PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Programa de Pós-Graduação em Informática

Kaio Henrique Andrade Ananias

An Approach based on Concept Approximation to search Triadic Concepts

Belo Horizonte

2020

Kaio Henrique Andrade Ananias

An Approach based on Concept Approximation to search Triadic Concepts

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática.

Advisor: Prof. Dr. Mark Alan Junho Song

Co-advisor: Prof. PhD Rokia Missaoui

Belo Horizonte

2020

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Ananias, Kaio Henrique Andrade
A533a An Approach based on Concept Approximation to search Triadic Concepts / Kaio Henrique Andrade Ananias. Belo Horizonte, 2020.
77 f. : il.

Orientador: Mark Alan Junho Song
Coorientador: Rokia Missaoui
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática

1. Conceitos - Análise. 2. Banco de dados da Web. 3. Algoritmos. 4. Arquitetura de computador. 5. Programação (Matemática). 6. Estruturas de dados. 7. Programação orientada a objetos (Computação). 8. Inteligência artificial. I. Song, Mark Alan Junho. II. Missaoui, Rokia. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática. IV. Título.

CDU: 681.3.011
Ficha catalográfica elaborada por Fernanda Paim Brito - CRB 6/2999

Kaio Henrique Andrade Ananias

An Approach based on Concept Approximation to searchTriadic Concepts

Dissertacao apresentada ao Programa de Pos-Graduacao em Informatica como requisito parcial para qualificacao ao Grau de Mestre em Informatica pela Pontificia Universidade Catolica de Minas Gerais.

Prof. Dr. Mark Alan Junho Song – PUC Minas (Advisor)

Prof. PhD Rokia Missaoui – Université du Québec en Outaouais (Co-advisor)

Prof. Dr. Luís Enrique Zárate Gálvez – PUC Minas

Prof. PhD. Renato Vimieiro – DCC-UFMG

Belo Horizonte, 1 de Setembro de 2020.

To my parents for the love and encouragement,
to my brother and friend for complicity,
and to my dogs for the faithful company

ACKNOWLEDGEMENTS

I would like to thank my parents for their support and love. I would also like to thank my brother Ryan Itálo for his friendship. I would like to thank Mr. Frodo and Mrs. Galadriel for being with me at this time of my life.

I would like to thank the support of my mentors, teachers, colleagues and the PUC-Minas institution.

"Wubba Lubba Dub-Dub."

Rick Sanchez

RESUMO

Formal Concepts Analysis (FCA) é uma teoria matemática orientada para representação e aquisição de conhecimento. Ela fornece um ferramental para o entendimento dos dados representando-os como uma estrutura de conceitos ou, mais precisamente, uma hierarquia conhecida como reticulado conceitual. Um dos potenciais da FCA está na possibilidade da análise e visualização das informações via conceitos. Entretanto, à medida que a base de dados utilizada cresce, a pesquisa e principalmente a visualização e exploração dos conceitos se torna proibitiva. Com a extensão da abordagem clássica FCA para *Triadic Concept Analysis* (TCA) ou 3FCA este problema se torna ainda mais evidente dada a complexidade das estruturas inerentes à relação triádica. Desta forma, este trabalho tem como objetivo propor uma abordagem, baseada em aproximações, para pesquisa de conceitos em reticulados triádicos possibilitando a visualização e exploração das informações presentes na base.

Palavras-chave: Análise Formal de Conceitos, Análise Formal de Conceitos Triádicos, Aproximação de Conceitos Triádicos.

ABSTRACT

Formal Concept Analysis (FCA) is a mathematical theory oriented to knowledge representation and acquisition, as well as data analysis and visualization. It provides a tool for understanding data by representing it as a conceptual framework or, more precisely, a concept hierarchy. FCA assists in processing by providing a framework for applying different data analysis techniques for knowledge acquisition. As pointed out, one of the potentials of FCA lies in the mathematical foundation that enables the generation, ordering, and visualization of information in the form of formal concepts described in a hierarchy known as a conceptual lattice. However, as the database used grows, research, and especially the visualization and exploration of concepts becomes prohibitive. With the extension of the classical approach FCA to TCA or 3FCA this problem becomes even more evident given the complexity of the structures inherent in the triadic relationship. Thus, this work aims to propose an approach, based on approximations, for research of concepts in triadic lattices allowing the visualization and exploration of the information present in the base.

