University of Miskolc

Faculty of Mechanical Engineering and Informatics

Efficiency Analysis and Optimization of Concept Lattice Reduction Methods

Summary of Ph.D. Dissertation

József Hatvany Doctoral School of Information Science, Engineering and Technology

Research Area
Applied Computer Science

Research Group
Data and Knowledge Bases

Author: **Mohammed Ali Daash Alwersh**, M.Sc. in Computer Science

Head Of Doctoral School: **Prof. Dr. Jenő Szigeti**

Academic Supervisor: **Prof. Dr. László Kovács**

Miskolc, Hungary 2025

Table of Contents

Preface

This dissertation embodies a focused endeavor in the realm of knowledge engineering, particularly at the intersection of data mining and Formal Concept Analysis (FCA). Since its inception in the early 1980s by Rudolf Wille, FCA has been recognized as a powerful mathematical tool for representing and analyzing the relationships between objects and attributes within formal contexts. By structuring information into concept lattices, hierarchical diagrams that capture the relationships between objects and attributes, FCA facilitates the discovery of meaningful patterns in diverse fields, including software engineering, information retrieval, e-learning systems, bioinformatics, and beyond.

Yet, as datasets expand in size and complexity, the concept lattices derived from them can grow exponentially, posing formidable computational and interpretive challenges. Traditional FCA methods, while theoretically elegant, often become computationally intensive and cognitively overwhelming, hindering the effective utilization of these structures in large-scale data analytics. This dissertation addresses these challenges head-on through a series of three integrated contributions, each representing a strategic step toward more scalable, efficient, and human-centered FCA methodologies.

A key groundwork is first laid out, establishing several foundational pillars that guide the methods proposed here. These include the notion of kernel concepts, specially chosen concepts that serve as anchors for understanding and reducing a concept lattice, alongside an asymmetrical distance metric that adapts Dijkstra's algorithm for cost-aware navigation. A baseline greedy framework for concept selection further sets the stage for the more specialized methods and cognitively aligned reduction strategies that follow.

The first contribution introduces two novel extensions of the k-means algorithm, K-Means Dijkstra on Lattice (KDL) and K-Means Vector on Lattice (KVL), to adapt clustering-based reduction strategies for FCA. KDL leverages the inherent hierarchical structure of categorical data by incorporating a graph-based distance measure derived from FCA. This ensures that reductions remain faithful to the underlying conceptual relationships, yielding more interpretable and structurally consistent lattices. In contrast, KVL transforms formal concepts into numerical vectors, allowing the application of conventional k-means clustering at scale. While this vectorization simplifies complexity and improves computational efficiency, careful consideration is given to preserving lattice quality. Together, KDL and KVL mark an initial leap toward practical, data-driven lattice reduction that balances complexity management with interpretability.

Building on these foundations, the second contribution, the Kernel Concept Set (KCS) approach proposes a frequency- and cost-based strategy for selecting a core subset of concepts. By determining a kernel that covers the most critical and frequently occurring attributes, KCS optimizes reduction while maintaining essential structure. This approach goes beyond the first step's clustering-centric views, providing a more refined selection mechanism that directly addresses the trade-off between completeness and efficiency.

Finally, the third contribution introduces cognitive and linguistic strategies for scalable concept lattice reduction. Inspired by human language optimization principles, this model employs a finite "vocabulary" of high-frequency conceptual units (kernel concepts) and an injective mapping function to ensure each concept is represented uniquely and meaningfully. By integrating Genetic Algorithms and Simulated Annealing alongside a learning-based module, the model identifies an optimal kernel subset that minimizes total generation cost, a measure reflecting both computational and cognitive resources. This interdisciplinary approach not only reduces lattice size but also aligns the resulting structures with human cognitive processes, making the reduced lattices both computationally feasible and intuitively comprehensible.

Collectively, these three contributions form a coherent research trajectory. Starting from harnessing clustering methods for initial complexity control (KDL and KVL), moving through a frequency- and cost-informed selection of pivotal concepts (KCS), and culminating in a linguistically and cognitively oriented optimization framework, this dissertation offers a comprehensive toolkit for addressing the scalability, efficiency, and interpretability challenges inherent in FCA.

By fusing computational heuristics, cognitive insights, and linguistic principles into the FCA reduction process, this work advances FCA from a theoretically compelling method to a practical, user-aligned analytical framework. It lays the groundwork for broader adoption of FCA in large-scale data analysis, equipping researchers and practitioners with strategies to navigate, understand, and ultimately derive more meaningful insights from complex and voluminous data

1. Background

Research Context

In recent decades, the exponential growth of data across diverse domains such as healthcare, finance, e-learning, and social media has created a need for increasingly sophisticated methods to extract, represent, and interpret meaningful patterns. The convergence of data mining, machine learning, and knowledge engineering has accelerated the search for frameworks that not only handle vast amounts of information efficiently but also facilitate human understanding of underlying structures and relationships.

Among these frameworks, Formal Concept Analysis (FCA) has emerged as a mathematically rigorous and conceptually rich approach for organizing and interpreting complex datasets [1]. FCA operates by mapping data into a concept lattice, a hierarchy of formal concepts that encodes relationships between objects and attributes. This structure supports clustering, knowledge representation, and discovery of dependencies. However, as datasets increase in size and complexity, the derived lattices can grow exponentially. This leads to computational demands and interpretability challenges, limiting FCA's practical use. Large lattices overwhelm analysts and require significant computational resources. In fields like bioinformatics or software engineering, where datasets are vast and evolving, this problem is especially critical [2].

These limitations highlight the urgent need for lattice reduction methods that retain essential structure and information while discarding redundancies. Such methods make FCA more computationally efficient and cognitively accessible.

Problem Statement

Despite FCA's potential, its application is hindered by the exponential growth of formal concepts, resulting in large and complex lattices. These pose challenges in:

- Computation – high runtime and memory requirements.
- Interpretability – dense lattices that are difficult for analysts to navigate.
- Reliability – oversimplification in some reduction methods, which discard critical information.

Existing reduction techniques (frequency-based pruning, abstraction mechanisms, etc.) often lack efficiency, oversimplify the lattice, or fail to improve interpretability. They remain fragmented and domain-specific, without a unified framework that integrates computational optimization, cognitive strategies, and systematic concept selection.

Thus, the core problem is to develop robust, scalable, and cognitively aligned lattice reduction methodologies that:

- Improve computational performance,
- Preserve interpretability and essential relationships, and
- Adapt flexibly to different datasets and application domains.

Research Objectives

The overarching aim of this dissertation is to advance FCA by making lattice reduction more efficient, scalable, and cognitively accessible. The objectives are:

1. Assess limitations of existing reduction methods in scalability, efficiency, and interpretability.
2. Enhance computational efficiency by designing methods that scale to high-dimensional datasets.
3. Preserve structural integrity, ensuring essential hierarchical relationships remain intact.
4. Improve interpretability, using principles inspired by human language efficiency to identify a minimal, expressive set of core concepts.
5. Empirically validate the proposed methods using standardized metrics and representative datasets.

Significance of the Study

This research is significant as it enhances both the theoretical and practical dimensions of Formal Concept Analysis (FCA). Theoretically, it introduces refined reduction methodologies that address longstanding challenges of computational complexity and interpretability, thereby advancing the core understanding of FCA's scalability. Practically, by producing more manageable and cognitively accessible lattices, the work broadens FCA's usefulness across various domains, ranging from knowledge management to data-driven decision-making, enabling clearer insights from large and complex datasets.

2. Related Work on Lattice Reduction

Since its conception by Wille in 1982 [1], FCA has become a powerful framework for knowledge extraction and structural data analysis. However, as FCA applications expanded into domains such as data mining [3], and social network analysis [8], the complexity of the resulting lattices emerged as a central challenge. Large lattices are often computationally demanding and cognitively overwhelming, motivating research into reduction techniques.

Redundancy removal methods eliminate unnecessary objects, attributes, or incidences while ensuring that the reduced lattice remains isomorphic to the original [9]. This includes merging objects with identical attributes or removing reducible attributes and objects [5]. Other strategies, such as the discernibility matrix approach [10], identify minimal subsets of attributes while preserving lattice isomorphism.

Simplification and abstraction methods approximate or restructure lattices to emphasize essential features, even at the cost of information loss. Examples include clustering objects/attributes [11], matrix factorization techniques like SVD [12]. Further innovations involve layered simplifications [13]. A notable abstraction approach is the box lattice [14], which isolates atomistic elements relevant to classification systems.

Selection-based strategies focus on extracting only the most relevant concepts or attributes [15]. Among these, the Iceberg Lattice [16] is one of the most influential. It retains only the "top-most" frequent concepts by applying a minimum support threshold. Formally, for an attribute set $B \subseteq M$, its support is defined as:

$$supp(B) = \frac{|\ B'\ |}{|\ G\ |}$$

where B' is the set of objects possessing BBB. If $supp(B) \geq minsup$, then B is considered frequent, and frequent concepts form the iceberg lattice. The TITANIC algorithm efficiently computes these lattices by pruning search space based on support. In the classic MUSHROOM dataset experiment, the full lattice contained 32,086 concepts, but at 85% support, it reduced to only the most frequent patterns (e.g., veil type: partial, 100%; veil color: white, 97.62%; gill attachment: free, 97.43%). Lower thresholds (70%, 55%) revealed more detailed associations. Thus, the iceberg lattice acts as a multi-resolution tool, balancing scalability and interpretability.

Beyond reduction, iceberg lattices have become central to knowledge discovery, enabling association rule mining, non-redundant bases, and efficient visualization of otherwise intractable datasets. However, they do not consider derivation costs all frequent concepts are equally retained, even if some are structurally more central than others.

