to spread data evenly across buckets, Locality-Sensitive Hashing (LSH) [18] functions are designed to place similar items into the same buckets. This "similarity-preserving" property works on a simple principle: the more similar two data points are, the higher the probability they will hash to the same value.

LSH achieves this by using mathematical functions tailored to specific distance measures:

- For Euclidean distance: Functions project data onto random directions and divide the resulting line into segments [19].
- For Angular distance: Functions create random hyperplanes and determine which side of the hyperplane the points fall on.

The LSH structure in IP.LSH.DBSCAN uses L independent hash tables, increasing the probability of finding neighbours. M hash functions are used per hash table to control the granularity. Each combination of L and M creates a tradeoff between precision and computational cost.

Although requiring slightly more pre-configuration, this approach transforms the computationally expensive neighborhood searches in traditional DBSCAN [10] into

simple bucket lookups, effectively pre-grouping points likely to belong to the same cluster.

Core principles and Innovations

The key innovation of IP.LSH.DBSCAN employs an integrated approach that leverages the natural grouping properties of locality-sensitive hashing to identify density-connected regions efficiently. The algorithm operates by hashing each data point into multiple hash tables, ensuring that points within ε-distance will, with high probability, be hashed into the same bucket at least once across all tables. This property allows the algorithm to approximate DBSCAN's neighborhood queries without performing exhaustive distance calculations.

Unlike previous approaches that used LSH as an indexing structure for neighborhood queries (VLSHDBSCAN [20], [21]), IP.LSH.DBSCAN's innovation comes from:

1. **Fusion of indexing and clustering:** The LSH structure creation is directly integrated with the cluster formation process.
2. **Core-point representation:** Each bucket contains at least minPts elements that identify a candidate core-point, which is the closest point to the centroid of all points in the bucket.
3. **Efficient merging strategy:** Rather than performing individual point-to-point neighborhood queries, IP.LSH.DBSCAN merges core points, representing larger entities (buckets), which significantly reduces computational complexity.

Algorithm Structure and Phases

IP.LSH.DBSCAN operates through a structured four-phase process integrating LSH indexing with density-based clustering. In this initial hashing and bucketing phase, each data point is hashed into multiple hash tables using Locality-Sensitive Hashing (LSH) functions for the specified distance metric. During this process, the algorithm identifies buckets containing at least minPts elements as candidate core buckets, establishing potential density centers.

Following the initial hashing, the core-point identification phase examines each candidate core bucket to determine its representative point. The algorithm selects the point closest to the centroid of all points in the bucket as the candidate core-point. This point becomes a true core point if at least minPts points within the bucket fall within its ε-neighborhood; at this point, it is inserted into a core-forest data structure that will later represent cluster relationships.

The algorithm then proceeds to the merge-task identification and processing phase, analyzing spatial relationships between core buckets. For each core bucket, the algorithm identifies pairs of core points within ε-distance of each other and merges their respective sets in the core forest. This merging is efficiently handled through a concurrent union-find data structure that maintains the evolving cluster topology.

In the final data labeling phase, points receive cluster assignments based on their relationship to identified core points. Core points belonging to the same set in the core forest are assigned identical cluster labels. Non-core points (border points) that fall within the ε-neighborhood of a core-point inherit the core-point's cluster label, while all remaining points are classified as noise, completing the DBSCAN-style clustering.

As shown in Pseudocode 2.2, the implementation utilizes L hash tables, each employing M randomly generated hyperplanes to partition the data space. Points are hashed into buckets, in which similar points are also hashed with a high probability. Dense buckets containing at least MinPts points become candidates for core point identification, then a concurrent union-find data structure merges clusters in parallel. The parameters L and M provide control over the accuracy-speed tradeoff, allowing balancing of computational efficiency against clustering precision.

```
IP.LSH.DBSCAN
Input: Dataset D, epsilon ε, minPts, numberOfhash tables L,
   hyperplanesPerTable M
Output: Clustering of the dataset with cluster labels
   assigned to each point

1. Initialize L hash tables with M random hyperplanes each

2. Populate hash tables:
For each point p in D:
  For each hash table HT:
    Calculate the hash value of p using LSH functions
    Add p to the appropriate bucket in HT

3. Identify core buckets:
For each hash table HT:
  For each bucket B in HT:
    If |B| ≥ minPts:
      Find the representative point r (closest to the
         centroid)
      If r has at least minPts neighbors within ε:
        Mark r as a core point
        Create a core bucket with r as representative

4. Merge core buckets:
For each core bucket CB:
  For each point p in CB:
    If p is a core point:
      Link p to the representative of CB

5. Relabel points:
For each point p in D:
  If p is connected to a core point:
    Assign p the label of its root core point
  Else:
```

```
33      Mark p as noise
```

Pseudocode 2.2: IP.LSH.DBSCAN Algorithm Implementation

Parallelization Strategy

The efficiency of IP.LSH.DBSCAN is based on its parallelization strategy, which enables concurrent processing across multiple threads. The algorithm employs logical partitioning by dividing the dataset into mutually disjoint batches, creating independent hash tasks that can be distributed among available computing threads. This partitioning enables parallel processing from the earliest stages of the algorithm.

Fine-grained synchronization mechanisms, utilizing atomic operations and concurrent data structures, enhance parallelism by enabling threads to operate concurrently without requiring data duplication. The algorithm's core forest implementation uses a concurrent union-find structure [22] with linearizable and wait-free operations, enabling multiple threads to perform cluster merges simultaneously without conflicts.

By working with pointers and references to datapoints and buckets rather than creating additional copies, the algorithm achieves in-place operations that significantly reduce memory overhead, a consideration for large datasets. This approach to memory management, combined with the work distribution among threads, allows IP.LSH.DBSCAN to process large, high-dimensional datasets more efficiently than traditional parallel DBSCAN implementations.

Performance and Theoretical Properties

IP.LSH.DBSCAN offers several key performance advantages over standard density-based clustering methods. The algorithm processes high-dimensional data much more effectively than traditional approaches, avoiding the typical performance degradation caused by the curse of dimensionality [11]. It also works well with skewed data distribution, a common issue in real-world datasets where points are not evenly spread.

From a theoretical perspective, IP.LSH.DBSCAN provides necessary probabilistic guarantees on clustering quality. The algorithm ensures that any point it identifies as a core point is a true core point according to DBSCAN's definition. Furthermore, for any core point in the exact DBSCAN solution, the probability that IP.LSH.DBSCAN also identifies that approaches one as the number of hash tables increases, demonstrating convergence to the exact solution.

2.3 Partitioning-Based Clustering (K-Means)

The k-means algorithm is one of the most influential and extensively studied clustering algorithms in data mining and machine learning. Initially proposed by MacQueen in 1967 [23], it has become the default choice in practical applications due to its conceptual simplicity, computational efficiency, and effectiveness in many real-world scenarios.

The k-means clustering problem can be formally stated as an optimization problem. Given a dataset where each element is represented as a d-dimensional data point, the objective is to partition the n data points into k clusters, such that the within-cluster sum of squares (WCSS) is minimized.

The most common k-means implementation, often referred to as Lloyd's algorithm [24], employs an iterative refinement approach based on alternating optimization. The algorithm alternates between two steps, corresponding to the optimization of the objective function for the assignments and the centroids separately.

Since this algorithm is guaranteed to converge to a local minimum of the objective function in a finite number of iterations, the quality of the local minimum highly depends on the initialization, which has led to extensive research on initialization strategies.

Poorly selected initial centroids can lead to several problematic scenarios:

- Slow Convergence: Poor initial positions may necessitate numerous iterations to achieve convergence, thereby impacting computational efficiency.
- Empty clusters: Some initial centroids may be placed in sparse regions, leading to empty clusters during iteration.
- Imbalanced Partitions: Initial centroids clustered in one region of the data space can result in highly imbalanced cluster sizes.

The k-means++ initialization algorithm, introduced by Arthur and Vassilvitskii [25], represents a theoretical and practical advancement in addressing the initialization problem. The key innovation lies in its probabilistic approach to centroid selection, which spreads initial centers through the data space while maintaining theoretical guarantees on solution quality.

The k-means algorithm with k-means++ initialization is described in Pseudocode 2.3 below.

```
Algorithm: K-Means
Input: Dataset D, number of clusters k, maxIterations
Output: k clusters with assigned points

1. Initialize centroids using K-Means++:
     Choose the first centroid randomly from D
     For i = 2 to k:
      Calculate D(p) = min distance to existing centroids for
          each point p
      Choose the next centroid with probability proportional
         to D(p)^2

2. While (iterations < maxIterations and not converged):
     Assign each point to the nearest centroid
     Update centroids as the mean of assigned points
     Check convergence

```

```
16 3. Return clusters
```

Pseudocode 2.3: K-Means Algorithm Implementation

The time complexity for traditional k-means and k-means++ algorithms are $O(i * n * k * d)$ where i is the number of iterations required for convergence, n is the number of data points, k is the number of clusters or centroids, and d is the dimensionality of the data points. The k-means++ initialization typically incurs the exact time cost as one iteration of the main algorithm; this overhead is justified since k-means++ generally requires fewer iterations before convergence and yields better-quality clusters.

2.4 Time Series Data Analysis

Time series data, characterized by sequences of observations collected chronologically, requires specialized analysis techniques that account for their temporal nature [26].

2.4.1 Representation Methods for Time Series Data

Time series vector data can be represented in various ways. The two most relevant are **Feature-based** representation and **Raw-data-based** representation. The selection of the representation approach impacts many other decisions regarding the clustering. For example, different distance metrics are more applicable to certain representations than others.

2.4.2 Raw-data-based

Raw-data-based representation performs no transformations or operations on the data and clusters it 'as-is'. This representation preserves all original information, but it can lead to high dimensionality and increased computational cost. Clustering typically involves treating each measurement point as a dimension, potentially leading to vectors with hundreds or thousands of dimensions. Depending on the data characteristics, this representation can lead to slow clustering speeds and issues inherent to high-dimensional data [8].

This representation is particularly valuable when the exact shape and timing of the time series are crucial for clustering and when the data dimensionality is manageable. Shape-based distance measures such as Dynamic Time Warping (DTW) [27] work well when coupled with raw-data-based representations.

2.4.3 Feature-based

Feature-based representation involves applying various feature extraction operations to the raw data to create feature vectors for clustering, rather than clustering the raw data directly. These features correlate with the raw data but take vastly different forms. For example, features can be statistics, transformations, or data aggregates

[28]. Furthermore, dimensionality reduction techniques such as *Principal Component Analysis (PCA)* [29] can be used to reduce the dimensions of the raw data into features semi-automatically. Feature extraction, as a dimensionality reduction technique, is beneficial when the dimensionality of the original data is expected to cause problems during clustering, such as those caused by high-dimensionality [30].

Feature-based representation can be considered when the loss of quality from feature extraction is regarded as an acceptable cost for dimensionality reduction. This approach should be employed if the extracted features are sufficient to capture the groupings accurately and when precise temporal ordering is less critical than the overall patterns or statistics.

2.5 Evaluation Methodology

Assessing the quality of time series clustering results requires evaluation methodologies that account for static quality and evolutionary dynamics. Evaluation approaches can be categorized into internal measures, which assess clusters without external references, and external measures, which compare against ground truth. Each category offers perspectives on clustering quality, with varying applicability depending on available data and analysis objectives.

2.5.1 Internal Validation Measures

Internal validation measures assess clustering quality using only the intrinsic properties of the data, without requiring external reference information or ground truth labels. These measures assess the quality of well-formed clusters by examining the structural characteristics of the clustering solution itself [31].

Internal validation approaches can be broadly categorized into measures that focus on cluster compactness and separation, comparative indices that evaluate relative clustering quality across different solutions, and density-based metrics that assess the identification of dense regions separated by sparse areas. These different perspectives offer complementary insights into clustering performance, particularly when dealing with complex time series data that exhibit irregular patterns or varying densities [7].

The advantage of internal validation lies in its independence from external information, making it applicable across diverse clustering scenarios where ground truth is unavailable or undefined.

2.5.2 External Validation Measures

When ground truth labels are available, external validation measures provide additional evaluation capabilities by comparing clustering results against known classifications or categorizations.

While external validation measures tend to provide a more accurate measure of cluster quality, the requirement for available ground truth limits their applicability

to datasets with predefined classifications.

3

Problem Definition

This chapter defines the general problem of efficiently clustering high-dimensional time series vectors while maintaining interpretability. We explore the fundamental challenges in this domain and illustrate them using electricity consumption data as a concrete example, creating a bridge between theoretical concerns and practical implementation.

3.1 Case Study: Göteborg Energi AB

Göteborg Energi is one of Sweden's largest regional energy companies, providing electricity, district heating, cooling, gas, and other energy-related services to businesses and residents in Gothenburg. The company serves over 260,000 customers and operates an electricity distribution network that spans both urban and suburban areas. In recent years, Göteborg Energi has expanded its usage of smart meters, which provide highly granular and voluminous data, increasing the need for sophisticated data analysis solutions.

Grid Capacity
The company's internal forecasts anticipate significant increases in electricity demand over the next decade. These projections suggest potential capacity limitations in specific grid segments, necessitating substantial investments in infrastructure or the implementation of innovative demand management strategies, such as pricing signals and general communication, to address these limitations and improve the utilization of the existing grid.

Peak Demand Management
While serving an industry-heavy customer base, Göteborg Energi Nät experiences significant variations in demand throughout the day, with pronounced peak periods. Understanding the composition of these peaks during days of the year when the strain on the system is extremely high (such as cold working days in winter) and the behavioral patterns that drive peak consumption are crucial to developing targeted demand response programs and optimizing grid operations.

Customer Behavior Knowledge
Energy consumption patterns often transcend industry boundaries, with similar behaviors observed across different business types, while significant variations can exist within the same category. Exploring consumption data and patterns enables the dis-

covery of natural behavioral groupings based on actual consumption, rather than predetermined classifications. This exploratory approach enables energy providers to extract deeper insights from existing data, potentially revealing hidden patterns that can inform more effective services and communication.

Objectives
The primary objective is to develop an understanding of Göteborg Energi's diverse customer base through clustering techniques. Identifying distinct groups based on electricity consumption patterns enables customer categorization methods that rely on meaningful behavioral differences in how customers consume electricity throughout the day.

This segmentation would enable Göteborg Energi Nät to:

- Identify customer groups with similar consumption that span industry boundaries
- Recognize emerging consumption trends that might indicate shifts in business operations or energy usage
- Establish baseline behavioral profiles, against which future consumption patterns can be compared
- Understand the diversity of consumption patterns within traditional customer categories
- Utilize specialized communication with their customers on a large scale by the use of cluster statistics

3.2 High-Dimensional Time Series Clustering Challenges

Time series data presents unique analytical challenges that traditional clustering approaches struggle to address effectively. These challenges become particularly acute when dealing with high-dimensional time series vectors in modern sensor networks, IoT devices, and AMI.

3.2.1 Dimensionality Effects on Distance Metrics and Computation

The challenges associated with high-dimensional data, originally coined the *"Curse of Dimensionality"* by Bellman [11], describe phenomena that occur when analyzing high-dimensional data, significantly altering the performance of clustering algorithms. These effects directly impact our ability to identify meaningful patterns in time series data [32].

The difference between the distances to the nearest and farthest neighbors tends to converge in high-dimensional spaces. Beyer et al. [33] demonstrated that all

pairwise distances between points become increasingly similar as dimensionality increases. For electricity consumption data collected at 15-minute intervals over a day, each daily vector has 96 dimensions (24 hours x 4 measurements per hour). In this 96-dimensional space, traditional distance metrics like Euclidean distance lose effectiveness [34], as even fundamentally different consumption patterns (such as morning peak vs. evening peak) may appear similarly distant.

Computational complexity represents another significant challenge. Most clustering algorithms scale poorly with dimensionality. DBSCAN [10] has $O(n^2)$ time complexity in its naive implementation, where n is the number of data points. When handling time series data, where a single week sampled at 15-minute intervals yields 672-dimensional vectors, the distance calculations become computationally expensive for large datasets.

3.2.2 Temporal Dependencies and Structure

Time series data contains inherent sequential structures and temporal dependencies that general-purpose clustering algorithms do not naturally accommodate. These characteristics create additional challenges for effective pattern discovery.

When comparing time series, similar patterns may occur with slight shifts in time or variations in duration. For example, in electricity consumption data, two businesses might exhibit nearly identical morning activity patterns, but one might start operations 30 minutes earlier than the other. Despite their structural similarity, these patterns might appear quite different under standard high-dimensional distance metrics. Dynamic Time Warping (DTW) [27] addresses this problem by finding the optimal matching between time points; however, it introduces higher computational costs.

The contextual nature of time series data introduces an additional layer of complexity. The meaning and importance of measurements depend on their temporal context. For example, consider measured electricity consumption; a spike of 10 kilowatts at 07:00 might represent regular household activity, while the exact measurement at 02:00 could indicate an anomaly. Effective clustering must preserve and account for these contextual relationships.

Time series data frequently exhibits patterns operating at different temporal resolutions simultaneously. Daily cycles, weekly patterns, monthly billing periods, and seasonal trends coexist in the data. Fixed-window approaches that treat each day or week as an independent vector may capture fine-grained patterns but miss longer-term trends. In contrast, aggregation approaches that summarize data at coarser resolutions lose detail that might be needed for specific analytical questions [35].

3.3 The Parameterization Problem

Clustering algorithms, particularly density-based methods such as DBSCAN [10], require careful parameter selection to yield meaningful results. This presents several interconnected challenges that are particularly relevant for time series analysis:

Parameters such as epsilon (ε) and the minimum number of points (MinPts) in DBSCAN interact in complex ways with the data. In electricity consumption clustering, the appropriate value depends on both the scale of consumption values and the distribution of consumption patterns. Similarly, the appropriate MinPts values depend on the data dimensionality and the relative sizes of meaningful customer segments. Parameters often have different appropriate values depending on the scale and distribution of the data, requiring domain knowledge or extensive experimentation to determine [36].