Formal Concept Analysis, Triadic Formal Concept Analysis, Triadic Concept Approximation.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

FCA – *Formal Concepts Analysis*

TCA – *Triadic Concept Analysis*

SUMMARY

1 INTRODUCTION

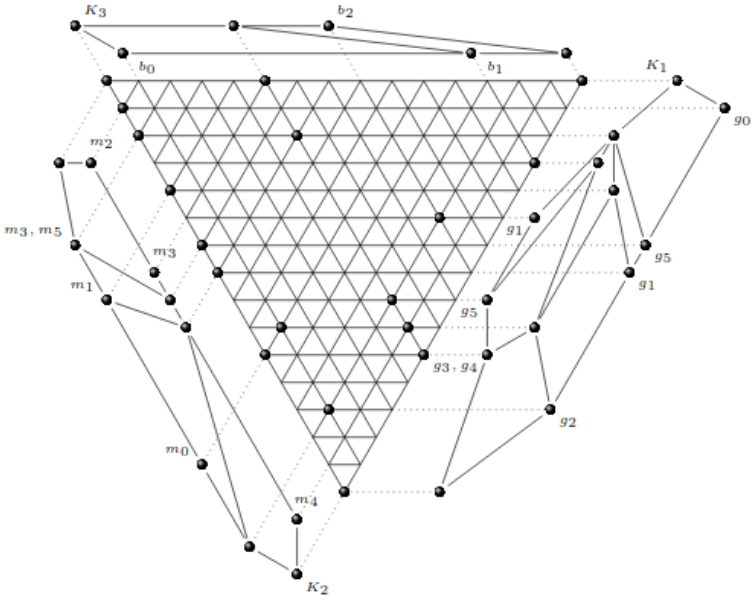
With the advent of the web and the large volume of data generated by different applications, the exploration and extraction of knowledge have become important ways to understand human behavior and assist in decision making in various scenarios. To obtain useful information, different approaches can be used, such as the Formal Concept Analysis (FCA) which provides a mathematical framework for representing knowledge from data.

Originally proposed by Ganter e Wille (1999), dyadic FCA (generally referred to as FCA) is based on the binary relationship between two sets of elements, named objects and attributes. The growing interest in the use of FCA for data mining is due to the fact that it represents and organizes what is known as formal concepts in a mathematical structure called conceptual lattice (WILLE, 1982). The lattice allows hierarchical visualization and understanding of data, as well as the extraction of implication rules, helping to discover useful and non-trivial knowledge.

Although successful in many applications, some situations require an extension of the classic approach, adding a third dimension for better characterization and representation of the data. The main example of such a scenario is the social resource sharing system represented by *folksonomies* (JAY; KOHLER; NAPOLI, 2008). This example consists of resources assigned with tags by users, this ternary relation can be represented as a triadic context.

The dyadic FCA was extended by Lehmann e Wille (1995) by adding a third dimension frequently called *modus* or conditions, featuring a ternary relationship between objects, attributes and conditions. This new approach is known as Triadic Concept Analysis (TCA or 3FCA). Lehmann e Wille (1995) proposed a graphical representation of the triadic network (trilattice) using a three-dimensional diagram and Biedermann (1997) showed how to read these diagrams and extract triadic implications from it. Despite coming from FCA, whose graphical representation is intuitive, the three-dimensional complexity of TCA makes the use of this structure prohibitive. Graphical representation is not intuitive, even in small data sets, due to the large number of triadic concepts that can be generated, losing the power of interpretation and knowledge extraction of FCA.

Figure 1 – Trilattice of "Hostels".