Clustering-based approaches have also been explored [17], adapting methods like k-modes and its extensions to categorical data and FCA structures. These aim to group related concepts into fewer representative ones, though challenges remain in integrating FCA's hierarchical nature.

Recent research has highlighted the potential of combining cognitive and linguistic principles such as the principle of least effort and Zipf's law [18], to optimize lattices not only computationally but also cognitively, making them more aligned with human information processing.

3. Foundational Pillars of the Proposed Strategies

As previously surveyed, many concept lattice reduction techniques have pushed the boundaries of FCA's applicability. However, critical limitations remain: existing methods often lack a dynamic understanding of concept interrelations within the lattice and may overlook derivation ease how readily one concept can be derived from another crucial to both algorithmic efficiency and semantic clarity. Clustering-based reductions frequently rely on geometric distance metrics ill-suited to FCA's relational hierarchy, leading to oversimplified or distorted structures. To address these challenges, this dissertation proposes strategies grounded in two core pillars: (1) a kernel concepts framework (KCS) and (2) a Dijkstra-based distance on the concept lattice.

Kernel Concepts in Concept Lattices

A kernel concept in FCA is a strategically chosen formal concept within a concept lattice that serves as a pivotal "building block" for efficiently representing and deriving other concepts. The notion of a kernel concept is introduced in this dissertation as a novel reduction strategy within Formal Concept Analysis. While the idea draws inspiration from clustering principles, particularly the use of centroids in K-means, the kernel concept framework uniquely adapts this principle to the structure of concept lattices. In contrast to frequency-only reductions such as iceberg lattices [16], kernel concepts combine structural centrality, frequency, and derivation cost into a unified optimization model. Kernel concepts are deliberately selected based on

additional criteria to minimize overall complexity. Typically, these criteria involve Frequency (how often or how prominently a concept appears) and Derivation Cost (the effort required to derive one concept from another).

Formally, let C be the set of all formal concepts and $C_M \subset C$ a subset limited by size (capacity) $| C_M |= S_c$. Each concept in C_M is a kernel concept. The selection minimizes:

$$min\{\sum_{c \in C} f(c)\, d(C_M, c) |\, C_M \subset C\, , |C_M| = S_c\},$$

where:

$f(c)$ indicates how "valuable" or "frequent" a concept c is,

$d(C_M, c)$ is the minimal cost to derive concept c from any concept in the kernel set C_M. Thus kernels act as anchor points (centroids) that can approximate or generate all other concepts with minimal overall cost. Kernel concepts serve as the structural backbone of large lattices; they reduce computational cost by enabling on-demand derivation of non-kernel concepts; they provide a balanced criterion beyond frequency alone by incorporating derivation effort; they improve interpretability and usability by offering navigable anchor points; and they facilitate downstream analysis (e.g., rule bases, queries) without storing the full lattice.

Dijkstra's Algorithm on Concept Lattices

Dijkstra's algorithm (1959) computes shortest paths in directed, weighted graphs and is adapted here to measure structure-aware distances within the concept lattice. In this adapted view, vertices are formal concepts and edges represent hierarchical relations (e.g., the partial order ≤); weights encode direction (up/down in the lattice), so path costs reflect not only frequency but also hierarchical effort.

The symbol $\mathcal{B}(C, <)$, and its corresponding graph $\mathcal{H}(C, E)$, where C represents the set of formal concepts and E denotes the edges signifying hierarchical relationships. Let C_s and C_e be two distinct formal concepts in C, with C_s serving as the starting point and C_e as the endpoint for the path calculation. Each concept $c \in C$ has an associated cost $d(c)$ that represents the cost of reaching c from C_s. To differentiate the directionality of traversal along the lattice edges, two cost parameters are defined: "UpCost" for moving from a concept to a more specific (child) concept, and "DownCost" for moving from a concept to a more general (parent) concept.

Within this framework, the Dijkstra-based distance measure relies on a priority queue Q, implemented as a min-heap keyed by $d(c)$, and a set V tracking visited nodes. The cost function $f: C \times C \rightarrow \mathbb{R} \cup \{\infty\}$ evaluates the cost of moving from one concept c to an adjacent concept c′ based on their relation:

$$f(c, c') = \begin{cases} UpCost, & if\ c \sqsupseteq c', \\ DownCost, & otherwise. \end{cases}$$

Combining these costs over a sequence of concepts forms the basis for calculating the shortest path. Thus, for all paths $(c_1, c_2, \ldots, c_n)$ from C_s to C_e, the Dijkstra-based distance measure $d(C_s, C_e)$ selects the path with the minimal cumulative cost:

$$d(C_s, C_e) = min\left\{ \sum_{i=1}^{n-1} f(c_i, c_{i+1}) \mid (c_1, c_2, \dots, c_n) \text{ is a path from } C_s \text{ to } C_e \right\},$$

Here, the measure $d(C_s, C_e)$ represents the minimal cost required to navigate the lattice from the starting concept C_s to the target concept C_e, effectively encapsulating both the structure of the concept lattice and the directional constraints inherent in the data's hierarchy.

4. Clustering-Based Reduction Strategies for FCA

K-means Dijkstra on Lattice (KDL)

The K-means Dijkstra on Lattice (KDL) method extends conceptual clustering to categorical data by integrating FCA with a customized Dijkstra algorithm. This approach leverages the hierarchical structure of concept lattices to ensure that clustering respects semantic relationships.

The process begins with converting categorical data into a formal context, represented as a binary incidence matrix. FCA then derives all formal concepts, forming a hierarchical lattice that captures attribute-object relationships. To compute distances, directional edge weights are assigned within the lattice e.g., downward transitions may cost more than upward ones. These weights are used in a modified Dijkstra's algorithm to determine the shortest paths (conceptual distances) between nodes.

Cluster centers, or kernel concepts, are formal concepts that minimize intra-cluster distances. Given a cluster $S = \{c1, \dots, c_{|S|}\}$, the centroid Z is defined as:

$$Z = argmin_{Z \in S}\left(\sum_{i=1}^{|S|} d(c_i, Z) \right).$$

where (c_i, Z) is the Dijkstra-based distance. This iterative process continues until centroids stabilize.

A key strength of KDL lies in the connectivity of the concept lattice: every concept pair is reachable via some path, ensuring the feasibility of distance computation and enabling high-quality, interpretable clustering. The method efficiently identifies representative concepts that preserve both structure and semantics.

K-Means Vector on Lattice (KVL)

The K-means Vector on Lattice (KVL) method transforms categorical data, originally structured as formal concepts, into **concept description vectors** that enable the use of classical numerical clustering. Each concept is represented as a vector where each dimension

corresponds to an attribute; attributes in the intent are given value 1, while others receive their average frequency across all objects.

Definition (Concept Description Vector):

For a concept $c = (X, Y)$ in context $= (G, M, I)$, with $| M |= q$ attributes and $| G |= r$ objects, the description vector is:

$$c_Y = (v_{m_1}, v_{m_2}, \ldots, v_{m_q})$$

with

$$v_{m_h} = \begin{cases} 1 & if\ m_h \in B, \\ \frac{1}{r}\sum_{j=1}^{r} I(g_j, m_h) & if\ m_h \notin B, \forall\ g_j \in G, \end{cases}$$

This ensures attributes in the intent are fully represented, while others capture their dataset prevalence (see Table 1).

Table 1. Matrix Corresponding to The Relation I

Objects/Attributes	m_1	m_2	…	m_q
g_1	$I(g_1, m_1)$	$I(g_1, m_2)$	…	$I(g_1, m_q)$
g_2	$I(g_2, m_1)$	$I(g_2, m_2)$	…	$I(g_2, m_q)$
…	…	…	…	…
g_r	$I(g_r, m_1)$	$I(g_r, m_2)$	…	$I(g_r, m_q)$

Definition . Concept Similarity (CS):
Let

$$V_{c_1} = \left(V_{c_1 m_1}, V_{c_1 m_2}, \ldots, V_{c_1 m_q}\right).$$

and

$$V_{c_2} = \left(V_{c_2 m_1}, V_{c_2 m_2}, \ldots, V_{c_2 m_q}\right).$$

be the concept description vectors of two distinct concepts c_1 and c_2. The Euclidean distance, which serves as the basis for CS, is given by:

$$CS(V_{c_1}, V_{c_2}) = \sqrt{\left(V_{c_1 m_1} - V_{c_2 m_1}\right)^2 + \left(V_{c_1 m_2} - V_{c_2 m_2}\right)^2 + \ldots + \left(V_{c_1 m_q} - V_{c_2 m_q}\right)^2}.$$

Armed with the concept description vectors and the associated similarity measure, we can apply the classical k-means clustering algorithm. In this process, each concept description vector is treated as a data point in a q-dimensional space. The algorithm groups these vectors into k clusters such that concepts within the same cluster share greater similarity than those in

different clusters. Each cluster has a centroid Z_i, defined as the mean of all concept description vectors assigned to that cluster:

$$Z_i = \frac{1}{|S_i|}\sum_{j=1}^{|S_i|} V_{Y_j}, V_{Y_j} \in S_i.$$

where S_i is the set of concept description vectors in the i-th cluster.

The objective of k-means is to minimize the within-cluster sum of squared distances (WCSS) from each concept description vector to its corresponding centroid:

$$\mathrm{Q} = \sum_{i=1}^{k}\sum_{j=1}^{|S_i|} ||V_{Y_j} - Z_i||^2,$$

where:

- S_i is the set of concept description vectors assigned to the *i-th* cluster,
- Z_i is the centroid of cluster i, defined as the mean of all vectors in S_i, and
- $\|\cdot\|$ denotes the Euclidean norm.