3.4 Feature Engineering vs. Raw Data Approaches

When clustering time series data, a fundamental choice exists between working with raw temporal measurements or extracting representative features, each with significant implications for cluster quality and interpretability.

Raw time series vectors from electricity meters often have relatively high dimensionality (96 dimensions for a day of 15-minute readings), making direct clustering computationally expensive and potentially less effective due to the effects of dimensionality. While feature extraction can reduce dimensions and potentially increase interpretability, it risks losing important information in the original consumption patterns.

The choice of features has a significant impact on the patterns detected. In electricity consumption analysis, features such as the peak-to-average ratio or load factor highlight specific operational characteristics that may interest grid managers. Without careful selection guided by domain knowledge, feature-based approaches may miss essential relationships in the data, such as rapid consumption changes that could indicate demand response participation.

Feature-based approaches may generalize better across different periods or customer types, but require a consistent feature extraction methodology. Ensuring this consistency across diverse consumption patterns and evolving grid conditions creates additional implementation challenges for electricity utilities.

3.5 Computational Efficiency vs. Accuracy

The scale of modern time series data creates a tradeoff between computational feasibility and clustering accuracy.

Exact clustering algorithms, such as traditional DBSCAN, provide precise cluster boundaries and complete noise identification; however, their $O(n^2)$ complexity makes them prohibitively expensive for high-dimensional electricity consumption data. Approximate methods like IP.LSH.DBSCAN [17] offers improved computational efficiency but introduces probabilistic approximations that may occasionally misclassify points near cluster boundaries or fail to identify some density connections. This tradeoff typically favors approximate methods for utilities with hundreds of thousands of customers, as the ability to analyze within operational timeframes outweighs

the need for perfect boundary precision.

Calculating pairwise distances accounts for the majority of overall processing time for time series data, particularly when using specialized time series distance measures such as Dynamic Time Warping. This bottleneck becomes particularly problematic when analyzing high-resolution consumption data (e.g., 15-minute intervals) or examining longer time windows (e.g., weekly patterns rather than daily). Simpler metrics, such as Euclidean or angular distance, sacrifice this temporal flexibility but enable the clustering of substantially larger datasets. The choice between these approaches depends on whether temporal alignment or computational scale is more critical for the specific analytical question at hand.

Although computational efficiency enables more frequent or larger-scale analysis, reduced accuracy can impact operational decisions. For example, inexact clustering may misclassify some customers' peak demand profiles, leading to suboptimal demand response targeting. However, approximate results that enable action often provide more operational value than perfect results delivered too late for implementation. This time-value tradeoff is particularly relevant for utilities managing day-ahead electricity markets or responding to emerging grid conditions.

3.6 Domain Knowledge

Practical analysis of time series data requires integrating domain expertise with advanced analytical techniques:

Domain experts possess crucial knowledge about significant patterns but often lack the technical expertise to implement advanced clustering algorithms. Technical implementers can create sophisticated analysis pipelines but may lack the domain knowledge to identify which patterns are operationally significant. Without domain guidance, technical implementations risk focusing on statistically interesting but practically irrelevant patterns. This expertise gap creates challenges in translating domain understanding into practical analytical approaches.

This expertise gap also challenges parameter selection, feature engineering, and the interpretation of results. Each aspect requires technical understanding of the algorithms and domain understanding of the data's meaning and significance. Clustering results must be translated into actionable insights that domain experts can apply. Transforming technical outputs into practical implications is essential for making clustering results useful in real-world applications.

3.7 Refined Problem Statement

Based on the challenges outlined above, we define the core problem addressed in this thesis as follows:

How can we develop a flexible, computationally efficient approach for clustering high-dimensional time series data that balances analytical accuracy

with practical interpretability, while accommodating domain-specific requirements and the expertise gap between data scientists and domain experts?

This problem statement can be broken down into four distinct sub-problems:

- **Efficient High-Dimensional Clustering**: Implementing clustering approaches that scale effectively with the quantity and dimensionality of time series data.
- **Automated Parameter Optimization**: Developing strategies to identify appropriate clustering parameters without requiring extensive manual tuning.
- **Usage Flexibility**: Allowing selections and combinations of algorithms and features to fit a variety of uses.
- **Interactive Exploration**: Enabling rapid exploration of clustering results to support iterative refinement and insight generation.

In the following chapters, we present our CLUE toolchain as a solution to these problems, demonstrating its application to electricity consumption data while highlighting its flexibility for other time series analysis tasks.

4

CLUE Toolchain Architecture

This chapter presents our **CLUE** (Clustering-based Load Understanding and Exploration) toolchain architecture and the motivation behind the selection of its components.

4.1 Overview and Design Goals

The CLUE toolchain has been designed as a modular and flexible framework for efficiently processing and analyzing high-dimensional time series vector data. Its architecture addresses the fundamental challenges identified in Chapter 3, particularly the computational complexity of high-dimensional clustering, the difficulties in parameter selection, and the need to bridge the gap between algorithmic capabilities and domain expertise.

CLUE's core design philosophy enables users to select and configure components based on their specific analytical requirements. This approach provides rapid exploration capabilities for initial data investigation and precise analytical tools for detailed examination of identified patterns.

Figure 4.1 illustrates the high-level architecture of the CLUE toolchain, showing the modular design with three core components at its center. The architecture employs a layered approach, where external pre-processing feeds data into the core components, which then output results for external analysis and evaluation. This design allows the toolchain to remain flexible while maintaining clear interfaces between components.

The toolchain consists of three primary components that can be independently configured and interchanged:

1. Clustering Algorithms
2. Distance/Similarity Metrics
3. (Optional) Parameter Optimization / Feature Standardization

The toolchain is designed to work with external pre-processing and analysis systems without imposing specific implementation requirements on those components, maximizing flexibility and adaptability to different operating environments.

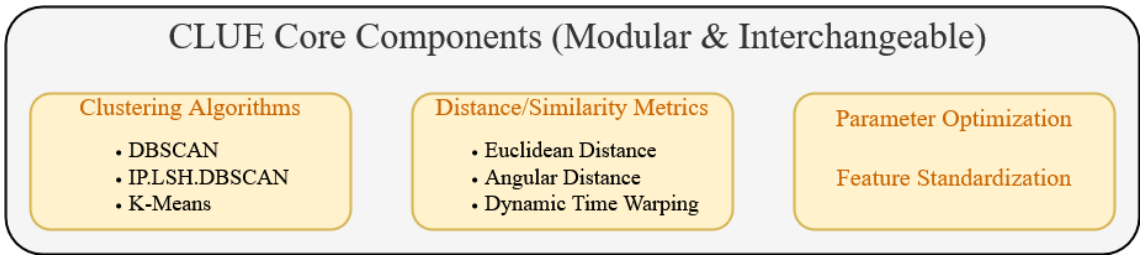


Figure 4.1: CLUE Architecture

Figure 4.2 illustrates the data flow through the CLUE toolchain, demonstrating how time series data progresses through various stages of transformation and analysis. The raw data is processed into time series vectors from which features are extracted. The time series vectors and the features form the input data to CLUE. CLUE then takes as input a selection of features to be clustered, alongside the CLUE configuration that details the clustering settings. Once the clustering has been performed, the intermediate output is fed back as input data into the next clustering round, possibly using a different configuration. This forms a cluster loop where the results are increasingly refined until the final loop, at which point the results are presented. The complete results comprise the intermediate results of each loop and the final clustering results, which are then analyzed using graphs, plots, and statistics.

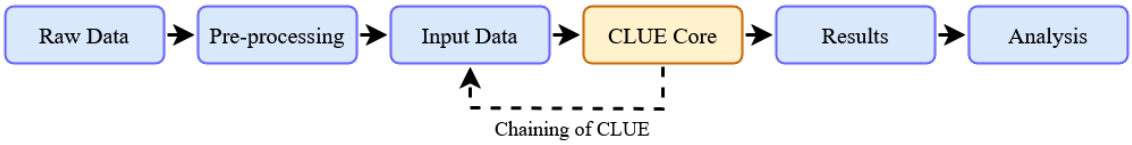


Figure 4.2: Data Flow through CLUE Toolchain

The cluster loop offers customizable data filtering based on a conceptual understanding of the data, rather than imposing direct limits on the data through regular filtering. It also allows for easy branching of cluster strategies depending on the output of these filterings, such as using different configurations for clusters found and outliers found in the same cluster step. This improves the understanding of the data from the data analysis perspective; rather than outputting direct data about a single round of clusterings, each round can select a subset of clusters to examine in greater detail if necessary.

4.2 Core Components

This section gives an overview of the core components of the CLUE toolchain.

4.2.1 Clustering Algorithms

The central idea of CLUE is the ability to cluster together a selection of data in various ways. The clustering algorithm component provides multiple implementa-

tions with a standard interface, allowing them to be interchanged based on specific requirements for accuracy, performance, or cluster characteristics.

CLUE primarily employs density-based methods for several reasons:

- **Arbitrary Cluster Shapes:** Unlike K-means, which assumes spherical clusters, density-based approaches can identify clusters of arbitrary shapes. This allows CLUE to explore the time series pattern without any predetermined notions of the data.
- **Automatic Noise Detection:** Data frequently contains outliers. Density-based methods inherently identify and isolate noise points, eliminating the need for separate outlier detection steps required by other clustering methods.
- **No Pre-specified Cluster Count:** Unlike k-means, which requires knowing k in advance, density-based methods discover the natural number of clusters, critical when exploring unknown patterns.

DBSCAN

The traditional DBSCAN algorithm serves as CLUE's baseline clustering implementation, providing exact density-based clustering results that establish the baseline for comparison. The CLUE implementation of DBSCAN is the reference standard against which approximate methods are evaluated. Furthermore, DBSCAN is a useful tool when the dimensionality and data size are low and/or there is a desire for consistent cluster results.

IP.LSH.DBSCAN

Integrated Parallel Locality-Sensitive Hashing DBSCAN (IP.LSH.DBSCAN), first developed by Keramatian et al. [17], represents a key inclusion in the toolchain. It provides significant performance improvements over traditional DBSCAN while maintaining sufficient clustering quality [17]. The necessity for an accelerated DBSCAN variant becomes apparent when considering practical scenarios in time series analysis. Standard DBSCAN, despite its theoretical advantages, faces scalability limitations. IP.LSH.DBSCAN distinguishes itself from other LSH-based DBSCAN variants.

Leveraging the rapid computational speed, IP.LSH.DBSCAN is the primary algorithm for rapid data exploration and parameter optimization. Its exceptional speed enables interactive analysis that would be impossible with exact methods in the same time frame.

K-Means

The K-means algorithm provides an alternative clustering approach when centroid-based methods are more appropriate for the specific analysis task. While less flexible than density-based methods regarding cluster shapes, K-means offers computational advantages and produces easily interpretable cluster centers that can serve as representative prototypes.

The rationale for incorporating a centroid-based partitioning algorithm alongside density-based approaches stems from practical analysis scenarios. For example, when density-based methods identify a single large cluster, which is frequent in high-dimensional data, k-means enables forced partitioning that can still generate valuable insight and reveal internal structures. This capability is particularly valuable during exploratory analysis, where domain experts seek to understand the data within larger, homogeneous groups or clusters.

The primary parameter is K, the number of clusters, which can be determined through domain knowledge or automated methods such as elbow or k-distance analysis. It is essential to note that this feature enables domain experts to partition the data according to their specific requirements.

4.2.2 Distance/Similarity Metrics

The distance metrics component offers various measures of similarity or dissimilarity between data points, addressing the challenges of calculating meaningful distances in high-dimensional spaces.

Euclidean Distance

The Euclidean distance serves as the standard metric for spatial distance. The Euclidean distance between two n-dimensional points $p = \{p_1, p_2, ..., p_n\}$ and $q = \{q_1, q_2, ..., q_n\}$, illustrated in Figure 4.3, is defined as follows:

$$d_{euclidean}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

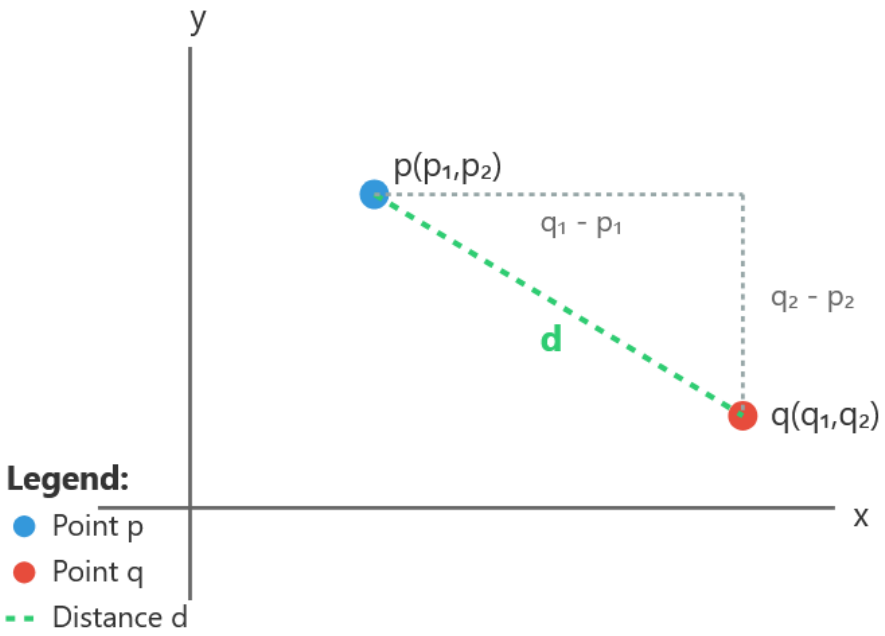


Figure 4.3: Illustration of Euclidean distance

The selection of Euclidean distance as a core metric reflects both its theoretical properties and practical advantages. The extensive optimization of clustering algorithms for Euclidean space ensures computational efficiency and well-understood behavior.

Including Euclidean distance acknowledges its limitations, particularly the curse of dimensionality. As vector dimensions increase, the ratio between the nearest and farthest neighbor distances converges, reducing the metric's discriminative power. This necessitates the inclusion of alternative metrics for high-dimensional scenarios, justifying CLUE's multi-metric approach.

Angular Distance

Angular distance offers a shape-focused alternative to magnitude-based metrics, which is helpful for high-dimensional time series analysis, where pattern similarity is more important than absolute values.

The motivation for incorporating angular distance emerges from the specific challenges of time series clustering. When analyzing consumption patterns, the shape of the temporal profile often carries more significance than absolute consumption levels. Furthermore, angular distance demonstrates greater robustness to the curse of dimensionality, maintaining discriminative power even as vector dimensions increase.

Calculated as the arc cosine of the normalized dot product between vectors (cosine similarity), this metric is normalized to the range $[0, 1]$ by dividing by π.

$$d_{angular} = \frac{cos^{-1}(\frac{V_p \cdot V_q}{||V_p|| \times ||V_q||})}{\pi}$$

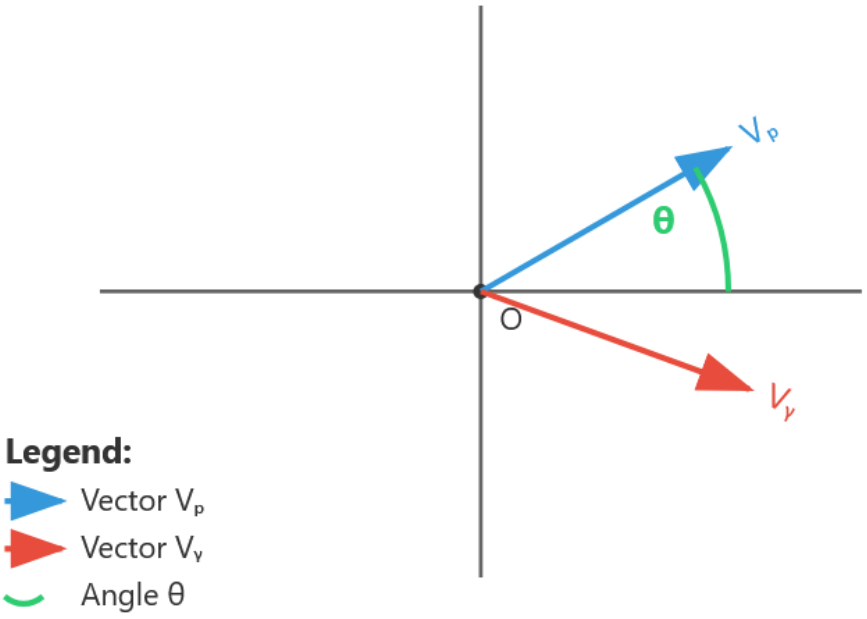


Figure 4.4: Illustration of Angular Distance

Where $V_p \cdot V_q$ is the inner product of vectors V_p and V_q, and $||x|| = d^{euclidean}(x, O)$ where O is the origin point.

Angular distance is particularly valuable for high-dimensional time series data, where the shape of the pattern is more important than its absolute values.

Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) offers a specialized distance metric for time series data that accounts for temporal shifts and variations in speed. The DTW algorithm finds the optimal alignment between two time series by warping the time axis, as shown in Figure 4.5, allowing for the recognition of similar patterns even when they are not perfectly aligned.