Fonte: (GLODEANU, 2013)

The graphical representation of a triadic lattice (Figure 1) is made through a 3-net whose central circles represent triadic concepts. Its extents, intents, and conditions can be read by following the dotted lines that connect the elements to the Hasse diagrams around and above the central triangle (WILLE, 1995). The exploration and knowledge extraction using this structure becomes extremely complex even in small data sets, as mentioned previously. In large data sets, the use of such a diagram as a data mining process is impractical.

Some works in the literature try to reduce the complexity of visualization and navigation of the triadic lattice (RUDOLPH; SĂCĂREA; TROANCĂ, 2015; MISSAOUI et al., 2020), by proposing strategies and graphic representation for the triadic setting based on the classic Hasse diagram. Missaoui et al. (2020) adapted the iPred algorithm (BAIXERIES et al., 2009) used to compute dyadic lattices and proposed a new algorithm called T-iPred to order and represent triadic lattices.

This new representation proposed in (MISSAOUI et al., 2020), simplify the visualization of the Hasse diagram using the classic FCA approach. Although this representation is easier to read and understand, stills a need for techniques of information and knowledge extraction from triadic data. One way to improve the use of TCA as a framework to manipulate data is by creating algorithms to query information related to the triadic concepts from the Hasse diagram. For example, it would be possible to identify in the lattice concepts formed by objects, attributes, and conditions that meet certain triadic relationships, present in the data set, as a result of a query mechanism. However, if the triadic relationship does not exist, it would be relevant to identify the concepts closest to the

one sought. This mechanism can use the simplified diagram produce by the T-iPred as a graphical representation for the results queried by the user.

1.1 Problem

This work aims to provide a mechanism to approximate concepts of a certain pattern defined as a query to the Hasse diagram of triadic concepts, where the elements sought can be specified as a triple. That said, the problem can be formulated with three questions:

1. It is possible to find a set of triadic concepts closest to a query where only one dimension is specified?
2. It is possible to find a set of triadic concepts closest to a query where two dimensions are specified?
3. It is possible to find a set of triadic concepts closest to a query where all the triadic dimensions are specified?

1.2 Objectives

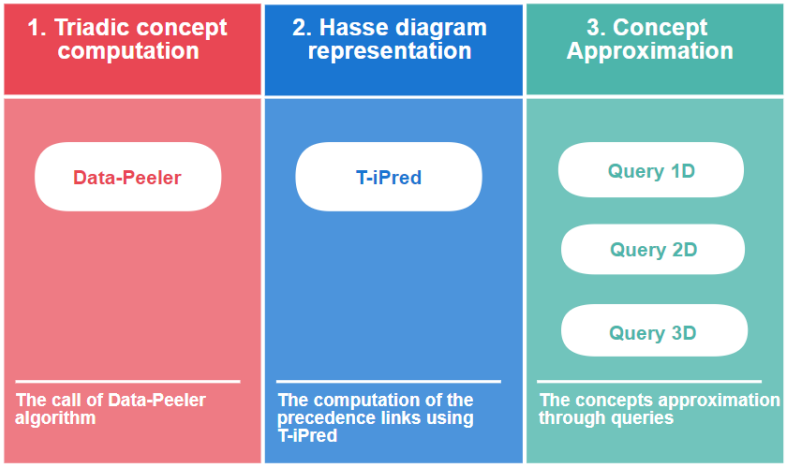
The objective of this work is to propose a method to extract information from triadic contexts using the T-iPred as a graphical base to display and retrieve information from a user query. We propose a method to search for patterns represented by formal concepts in order to either retrieve or approximate a triadic concept. Through queries of up to three dimensions, it is possible to search for concepts that are closest to the information specified. The searching strategy proposed uses the diagram generated by T-iPred to display the result of a query through upper and lower covers. Figure 2 shows our proposal. We proposed this architecture to aim the exploration of the triadic context using three types of queries that can be made to the Hasse diagram (second module) of triadic concepts (first module). In the third module, we proposed three approaches to approximate concepts from one of the triadic dimensions, or any subset of them. Despite the pipeline showed here, our main contributions rely on the Concept Approximation.