By repeatedly assigning vectors to their nearest centroids (based on the CS measure) and then recalculating the centroids, the algorithm proceeds until it converges to a stable configuration, thereby optimally partitioning the concept vectors into coherent, meaningful clusters.

Clustering Algorithm

The clustering procedure unfolds as follows. Consider a formal context $T = (G, M, I)$ and let $V(T)$ represent the set of all derived concept description vectors. Suppose we aim to form K clusters. Initially, randomly select K initial centroids, $Z_t^0 = (A_t, B_t)$ for $(t = 1,2,\ldots K)$, each corresponding to a preliminary cluster $S_t^0 = \{Z_t^0\}$.

Next, assign each concept description vector $v \in V(T)$ to the cluster whose current centroid is nearest to v based on the chosen distance measure. After this initial assignment, recompute each cluster's centroid by taking the average of all vectors assigned to it, thereby updating each cluster center.

This reassignment and centroid calculation process is repeated iteratively. In each iteration, vectors may shift clusters if doing so reduces the overall clustering cost. The process continues until the cluster memberships and their centroids remain stable across consecutive iterations, indicating that the algorithm has converged. The algorithm steps are as follows:

Algorithm 2. K-means clustering of concepts

Input: All the description vectors of concepts in $V(T)$, K.
Output: The clusters and corresponding centers.
Initialize:
Set $S_1^i \leftarrow \emptyset$, $S_2^i \leftarrow \emptyset$, ..., $S_k^i \leftarrow \emptyset$;
$i \leftarrow 0$,

Select initial center vectors of K clusters: $Z_1^i, Z_2^i, \ldots, Z_k^i$;

Assignment:

For each $v \in V(T)$ *do*:

-Find t such that $CS(distance)\left(v, Z_t^i\right) \leq CS(distance)\left(v, Z_j^i\right),\ (j = 1,2, \ldots, k)$ then,

$v \in S_t^i$;

EndFor

Centroid Update:

For each S_t^i *do*:

$Z_t^{i+1} = \frac{1}{|S_t|} \sum_{s=1}^{|S_t|} v_s\ , v_s \in S_t$

$S_t^{i+1} = \left\{v \in V(T) | CS\left(v, Z_t^{i+1}\right) \leq CS\left(v, Z_j^{i+1}\right)\right\}$

EndFor

Convergence Check:

If $Z_t^i = Z_t^{i+1}$, $S_t^i = S_t^{i+1}$, $t = 1,2, \ldots, K$, *then*

Go to "Stop and output the clusters".

Else:

$i = i + 1$,

Go to "Repeat the assignment step".

Output: clusters $S_1^i, S_2^i, \ldots, S_k^i$ and the corresponding centers $Z_1^i, Z_2^i, \ldots, Z_k^i$.

Once the clustering process is complete and stable clusters are formed, the concept description vectors in each cluster can be mapped back to their corresponding original concepts from the formal context. This backward mapping leverages the initial construction of concept description vectors, ensuring that the clustering results can be interpreted and analyzed in terms of the actual concepts they represent.

Algorithm 3: Mapping Description Vectors Back to Original Concepts

Input: The clusters $S_1^i, S_2^i, \ldots, S_k^i$ and the corresponding centers $Z_1^i, Z_2^i, \ldots, Z_k^i$.
Output: Clusters of original concepts.
Initialize:
For each $t = 1$ to K, set $NS_t = \emptyset$,
Mapping:
For each vector $v \in S_t^i$:

- Retrieve the corresponding original concept c associated with vector v
- Add concept C to NS_t

Output: the new clusters $NS_1, NS_2, \ldots NS_k$, each containing the original concepts.

This approximation and mapping technique enables efficient and interpretable clustering of concepts within a given context, thereby clarifying the intricate relationships and similarities among the different concepts.

Experimental Results

The experiments evaluated the effectiveness and scalability of the Dijkstra-Based Distance Measure and the two clustering approaches: K-means Dijkstra on Lattice (KDL) and K-means Vector on Lattice (KVL). All algorithms were implemented in Python (3.11) with NetworkX, scikit-learn, and Matplotlib. Formal concepts were generated using a tailored NextClosure routine and lattices were constructed with iPred.

Results show that runtime grows with lattice size (Figure 1), while mean distance behaves non-linearly, sometimes peaking in mid-sized, fragmented lattices (Figure 2). Real datasets revealed similar patterns (Figures. 3– 4):

- Car Evaluation: large lattice but shorter mean distances due to dense interconnections.
- Balance-Scale & Breast Cancer: fewer concepts but longer paths, reflecting fragmented lattices.

These findings confirm that the Dijkstra-based measure captures structural coherence**,** density effects**,** and topological differences more robustly than Euclidean alternatives.

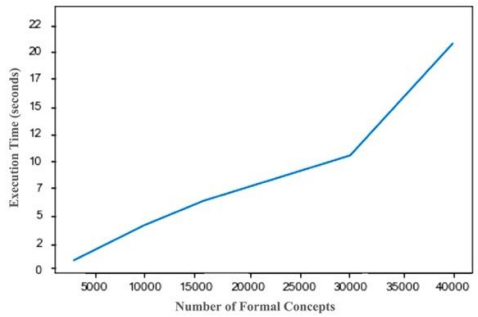

Figure 1. Average Runtime vs. Lattice Size for Random Contexts

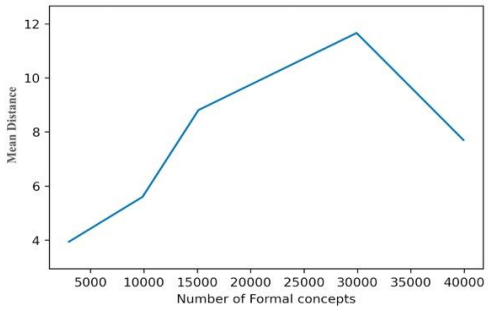

Figure 2. Mean Distance vs. Lattice Size for Random Contexts

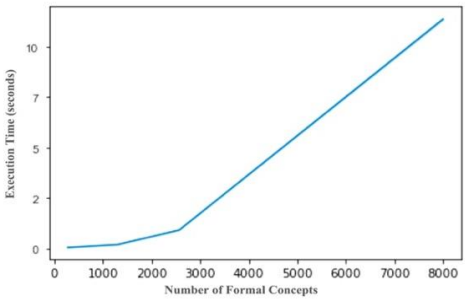

Figure 3. Average Runtime vs. Lattice Size for Real-World Datasets

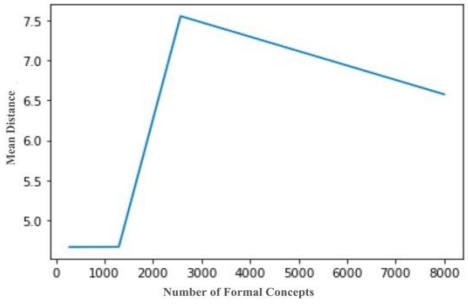

Figure 4.. Mean Distance vs. Lattice Size for Real-World Datasets

Cluster quality was evaluated with the Silhouette Coefficient and Davies–Bouldin Index (DBI)**.**

- KDL consistently outperformed KVL: higher Silhouette scores (Figure 5) and lower DBI values (Figure 6).
- Example: On Car Evaluation, KDL achieved a Silhouette of 0.563 vs. 0.106 for KVL.
- This demonstrates that KDL better preserves conceptual structure, while KVL's vector simplification sacrifices interpretability.

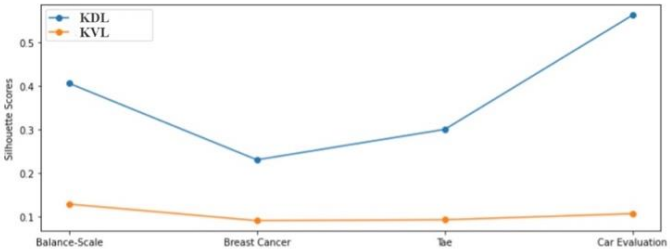

Figure 5. Silhouette Scores by Dataset and Method

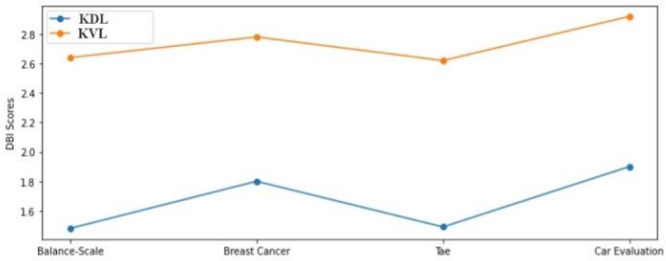

Figure 6. DBI Scores by Dataset and Method

Scalability was examined from two perspectives. First, when varying the number of clusters from 2 to 18 on the Car Evaluation dataset, the KVL method exhibited almost linear growth, with runtimes remaining efficient between 44 and 52 seconds (Figure 7). In contrast, the KDL method showed a steep increase, rising from approximately 1,900 seconds for 2 clusters to nearly 49,600 seconds for 18 clusters (Figure 8). Second, scalability was tested with respect to the number of formal concepts across four datasets (Balance-Scale, Breast Cancer, Tae, and Car Evaluation). Here, KVL maintained stable runtimes in the narrow range of 43–46 seconds, demonstrating strong scalability. On the other hand, KDL performed poorly, with execution times escalating rapidly and surpassing 2,000 seconds for 8,001 formal concepts (Figures. 9–10).Summary

- KDL yields conceptually richer clusters by fully exploiting lattice structure, but at high computational cost.
- KVL offers strong scalability and efficiency, but loses hierarchical nuance.
- Both methods are viable FCA reduction strategies depending on whether the goal is conceptual fidelity (KDL) or scalability (KVL).