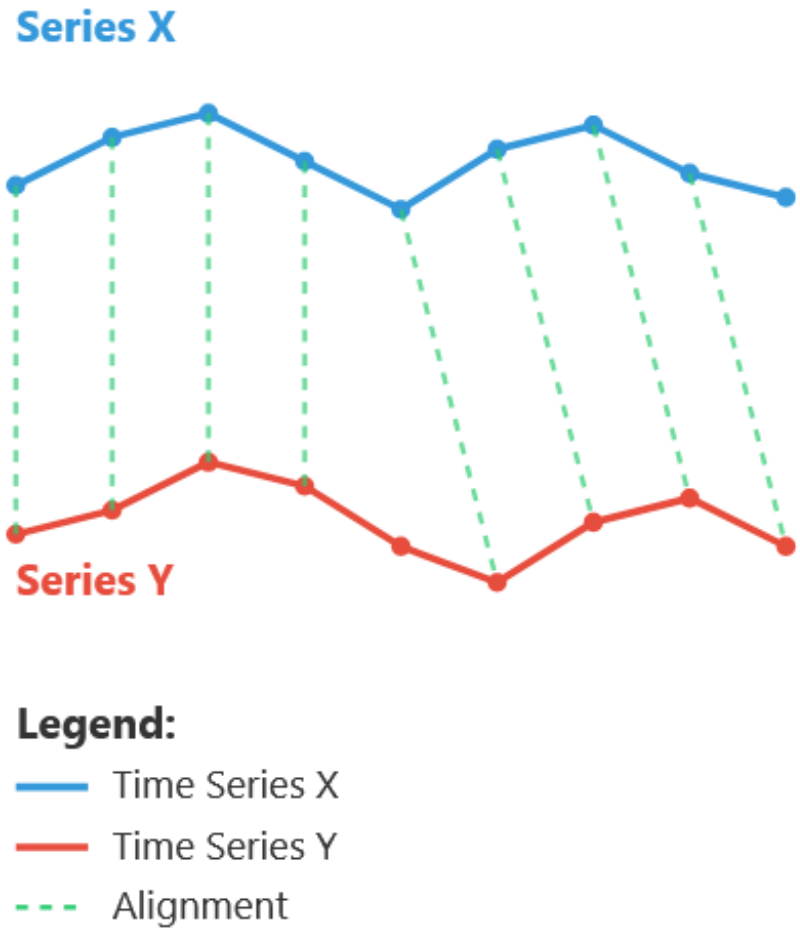


Figure 4.5: Illustration of DTW

DTW addresses the shortcomings of Euclidean and angular distances, specifically their inability to recognize similar patterns occurring at different times or speeds. In time series data, identical wavelike patterns may occur with temporal shifts, and a maximum or peak during a specific time of day may be shifted by a few hours.

DTW's warping capability handles the differences and groups similar-shaped time series together, and these variable differences occur frequently in real-world data.

The computational complexity of DTW for the default naive implementation, $O(d^2)$ where d is the dimensionality of each data point, presents a computational challenge compared to the other distance metrics in CLUE, which operate in $O(d)$ time complexity. To combat the much worse time complexity, CLUE implements three optimization strategies:

1. Sakoe-Chiba band constraints [37] limit the warping path to a fixed window around the diagonal, reducing computation time and preventing pathological alignments, described in more detail in Pseudocode 4.2. In practice, this sets a boundary for how far the alignment is allowed to warp, for example, if we have a bandwidth of 5 hours, for 07:00 in time series A, the only legal alignments are within 02:00 and 12:00 in time series B. Speeding up computation time, and allowing for more control in pattern alignments.
2. LB_Keogh lower bound [38] provides efficient pruning for early abandoning of distant comparisons. If the distances exceed the clustering threshold (ε), the computations are abandoned, which reduces unnecessary calculations and improves computational efficiency of the entire clustering algorithm.
3. Distance caching prevents redundant calculations when comparing the same time series repeatedly. When computing DTW, the algorithm must calculate distances between individual points from both series multiple times. By storing these point-to-point distance calculations in a lookup table, subsequent DTW operations involving the same series can retrieve these values instantly, rather than having to recompute them.

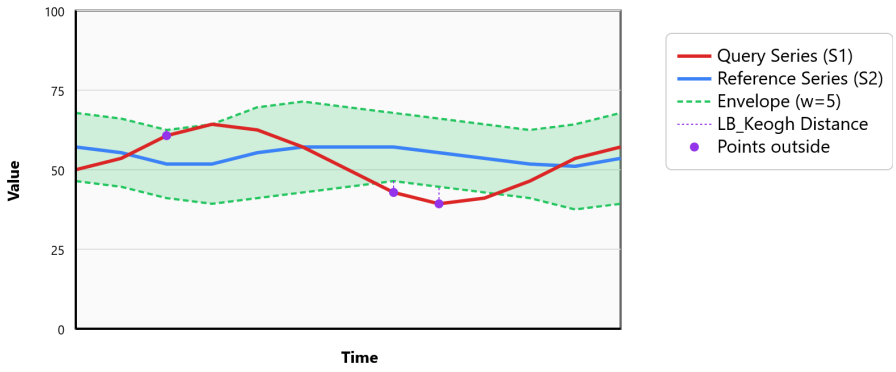


Figure 4.6: Illustration of LB_Keogh lower bound distance,

Despite its higher computational cost compared to simpler metrics, these optimizations make DTW more comparable to the previously mentioned alternatives and

practical for real-world applications that require temporal pattern matching.

```
DTW(X, Y):
1. Initialize matrix DTW[n+1][m+1] with infinity
2. Set DTW[0][0] = 0
3. For i = 1 to n:
   For j = 1 to m:
      cost = (x_i - y_j)^2
      DTW[i][j] = cost + min(DTW[i-1][j], DTW[i][j-1], DTW[i
         -1][j-1])
4. Return sqrt(DTW[n][m])
```

Pseudocode 4.1: DTW

```
DTW_Constrained(X, Y, window_size w):
1. Initialize matrix DTW[n+1][m+1] with infinity
2. Set DTW[0][0] = 0
3. For i = 1 to n:
   For j = max(1, i-w) to min(m, i+w):
      cost = (x_i - y_j)^2
      DTW[i][j] = cost + min(DTW[i-1][j], DTW[i][j-1], DTW[i
         -1][j-1])
4. Return sqrt(DTW[n][m])
```

Pseudocode 4.2: DTW with Sakoe-Chiba band constraint

4.2.3 Parameter Optimization

The parameter optimization component provides automated methods for identifying appropriate clustering parameters, addressing the parameterization challenges described in Section 3.3. The technique described in this section represents one of our main contributions to CLUE, explicitly developed to address the challenge of parameter selection in density-based clustering. This method relies on established cluster quality metrics to provide parameters best suited for the data. This process has historically been challenging and requires a deeper understanding of the data, making exploratory data analysis a daunting task, especially for domain experts with limited knowledge of clustering.

Crucially, this parameter analysis is only feasible because of IP.LSH.DBSCAN's exceptional speed. Traditional DBSCAN implementations would require minutes or hours for each parameter evaluation, making it impractical to explore multiple combinations of epsilon and MinPts. With IP.LSH.DBSCAN, our implementation, analyzes hundreds of parameter combinations in a reasonable time, enabling optimization that would be impossible with exact methods such as DBSCAN.

K-Distance Plot Analysis

The K-distance plot analysis module implementation provides a methodical approach to determining appropriate epsilon (ε) values, addressing perhaps the most challenging aspect of DBSCAN parameter selection.

K-distance plot analysis offers a systematic approach to determining suitable epsilon values for density-based clustering algorithms. The method calculates the distance to the k-th nearest neighbor for each point in the dataset, sorts these distances, and plots them to identify "elbow points" where the curve shows significant change, as illustrated in Figure 4.7.

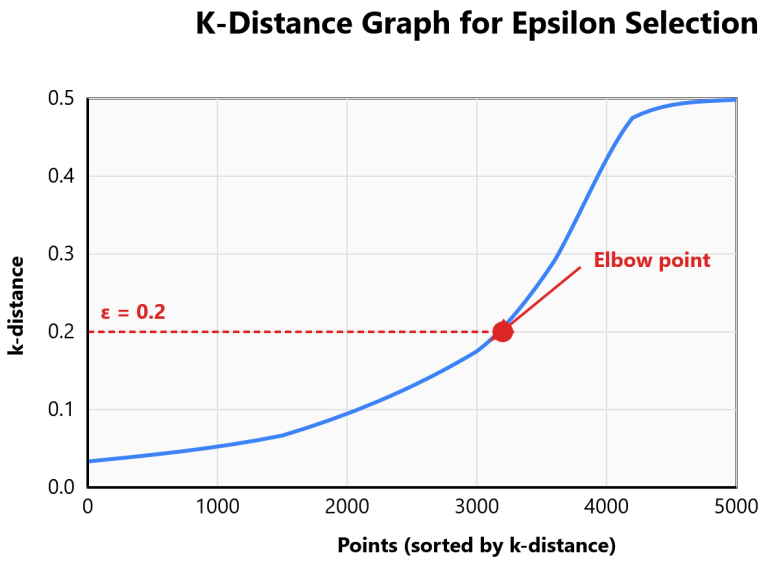


Figure 4.7: K-Distance plot showing the identified "elbow point"

Our implementation, as illustrated in Pseudocode 4.3, extends traditional k-distance analysis with specific methods for high-dimensional data scenarios. Rather than using only $k = 4$, as the authors of the original DBSCAN paper suggest, we adaptively calculate distances for multiple values of k, scaled to the data dimensionality d ($k = 4$, $k = 2d$, $k = 4d$), providing more robust parameter estimation across varying dimensionalities. A stratified sampling technique for large datasets enables computation on datasets that would otherwise overwhelm the implementation.

Rather than requiring manual inspection of k-distance plots, the implementation employs curvature analysis to identify transition points in the data distribution, in other words, finding the "elbow" mathematically. By calculating the curvature at each point in the sorted k-distance plot using the formula:

$$\text{Curvature}(i) = \frac{|y_i''|}{(1 + (y_i')^2)^{3/2}}$$

Where y_i' and y_i'' represent the first and second derivatives estimated using central difference approximations, this identifies points where the k-distance curve changes

direction most sharply - the characteristic "elbow" that indicates a natural separation between points within clusters and points between clusters (noise).

This automation enables non-expert users to obtain recommended epsilon values without requiring a deep understanding of DBSCAN's mathematical foundations. Furthermore, the algorithm produces top-3 recommendations and a range of candidate values reflecting different k-values, providing multiple starting points for exploration.

```
K_Distance_Analysis(Dataset D, k_values [k_1, k_2, ...]):
1. distances = {}

2. For each k in k_values:
   k_distances = []
   For each point p in D (or sampled subset for large D):
      Calculate distances from p to all other points
      Sort distances in ascending order
      Add the k-th distance to k_distances
   Sort k_distances in ascending order
   distances[k] = k_distances
   Find the elbow point in k_distances using curvature
      measurement
   Add the detected epsilon value to recommended_epsilons

3. Return recommended_epsilons
```

Pseudocode 4.3: K-Distance Analysis

Grid Search Framework

The grid search framework systematically evaluates combinations of clustering parameters to identify optimal configurations. Starting with candidate epsilon values derived from k-distance analysis, the framework explores combinations with different MinPts values (for density-based methods) and hash table configurations (L and M for IP.LSH.DBSCAN).

The implementation generates parameter combinations using both the k-distance results and multiplicative factors to ensure broader exploration around promising starting values. For MinPts, values are typically derived from the data dimensionality, following established heuristics [36].

For each parameter combination, the framework executes clustering through IP.LSH.-DBSCAN and calculates quality metrics, which will be explained in greater detail in the following section, to evaluate the results. The process identifies optimal parameter sets and potentially problematic configurations through "red flag" heuristics that detect degenerate results, highlighted in Pseudocode 4.4 below.

```
Grid_Search(Dataset D, epsilon_values, minPts_values,
   hash_table_values, hyperplanes_values):
1. best_score = -infinity
2. best_params = null
```

```
4
5  3. For each epsilon in epsilon_values:
6     For each minPts in minPts_values:
7        For each L in hash_table_values:
8           For each M in hyperplanes_values:
9              Run clustering with parameters (epsilon, minPts,
                  L, M)
10             Evaluate clustering quality using:
11                - Davies-Bouldin Index
12                - Silhouette Coefficient
13                - Calinski-Harabasz Index
14                - Dunn Index
15                - DBCV
16
17             Check for degenerate cases:
18                - Too much noise (>40%)
19                - Too little noise (<1%)
20                - Single dominant cluster (>90%)
21
22             Calculate weighted score
23             If score > best_score:
24               best_score = score
25               best_params = (epsilon, minPts, L, M)
26
27 4. Return the best parameters
```

Pseudocode 4.4: Grid Search for Optimal Parameters

Internal Cluster Quality Metrics

For each parameter configuration, the framework executes clustering through IP.LSH.-DBSCAN and calculates multiple internal quality metrics:

1. **Davies-Bouldin Index** [39] measures the ratio of within-cluster scatter to between-cluster separation. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{S_i + S_j}{d_{ij}}\right)$$

   where S_i is the average distance between each point in cluster i and its centroid, and d_{ij} is the distance between the centroids of clusters i and j. Lower values indicate better clustering, characterized by compact and well-separated clusters. The DB index is particularly effective for comparing different clustering solutions on the same dataset.

2. **Silhouette Coefficient** [40] evaluates how similar points are to their cluster compared to other clusters. For each point i, we calculate:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$a(i)$ is the average distance between i and all other points in the same cluster, and $b(i)$ is the minimum average distance from i to any other cluster. The Silhouette Coefficient is the mean of $s(i)$ across all points, with values ranging from -1 to 1. Higher values indicate better-defined clusters where points are well-matched to their clusters and poorly matched to neighboring clusters.

3. **Calinski-Harabasz Index [41]** evaluates cluster validity based on the ratio of between-cluster variance to within-cluster variance, also known as the Variance Ratio Criterion:

$$CH = \frac{SS_B}{SS_W} \times \frac{N-k}{k-1}$$

SS_B is the between-cluster variance, SS_W is the within-cluster variance, N is the number of data points, and k is the number of clusters. Higher values indicate better clustering solutions, characterized by dense and well-separated clusters. This index works best for convex clusters and tends to prefer solutions with larger clusters.

4. **Dunn Index [42]** calculates the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance:

$$DI = \frac{\min_{i \neq j} d(C_i, C_j)}{\max_k diam(C_k)}$$

Where $d(C_i, C_j)$ is the distance between clusters i and j, and $diam(C_k)$ is the diameter of cluster k (maximum distance between any two points in the cluster). Higher values indicate better clustering, with compact and well-separated clusters. The Dunn Index is particularly sensitive to outliers, as it utilizes minimum and maximum distances.

5. **Density-Based Clustering Validation (DBCV) [43]** provides a specialized metric for assessing density-based clustering quality. Unlike traditional metrics that assume spherical or convex clusters, DBCV evaluates clusters based on their density connectedness:

$$DBCV = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{DSC(C_i) - DSC_{min}}{DSC_{max} - DSC_{min}}$$

where $DSC(C_i)$ is the density sparseness of cluster i, and DSC_{min} and DSC_{max} are the minimum and maximum sparseness values respectively. DBCV ranges from -1 to 1, with higher values indicating better clustering. This metric is particularly valuable for evaluating DBSCAN (and density-based variants) results as it can handle arbitrarily shaped clusters and directly incorporates the concept of density connectivity into its evaluation.

Each of these metrics is visualized in Figure A.1 in Appendix 1.

The findings of Arbelaitz et al. [44] suggest that no single internal clustering quality metric can capture all aspects of "good" clustering. Rather than forcing a single optimization criterion, our toolchain enables weighted multi-metric evaluation that balances different quality aspects according to domain-specific priorities.

CLUE combines these metrics through configurable weights, allowing users to prioritize specific aspects of clustering quality according to their domain requirements. Additionally, the metrics are complemented by "red flag" heuristics proposed by Schubert et al. [36], that identify potentially problematic clustering results, such as excessive noise ($> 40\%$ of points), single dominant clusters ($> 80\%$ of non-noise points), or no substantial clusters found.

4.2.4 Feature Standardization

CLUE's feature standardization component [45] enables the effective clustering of externally engineered features, ensuring equal influence across features while maintaining high flexibility in data representation.

A design decision in CLUE's architecture was to separate feature engineering (external to the toolchain) and clustering operations. While CLUE does not perform feature extraction, it provides robust standardization capabilities that ensure externally engineered features can be effectively used in clustering operations regardless of their original scales or units.

When clustering features with different scales, units, or value ranges, standardization ensures that each feature contributes equally to the distance calculations. Without standardization, features with larger numeric ranges would dominate the clustering process, potentially obscuring essential patterns in smaller-scaled features.

CLUE provides Z-score normalization, transforming features to have zero mean and unit variance, and is suitable for most feature sets [45]:

$$z = \frac{x - \mu}{\sigma}$$

Where z is the standardized value (z-score), x is the original feature value, μ is the mean for that feature of all values across the dataset, and σ is the standard deviation of all values for that feature. The resulting z-scores represent the number of standard deviations each value is from the mean. In the context of CLUE, this ensures that features measured in different units (e.g., kilowatts vs. percentages) or with different ranges (e.g., 0-100 vs. 0-10000) contribute equally to the clustering distance calculations.

When enabled, standardization is applied automatically before clustering operations, ensuring consistent treatment of all features regardless of their original scales.

5

Evaluation

This chapter evaluates the CLUE toolchain across multiple dimensions: computational performance, clustering quality, parameter optimization effectiveness, and practical applicability to real-world electricity consumption data. The evaluation methodology combines quantitative performance metrics, internal clustering validation measures, and application-specific assessments, utilizing data from Göteborg Energi, as well as synthetic data when necessary.