For this, it was necessary to achieve the following specific objectives:

- Propose an approach to approximate concepts using one dimension based on object, attribute or condition;
- Propose an approach to approximate concepts using two dimensions based on combinations of objects/attributes, attributes/conditions and objects/conditions;

- Propose an approach to approximate concepts using three dimensions based on objects/attributes/conditions;

Figure 2 – The architecture modules of our proposed solution.



1.3 Justification

The organization and manipulation of information through FCA and TCA are objectives of several studies in the literature. The classical approach has techniques for organizing, hierarchizing, and searching for formal concepts. But, in the triadic approach, the complexity of adding the third dimension makes the visualization and manipulation of concepts a complex task with a few understandable knowledge, even in small data sets. Although the TCA can be used to map data in various scenarios, such as collaborative tagging, also called folksonomies (JASCHKE et al., 2006), the problem of searching for information when it is ordered hierarchically, to help humans understand the structure of the triconcepts, is not a trivial task and, for this reason, is the main focus of this work.

1.4 Organization

This master dissertation is organized as follows. Chapter 2 presents the theoretical foundation on FCA and TCA as well as their formal structures. Chapter 3 describes the revision of the literature with a focus on the exploration and approximation of data through triadic concepts. Chapter 4 describes the contributions of this work defining queries of up to three dimensions and the method of approximating triadic concepts. Finally, Chapter 5 presents the conclusions and future works.

2 THEORETICAL FOUNDATIONS

2.1 Formal Concept Analysis

Proposed by Rudolf Wille in the 1980, Formal Concept Analysis is a branch of applied mathematics based on the mathematization of the concept and conceptual hierarchy (GANTER; WILLE, 1999). The formalization of concepts must be transparent and simple, but also comprehensive so that the main aspects of a concept can have explicit references in the formal model (LEHMANN; WILLE, 1995).

2.1.1 Formal Context

In the traditional approach (dyadic), a formal context is a triple $\mathbb{K} = (G, M, I)$, where G is the set of objects, M a set of attributes and I the binary incidence relation between G and M defined by $I \subseteq G \times M$ indicating that an object $g \in G$ has a certain attribute $m \in M$ (gIm or $(g, m) \in I$).

Formal contexts are often represented by a two-dimensional table where the lines represent the objects of the context and the columns the attributes. When an object has a certain attribute, " × " is placed in the respective row and column of the table. The Table 1 represents a dyadic context with three objects $\{O_1, O_2, O_3\}$, six attributes $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ and ten incidences. That is, $|\boldsymbol{G}| = 3$, $|\boldsymbol{M}| = 6$ and $|\boldsymbol{I}| = 10$. The column 1, line 2 indicates that the object O_1 has the attribute a_1 or simply (O_1, a_1).

Table 1 – Dyadic context represented by a cross table.

$\boldsymbol{G/M}$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
O_1	×		×		×	×
O_2		×		×		×
O_3	×		×		×	

2.1.2 Formal Concepts

Given a dyadic context $\mathbb{K} = (G, M, I)$ and a subset of objects $A \subseteq G$, there is a subset of attributes shared by all objects in A. Similarly, given a subset of attributes $B \subseteq M$ there is a set of objects where all elements share all attributes in B. This relationship can be expressed through the derivation operators defined below (GANTER; WILLE, 1999):

$$A' = \{m \in M \mid gIm \forall\, g \in G\} \tag{2.1}$$

$$B' = \{g \in G \mid gIm \forall\, m \in M\} \tag{2.2}$$

For instance, take the dyadic context represented by Table 1. The derivation operator can be applied to a set of objects or attributes as follows. Given the set of objects $\{o_1, o_2\} \subseteq G$, applying the derivation operator $(o_1, o_2)'$ we get the attribute set $\{a_6\}$. That is, only the attribute $a_6 \in M$ is shared by both objects. The derivation can also be applied to a set of attributes. The result of the $(a_6)'$ operation is exactly the two objects $\{o_1, o_2\}$ previously derived.