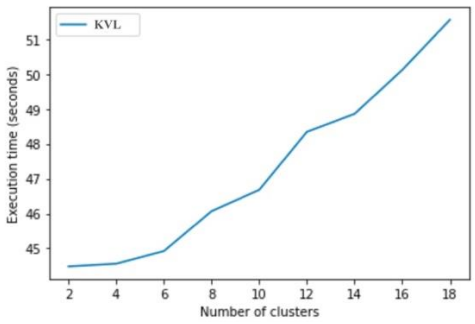

Figure 7. KVL Scalability vs. Cluster Count (Car Evaluation Dataset with 8001 Concepts)

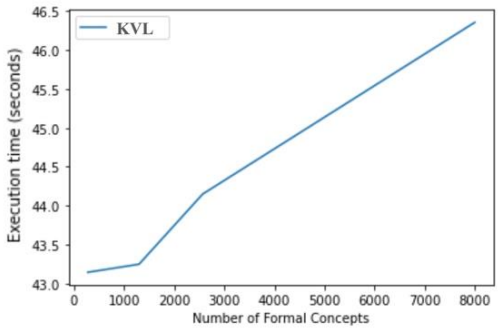

Figure 8. KVL Scalability with an Increasing Number of Formal Concepts

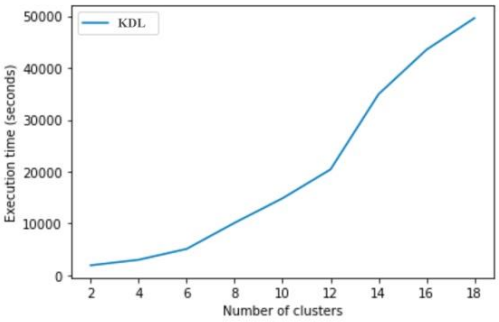

Figure 9. KDL Scalability with Increasing Number of Clusters

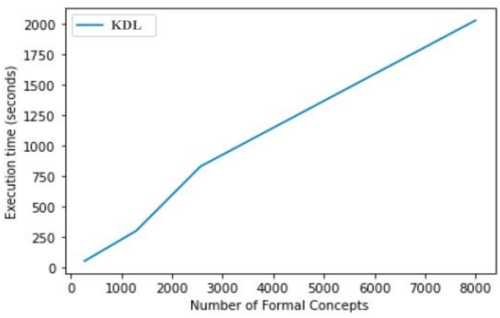

Figure 10. KDL Scalability with Increasing Number of Formal

5. Kernel Concepts Selection for Efficient Lattice Reduction

FCA provides a powerful framework for conceptualization, its derived concept lattices can become unwieldy, limiting both scalability and insight. Traditional approaches to simplifying these lattices, be they the removal of redundant elements, structural simplifications, or selective filtering, can still struggle to accommodate the dynamic, complex nature of many real-world datasets.

.

Kernel Concept Set Approach

The Kernel Concept Set (KCS) method addresses the inherent complexity of concept lattices in FCA, particularly when managing extensive lattices where conventional techniques, such as removing arbitrary elements or selecting objects ad hoc, may overlook critical structures. KCS focuses on two core attributes of each concept: its frequency and the cost of deriving one concept from another. Frequency gauges a concept's prevalence and importance in the dataset, while the derivation cost assesses the effort required to navigate between concepts in the lattice.

Central to KCS is the idea of identifying "kernel concepts," high-frequency concepts strategically positioned in the lattice. By singling out these pivotal elements, KCS preserves both structural coherence and meaningful data relationships during lattice simplification. Furthermore, KCS employs a flexible derivation cost function to measure similarity, thereby accommodating both the real-world usage level of concepts and their internal structure. This dual perspective enriches analysis by spotlighting concept clusters and pinpointing the most essential information within the lattice.

In addition, KCS treats kernel concepts as cluster centroids, making it a powerful clustering approach for formal concepts. This strategy operates in a general metric space, avoiding the need for a vector space, and can yield cost savings relative to typical agglomerative methods. Crucially, KCS not only isolates cluster members but also designates central concepts as cluster representatives, highlighting the lattice's crucial "backbone." Consequently, the KCS method offers a balanced, efficient means to reduce and interpret large FCA lattices while protecting the most valuable insights embedded in the data.

Building upon the standard concept lattice model described in Definition 2.5, the Extended Concept Lattice introduces additional elements to enrich FCA. Specifically, this extension incorporates two core components:

- *A* Frequency Value function, reflecting how often each concept appears or how central it is within the dataset.
- A Derivation Cost function, quantifying the cost or complexity of reaching one concept from another within the lattice's structure.

Definition 5.5 (Kernel Concept Set).

An extended lattice $\mathfrak{B}(d, f, d^f)$ uses these components to identify a *Kernel Concept Set* K_s that satisfies the following:

- Capacity Constraint:

  $|K_s| = S_c$, where S_c is a predefined size limit.

- Optimization Constraint:

  K_s should minimize the cumulative derivation cost across the lattice. Formally:

  $$K_s = argmin_{K_s \subset K}\{ \sum_{c \in K} d^f(K_s, c) | \, |K_s| \leq S_c \}.$$

  This enforces an optimal coverage of the concept set using only S_c kernel concepts.

- Role in Lattice Simplification:

  By focusing on these kernel concepts which both appear often (high frequency) and are strategically positioned (low derivation cost) the approach zeroes in on the lattice's structural "backbone." It thereby condenses the lattice into its most informative subset, enhancing manageability and preserving core relationships during analysis.

Overall, these definitions provide a systematic framework for extending an FCA concept lattice with frequency-based prioritization and cost-aware navigation, enabling more powerful reduction, clustering, and insight extraction in complex or large datasets.

Optimized Greedy Algorithm for Determining a Kernel Concept Set

Optimized Greedy Algorithm (Algorithm 4) efficiently identifies a Kernel Concept Set (KCS) by selecting a subset of pivotal concepts that minimize total derivation costs across the lattice. This reduces lattice size and complexity while preserving key structural insights and interpretability.

Algorithm 4: Optimized Greedy Algorithm

Input:

- Concept Lattice $\mathfrak{B}$ *(K, ≤)*
- Frequency Value Function $f: C \rightarrow R^+$
- Maximum Core Set Size S_c
- Transition Cost: $upward \leftarrow 2, downward \leftarrow 1$

Output:

- Kernel Concept Set K_s

Algorithm Steps:

1. Initialization:
   - Construct the Concept Lattice $\mathfrak{B}(C, \leq)$.

- Initialize Kernel Set K_s as an empty set.
- Assign Frequency Values $f(c)$ to each concept c in the lattice.

2. Ancestors and Descendants Preprocessing:
   - For each concept c in the lattice, identify its ancestors and descendants.
   - Prepare a memoization dictionary to store the minimal derivation costs.
3. Derivation Cost Calculation:
   - For each concept c in the lattice:
     - Use Dijkstra's algorithm to calculate the minimal derivation cost $d\ (K_s, c)$ to every other concept.
     - Store the costs in a structured way for quick retrieval and use memorization to avoid redundant calculations.
4. Core set identification with Sub-Lattice Optimization:
   - Define S_c as the maximum size for the Kernel set.
   - Initialize
     $best\_cost \leftarrow \infty$, *best_candidate* ← None.
   - Iteratively expand K_s:
     - For each candidate concept not in K_s, construct or retrieve a relevant sub-lattice Algorithm 5.2.
     - Calculate the potential reduction in aggregated derivation cost if the candidate were added to K_s.
     - Update $best\_cost$ and *best_candidate* accordingly.
     - Add the *best_candidate* to K_s and update the cost.
     - Continue until $|K_s|=S_c$ or no further reduction in cost is possible.
5. Result Analysis:

Return the final K_s as the kernel concept set that minimizes the aggregated derivation cost while adhering to the size constraint $|K_s|=S_c$.

Algorithmic routines such as sub-lattice construction (presented in Algorithm 5) are crucial for reducing the size of the problem space:

1. Defining the Sub-Lattice
   - Identify a compact subset of concepts (and their interconnections) directly relevant to the current calculation.
   - This subset often centers on the target concept(s) and the kernel set members.
2. Selective Inclusion
   - Only nodes (concepts) and edges (relationships) pertinent to the cost evaluation or kernel set update are included, minimizing overhead.
3. Dynamic Construction
   - As the algorithm updates the kernel set or refines potential candidates, sub-lattices are rebuilt or adjusted to ensure accuracy and relevance.
4. Scalability
   - By confining computations to smaller sub-lattices, the method accommodates lattices of larger overall size without incurring prohibitive computational costs.

Algorithm 5: Steps for Building a Sub-Lattice

1. Initialize Relevant Concepts:
   - Start with an empty set to hold all relevant concepts.
   - Add the two concepts, A and B, to the relevant concepts set.
2. Add Ancestors and Descendants:
   - Include all ancestors of A into the relevant concepts set.
   - Include all descendants of A into the relevant concepts set.
   - Repeat the process for node B, adding both its ancestors and descendants to the relevant concepts set.
3. Create Sub-Lattice:
   - Initialize an empty dictionary to represent the sub-lattice.
   - For each concept in the relevant concepts set, do the following:
     - Initialize an empty list to store the neighbors of the concept.

- Retrieve the list of neighbors from the full lattice dictionary.
- Include a neighbor in the concept's neighbor list only if the neighbor is also in the relevant concepts set.

- Assign the neighbor list to the concept in the sub-lattice dictionary.