5.1 Evaluation Methodology

This section presents an evaluation methodology for assessing the performance, accuracy, and practical utility of the CLUE toolchain's clustering techniques for high-dimensional time series data. As noted by Aghabozorgi et al. [8], evaluation of time series clustering presents unique challenges due to the lack of standardized benchmarks and the domain-specific nature of "good" clustering results. The methodology addresses these challenges through a multi-faceted approach that combines quantitative performance metrics, quality assessment techniques, and application-specific validation methods.

5.1.1 Technical Performance Metrics

The technical evaluation focuses on three key aspects of clustering system performance: computational efficiency, accuracy, and scalability. These metrics provide a foundation for assessing the practical viability of our approach in real-world applications.

Performance and Latency

Following the methodology of Keramatian et al. [17], execution time is measured from the initiation of the clustering algorithm to the point at which the final clustering results are produced. This metric is used to assess the feasibility of the algorithms for interactive exploration, where rapid feedback is essential for iterative analysis. Evaluations are conducted as follows:

- **Component-level microbenchmarks:** The performance of individual clustering components and distance calculations is isolated for comparative evaluation.

- **End-to-end CLUE macrobenchmarks:** Total execution time, including data preparation, clustering, and output writing, is recorded to evaluate overall workflow latency.
- **Execution Time Ratio:** Execution Time Ratio between baseline (DBSCAN) and optimized implementations (IP.LSH.DBSCAN) across various parallelization settings are computed.

The focus on performance has been motivated by the need for interactive exploration in time series analysis and the utilization of advanced functionality such as parameter search. When clustering execution times extend into minutes or hours, exploratory cycles are disrupted, and analytical iteration becomes impractical.

Accuracy Assessment

Clustering accuracy is evaluated through a combination of internal and external validation techniques:

- **External validation:** When ground truth labels are available, the Adjusted Rand Index [46] measures clustering accuracy. This is done by comparing the resulting clustering labels with the true labels provided by the ground truth. How ARI is calculated will be explained in a later section.
- **Comparative accuracy:** Clustering outcomes from IP.LSH.DBSCAN is compared against DBSCAN, highlighting the tradeoff between execution time and accuracy.
- **Parameter sensitivity:** The sensitivity of clustering outcomes to variations in parameters is assessed, serving to validate the robustness of the parameter finding strategy.

Accuracy measurements for IP.LSH.DBSCAN uses parameters that the proposed optimization framework selects, reflecting real-world usage behavior. Following Kriegel et al. [16], the influence of parameters on density-based methods is examined to assess cluster quality.

Scalability Analysis

Scalability is evaluated by measuring algorithm behavior under varying dataset sizes, dimensionalities, and computational resources:

- **Dataset size scaling:** Performance changes are measured as data points increase.
- **Dimensionality scaling:** Vector dimensionality varies from 24 to 96 to observe performance shifts.
- **Resource scaling:** Sequential and parallel performance is compared across a range of thread counts.

This analysis is crucial in time series applications, where increasing monitoring resolution yields data with potentially hundreds of dimensions. Without scalable algo-

rithms, analyzing these higher-resolution datasets becomes computationally infeasible.

5.1.2 Application Specific Validation of CLUE

The practical value of the clustering approach is evaluated through application-focused methods:

- **Pattern Discovery Assessment:** The extent to which clustering results uncover meaningful patterns in real-world electricity consumption data.

This validation ensures that algorithmic improvements yield meaningful benefits for domain experts, data analysts, and grid operators. Following Wang et al. [35], it is acknowledged that the actual impact of clustering is determined by the actionable insights it enables in domain-specific contexts.

5.2 Experimental Setup

The experiments were conducted on a virtualized Red Hat Enterprise Linux 8.10 server. The system utilized an Intel Xeon Gold 6226R processor operating at 2.90 GHz, with four virtual CPUs configured in a single-core-per-socket topology. The processor featured a three-level cache hierarchy consisting of 32 KB L1, 1 MB L2, and 22 MB L3 caches. The system operated within a complete virtualization environment provided by VMware, which should be taken into account when interpreting performance measurements related to computational throughput and storage access patterns.

CLUE was evaluated using four datasets with varying characteristics:

Dataset 1 consists of hourly consumption data from Göteborg Energi, comprising approximately 7500 small-to-medium business customers. This dataset was selected as these customers are expected to have the most impact on the electricity grid and are considered the most important customers for identifying patterns and behaviors. Each data point contains 24 measurements, representing hourly daily readings, which are typical operational data with moderate dimensionality.

Dataset 2 shares the same customer base as Dataset 1 but increases the resolution to 15-minute intervals, resulting in 96-dimensional vectors per data point. This dataset enables testing of performance on higher-dimensional data while maintaining the same domain context.

Dataset 3 was synthetically generated with 50,000 data points, each containing 96-dimensional vectors that represent daily consumption patterns. The dataset was synthetically generated with seven distinct consumption patterns (clusters) and the true labels for each data point.

Dataset 4 is similar to Dataset 3; it is a synthetically generated dataset with 10,000 data points generated according to the same rules as Dataset 3 but with eight distinct

clusters instead of seven. The additional cluster is far more spread out and more challenging to pinpoint.

5.3 Core Algorithm Performance Evaluation

This section evaluates the performance of the core algorithms, both in comparison to one another and in relation to the CLUE toolchain as a whole.

5.3.1 Computational Efficiency Comparison

This evaluation assesses the computational performance of IP.LSH.DBSCAN against traditional DBSCAN across datasets of varying sizes and dimensionalities. The primary objective is to quantify the speedup achieved by IP.LSH.DBSCAN.

Each algorithm was executed on all three datasets using identical parameters (ε = 0.1, minPts = 25, L = 50, M = 10). Execution time was measured from the algorithm's initialization to the completion of cluster assignment. Single-threaded (sequential) and multi-threaded (4-core) configurations were tested to assess the effectiveness of parallelization.

Execution Time (s)			
Algorithm (Configuration)	**Dataset 1 (s)**	**Dataset 2 (s)**	**Dataset 3 (s)**
DBSCAN	25.40	72.21	3353 (56 min)
IP.LSH.DBSCAN (1 thread)	0.62	1.43	10.03
IP.LSH.DBSCAN (4 threads)	0.41	0.62	3.53

Table 5.1: Execution times (in seconds) for DBSCAN and IP.LSH.DBSCAN with parameters ε = 0.1, minPts=25, L=50, M=10, and different parallelization configurations across the first three datasets. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Execution Time Ratio compared to DBSCAN			
Algorithm (Configuration)	**Dataset 1**	**Dataset 2**	**Dataset 3**
DBSCAN	1.0×	1.0×	1.0×
IP.LSH.DBSCAN (1 thread)	41.2×	50.5×	334.3×
IP.LSH.DBSCAN (4 threads)	61.4×	115.6×	948.9×

Table 5.2: Execution Time Ratio for IP.LSH.DBSCAN compared to DBSCAN, with parameters ε = 0.1, minPts=25, L=50, M=10, and different parallelization configurations across the first three datasets. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Table 5.1 presents the raw execution times for both algorithms across the first three datasets. The results demonstrate substantial performance improvements with IP.LSH.DBSCAN, particularly as dataset complexity increases. For Dataset 1 (24-dimensional, 7,500 points), IP.LSH.DBSCAN reduces execution time from 25.40

seconds to just 0.41 seconds with four threads. The improvement becomes even more pronounced with larger datasets. Dataset 3 (96-dimensional, 50,000 points) drops execution time from nearly 56 minutes to under 4 seconds.

The Execution Time Ratio shown in Table 5.2 reveal that IP.LSH.DBSCAN delivers improvements ranging from 41.2× to 948.9× depending on dataset characteristics and parallelization. Notably, the speedup increases with both dataset size and dimensionality. Dataset 3 achieves the highest speedup (948.9× with four threads), demonstrating that IP.LSH.DBSCAN becomes more effective in complex, high-dimensional scenarios, which are particularly challenging for traditional DBSCAN.

The parallel efficiency is also noteworthy. Moving from 1 to 4 threads provides additional speedup across all datasets, with the most significant absolute gains on Dataset 3 (334.3× to 948.9×).

5.3.2 LSH Parameter Impact

Investigating how different LSH configuration parameters (L hash tables and M hyperplanes per table) affect computational performance is crucial to understanding the tradeoff between accuracy and speed as LSH parameters vary.

IP.LSH.DBSCAN was executed with four different LSH configurations, while keeping the clustering parameters constant ($\varepsilon = 0.1$, minPts = 25, and four threads). The configurations tested represent distinct points along the accuracy-speed tradeoff spectrum, from aggressive speedup (L=15, M=5) to more conservative approximation (L=100, M=50).

Execution Time Ratio with different LSH parameters

Algorithm (Configuration)	Dataset 1	Dataset 2	Dataset 3
DBSCAN	1.0×	1.0×	1.0×
L=15, M=5	120.5×	266.2×	2012.3×
L=50, M=10	61.4×	115.6×	948.9×
L=100, M=10	41.1×	67.8×	559.1×
L=100, M=50	17.4×	27.5×	133.5×

Table 5.3: Execution Time Ratio for IP.LSH.DBSCAN compared to DBSCAN, with parameters $\varepsilon = 0.1$, minPts=25, varying L and M, and four threads parallelization configurations across the first three datasets. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Table 5.3 demonstrates the relationship between LSH parameter selection and computational performance. The most aggressive configuration (L=15, M=5) achieves the highest speedups, 120.5× for Dataset 1, 266.2× for Dataset 2, and 2012.3× for Dataset 3. The most conservative configuration (L=100, M=50) still provides substantial speedups (17.4× to 133.5×) while likely offering higher accuracy through increased hash table coverage.

The results reveal a consistent pattern across all datasets, showing that significantly reducing the number of hash tables and hyperplanes improves computational performance. However, this creates a practical tradeoff where users must balance execution speed against clustering accuracy. The moderate configuration (L=50, M=10) used in our main evaluation represents a balanced approach, providing substantial speedups (61.4× to 948.9×) while maintaining reasonable accuracy guarantees.

5.4 CLUE Macrobenchmarks - Performance

This evaluation assesses the performance of a CLUE clustering pipeline that integrates two clustering algorithms in sequence. The two stages of the tested pipeline are as follows: (1) DBSCAN or IP.LSH.DBSCAN for outlier detection and removal, and (2) K-means clustering applied to the remaining non-outlier points. Total execution time includes data loading, clustering stages, and result output. Parameters were set to ε=0.2, minPts=25, L=50, and M=10 for the density-based stage.

Macrobenchmark - Execution Time (s)

Algorithm (Configuration)	Dataset 1 (s)	Dataset 2 (s)	Dataset 3 (s)
DBSCAN+K-Means	26.59	75.91	3367 (56 min)
IP.LSH.DBSCAN+K-Means (1T)	2.73	4.92	26.20
IP.LSH.DBSCAN+K-Means (4T)	2.37	4.22	18.10

Table 5.4: End-to-end execution times for complete CLUE pipelines combining density-based clustering with K-means. Times include data loading, both clustering stages, and result output. Parameters T=threads, $\varepsilon = 0.2$, minPts=25, L=50, M=10. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Macrobenchmark - Execution Time Ratio compared to DBSCAN

Algorithm (Configuration)	Dataset 1	Dataset 2	Dataset 3
DBSCAN+K-Means	1.0×	1.0×	1.0×
IP.LSH.DBSCAN+K-Means (1 thread)	9.74×	15.42×	128.56×
IP.LSH.DBSCAN+K-Means (4 threads)	11.22×	18.01×	186.04×

Table 5.5: Execution Time Ratio for complete CLUE pipelines using IP.LSH.DBSCAN versus DBSCAN in the first stage. Values show total pipeline acceleration, including K-means overhead. Parameters $\varepsilon = 0.2$, minPts=25, L=50, M=10. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Table 5.4 shows the end-to-end execution times for complete CLUE pipelines. Even when including the overhead of K-means clustering and data I/O operations, IP.LSH.-DBSCAN maintains substantial performance advantages. The pipeline with traditional DBSCAN requires 26.59 seconds for Dataset 1 and over 56 minutes for Dataset

3, while the IP.LSH.DBSCAN variant completes the same operations in 2.37 and 18.10 seconds, respectively.

The macrobenchmark speedups presented in Table 5.5 are naturally lower than the microbenchmark results due to the additional computational overhead of K-means and I/O operations. However, the improvements remain significant, ranging from 9.74 times to 186.04 times across different datasets and configurations. These results demonstrate the benefits of IP.LSH.DBSCAN are effectively translated to real-world analysis pipelines that combine multiple algorithms.

Importantly, the macrobenchmark results confirm that IP.LSH.DBSCAN enables interactive analysis workflows that would be impractical with traditional DBSCAN. Dataset 3's reduction from 56 minutes to 18 seconds transforms a batch processing scenario into an interactive exploration opportunity.

Comparing the microbenchmark results in Table 5.2 with the macrobenchmark results in Table 5.5, we observe that while absolute speedups decrease when additional algorithmic overhead is included, the relative performance benefits remain substantial and practically significant.

5.4.1 CLUE performance with LSH parameter variation

By extending the pipeline evaluation to examine how LSH parameter choices affect performance when multiple algorithms are involved. The same two-stage pipeline (density-based + K-means) was executed with varying LSH configurations. All other parameters remained constant ($\varepsilon = 0.2$, minPts = 25, four threads) to isolate the impact of LSH parameter selection on overall pipeline performance.

Macrobenchmark - Execution Time Ratio with different LSH parameters

Algorithm (Configuration)	Dataset 1	Dataset 2	Dataset 3
DBSCAN	1.0×	1.0×	1.0×
L=15, M=5	11.92×	22.22×	200.9×
L=50, M=10	11.22×	18.01×	186.0×
L=100, M=10	9.57×	16.01×	135.67×
L=100, M=50	7.32×	12.64×	81.43×

Table 5.6: Impact of LSH parameters on complete pipeline performance when combining IP.LSH.DBSCAN with K-means. Higher L and M values trade speed for accuracy. Dataset 1 (7,500 points, 24 dimensions), Dataset 2 (7,500 points, 96 dimensions), Dataset 3 (50,000 points, 96 dimensions).

Table 5.6 shows that while the k-means stage and the overhead cut the speedup achieved in half with L=100 and M=50 when compared with Table 5.3, it still achieves a significant enough speedup compared to the base DBSCAN. This further emphasizes CLUE's ability to quickly generate output information for analysis, even for large datasets.

5.5 Accuracy and Parameter Optimization Effectiveness

This section evaluates the accuracy and quality of DBSCAN and IP.LSH.DBSCAN cluster results when using the parameter optimizer against basic k-distance elbow plotting to find epsilon.

5.5.1 DBSCAN/IP.LSH.DBSCAN Accuracy

Here, we evaluate the accuracy and effectiveness of CLUE's automated parameter optimization by comparing traditional DBSCAN parameter selection with systematic parameter exploration using IP.LSH.DBSCAN. Both approaches utilize k-distance analysis (k = 4) to ensure a consistent comparison.

To evaluate the accuracy, we calculate the Adjusted Rand Index (ARI). Given a set of i elements $S = \{e_1, e_2, ..., e_i\}$ and two partitions of S, $P = \{P_1, P_2, ..., P_n\}$ and $Q = \{Q_1, Q_2, ...Q_m\}$, partitioned into n and m subsets of S respectively, the ARI is calculated with the formula:

$$ARI = \frac{RI - RI_{ex}}{RI_{max} - RI_{ex}}$$

Where RI is the number of pairs in which the two elements are found to be in the same subset in the different partitions, RI_{ex} is the expected number of pairs if the labeling were to be performed randomly, and RI_{max} is the maximum number of pairings for the two partitions assuming they are all in the same subset [46]. The ARI spans -0.5 to 1, with a higher value indicating a better match. Any value lower than 0 suggests that the labeling was worse than a random labeling.

The evaluation was conducted using Dataset 4, which contains 10,000 data points, 96 dimensions, and known ground truth labels. As seen in Table 5.7, DBSCAN achieved an ARI of 0.4483 using parameters from the k-distance plot elbow point (ε=1.626, minPts=24). Systematic grid parameter exploration using IP.LSH.DBSCAN evaluated 180 combinations across epsilon values that k-distance analysis suggested ε={0.8131, 1.211, 1.626, 2.019, 2.423}, minPts values {10, 15, 20, 25, 50, 100}, and LSH-specific parameters (L={10, 50, 100}, M={10, 50}). Of these 180 combinations, 66 passed the "red flag" heuristics control, resulting in non-degenerate clustering results.

The parameter exploration revealed substantial variability in accuracy. While the baseline DBSCAN achieved moderate accuracy, systematic grid exploration revealed configurations with seemingly improved clustering quality. The best configuration (L=50, M=10, ε=1.626, MinPts=15) achieved an ARI of 0.932, a 108% improvement over the baseline. Overall, 48.5% of evaluated combinations exceeded baseline performance, with 28.8% achieving ARI scores above 0.7. It should be noted that, due to the synthetic nature of the dataset, the improvements in ARI are most likely a result of the inherent inaccuracies found in clustering results when using IP.LSH.DBSCAN. This will be further explored later in this section.

To further evaluate the effectiveness of parameter optimization, we extended the

Clustering Accuracy - IP.LSH.DBSCAN vs Baseline

Metric	Value	Significance
DBSCAN Baseline ARI	0.448	–
Best IP.LSH.DBSCAN ARI	0.932	2.08× improvement
Configurations Exceeding Baseline	32/66	48.5%
High-Performance Configurations (ARI > 0.7)	19/66	28.8%

Table 5.7: Clustering accuracy comparison showing Adjusted Rand Index (ARI) scores for IP.LSH.DBSCAN parameter configurations. Of 180 tested combinations, 66 produced valid results, with 32 exceeding baseline DBSCAN performance.

evaluation by testing the exact DBSCAN algorithm using the same parameter combinations that showed superior performance with IP.LSH.DBSCAN. This analysis provides valuable insights into the relationship between LSH approximation and clustering quality.