When $A = B'$ and $B = A'$ we say that the pair of objects and attributes *(A, B)* is a formal concept and its elements are called *extent* and *intent* respectively. The extent contains all the objects g that have all the attributes $m \in B$. On the other hand, the intent contains all the attributes m shared by all objects in $g \in A$. The pair $(\{o_1, o_2\}, \{a_6\})$ from the context represented by Table 1 is a formal concept, since the objects $\{o_1, o_2\}$ have exactly the attribute $\{a_6\}$ and the only objects shared by $\{a_6\}$ are $\{o_1, o_2\}$. The Table 2 presents all the dyadic concepts from the formal context in Table 1.

Table 2 – Formal concepts of the Table 1.

Extent	**Intent**
$\{o_1, o_2, o_3\}$	$\{\}$
$\{o_2\}$	$\{a_2, a_4, a_6\}$
$\{o_1, o_2\}$	$\{a_6\}$
$\{o_1, o_3\}$	$\{a_1, a_3, a_5\}$
$\{o_1\}$	$\{a_1, a_3, a_5, a_6\}$
$\{\}$	$\{a_1, a_2, a_3, a_5, a_6\}$

Notice that the Table 1 has two concepts with empty sets. The first and last concept are called *infimum* and *supremum*.

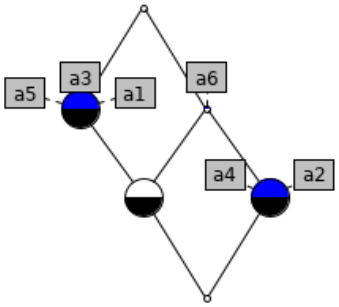
2.1.3 Concept Lattice

The set of all formal concepts of a context (G, M, I) ordered by a partial order $\leq$, constitutes a complete lattice, also called conceptual lattice denoted by $\mathfrak{B}(G, M, I)$ (DAVEY; PRIESTLEY, 2002). Consider the context (G, M, I) presented in Table 1. The set of all concepts can be ordered by the partial order $\leq$ such that, for any two concepts $(A, B), (C, D) \in (G, M, I)$, we have the following relation:

$$(A, B) \leq (C, D) \iff C \subseteq A \text{ and } B \subseteq D \tag{2.3}$$

Figure 3 presents the Hasse diagram of the conceptual lattice $\mathfrak{B}(G, M, I)$ obtained from Table 1.

Figure 3 – Conceptual Lattice from context 1.



The basic theorem of conceptual lattices states that a conceptual lattice defined by $\mathfrak{B}(G, M, I)$ is a complete lattice in which any subset $C \subseteq \mathfrak{B}(G, M, I)$ the *supremum* and *infimum* are given by:

$$\begin{aligned} \bigwedge C &= (\bigcap X, (\bigcup Y)'') \\ \bigvee C &= ((\bigcup X)'', \bigcap Y) \end{aligned} \tag{2.4}$$

where $X = \{A|(A, B) \in C\}$ e $Y = \{B|(A, B) \in C\}$ (GODIN; MISSAOUI; ALAOUI, 1991).

2.2 Triadic Concept Analysis

The Triadic Concept Analysis (TCA), introduced by Lehmann e Wille (1995), extends the classic FCA with the insertion of a new dimension. The primitive notion of a formal context is defined by a quadruple (K_1, K_2, K_3, Y) where K_1, K_2 and K_3 are sets and Y the ternary relationship between K_1, K_2 and K_3. The elements of K_1, K_2 and K_3 are called objects, attributes and conditions respectively and $(a_1, a_2, a_3) \in Y$ defines that the object a_1 has the attribute a_2 under the condition a_3 (LEHMANN; WILLE, 1995; WILLE, 1995).