4. Return Sub-Lattice:
   - The sub-lattice containing only the relevant concepts and edges is now constructed.
   - Return the sub-lattice dictionary.

By applying these optimization methods, the algorithm strategically narrows the scope of its computations while still preserving a comprehensive view of the lattice. This balanced approach results in a kernel set that is both cost-effective and representative, exemplifying how depth and breadth can be maintained in the analysis of large and intricate concept lattices.

Experimental Setup and Methodology

Implementation used Python on macOS (Apple M1, 8 GB RAM, macOS 14.3.1). Experiments evaluate KCS against K-means Dijkstra on Lattice (KDL) using four real-world datasets (see Table 2 in your thesis). We assess clustering without labels using Silhouette Coefficient and Davies–Bouldin Index (DBI).

Across all datasets, KCS consistently surpasses KDL with higher Silhouette and lower DBI, indicating tighter, better-separated clusters. KCS achieves, for example, 0.406 (Balance-Scale) and 0.680 (Car Evaluation) in Silhouette, and 1.72 and 1.41 in DBI, respectively. See Figure 11 (Silhouette) and Figure 12 (DBI) for visual comparison.

Why KCS wins**.** KCS centers clusters around kernel concepts chosen by both frequency and derivation cost, capturing the lattice's essential structure. It operates in a general metric space (no vector conversion), reduces overhead compared to some traditional approaches, and directly yields cluster hubs (centroids) and memberships.

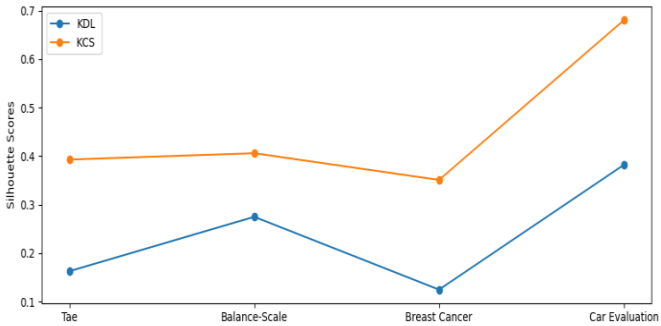

Figure 11. Silhouette Scores by Dataset and Method.

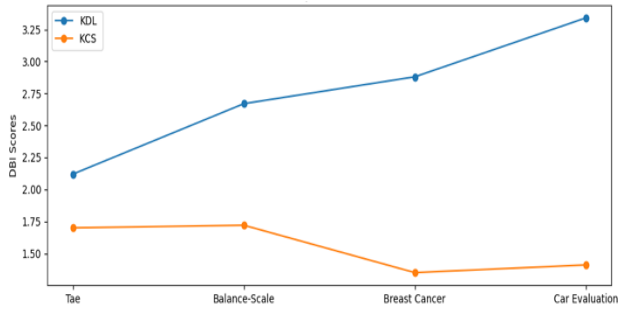

Figure 12. DBI Scores by Dataset and Method

Using the datasets in Table 2, we compared KCS vs. KDL runtimes as lattice size grows. Figure 13 shows a marked difference: KDL is acceptable on modest lattices but scales poorly, while KCS maintains strong efficiency across sizes. Examples: Tae (276 concepts) KDL: 1210.14 s vs. KCS: 9.35 s; Car Evaluation (3596 concepts) KDL: 781,799.93 s vs. KCS: 8,361.93 s.

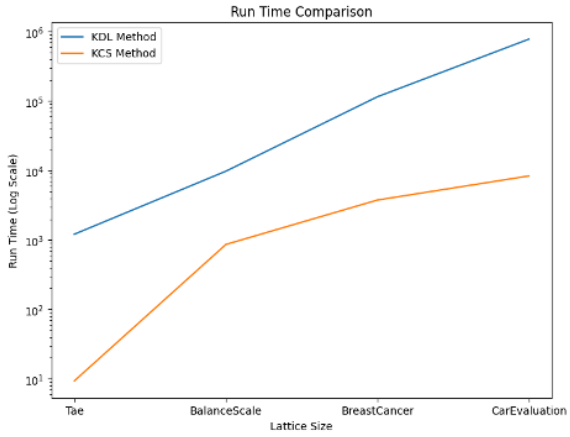

Figure 13. Comparative Performance Analysis of KCS and KDL Methods Across Diverse Lattice Sizes

Experiment with the Teaching Assistant Evaluation Dataset

The TAE dataset (UCI KDD) records 151 TA assignments with six categorical attributes (language background, instructor, course type, semester, class size). Transformed to Boolean form, the formal context has 151 objects × 101 attributes at 0.05 density. Table 5 shows a 10×8 subset; Figure 14 displays the line diagram of the resulting lattice.

Table 5. Formal Context about Subset of Tas Dataset.

	Class_Size_17	Eng_Nat_spk_1	Eng_Nat_Spk_2	Summer_or_Regular_1	Summer_or_Regular_2	Course_3	Course_Instructor_13	Course_Instructor_23
TA 1	X	X		X		X		X
TA 2	X	X			X	X		X

TA 3	X		X		X	X	X	
TA 4	X		X	X		X		X
TA 5	X	X			X	X	X	
TA 6	X		X		X	X		X
TA 7	X		X		X	X		X
TA 8	X	X		X		X	X	
TA 9	X	X		X		X	X	
TA 10	X	X		X		X	X	

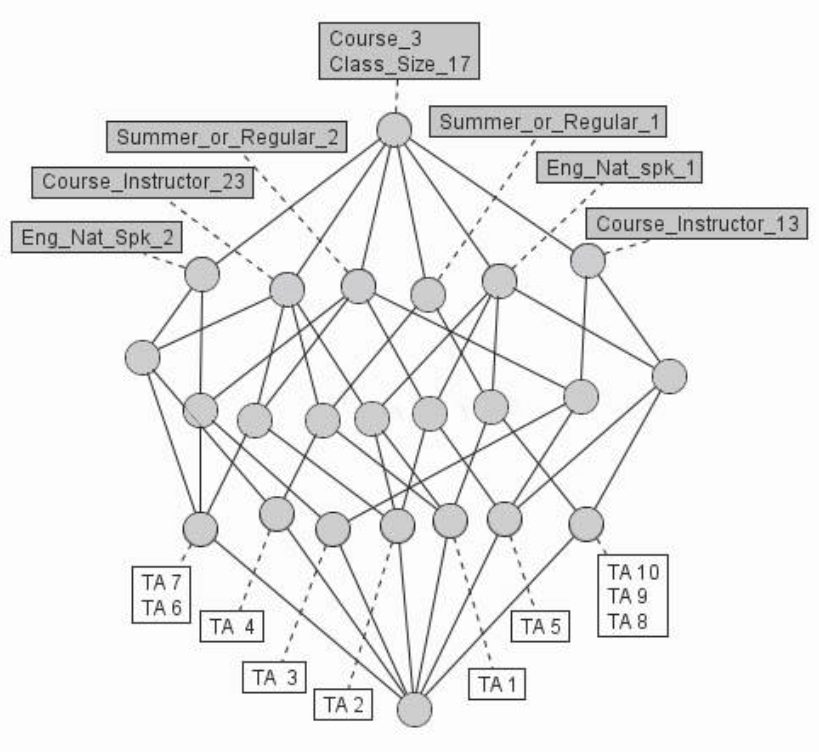

Figure 14. Concept Lattice Derived from the Formal Context of Tae Dataset Table 4.

Applying KCS:

- With S_c=5, KCS selects 14 kernel concepts, total derivation cost 30,808. Two concepts alone cover 138/151 TAs, surfacing patterns such as non-English-speaking TAs in regular semesters.
- Increasing to S_c=8% retains those 14 and adds 8 more (22 kernels), reducing cost to 26,768 and revealing finer structure (class sizes, course types, language patterns). • As S_c grows from 5% → 20%, cost decreases monotonically (30,808 → 16,132) while preserving a streamlined structure. See Figure 15.

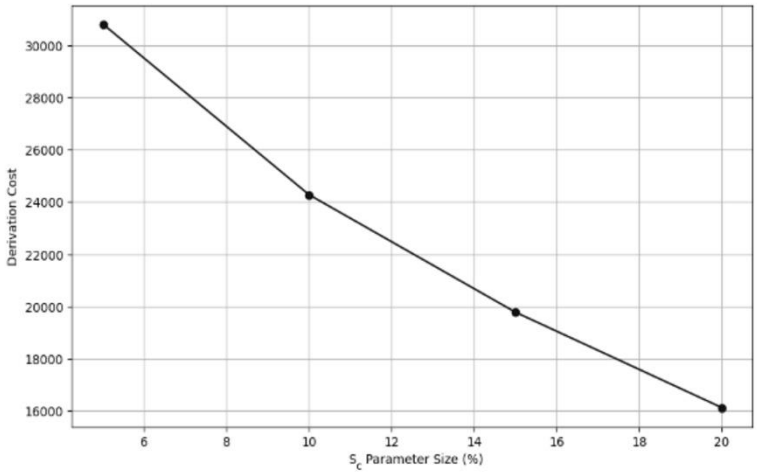

Figure 15. Trend of Decreasing Derivation Cost with Incremental Expansion of Kernel Set Size (S_c)

6. Mining Kernel Concepts: A Cost-Optimized Concept Set Generation Method

This section introduces a new framework for concept lattice reduction, focusing on an optimal balance between expressive power and computational feasibility. Unlike conventional methods that emphasize frequency filters or attribute-based pruning, our model employs a heuristic and machine learning–assisted strategy to pinpoint a small "kernel" of high-frequency concepts. These selected kernel concepts form a finite memory structure, with a specialized mapping function ensuring each concept is uniquely and transparently represented. The method is further bolstered by a Genetic Algorithm (GA) tasked with optimizing the kernel selection, aiming to minimize a global generation cost while preserving lattice integrity. Extensive tests confirm that our GA-based approach outperforms a benchmark Simulated Annealing method in both speed and scalability. The chapter also demonstrates a linguistic-based cost model for defining kernel vocabularies, showcasing the versatility of our solution for diverse contexts and data domains. Our Main Contributions:

- Development of a Novel Reduction Model: We introduce a mechanism that integrates a derivation cost function with a robust optimization procedure, enabling the construction of a simplified yet expressive concept lattice.
- Genetic Algorithm with Machine Learning Support: A neural network module predicts chromosome segment fitness, generating an efficient starting population for the GA, thus enhancing convergence speed.
- Flexible Probability Distribution for Concept Prioritization: Our system accommodates various probability distributions $P(s)$ across concepts, enabling tailored solutions in domains with different analytical requirements.
- Injective Mapping Function: By ensuring each concept is encoded as a unique word sequence, the mapping function prevents ambiguity and preserves clarity during lattice reduction.