Parameter Configuration Performance Comparison

ε	minPts	DBSCAN ARI	Performance vs Baseline
1.211	{10, 15, 20, 25}	0.773	+72.7% improvement
1.211	50	0.771	+72.1% improvement
1.626	10	0.445	-0.7% (baseline level)
1.626	{15, 20, 25, 50, 100}	0.448	+0.0% (baseline level)
2.019	{50, 100}	0.289	-35.5% (below baseline)

Table 5.8: DBSCAN clustering accuracy (ARI scores) for parameter combinations identified by IP.LSH.DBSCAN optimization. Results show parameter search can identify configurations yielding a 72.7% improvement over the k-distance baseline.

The DBSCAN ARI scores for parameter configurations that demonstrated above-baseline performance in the IP.LSH.DBSCAN evaluation is presented in Table 5.8, where the 32 IP.LSH.DBSCAN configurations that exceeded the baseline performance (ARI = 0.448) are consolidated into 13 unique parameter combinations and tested using DBSCAN.

As previously mentioned, while the parameter optimization demonstrates substantial ARI improvements, in many cases, the IP.LSH.DBSCAN approximation achieves even higher ARI scores than exact DBSCAN with identical (ε, minPts) parameters. This counterintuitive result suggests that if L is set too low, the results may be unreliable and would require further exploration at larger L to confirm actual improvement in cluster quality (with respect to ARI).

Figure 5.1 illustrates the convergence of the ARI with the DBSCAN results as ground truth labels and IP.LSH.DBSCAN results as comparison labels. As the number of hash tables increases, the ARI converges to one at varying speeds, depending on the parameter combinations. Figures 5.2a, 5.2b, and 5.2c show this convergence by comparing the respective results from DBSCAN and IP.LSH.DBSCAN cluster ARI when compared to the true labels of the dataset.

The parameter combinations selected for each of the graphs in Figure 5.2 were chosen as follows:

1. First, the parameter optimizer was run to find which epsilon values result in an improved ARI from the baseline at any MinPts, hash tables, and Hyperplanes.

2. Then MinPts was selected by looking at which Epsilon/MinPts/Hyperplanes combination had the largest ARI *for each epsilon found in the previous step*, regardless of hash tables

3. Finally, each of these combinations (in this case, three epsilons were found, resulting in three combinations or graphs) was tested at a selection of hash table values to evaluate the convergence rate.

For this part of the evaluation, Euclidean distance was used. The reasoning behind this was that since the synthetic dataset was easy to cluster, Euclidean distance proved to be very effective in measuring the ARI differences between hash table values. The set of tested hash table values was [10, 15, 25, 50, 75, 100, 200]

For Figures 5.2b and 5.2c the IP.LSH.DBSCAN ARI exceeds the DBSCAN ARI and decreases until convergence. Notably, in Figure 5.2a, the IP.LSH.DBSCAN ARI increases towards the DBSCAN ARI. This further suggests that at lower hashtable values, the unpredictability of IP.LSH.DBSCAN is more pronounced. Furthermore, each of the IP.LSH.DBSCAN ARI at 10 hash tables exceeds the ARI of a DBSCAN (0.44) configuration set by only using the k-distance method (as opposed to using the parameter optimizer).

This introduces the problem of erroneously selecting a configuration with a high ARI using IP.LSH.DBSCAN, but when increasing the number of hash tables, IP.LSH.DBSCAN converges to a lower ARI, resulting in a poorer overall DBSCAN clustering result.

5.5.2 Cluster Quality

Finally, we evaluated the cluster quality metrics discussed in Chapter 4.2.3 by comparing the highest-ranked parameter combination found by the parameter optimizer with the combination found using k-distance for DBSCAN.

Table 5.9 shows the results from the cluster quality evaluation. This part of the evaluation was conducted on Dataset 2, clustering features extracted from the raw time series data using Euclidean distance. As the parameter optimizer evaluates each non-degenerate combination by aggregating the weighted cluster quality metrics into a single score and selects the best one, it is expected that the parameter optimizer finds a combination with superior cluster quality metrics compared to k-distance. This was indeed the case with mean improvements ranging from 1.16× improvement to 1.50× improvement.

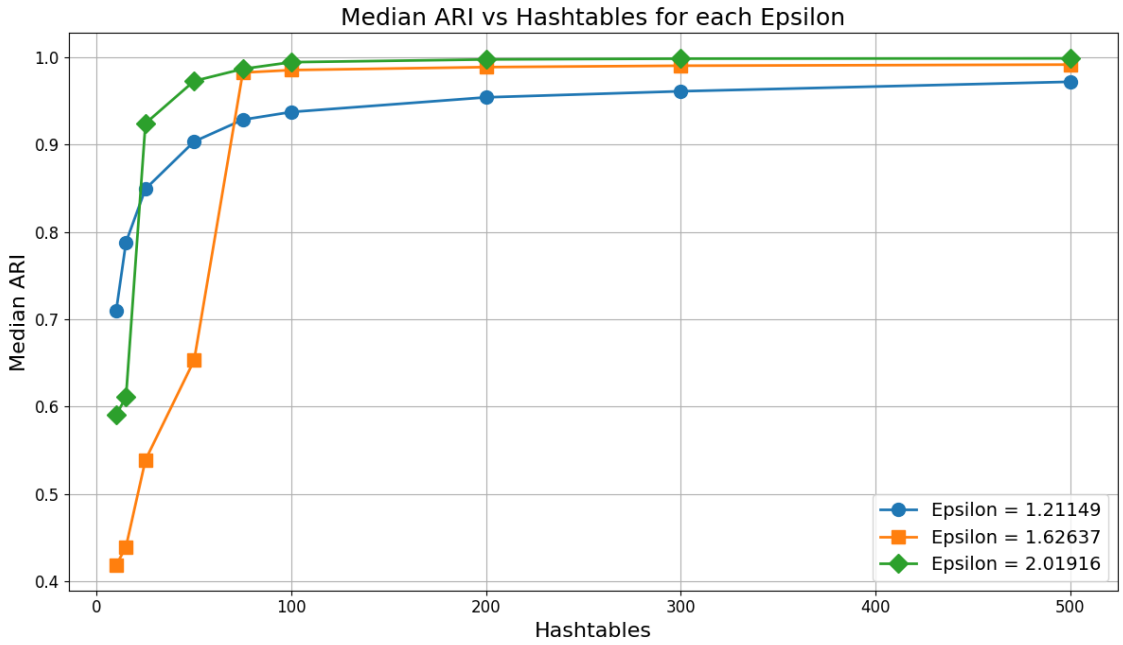
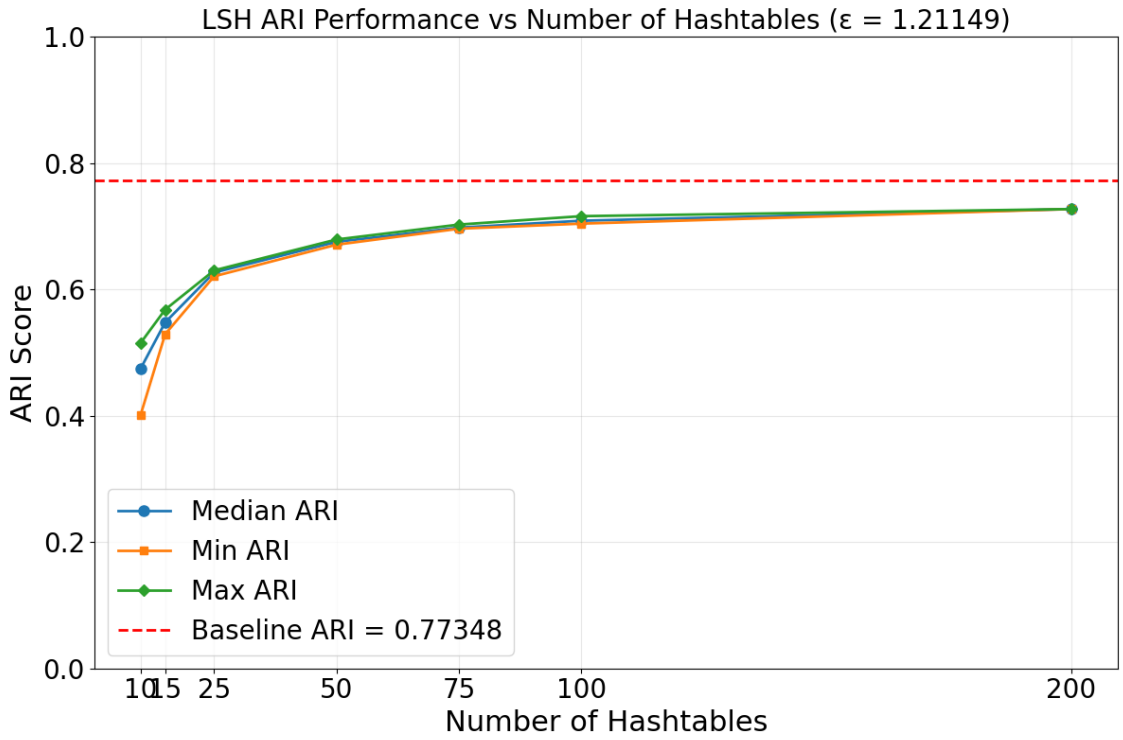


Figure 5.1: Convergence analysis of IP.LSH.DBSCAN clustering results compared to exact DBSCAN results. The graph shows how the ARI between IP.LSH.DBSCAN and DBSCAN clustering results approach 1.0 as the number of hash tables (L) increases.

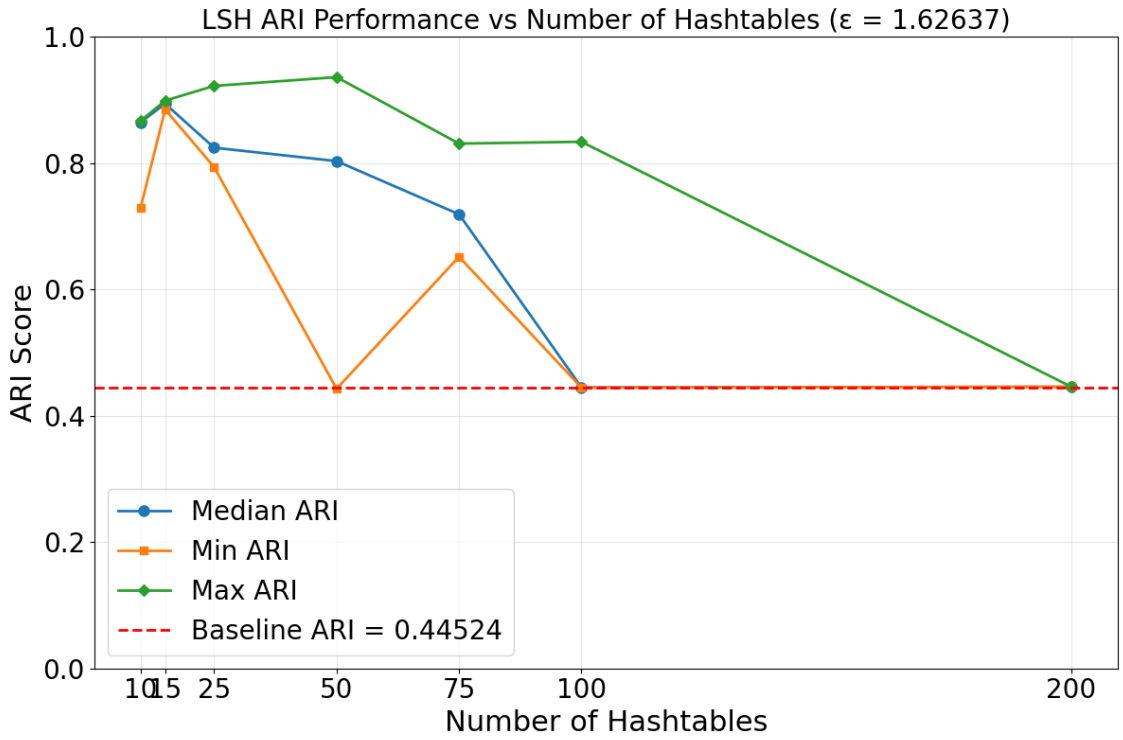
Cluster Quality Metrics

Measurement	**DBI**	**Silhouette**	**CH**	**Dunn**	**DBCV**
KD-Min	0.4965	0.6274	194	0.2241	0.3528
PO-Min	0.3868	0.7233	230	0.2873	0.4405
Min Improvement	1.28×	1.15×	1.19×	1.28×	1.25×
KD-Max	0.5525	0.6690	230	0.3113	0.4307
PO-Max	0.4555	0.7900	302	0.4526	0.5172
Max Improvement	1.21×	1.18×	1.31×	1.45×	1.33×
KD-Mean	0.5410	0.6455	211	0.2474	0.3901
PO-Mean	0.4231	0.7515	255	0.3709	0.4897
Mean Improvement	1.28×	1.16×	1.21×	1.50×	1.26×

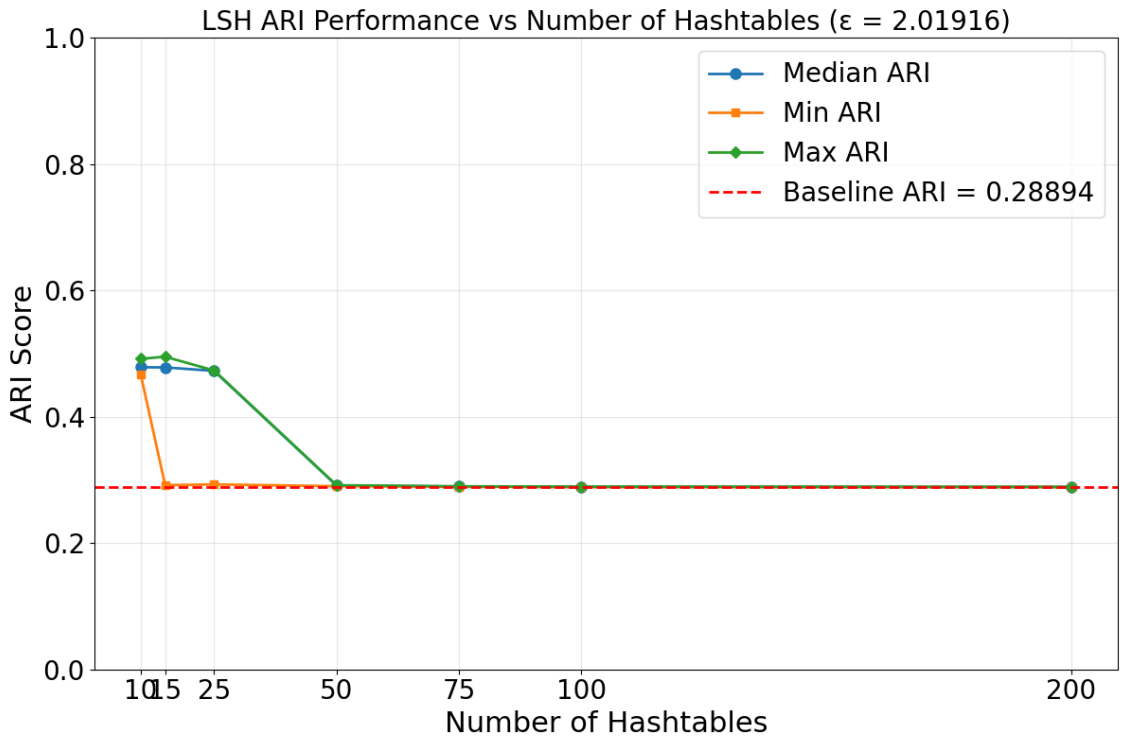
Table 5.9: Cluster quality metrics comparing k-distance parameter selection versus automated parameter optimization. Mean improvements range from 16% to 50% across five quality measures. KD = K-Distance, PO = Parameter Optimizer.



(a) IP.LSH.DBSCAN ARI comparison with DBSCAN ARI for epsilon = 1.21149 and minPts = 10



(b) IP.LSH.DBSCAN ARI comparison with DBSCAN ARI for epsilon = 1.62677 and minPts = 10



(c) IP.LSH.DBSCAN ARI comparison with DBSCAN ARI for epsilon = 2.01916 and minPts = 50

Figure 5.2: Comparison of clustering accuracy (ARI) between IP.LSH.DBSCAN and DBSCAN across varying hash table counts. Three parameter configurations shown: (a) ε=1.211, minPts=10; (b) ε=1.627, minPts=10; (c) ε=2.019, minPts=50

5.6 Application and Evaluation of CLUE in a Real-world Scenario

In this section, we demonstrate and evaluate the CLUE toolchain applied to real-world electricity consumption data using Dataset 1, as previously introduced in this chapter. The assessment will consist of several rounds of CLUE, using different configurations and components to walk through an example pipeline and using CLUE to gain valuable insights extracted from electricity consumption time series data.

5.6.1 Round 1: Outlier Detection and Separation

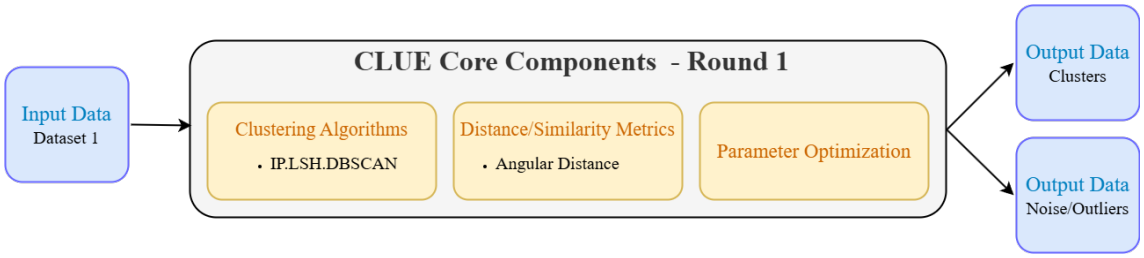


Figure 5.3: Round 1 CLUE pipeline workflow showing IP.LSH.DBSCAN processing 7,500 customer electricity consumption profiles. Output separates regular consumption patterns (clusters) from anomalous usage (noise or outliers).

The initial execution of CLUE, referred to as Round 1, utilized IP.LSH.DBSCAN to partition the dataset effectively by executing the parameter optimization module and applying the IP.LSH.DBSCAN algorithm using the top-performing feature combination result. The intent is to separate anomalous consumption patterns from the rest, thereby achieving anomaly detection and separating the regular and high-consuming customers.

As visualized in Figure 5.4, distinct consumption profiles can be observed after an execution of IP.LSH.DBSCAN:

- **Cluster 0:** Represents a group with relatively low and stable consumption throughout the day, with slight variations around midday.
- **Cluster 1:** Exhibits moderate consumption levels with an uptick in the morning and a gradual decline during the evening hours.
- **Cluster 2:** Identified as the noise/outlier cluster by IP.LSH.DBSCAN, showcasing significantly higher variability and average consumption than the other clusters, indicates unusual or irregular consumption patterns.

Table 5.10 quantifies the distribution of data points among these clusters. Cluster 0 dominates with 5062 samples, cluster 1 includes 421 samples, while the noise cluster (cluster 2) consists of 2083 samples. This substantial number of outliers highlights the effectiveness of CLUE in distinguishing between typical usage and anomalous patterns.

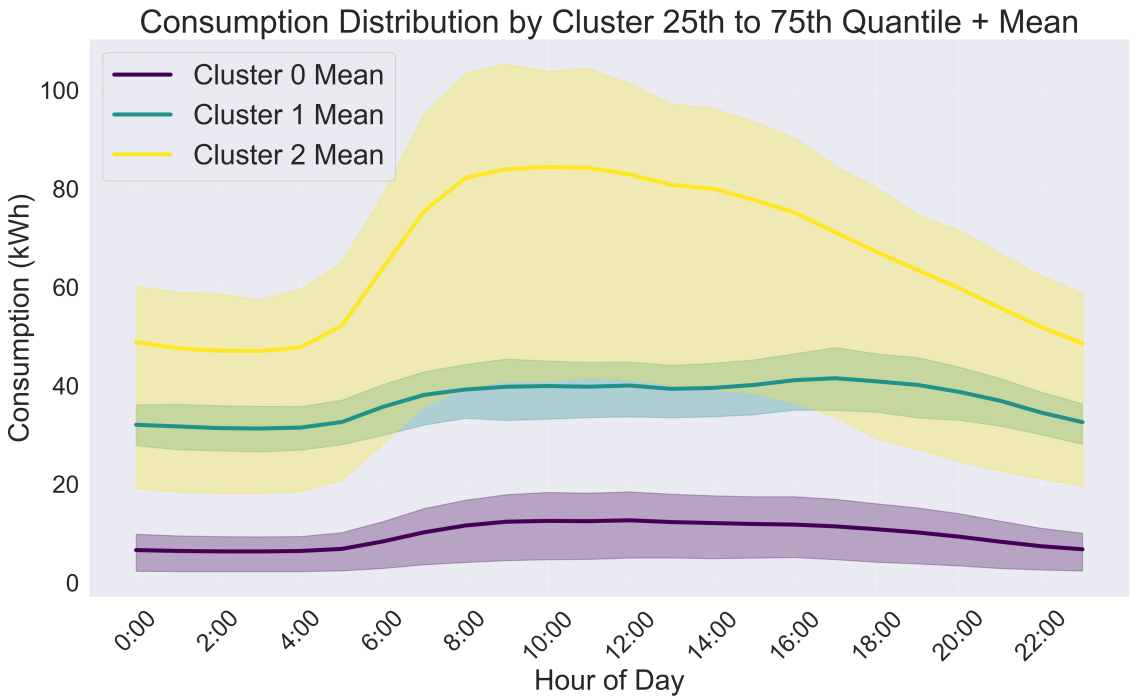


Figure 5.4: Daily electricity consumption patterns for three customer clusters identified in Round 1. Lines represent the mean consumption, while shaded areas indicate the ranges of the 25th to 75th percentiles. Cluster 2 (noise) shows significantly higher consumption and variability compared to regular clusters.

DBSCAN Cluster Size Distribution

Cluster	Count
0	5062
1	421
2	2083

Table 5.10: Distribution of customer data points across clusters identified by DBSCAN in Round 1. Cluster 2 represents noise points (outliers) detected by the density-based algorithm.

To further interpret the clustering results, Figure 5.5 presents a feature-based comparison of the distinctive attributes of electricity consumption. Key differences are highlighted using a color gradient in each column.

- **Peak-to-Average Ratio:** Cluster 0 displays the highest peak-to-average ratio, suggesting pronounced peak consumption periods relative to baseline consumption.
- **Number of Significant Peaks and Peak Width:** Cluster 1 has notably higher values for both these features, indicative of more frequent and sustained peak periods.
- **Base Load and Maximum Ramp-up:** Cluster 2 distinctly stands out with elevated base load averages and maximum ramp-up rates, reinforcing its classification as anomalous with frequent and abrupt consumption changes.

Figure 5.5: Feature comparison heatmap showing average consumption characteristics per cluster from Round 1. Color intensity indicates normalized feature values across clusters.

This initial segmentation validates CLUE's ability to differentiate between standard consumption behaviors and anomalous patterns effectively, establishing a foundation for deeper analysis. Users can target specific interventions or further specialized analyses, enhancing demand management and operational strategies.

Future rounds will leverage insights from Round 1 to iteratively refine clustering approaches and explore additional consumption behaviors in greater detail.

5.6.2 Round 2A: Exploring the Clusters Data

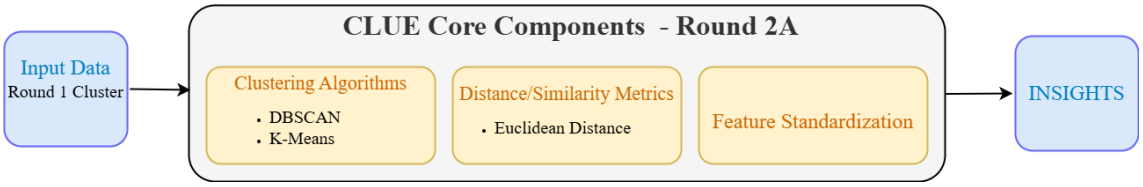


Figure 5.6: Round 2A CLUE pipeline processing 5,482 non-outlier customers from Round 1. Applies both feature-based DBSCAN and raw time series K-means for comparative analysis.

Building upon the initial segmentation from Round 1, Round 2A applies two distinct clustering strategies to the 5,482 non-outlier data points (clusters 0 and 1 from Round 1). This comparative analysis examines feature-based clustering using DBSCAN versus raw time series clustering using K-Means, demonstrating how different data representations and algorithms reveal complementary insights about electricity consumption patterns.

Feature-Based DBSCAN Analysis

The feature-based DBSCAN approach, configured with parameters derived from the optimization module, produced four distinct clusters with markedly different characteristics. As shown in Figure 5.7, the chart reveals clear differentiation across multiple consumption attributes.

CLUE identified significantly varying-sized clusters, as shown in Table 5.11, with cluster 0 dominating with 3762 data points and cluster 1 (labeled as noise/outliers) with 1619 points. Clusters 2 and 3 are notably smaller, containing only 79 and 22 data points, respectively. This highly skewed distribution suggests that most customers share relatively common consumption patterns, while clusters 2 and 3 represent rare but distinct behavioral patterns.

DBSCAN Cluster Size Distribution

Cluster	Count
0	3762
1	1619
2	79
3	22

Table 5.11: Distribution of non-outlier customers across feature-based DBSCAN clusters in Round 2A. Highly imbalanced distribution with 68.7% in cluster 0 and only 0.4% in cluster 3.

Analysis of the feature profiles in Figure 5.7 reveals distinct peak consumption behaviors, which can also be observed in Figure 5.8. Examining peak behavior consumption, Cluster 0 shows extreme variations in each behavior compared to the others, with a peak-to-average ratio of 2.05, the narrowest peak widths, and the

lowest base load. Compared to clusters 2 and 3, with a base load average of 24.34 and 29.19, respectively, and a lower peak-to-average ratio.

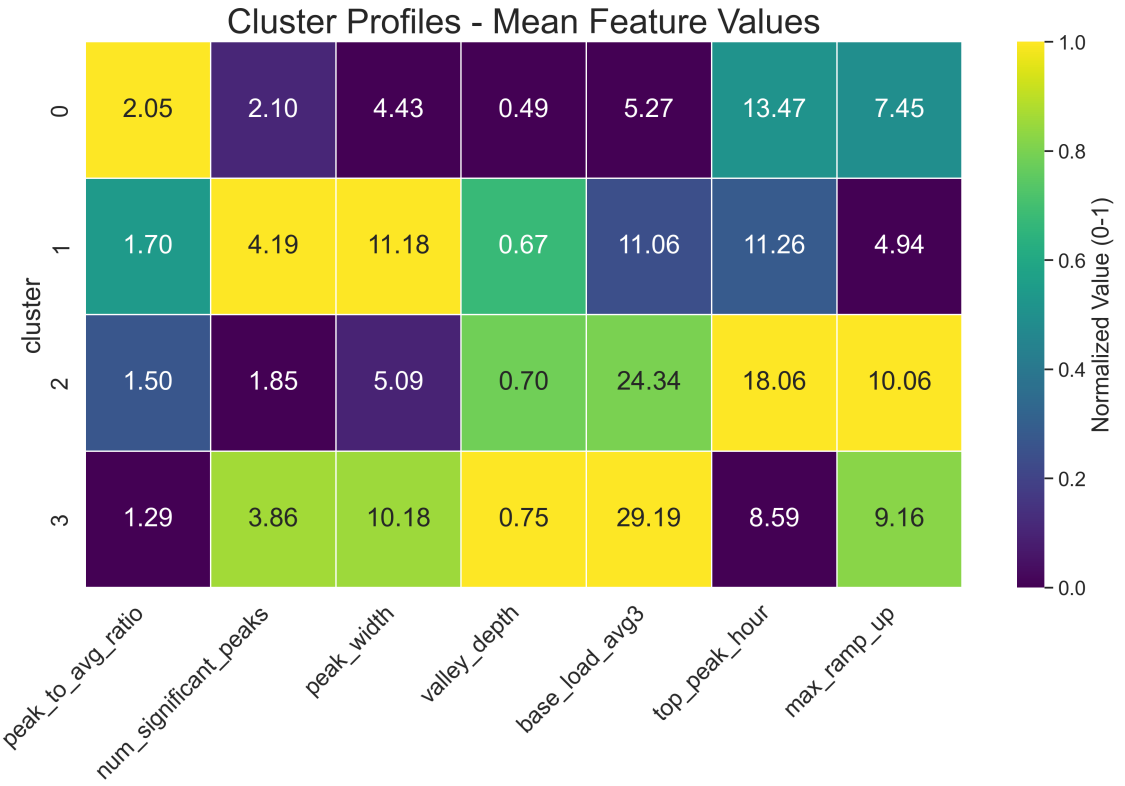


Figure 5.7: Round 2A feature-based clustering results showing consumption characteristics per cluster. Cluster 0 exhibits the highest peak-to-average ratio, while clusters 2-3 show elevated base loads.

The consumption profiles shown in Figure 5.8 show distinct cluster variations. There is a clear separation in amplitude between cluster 0 and cluster 1, as well as between cluster 2 and cluster 3, with the latter exhibiting significantly higher consumption on average. Cluster 2's profile shows a clear behavior of a sharp ramp-up in the afternoon, high peak consumption during evening hours, and a drastic ramp-down. It differs from the other three profiles, which all show a similar curve, albeit with different amplitudes.

The feature-based DBSCAN successfully identifies consumption patterns that correlate with operational intensity rather than just magnitude. The algorithm's ability to produce such an imbalanced cluster, with two dominant groups and two small outlier groups, demonstrates its effectiveness in identifying genuinely distinct behavioral patterns, even when they represent only a small fraction of the customer base. This segmentation provides valuable insights for targeted grid management strategies, particularly for the high-impact customers in clusters 2 and 3.

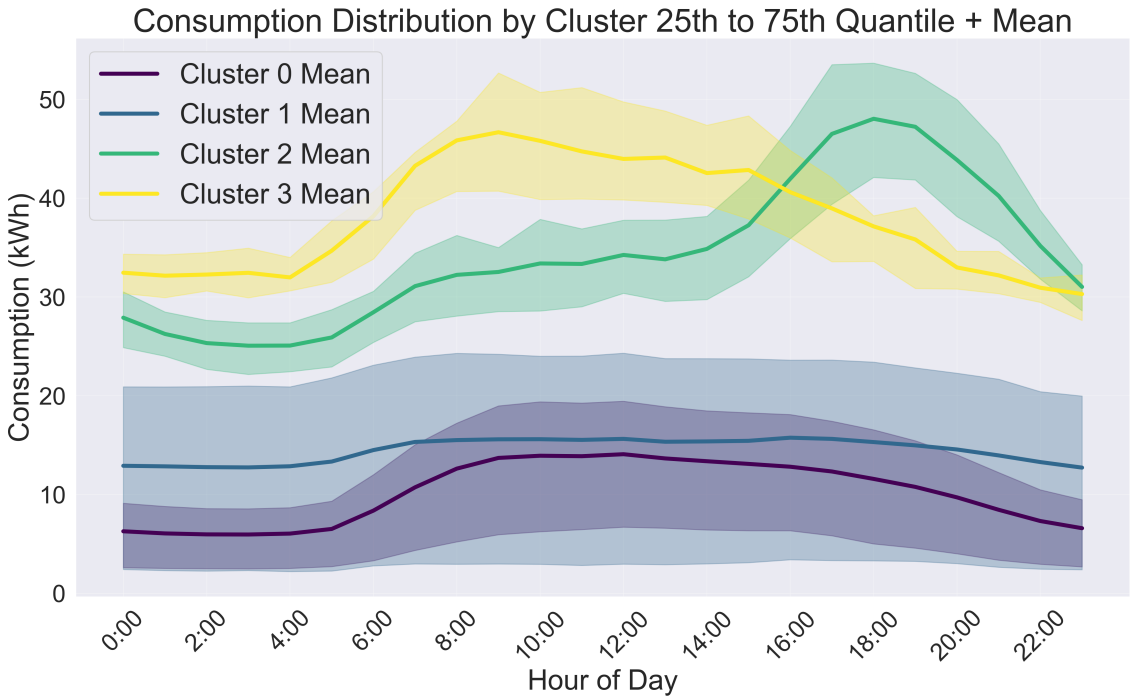


Figure 5.8: Consumption profiles from Round 2A feature-based DBSCAN showing four distinct patterns. Amplitude separation is visible between cluster pairs (0-1 and 2-3).

Raw time series K-Means Analysis

The K-Means clustering, applied directly to the 24-dimensional time series vectors (k = 66), provides an alternative perspective on the data structure. This approach segments based on the complete temporal profile rather than extracted features.

K-Means Cluster Size Distribution

Cluster	Count
0	1620
1	1519
2	926
3	417
4	435
5	565

Table 5.12: Customer distribution for Round 2A K-means clustering on raw time series data. More balanced distribution (7.6% to 29.6%) compared to the feature-based approach.

As shown in Table 5.12, K-means produces a relatively balanced distribution across six clusters. Clusters 0 and 1 are the largest, containing 1620 and 1519 data points, respectively. Cluster 2 is of medium size, with 926 points, while Clusters 3, 4, and 5 are smaller, with 417, 435, and 565 points, respectively. This more balanced distri-

bution contrasts sharply with the feature-based DBSCAN results, suggesting that raw-time series clustering identifies more gradual transitions between consumption patterns.

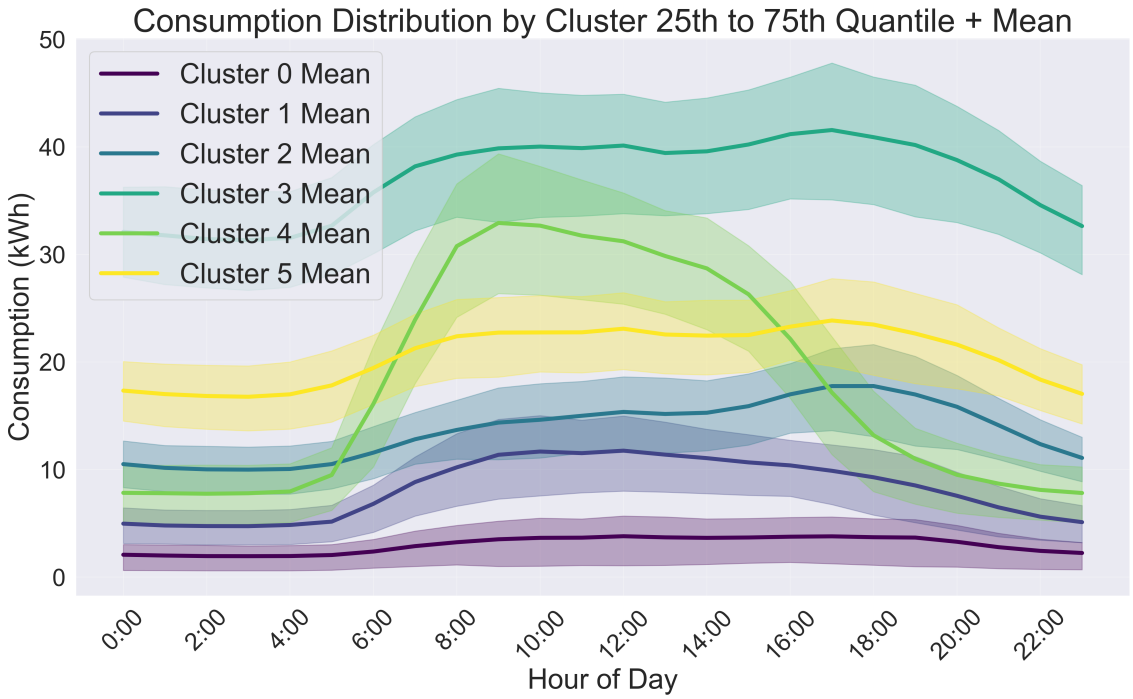


Figure 5.9: Six consumption patterns identified by K-means clustering of raw time series in Round 2A. Clusters are primarily separated by consumption magnitude, with Cluster 4 exhibiting unique morning peak behavior.