Our approach provides multiple benefits that significantly improve both the scalability and usability of FCA:

- Scalability: Adjustable kernel concept selection through input parameters allows users to generate compact or more expansive concept sets, matching specific data complexity.
- Approximation of Full Lattice: The resulting kernel concepts effectively approximate the entire concept lattice, retaining crucial relational patterns while minimizing overall complexity.
- Enhanced Clarity: The injective mapping function, coupled with the kernel's high-frequency elements, yields a more interpretable representation of concepts.
- Cognitive Alignment: Aligning the reduced structure with linguistic and cognitive principles lowers the mental overhead for understanding and navigating the lattice.
- Adaptability: Configurable memory sets W_M and selection thresholds facilitate broad adaptability across various domain-specific vocabularies and semantic demands.

Against this backdrop, the following sections detail the design of our reduction method, elaborate on the Genetic Algorithm for kernel concept selection, and evaluate the resulting model through comprehensive experiments.

Proposed Method

To systematically reduce a concept lattice while maintaining both expressiveness and derivational efficiency, we propose selecting a targeted kernel subset of concepts. Guided by the compactness and clarity inherent in human language, our method relies on a finite "memory" of frequently used concepts, applies an injective mapping function to guarantee a unique representation for each concept, and utilizes optimization algorithms focused on minimizing overall generation cost. By aligning with cognitive and linguistic principles, this strategy not only streamlines computational tasks but also enhances the interpretability and practical utility of the resulting lattice.

We begin by assigning a probability value to every concept in the concept lattice $L = (C, \leq$), These probabilities form a distribution $p: C \rightarrow [0,1]$ such that

$$\sum_{c \in C} p_c = 1.$$

Each probability reflects how frequently a given concept is used. For instance, the concept "bread" is typically used more often than "petrichor." In addition to the concept lattice, this probability distribution serves as an integral part of the input data.

The first step in reducing the concept set relies on probability-based filtering. Specifically, we introduce a probability threshold p_F. Any concept whose probability value is below this threshold is removed from consideration, leaving us with the set of frequent concepts,

$$C_F = \{c \in C \mid p(c) \geq p_F\}.$$

Note that, in general, C_F does not form a lattice. From C_F, we select a finite subset of concepts, known as the kernel C_M,

$$C_M = \{C_{M,1}, C_{M,2}, \dots, C_{M,D}\} \subset C_F, .$$

where D is the size of the kernel set. This finite size is a key attribute: it is chosen based on the specific requirements of an application and the limitations of available resources, thereby ensuring representations that are both scalable and manageable. Moreover, the kernel set's properties help guarantee its effectiveness and dependability in the model.

The kernel concepts act as special cluster centroids within the target concept set. Clustering, commonly employed in data analysis, reduces data volume such that subsequent analyses can target whole clusters rather than individual items, thereby optimizing resource usage. In particular, conceptual clustering refines standard clustering methods (like k-means or hierarchical agglomerative clustering) to work with semantic concept domains. In this study, we use an evolutionary strategy to optimize the positions of the cluster centers.

One application of this kernel concept model lies in refining linguistic concept representations. In the language model considered here, each kernel concept corresponds to a single word in the available vocabulary, each of these words is a single-word linguistic unit that forms the foundation for representing the broader set of concepts.

Kernel Selection Method

Given a kernel set C_M, we define a cost function h_{C_M}:

$$h_{C_M}: \mathrm{C} \rightarrow \mathbb{R}^+ \ .$$

where,

$$h_{C_M}(c) = g(\{d(c_k \in C_M, c\ )\}).$$

where $d(c_k, c\ )$ represents the cost of deriving a representation of c from c_k, and g is a function applied to the set of these distances. A common choice for g is the min function. The main objective is to identify the kernel that minimizes the overall mapping costs, which is calculated as

$$h(C_M) = \sum\nolimits_{c \in C} p_c \, h_{C_M}(c).$$

Additionally, there is a constraint on the size of the kernel set:

$$|C_M| \ \leq K.$$

where K is a predefined integer. Minimizing $h(C_M)$ by optimally determining the kernels C_M is the core goal. Through this approach, we significantly enhance FCA by reducing the complexity of the concept lattice via a careful selection of key concepts. This, in turn, supports more efficient knowledge representation and further broadens the potential applications of FCA across various complex domains.

If, in a particular case, $h(C_M)$ is defined as the sum of element-wise costs

$$h(C_M) = \sum\nolimits_{c_k \in C_M} d(c, c_k).$$

and taking the following weight value:

$$w_c = 1,$$

the problem becomes analogous to the well-known knapsack problem. Specifically, if we use an indicator variable x_i to denote whether a concept c_i is part of the kernel, then the cost function can be expressed as:

$$h(C_M) = \sum_{c \in C} p_c \sum_{i \in C} d(c,i)\, x_i = \sum_{i \in C} (d(c,i) \sum_{c \in C} p_c)\; x_i = \sum_i v_i x_i,$$

with a capacity constraint of

$$\sum_{i \in C} w_i x_i \leq K.$$

Since the knapsack problem is NP-complete, we rely on heuristics—Genetic Algorithm (GA) and Simulated Annealing (SA)—to optimize kernel selection. GA efficiently searches for near-optimal subsets by balancing exploration with refinement, while SA explores complex spaces through probabilistic acceptance of neighbor solutions, enabling escape from local optima and improved results.

Practical Application in Word-Level Concept Representation

In natural language, we use words to describe the concepts that exist in our world. However, it is evident that not every concept has a dedicated single word; many concepts require more elaborate descriptions to differentiate them. In this context, words that function as "identifiers" can be thought of as memory, or kernel concepts. For other concepts, we often rely on a combination of these memory words when referring to them in conversation. Together with the kernel set, these additional concepts form the set C_F. As for any remaining concepts, we do not assign them separate expressions for unique identification. In this work, we utilize the kernel concept set mining algorithm to tackle the problem of selecting an optimal vocabulary.

To formalize this, let f be the mapping function that represents concepts at the word level:

$$f: C_F \rightarrow W^*.$$

where W^* is the set of all possible word sequences constructed from a finite collection of words W. The pool W includes the words corresponding to the kernel concepts; we denote W_c as the word linked to a specific kernel concept c.

Concerning the cost function h_{C_M}, we take a straightforward approach:

$$h_{C_M}(c) = |f(c)|,$$

where $|f(c)|$ indicates the length (in words) of the representation of concept c. Therefore, for every $c \in C_M$, we have

$$h_{C_M}(c) = 1.$$

If we assume C_M includes all attribute concepts $c_a = (\{a\}'', \{a\}')$ and

$$\forall a \in M: \{a\} = \{a\}'',$$

then we can specify a unique word-level representation:

$$f(c) = \{f(c_k)\} \cup \{f(c_a) | a \in attr(c) \setminus attr(c_k)\} = W_{c_k} \cup \{W_{c_a} | a \in attr(c) \setminus attr(c_k)\}.$$

where c_k denotes the nearest kernel concept to c, and $attr(c)$ is the set of attributes (the intent) of c.

Proposition 1

The above mapping function guarantees an unambiguous representation at the word level.

Example 1

For illustration, consider the Live in Water ontology provided at: https://upriss.github.io/fca/examples.html. This ontology includes 18 concepts in total. Their frequencies are compiled in Table A.4 of Appendix A, and the frequency threshold is set at 0.4. Figure 19 shows the resulting concept lattice; concepts not in C_F appear with a gray background.

In this scenario, only the "specialization" operation is allowed, so

- $d(c_1, c_2) = 1$ if is a direct parent of c_2,
- $d(c_1, c_2) = \infty$ otherwise.

Using these cost settings, the kernel concept mining algorithm yields:

- Kernel concepts**:** $\{8, 9, 15\}$
- Total cost: 14.66

Within the lattice shown in Figure 7.4, these kernel concept nodes are colored orange.