The consumption pattern plot shown in Figure 5.9 reveals six distinct patterns with clear separation in consumption magnitude. Clusters 0 and 1, the most significant clusters, can be observed to have the lowest average consumption throughout the day. Interestingly, cluster 4 stands out from the rest, as observed from its consumption profile curve, which features a very high morning ramp-up followed by high consumption and a steep ramp-down, in contrast to the apparent uptick and peak during the evening in most clusters.

Comparing the mean feature values for the k-means clustering shown in Figure 5.10, it should be observed how the top peak hour values are much closer in range than in Figure 5.7 for the feature-based DBSCAN clustering.

The base load average progresses from Cluster 0 (1.64) to Cluster 3 (29.09), indicating that K-Means effectively separates customers based on their baseline consumption levels. Following the observations about cluster 4, a significantly higher maximum ramp-up rate of 14.86 was observed compared to the other clusters, with an average of around 7.

This approach proves valuable for load forecasting and capacity planning, where understanding the complete temporal profile and absolute consumption levels is crucial. The apparent clustering by magnitude facilitates tiered service offerings

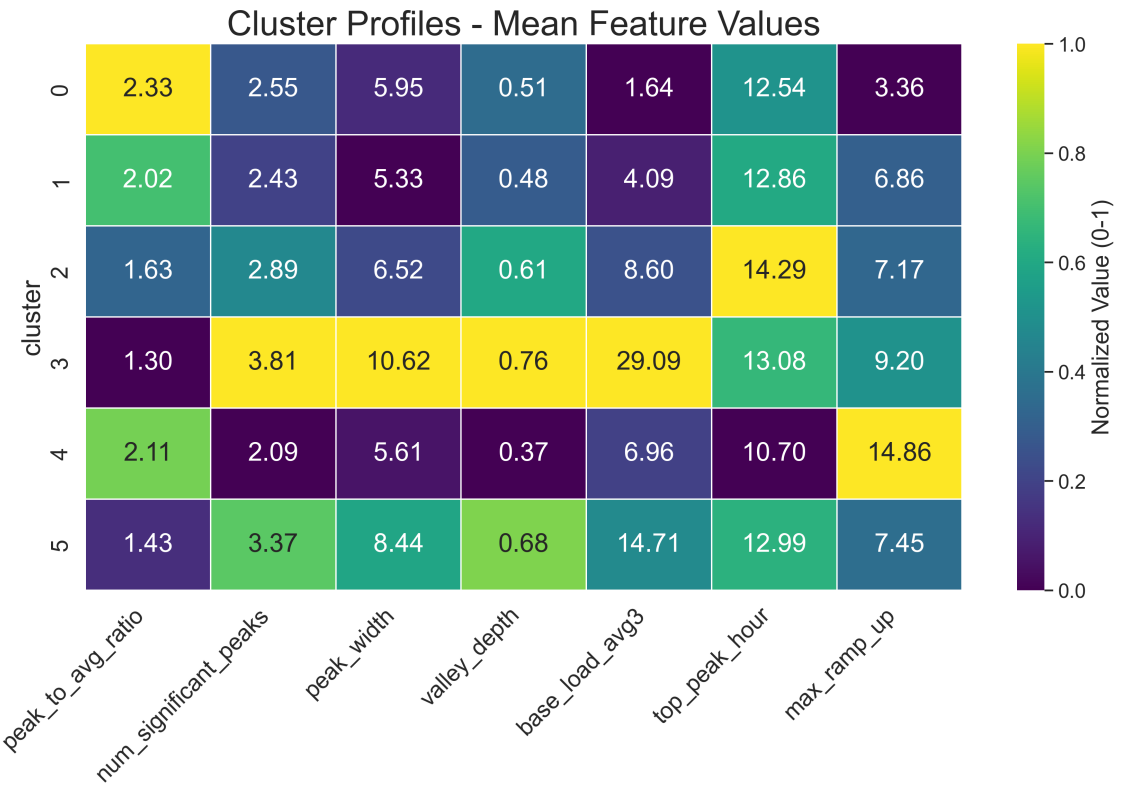


Figure 5.10: Feature comparison for Round 2A K-means results showing progression of base load from 1.64 kW (cluster 0) to 29.09 kW (cluster 3). Cluster 4 exhibits the highest ramp-up rate at 14.86 kW/hour.

and differential pricing strategies based on overall consumption intensity.

Comparative Analysis and Insights

The comparison of feature-based DBSCAN and raw time series K-Means clustering reveals fundamental differences in how each method interprets the consumption data.

DBSCAN produces a highly imbalanced distribution (68.7%, 29.5%, 1.4%, 0.4%), while K-Means creates balanced clusters (ranging from 7.6% to 29.6%). This reveals that while most customers share similar operational patterns, their consumption magnitudes vary continuously. Simultaneously, DBSCAN successfully isolated 101 customers with anomalous patterns, characterized by extreme peak-to-average ratios and unusual base loads. K-Means distributed these outliers across magnitude-based clusters, but could not recognize their behavioral uniqueness.

The analysis reveals that the customer base is behaviorally homogeneous but varies significantly in terms of consumption scale. Approximately 98% of customers follow standard operational patterns (morning ramp-up, midday plateau, evening decline) but at different magnitudes. This suggests that business operations are standardized while company sizes differ significantly.

5.6.3 Round 2B: Exploring the noise/outlier Data

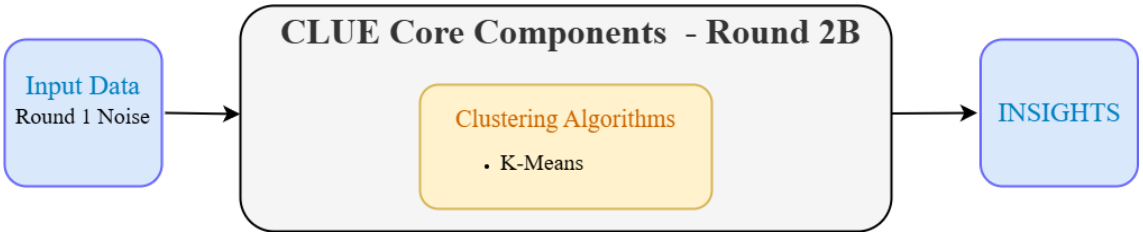


Figure 5.11: Round 2B pipeline analyzing 2,083 outlier customers identified in Round 1. K-means clustering reveals internal structure within previously identified noise points.

Round 2B utilizes the clustering result from Round 1 to further investigate the identified noise and outlier points. It performs K-means clustering on the time series data, further explores the outliers, and potentially reveals interesting behavioral changes to find valuable insights.

Table 5.13 presents the size distribution of clusters, with a wide range between the largest cluster 0 with 916 data points and the smallest, cluster 5, with only 10 data points.

Figure 5.12 shows cluster consumption patterns. This visualization provides immediate insights into cluster behaviors, highlighting the distinct temporal patterns captured through clustering. Cluster 5 exhibits the highest average consumption and a pronounced daily pattern, peaking around midday, with a magnitude significantly higher than that of the other clusters. Clusters 2, 3, and 4 exhibit medium to

K-Means Cluster Size Distribution	
Cluster	**Count**
0	916
1	675
2	137
3	295
4	48
5	10

Table 5.13: Distribution of outlier customers across six K-means clusters in Round 2B. Ranges from 916 customers in cluster 0 to only 10 in cluster 5.

high consumption (for this dataset), with clear peaks and wider consumption bands, indicating higher variability in consumption. While clusters 0 and 1, representing the most significant customer segments, as seen in Table 5.13, show the significantly lowest consumption levels throughout the day.

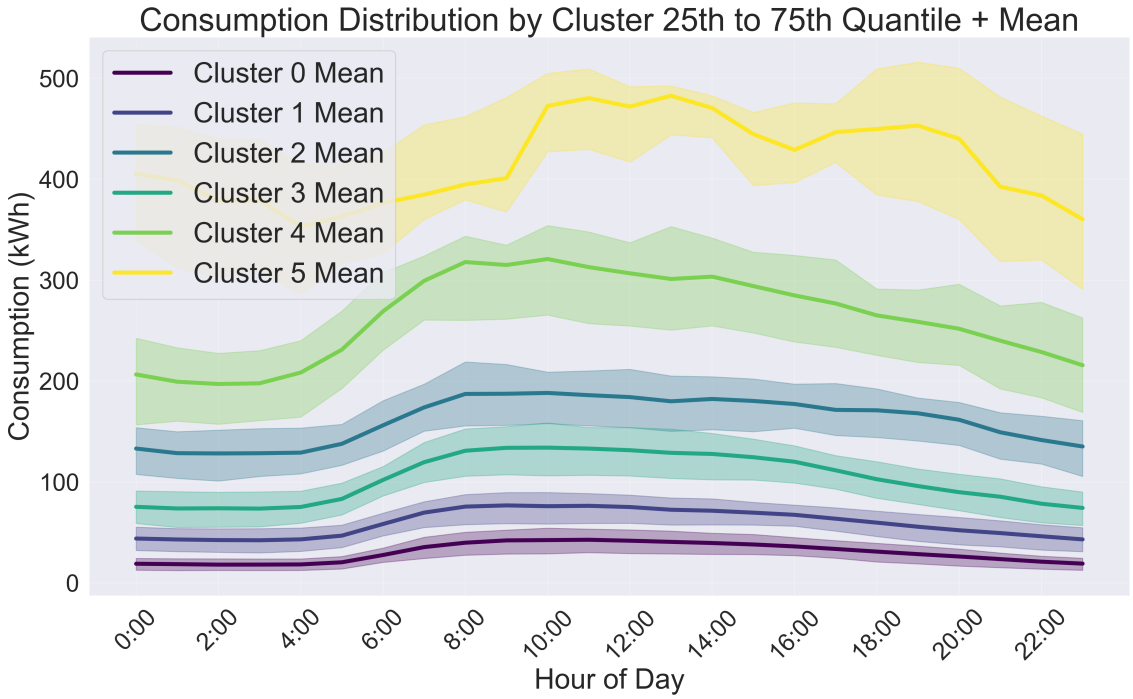


Figure 5.12: Consumption patterns for outlier customer clusters showing extreme behavioral diversity. Cluster 5 peaks at 400-500 kW while clusters 0-1 remain below 100 kW.

The detailed characteristics of each cluster are illustrated in Figure 5.13, showing the mean values of key consumption-related attributes. Some interesting observations include that cluster 5 experiences significantly higher values in features base load average, maximum ramp-up, and peak width, indicating consistent and significant electricity consumption along with rapid increases and sustained peak demand periods. Clusters 0 and 1 are characterized by lower overall consumption

with significantly lower base load average, indicative of steadier, less volatile energy consumption patterns.

Figure 5.13: Feature analysis of Round 2B outlier clusters revealing cluster 5's extreme characteristics. A base load of 319.99 kW and a maximum ramp-up of 107.25 kW/hour indicate possible industrial-scale operations.

5.7 Summary

The evaluation demonstrated significant computational performance advantages of the CLUE framework. Using IP.LSH.DBSCAN, the clustering algorithm, achieved speedups of several magnitudes compared to standard DBSCAN, improving scalability for high-dimensional data. The technical analysis further revealed that parameter settings (ε, minPts, L, M) have a considerable impact on both runtime and clustering accuracy, with well-chosen configurations consistently delivering efficient and accurate clustering outcomes.

Applied evaluation through iterative clustering rounds provided clear evidence of the CLUE toolchain's practical utility. Initial rounds successfully isolated meaningful clusters by effective outlier detection. Subsequent analyses differentiated customer segments based on detailed consumption patterns, highlighting distinct operational profiles. These objectively identified clusters demonstrated CLUE's capability to reveal valuable behavioral groupings in real-world electricity grid data, providing essential insights for operational decision-making.

6

Discussion

In this chapter, we discuss the findings presented in the evaluation chapter and their implications.

6.1 Performance

In this section, we discuss the implications of the performance evaluation findings. The discussion primarily revolves around the differences in evaluation results between IP.LSH.DBSCAN and DBSCAN enable various proper functionalities for clustering frameworks such as CLUE.

The speed and accuracy of IP.LSH.DBSCAN offers several advantages over basic DBSCAN for clustering, including rapid parameter search, clustering of large and high-dimensional datasets, and consistently fast clustering.

6.1.1 Large high-dimensional datasets

Large, high-dimensional datasets typically pose challenges for density-based algorithms, such as DBSCAN. While IP.LSH.DBSCAN has already been proven to have superior complexity [17]. Our results demonstrate this speedup in practice, as shown in Table 5.2, when applied to moderately sized real-world and larger synthetic datasets.

The speedup factors ranging from 41x to 949x (Table 5.2) demonstrate that the algorithm scales effectively with both dataset size and dimensionality. For Dataset 3, the reduction from 56 minutes to under 4 seconds transforms clustering from a batch processing task to an interactive analysis tool.

This performance improvement has important implications for practical applications. Grid operators can now analyze consumption patterns across their entire customer base in near real-time, enabling them to make responsive decisions. The ability to process 96-dimensional vectors (representing 15-minute interval data) as quickly as 24-dimensional vectors (hourly data) removes the traditional tradeoff between temporal resolution and computational feasibility.

6.1.2 Parameter Search

Density-based clustering algorithms have traditionally been hindered by the need to determine the appropriate parameter values manually. The k-distance plot approach, while theoretically sound, often provides only a starting point rather than optimal parameters. Our results in Table 5.8 demonstrate that systematic parameter exploration can yield substantial improvements over single-point estimates.

The effectiveness of the parameter optimization framework stems directly from IP.LSH.DBSCAN's computational efficiency. While it would be possible to use DBSCAN for parameter optimization, our results indicate that for sufficiently large datasets (such as dataset 3), traditional DBSCAN would require hours to evaluate the parameter combinations we tested, whereas IP.LSH.DBSCAN achieved this in minutes. Furthermore, the speed of IP.LSH.DBSCAN allows for parameter optimization as a preparatory step, without significantly increasing execution time, provided the dataset is sufficiently large.

The 72.7% improvement in clustering accuracy (ARI from 0.448 to 0.773) for DBSCAN, achieved through systematic parameter exploration, demonstrates that the traditional approach of selecting parameters based solely on k-distance plots can yield suboptimal results. The ability to rapidly evaluate dozens of parameter combinations enables analysts to identify configurations that better align with the underlying structure of their data.

The parameter optimization results reveal that optimal configurations often differ significantly from theoretical recommendations. The best-performing parameters in our evaluation used different epsilon and MinPts combinations than those suggested by standard heuristics, highlighting the value of data-driven parameter selection. However, it is worth noting that these tests were performed on synthetic datasets, which are among the easiest datasets to cluster.

The parameter optimizer utilizes five different cluster metrics to find superior parameter combinations. As shown in Table 5.9, the parameter optimizer successfully improves all cluster metrics using our method. However, the selected parameter combination is the one with the highest weighted aggregate score of these five metrics, which means that the ARI is never directly considered. This presents a "missing link" between the parameter combination that the parameter optimizer considers the best (the one with the highest aggregate weighted score of the cluster quality metrics) and the optimal parameter combination in terms of the ARI. Preliminary testing indicates that the best weighted aggregate score does not yield a superior ARI when compared to using k-distance for parameter selection alone. This missing link in the parameter optimizer will be discussed further in the future work section.

Furthermore, the parameter search space was partially determined manually. The values of L, M, and, to a lesser extent, epsilon and MinPts were set based on our experience with the data itself. While each parameter range searched had some starting point based on previous research, the ranges were somewhat arbitrary. Nonetheless, the parameter optimization results showed that parameter optimization is a promising application of IP.LSH.DBSCAN to improve the performance of DBSCAN.

6.2 Applicability

This section discusses the applicability of the CLUE toolchain, considering the cluster results presented in the previous chapter and the functionality described in Chapter 4.

The CLUE toolchain's potential speed, as shown in Table 5.4, enables a fundamentally different approach to clustering analysis. Rather than requiring analysts to commit to specific parameter choices upfront, CLUE supports rapid iteration cycles, during which users can explore different configurations, assess results, and refine their approach based on domain knowledge.

The usefulness of various tools for interactive cluster exploration is exemplified by the use of IP.LSH.DBSCAN for outlier detection is shown in Section 5.5.1. By utilizing clustering as a tool to define outliers in the data, we obtain a clear, yet previously undefined, split between Cluster 0 and Cluster 2 in Figure 5.5, where Cluster 2 is defined by its base load and maximum ramp-up. By further observing the consumption profiles in Figure 5.4, it is clear that Cluster 2 indeed consists of users with higher consumption than Cluster 1.