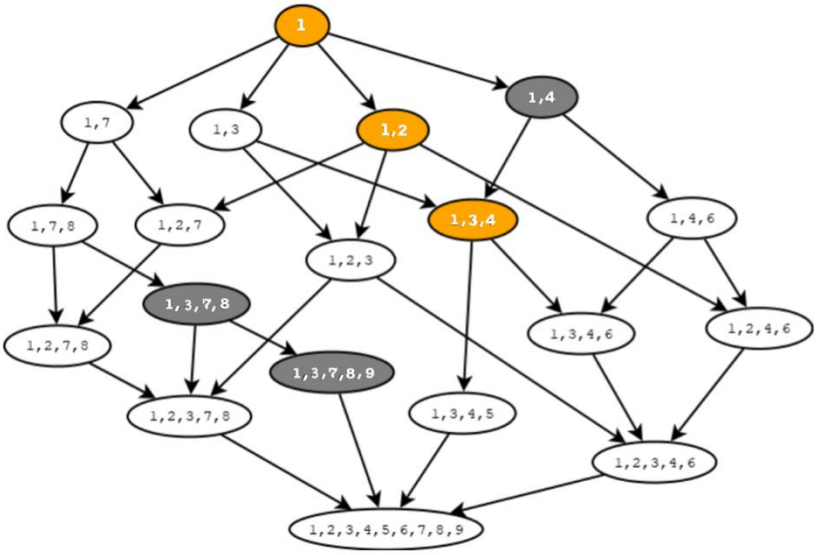

Figure 19. Structure of the Live in Water Ontology

Attribute Reduction

Although the mapping function introduced above ensures a valid word-level representation, there might be instances where some elements are redundant. In other words, certain attributes and words might be superfluous for distinguishing a particular concept, so only a subset of $attr(c) \setminus attr(c_k)$ would be needed to create an unambiguous representation. By removing these unnecessary attributes, we can streamline our overall vocabulary.

The proposed attribute reduction technique uses the attribute relevance test outlined in Algorithm 7.3. This procedure follows a greedy strategy that identifies redundant attributes in a loop. Candidate attributes are temporarily deactivated, and we check whether the remaining attributes in $attr(c) \setminus attr(c_k)$ still provide unique sets for all concepts attached to a kernel concept.

Algorithm 7: Attribute Reduction Algorithm

Input:
- Concept Lattice: L
- Kernel Set: P

Output:
- Reduced C_M concept set

Procedure:
1. For each kernel concept, gather all items in its cluster along with their respective sets $A(c) = attr(c) \setminus attr(c_k)$.
2. loop on all attributes a ∈ M for relevance test
   - For all concepts c and for attributes sets in $A(c)$, we remove a from the attribute sets. The result set is denoted by $A'(c)$.
   - We check, whether all sets in $A'(c)$ are unique or not.
4. . If the reduced set $A'(c)$ is unique for each concept c, then we can remove c from the kernel set

Example 2

Continuing the Live in Water example, we perform attribute reduction after computing the "winner" kernel concept for each concept. This computation groups concepts by kernel concept, forming separate hierarchies whose roots are the kernel concepts. Figure 20 visualizes these hierarchies.

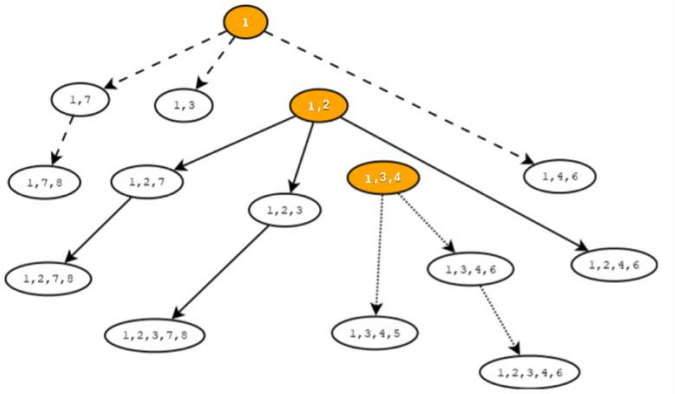

Figure 20. Structure of the resulted tree structures after selection of the kernel concepts

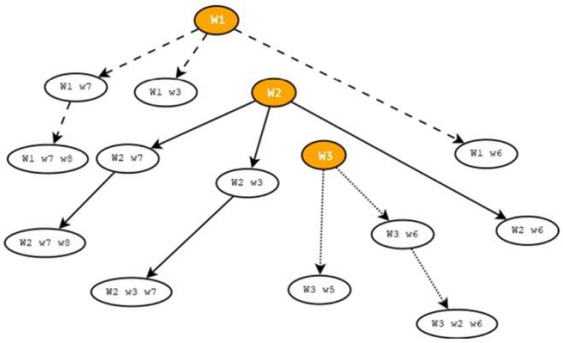

Figure 21. Word-Level Representation of the Concepts After Attribute Reduction

Next, the algorithm pinpoints redundant attributes, and in this scenario, the attributes $\{1, 4, 9\}$ are identified as extraneous. With these removed, we obtain a reduced attribute set and reconstruct the word-level representations of all concepts. Figure 21 illustrates the resulting representation tree. Here, W_i denotes the word assigned to each kernel concept, while w_i stands for the words of the attribute concepts.

Experimental Evaluation

We implemented the algorithm in Python and ran all experiments on macOS 14.3.1 (Apple M1, 8 GB RAM). Four UCI datasets Balance Scale, Breast Cancer Wisconsin, Teaching Assistant Evaluation (Tae), and Car Evaluation were converted to FCA **formal contexts** by binarizing categorical variables into Boolean attributes, then used to build their concept lattices (see Table 2). These datasets differ in size, attribute count, density, and lattice complexity, providing a comprehensive testbed for performance and scalability. The variation in objects/attributes and densities stresses the method across both sparse and dense settings, enabling a robust assessment of scalability, efficiency, and overall effectiveness.

A comprehensive set of experiments evaluated the computational time of the Genetic Algorithm (GA) and Simulated Annealing (SA) across lattices of varying sizes, differing in objects, attributes, and densities. An exponential decay distribution P(s) prioritized higher-level concepts to simulate frequent usage in natural language. Both methods were run under identical conditions. GA parameters were: population size 100, 50 generations, crossover rate 0.8, mutation rate 0.05, and tournament size 5. SA parameters were: initial temperature 1500.0, final

temperature 1.0, cooling rate 0.95, and 200 iterations per temperature. Results (Figure 22) show GA achieves substantial efficiency gains, scaling nearly linearly with lattice size, while SA grows more steeply. This highlights GA's stronger scalability and suitability for larger, more complex lattices.

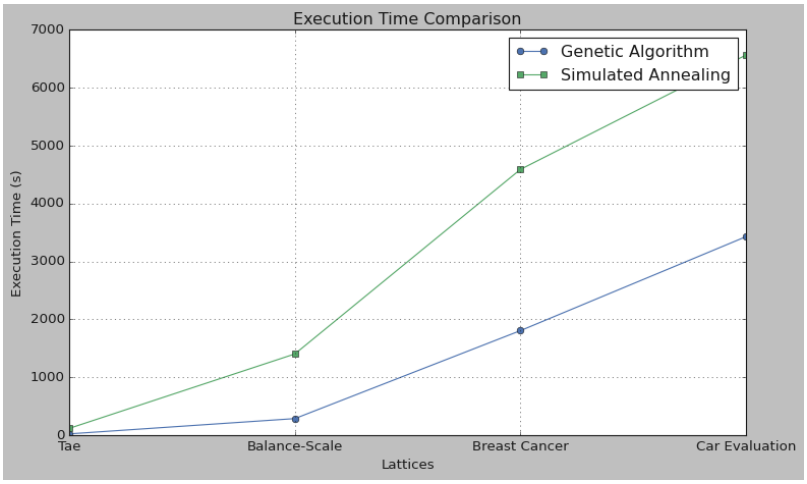

Figure 22. A Runtime Comparison of Genetic Algorithm (GA) and Simulated Annealing (SA) on Multiple Datasets

This section examines how varying kernel concept sizes (20%, 25%, 30%) affect the Total Expected Generation Cost in the Car Evaluation dataset (3,542 concepts). Both Genetic Algorithm (GA) and Simulated Annealing (SA) were tested under consistent parameters. Results (Table 6, Figure 23) show that as kernel size increases, costs steadily decrease. At 20%, GA achieved 2.0846 vs. SA's 2.0932; at 25%, GA 1.9486 vs. SA 1.9612; and at 30%, GA 1.8343 vs. SA 1.8411.

Overall, GA consistently outperformed SA, offering lower costs and demonstrating stronger scalability. Enlarging kernel size reduced generation costs further, confirming GA's robustness and efficiency in simplifying concept lattices while preserving essential structure.

Table 6. Impact of Kernel Concept Size on Optimization Performance of GA and SA

Kernel Concept Size (%)	Algorithm	Core Concepts Selected	Cost of the Kernel
20.0	GA	725	2.08461
20.0	SA	725	2.09324
25.0	GA	901	1.94862
25.0	SA	901	1.96122
30.0	GA	1,077	1.83434
30.0	SA	1,077	1.84108

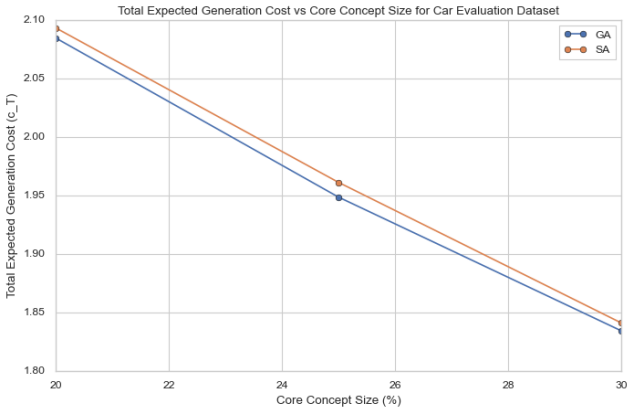

Figure 23. Variation of Total Generation Cost () with Kernel Concept Size (%) for GA and SA