In the second round of clustering, the two different approaches used for clustering the non-outliers show the benefits of both K-Means and DBSCAN feature clustering. The consumption profiles shown in Figure 5.9 exhibit a partitioning that focuses on the amplitude of usage. This partitioning aligns with what is expected of K-Means; the k partitions are evenly distributed across the entire space, forming clear distinctions between consumption amplitudes. However, when considering the results from the DBSCAN feature-based clustering shown in 5.8, the influence of the features themselves becomes clearer. The consumption profiles do not adhere to consumption patterns alone, but instead follow the directed patterns by the feature selection. These results further establish that a variety of options enhances the applicability of the toolchain as an interactive exploration tool for clustering.

The usefulness of the CLUE toolchain is directly linked to the ability to derive insightful information from the clustering results. Figure 5.7 shows an example of such a situation where, as mentioned in previous sections as insightful information, peak usage is split across clusters. Customers in Cluster 0 tend to peak around 2:00 PM, while those in Cluster 2 tend to peak around 6:00 PM. Furthermore, splits within peak width suggest that customers in Cluster 1 are active for almost half of the day, while customers in Cluster 0 tend to be active for only a few hours.

7

Future Work

The CLUE toolchain establishes a foundation for efficient high-dimensional time series clustering, but several areas offer opportunities for extension and improvement.

While our results have shown potential for using IP.LSH.DBSCAN for parameter optimization, the tests were performed on synthetic data due to the lack of accessible, high-dimensional, and large-scale data with ground truth. To establish the robustness of these methods, they must be explored on more chaotic data, either on far more complex synthetic data, such as data created from a mold of existing data, or on real-world data with ground truth.

The parameter optimization results suggest that IP.LSH.DBSCAN-based parameter search can identify parameter combinations that outperform those found using k-distance and heuristics when applied to basic DBSCAN. The search space of this parameter search still needs to be established with more certainty. We arbitrarily chose some combinations, such as taking 0.5, 0.75, 1.25, and 1.5 of the epsilon found by k-distance. Establishing a more motivated set of epsilons could further improve the performance of the parameter optimizer.

As mentioned in Chapter 6, there is a missing link between the ARI results and the cluster quality metrics we employ to find the 'best' parameter combinations without ground truths available. More work needs to be done to explore how to correlate cluster quality results with ARI results. This could involve exploring combinations of cluster quality metrics or evaluating different weights for various combinations of metrics and finding correlations between the ARI found for that dataset and the cluster quality metric results.

Furthermore, these results must be explored with similarly constructed datasets. These datasets could be a set of daily readings for electricity consumption or synthetically constructed datasets built using the same rules but with varying levels of randomness. This research aims to establish whether a correlation between ARI and cluster metrics, as found in a dataset with ground truth, can be used to obtain good cluster results (specifically concerning ARI, but utilizing the metrics) in a dataset of similar construction.

The selection of hash tables for the parameter optimizer should also be explored further. The results show that fewer hash tables improve latency but make the results less reliable. This could be leveraged using a more complex parameter exploration algorithm, where more runs are performed at lower hash tables, and the

most promising results are selected for runs with higher hash tables.

While the usefulness of CLUE lies primarily in its ability to efficiently utilize clustering for interactive and quick cluster exploration, for time series data, many insights are hidden across multiple periods. To capture this hidden information, it would be necessary to expand the toolchain further to automatically evaluate several runs of the same configuration on different datasets, such as those from other days. Developing techniques for tracking clusters across periods would enable the creation of timelines of clusters, capturing not only states but also the changes themselves as information.

8

Conclusion

This thesis proposed the CLUE toolchain to provide solutions for data analysis of high-dimensional time series vector data. The CLUE toolchain incorporates a variety of clustering methods and techniques to cover a wider area than individual clustering algorithms traditionally allow. The combined use of density-based clustering algorithms, k-means, selection of distance metrics, types of data representation, round-based clustering, and parameter optimization functionality culminates in a toolchain that strikes a balance between versatility and usability.

IP.LSH.DBSCAN is employed in CLUE, not only as a fast alternative to regular DBSCAN when a density-based algorithm is desired for large or wide datasets, but also as the core component of a parameter optimizer. The results presented suggest that the parameter optimizer enhances both the effectiveness and usability of base DBSCAN by providing a tool to automatically establish parameters that outperform those established by other automatic and heuristic methods. This is demonstrated by both the use of clustering evaluation against synthetic data with true labels and the application of cluster quality metrics to real-world high-dimensional time series data.

The CLUE toolchain was demonstrated using a configuration consisting of IP.LSH.-DBSCAN is used to split outliers, and then another round of clustering is performed for the outliers and non-outliers, respectively. This was achieved by both clustering the raw data using K-Means and the extracted features using DBSCAN, allowing for the visualization of their differences in application. This example run demonstrated the versatility of the toolchain and the effectiveness of round-based clustering in targeting specific subsets in the dataset.

CLUE effectively solves the sub-problems described in Section 3.7. CLUE manages to cluster both high-dimensional and voluminous data in a timely manner by the use of IP.LSH.DBSCAN and K-means. CLUE also addresses the issue of parameter optimization by leveraging IP.LSH.DBSCAN combined with cluster quality metrics. Although further research is needed when utilizing external validation metrics, such as ARI, for this process. CLUE's modular architecture provides interchangeable components across clustering algorithms and distance metrics. The round-based clustering approach enables iterative refinement of clustering results, allowing for rapid iteration cycles that facilitate interactive data exploration and analysis. Analysts can explore configurations, assess results, and refine approaches based on domain knowledge within a reasonable time.

Bibliography

[1] I. Assent, "Clustering high dimensional data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 340–350, 2012. DOI: https://doi.org/10.1002/widm.1062.

[2] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012. DOI: https://doi.org/10.1002/sam.11161.

[3] J. W. Tukey *et al.*, *Exploratory data analysis*. Springer, 1977, vol. 2. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-031-20719-8_2?pdf=chapter%20toc.

[4] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003. DOI: 10.1023/A:1024988512476. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0042711018&doi=10.1023%2fA%3a1024988512476&partnerID=40&md5=117702dd7430230cf54df7619b591f4b.

[5] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010. DOI: https://doi.org/10.1016/j.patrec.2009.09.011.

[6] J. Han, M. Kamber, and J. Pei, "10 - cluster analysis: Basic concepts and methods," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Third Edition, Boston: Morgan Kaufmann, 2012, pp. 443–495, ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00010-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123814791000101.

[7] C. C. Aggarwal and C. K. Reddy, Eds., *Data Clustering: Algorithms and Applications*, 1st ed. Chapman and Hall/CRC, 2014. DOI: 10.1201/9781315373515.

[8] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information systems*, vol. 53, pp. 16–38, 2015. DOI: https://doi.org/10.1016/j.is.2015.04.007.

[9] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction To Cluster Analysis*. Jan. 1990, ISBN: 0-471-87876-6. DOI: 10.2307/2532178.

[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96,

1996, pp. 226–231. [Online]. Available: https://dl.acm.org/doi/10.5555/3001460.3001507.

[11] P. C. Hammer and R. Bellman, "Dynamic programming. princeton university press, princeton, n.j., 1957. xxv+ 342 pp. $6.75.," *Science*, vol. 127, no. 3304, pp. 976–976, 1958. DOI: 10.1126/science.127.3304.976.a. eprint: https://www.science.org/doi/pdf/10.1126/science.127.3304.976.a. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.127.3304.976.a.

[12] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics: econophysics, bioinformatics, and pattern recognition*, Springer, 2004, pp. 273–309. DOI: https://doi.org/10.1007/978-3-662-08968-2_16.

[13] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011. DOI: https://doi.org/10.1016/j.engappai.2010.09.007.

[14] T. Warren Liao, "Clustering of time series dataa survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2005.01.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320305001305.

[15] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 262–270. DOI: https://doi.org/10.1145/2339530.2339576.

[16] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 3, pp. 231–240, 2011. DOI: https://doi.org/10.1002/widm.30.

[17] A. Keramatian, V. Gulisano, M. Papatriantafilou, and P. Tsigas, "IP.LSH.DBSCAN: Integrated parallel density-based clustering through locality-sensitive hashing," in *European Conference on Parallel Processing*, Springer, 2022, pp. 268–284.

[18] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, pp. 117–122, 2008. DOI: https://doi.org/10.1145/1327452.1327494.

[19] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262. DOI: https://doi.org/10.1145/997817.997857.

[20] Y.-P. Wu, J.-J. Guo, and X.-J. Zhang, "A linear dbscan algorithm based on lsh," in *2007 International Conference on Machine Learning and Cybernetics*, IEEE, vol. 5, 2007, pp. 2608–2614.

[21] Y. Shiqiu and Z. Qingsheng, "Dbscan clustering algorithm based on locality sensitive hashing," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1314, 2019, p. 012 177. DOI: 10.1088/1742-6596/1314/1/012177.

[22] S. V. Jayanti and R. E. Tarjan, "A randomized concurrent algorithm for disjoint set union," in *Proceedings of the 2016 ACM Symposium on Principles of*

Distributed Computing, 2016, pp. 75–82. DOI: https://doi.org/10.1145/2933057.2933108.

[23] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, University of California press, vol. 5, 1967, pp. 281–298. [Online]. Available: https://scispace.com/pdf/some-methods-for-classification-and-analysis-of-multivariate-4pswti19oz.pdf.

[24] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[25] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006. [Online]. Available: https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf.

[26] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. Springer, 2016, ISBN: 978-3-319-29852-8. DOI: https://doi.org/10.1007/978-3-319-29854-2.

[27] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd international conference on knowledge discovery and data mining*, 1994, pp. 359–370. DOI: https://doi.org/10.1145/3230734.

[28] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. DOI: https://doi.org/10.1016/j.neucom.2018.03.067.

[29] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010. DOI: https://doi.org/10.1002/wics.101.

[30] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 758–770, ISBN: 978-3-540-32106-4. DOI: https://doi.org/10.1007/11494669_93.

[31] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*, SIAM, 2014, pp. 839–847. DOI: https://doi.org/10.1137/1.9781611973440.96.

[32] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, no. sept, pp. 2487–2531, 2010. [Online]. Available: https://www.jmlr.org/papers/volume11/radovanovic10a/radovanovic10a.pdf.

[33] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" In *Database Theory — ICDT'99*, C. Beeri and P. Buneman, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235. DOI: https://doi.org/10.1007/3-540-49257-7_15.

[34] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*, Springer, 2001, pp. 420–434. DOI: https://doi.org/10.1007/3-540-44503-X_27.

[35] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, pp. 275–309, 2013. DOI: https://doi.org/10.1007/s10618-012-0250-5.

[36] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, Jul. 2017, ISSN: 0362-5915. DOI: 10.1145/3068335. [Online]. Available: https://doi.org/10.1145/3068335.

[37] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978. DOI: 10.1109/TASSP.1978.1163055.

[38] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, pp. 358–386, Jan. 2005. DOI: 10.1007/s10115-004-0154-9.

[39] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

[40] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987, ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[41] T. Caliski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974. DOI: https://doi.org/10.1080/03610927408827101.

[42] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," 1973. DOI: https://doi.org/10.1080/01969727308546046.

[43] D. Moulavi, P. Andretta Jaskowiak, R. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," Apr. 2014. DOI: 10.1137/1.9781611973440.96.

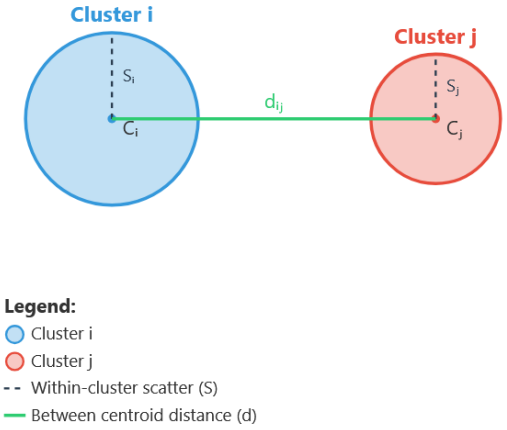
[44] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern recognition*, vol. 46, no. 1, pp. 243–256, 2013. DOI: https://doi.org/10.1016/j.patcog.2012.07.021.

[45] A. Zheng and A. Casari, *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018, ISBN: 1491953241. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3239815.

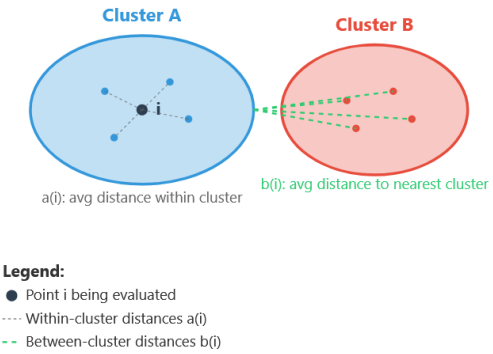
[46] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, pp. 193–218, 1985. DOI: https://doi.org/10.1007/BF01908075.

A

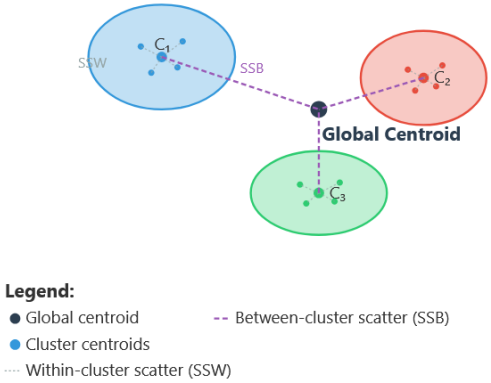
Appendix 1



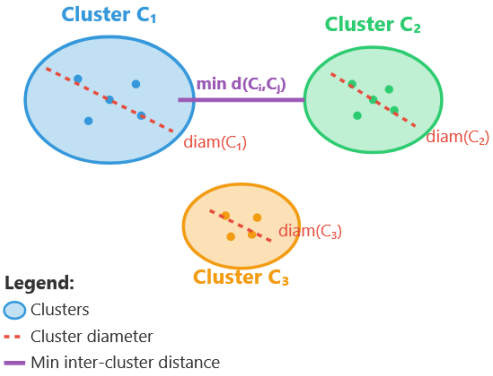
(a) Davies-Bouldin Index



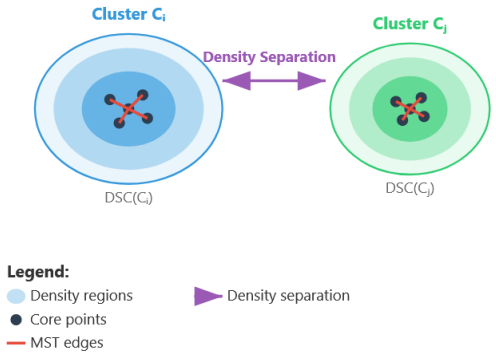
(b) Silhouette Coefficient



(c) Calinski-Harabasz Index



(d) Dunn Index



(e) Density-Based Clustering Validation

Figure A.1: Comparison of different clustering validation metrics: (a) Davies-Bouldin Index minimizes the ratio of within-cluster scatter to between-cluster separation, (b) Silhouette Coefficient measures how similar an object is to its cluster compared to others, (c) Calinski-Harabasz Index evaluates cluster validity based on the ratio of between-clusters to within-clusters dispersion, (d) Dunn Index identifies compact and well-separated clusters, and (e) Density-Based Clustering Validation assesses clusters based on density connectedness.

B

Appendix 2: Synthetic Dataset Construction

This appendix describes the methodology used to generate the synthetic electricity consumption datasets (Datasets 3 and 4) employed in the evaluation of the CLUE toolchain.

The dataset comprises 50,000 individual electricity meters, each representing a distinct consumer. Each meter records consumption data at 15-minute intervals throughout 24 hours, resulting in 96 measurement points per meter.

Consumption Pattern Design

The generator creates seven distinct consumption patterns that represent realistic customer behaviors observed in electricity grids. Each pattern is mathematically constructed to exhibit specific temporal characteristics:

Morning Peak Pattern generates consumption profiles with elevated usage between 06:00 and 09:00. The pattern uses a sinus function to create a smooth peak during morning hours, representing businesses or households with significant morning activity.

Evening Peak Pattern produces profiles with maximum consumption between 17:00 and 20:00. This pattern captures typical commercial establishments or residences with evening-focused energy usage.

Low Consumption Pattern maintains consistently minimal electricity usage throughout the day with only slight sinusoidal variations. This represents energy-efficient customers or premises with minimal electrical demands.

High Consumption Pattern generates profiles with sustained high electricity usage across all hours, varying sinusoidally around an elevated baseline. This pattern represents energy-intensive operations or large facilities.

Dual Peak Pattern combines both morning and evening peaks, with consumption surges during 06:00-09:00 and 17:00-20:00 periods. This represents mixed-use facilities or businesses with bifurcated operational schedules.

Midday Peak Pattern creates consumption profiles with maximum usage between 10:00 and 14:00, representing businesses with midday operational peaks such as

restaurants or retail establishments.

Night Activity Pattern generates profiles with elevated consumption from 22:00 to 04:00, representing facilities with overnight operations such as data centers or manufacturing plants with night shifts.

Each consumption pattern is generated using mathematical functions that create realistic temporal profiles. The base consumption level for each pattern is established, and then temporal variations are applied using sinusoidal functions to create smooth transitions between different consumption levels.

To enhance realism, the generator applies Gaussian noise to all data points. The noise level varies by pattern type, with some patterns receiving additional variability to simulate real-world measurement fluctuations and consumption irregularities. Twenty percent of meters within each pattern category receive enhanced noise levels to represent outliers within their respective groups.

The final consumption values are scaled by an amplitude factor of 100 to produce realistic kilowatt-hour measurements. All values are constrained to remain non-negative, ensuring physical validity of the generated data.

Alongside the consumption data, the generator produces a ground truth labels file that records the actual cluster membership for each meter.

The generator uses a fixed random seed (42) to ensure reproducibility of results across different executions.