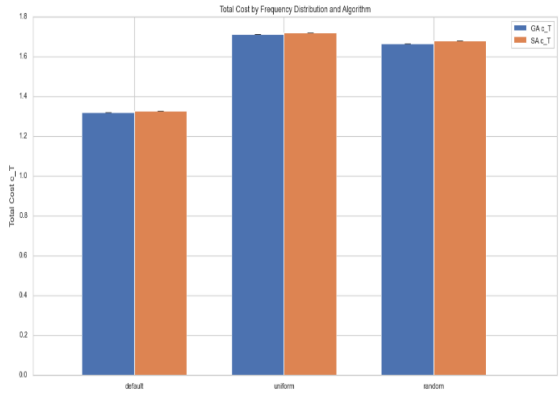

Figure 24. Average Cost Comparison of GA and SA Across Frequency Distributions

7. Contributions

The main scientific results achieved during the completion of this research are summarized below in three theses:

Thesis 1
Related Publications: [P_1, P_4]

I have introduced two new clustering algorithms for lattice reduction in FCA: K-Means Dijkstra on Lattice (KDL) and K-Means Vector on Lattice (KVL). Both approaches adapt the standard k-means clustering framework to the specific structure of concept lattices, where the relationships between formal concepts are hierarchical rather than purely numerical.
In the case of KDL, the method leverages a Dijkstra-based distance measure that assigns direction-sensitive costs to lattice traversal, ensuring that concept proximity is measured in terms of structural and hierarchical effort. This allows clusters to reflect the intrinsic organization of the lattice, thereby capturing semantic similarity more faithfully.
By contrast, KVL embeds each concept into a vector space representation based on its intent and attribute frequencies. This transformation enables the direct application of standard k-means clustering, providing a faster and more computationally scalable alternative while still preserving meaningful groupings.
Experimental evaluations conducted on benchmark datasets from the UCI Machine Learning Repository demonstrated that both KDL and KVL improve the balance between fidelity of conceptual structure and scalability of computation. KDL was shown to be particularly effective in producing structure-aware clusters, while KVL provided a robust and efficient method for handling larger datasets. Together, these two algorithms extend FCA into the realm of modern clustering applications, offering practical solutions for concept lattice reduction.

Thesis 2
Related Publications: [P_2, P_1, P_4]

I have introduced the Kernel Concept Set (KCS) approach, a selection-based strategy for reducing concept lattices by identifying a small but representative subset of formal concepts. This approach is original to the present research and defines kernel concepts as those that combine high frequency of occurrence with low derivation cost, making them both semantically central and computationally efficient.
KCS thus balances two competing objectives: preserving interpretability while reducing computational complexity. By retaining kernel concepts as structural "anchors," the lattice can be effectively approximated without losing essential relationships. This represents a departure from earlier methods such as iceberg lattices, which rely solely on frequency thresholds and therefore risk discarding structurally important but less frequent concepts.
Comparative experiments confirmed that KCS yields smaller, more interpretable lattices while still covering the most significant conceptual structures. Furthermore, the approach enhances usability by aligning with human cognitive processes of focusing on "core" concepts, making the reduced lattices easier to visualize and analyze. In this way, KCS offers both theoretical novelty and practical utility, bridging a gap between efficiency and semantic clarity in lattice reduction.

Thesis 3
Related Publications: [P_3]

I proposed an optimized Genetic Algorithm (GA) solution for mining kernel concepts in FCA. This method introduces a hybrid strategy where the GA is enhanced by a neural network module to accelerate fitness evaluation, thereby reducing the computational overhead typically associated with evolutionary approaches. The genetic optimization process was specifically tailored to select kernel sets that minimize overall derivation cost while respecting constraints on set size and interpretability. Through extensive testing on benchmark datasets, the GA-based method consistently outperformed existing approaches in terms of both efficiency and quality of selected kernel sets.
Beyond pure efficiency, the method also demonstrated adaptability to application domains such as computational linguistics, where kernel concepts can be used to represent core semantic structures in textual data. This illustrates the broader potential of kernel-based reduction beyond formal lattice theory, highlighting its utility in interdisciplinary research contexts.
Taken together, these three theses establish a coherent research program that advances the state of the art in Formal Concept Analysis. By introducing two novel clustering methods (KDL and KVL), formulating the original concept of Kernel Concept Sets, and designing an optimized evolutionary algorithm for kernel selection, this dissertation provides a comprehensive framework for scalable and interpretable lattice reduction. The results open pathways for applying FCA to increasingly complex and large-scale data, bridging theory, computation, and real-world application.

8. Author's Publications

Publications Related to the Dissertation

[P_1] M. Alwersh and L. Kovács, "K-Means Extensions for Clustering Categorical Data on Concept Lattice," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, 2023. ***Scopus Indexed [Q3]***.

[P_2] ALWERSH, Mohammed; KOVÁCS, László. Enhancing Formal Concept Analysis with the Kernel Concept Set Approach: A Novel Methodology for Efficient Lattice Reduction. International Journal of Intelligent Engineering & Systems, 2024, 17.4. ***Scopus Indexed [Q3]***.

[P_3] L. Kovács and M. Alwersh, "Mining of Kernel Concepts based on Optimization of Concept Set Generation Costs," *Knowledge and Information Systems*, **manuscript submitted for publication and currently under peer review**. ***Scopus Indexed [Q1]***.

[P_4] M. Alwersh and L. Kovács, "Survey on attribute and concept reduction methods in formal concept analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 366–387, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp366-387. . ***Scopus Indexed [Q3]***.

[P_5] ALWERSH, Mohammed; KOVÁCS, László. Fuzzy formal concept analysis: approaches, applications and issues. Computer Science and Information Technologies, 2022, 3.2: 126-136.

[P_6] ALWERSH, Mohammed. Integration of FCA with Fuzzy logic: a survey. Multidiszciplináris Tudományok, 2021, 11.5: 373-385.

References

[1] R. Wille, “Restructuring lattices theory: an approach on hierarchies of concepts,” 1982, *Dordrecht, Holland: Springer*.

[2] S. Roscoe, M. Khatri, A. Voshall, S. Batra, S. Kaur, and J. Deogun, “Formal concept analysis applications in bioinformatics,” *ACM Comput Surv*, vol. 55, no. 8, pp. 1–40, 2022.

[3] K. Sumangali and C. A. Kumar, “Critical analysis on open source LMSs using FCA,” *International Journal of Distance Education Technologies (IJDET)*, vol. 11, no. 4, pp. 97–111, 2013, doi: 10.4018/ijdet.2013100107.

[4] K. Sumangali and C. Aswani Kumar, “Knowledge reduction in formal contexts through CUR matrix decomposition,” *Cybern Syst*, vol. 50, no. 5, pp. 465–496, 2019.

[5] R. Ganter and R. Wille, “Formal concept analysis: Mathematical foundations Springer-Verlag Berlin Germany,” 1999.

[6] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, and G. Dedene, “Formal concept analysis in knowledge processing: A survey on applications,” *Expert Syst Appl*, vol. 40, no. 16, pp. 6538–6560, 2013.

[7] K. Sumangali and C. A. Kumar, “A comprehensive overview on the foundations of formal concept analysis,” *Knowledge Management & E-Learning: An International Journal*, vol. 9, no. 4, pp. 512–538, 2017, doi: 10.34105/j.kmel.2017.09.032.

[8] F. Hao, Y. Yang, G. Min, and V. Loia, “Incremental construction of three-way concept lattice for knowledge discovery in social networks,” *Inf Sci (N Y)*, vol. 578, pp. 257–280, 2021, doi: 10.1016/j.ins.2021.07.031.

[9] J. Medina, “Relating attribute reduction in formal, object-oriented and property-oriented concept lattices,” *Computers & Mathematics with Applications*, vol. 64, no. 6, pp. 1992–2002, 2012, doi: 10.1016/j.camwa.2012.03.087.

[10] W. X. Zhang, L. Wei, and J. J. Qi, “Reduction Theory and Approach to Concept Lattice. China Ser,” *E Inform. Sci*, vol. 35, pp. 628–639, 2005, doi: 10.1360/122004-104.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[12] K. S. K. Cheung and D. Vogel, “Complexity reduction in lattice-based information retrieval,” *Inf Retr Boston*, vol. 8, pp. 285–299, 2005.

[13] K. Pang, P. Liu, S. Li, L. Zou, M. Lu, and L. Martínez, “Concept lattice simplification with fuzzy linguistic information based on three-way clustering,” *International Journal of Approximate Reasoning*, vol. 154, pp. 149–175, 2023.

[14] A. Körei and S. Radeleczki, “Box elements in a concept lattice,” *Contributions to ICFCA*, vol. 2006, pp. 41–56, 2006.

[15] M. Alwersh and L. Kovács, “Survey on attribute and concept reduction methods in formal concept analysis,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 366–387, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp366-387.

[16] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal, “Computing iceberg concept lattices with titanic,” *Data Knowl Eng*, vol. 42, no. 2, pp. 189–222, 2002, doi: 10.1016/S0169-023X(02)00057-5.

[17] V. Ganti, J. Gehrke, and R. Ramakrishnan, “CACTUS—clustering categorical data using summaries,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 73–83.

[18] S. T. Piantadosi, H. Tily, and E. Gibson, “Word lengths are optimized for efficient communication,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 9, pp. 3526–3529, 2011.

[19] S. O. Kuznetsov and S. A. Obiedkov, “Comparing performance of algorithms for generating concept lattices,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 14, no. 2–3, pp. 189–216, 2002.

[20] F. Beil, M. Ester, and X. Xu, “Frequent term-based text clustering,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 436–442.

[21] R. Poli, M. Healy, and A. Kameas, *Theory and applications of ontology: Computer applications*. Springer, 2010.

[22] M. Alwersh and L. Kovács, “K-Means Extensions for Clustering Categorical Data on Concept Lattice,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, 2023.