2.2.1 Triadic Formal Context

Just as dyadic contexts are often described by two-dimensional tables, triadic contexts can be presented by three-dimensional tables (LEHMANN; WILLE, 1995). Table 3 represents a triadic context where $K_1 = \{o_1, o_2, o_3, o_4\}$, $K_2 = \{a_1, a_2, a_3\}$ e $K_3 = \{c_1, c_2\}$.

Table 3 – Triadic Context represented by a three-dimensional table.

	c_1			c_2		
	a_1	a_2	a_3	a_1	a_2	a_3
o_1	×		×	×		
o_2					×	
o_3	×	×				×
o_4			×		×	×

Every triadic context (K_1, K_2, K_3, Y) gives rise to the following dyadic contexts:

$$\begin{aligned}
\mathbb{K}^{(1)} &:= (K_1, K_2 \times K_3, Y^{(1)}) \quad \text{with} \quad a_1 Y^{(1)} (a_2, a_3) \Leftrightarrow a_1, a_2, a_3 \in Y \\
\mathbb{K}^{(2)} &:= (K_2, K_1 \times K_3, Y^{(2)}) \quad \text{with} \quad a_2 Y^{(2)} (a_1, a_3) \Leftrightarrow a_1, a_2, a_3 \in Y \\
\mathbb{K}^{(3)} &:= (K_3, K_1 \times K_2, Y^{(3)}) \quad \text{with} \quad a_3 Y^{(3)} (a_1, a_2) \Leftrightarrow a_1, a_2, a_3 \in Y
\end{aligned} \tag{2.5}$$

Intuitively the contexts $\mathbb{K}^{(i)}$, i = *{1,2,3}*, represent the "flattened"versions of the triadic context, obtained through the combinations of triadic dimensions (LEHMANN; WILLE, 1995). These contexts are useful for deriving elements, as will be seen in future sections.

Table 4 represents the generated dyadic context $\mathbb{K}^{(1)} = (K_1, K_2 \times K_3, Y^{(1)})$ from the context $\mathbb{K}$ presented in Table 3. $\mathbb{K}^{(1)}$ is a dyadic context whose set of objects is the same as the original context but its attributes are combinations of the attributes and conditions from $\mathbb{K}$, i.e., $K_2 \times K_3$. This new generated context has six attributes represented by pairs.

Table 4 – Dyadic context $\mathbb{K}^{(1)} = (K_1, K_2 \times K_3, Y^{(1)})$

	(a_1, c_1)	(a_2, c_1)	(a_3, c_1)	(a_1, c_2)	(a_2, c_2)	(a_3, c_2)
o_1	×		×	×		
o_2					×	
o_3	×	×				×
o_4			×		×	×

Therefore, the contexts $\mathbb{K}^{(2)}$ and $\mathbb{K}^{(3)}$, are dyadic contexts where the sets of objects are the attributes and conditions, respectively, of the original context. Thus, $\mathbb{K}^{(2)}$ has only three objects (a_1, a_2, a_3) and eight attributes resulting from the combination of $K_1 \times K_3$ (Table 5).

Table 5 – Dyadic context $\mathbb{K}^{(2)} = (K_2, K_1 \times K_3, Y^{(2)})$

	(o_1, c_1)	(o_2, c_1)	(o_3, c_1)	(o_4, c_1)	(o_1, c_2)	(o_2, c_2)	(o_3, c_2)	(o_4, c_2)
a_1	×		×		×			
a_2			×			×		×
a_3	×			×			×	×

Finally, the context $\mathbb{K}^{(3)}$ has two objects (c_1, c_2) and twelve attributes since this is the number of pairs resulting from the combination of $K_1 \times K_2$ of objects and attributes of the original context. Table 6 represents the dyadic context $\mathbb{K}^{(3)}$.

Table 6 – Dyadic context $\mathbb{K}^{(3)} = (K_3, K_1 \times K_2, Y^{(3)})$

	(o_1, a_1)	(o_2, a_1)	(o_3, a_1)	(o_4, a_1)	(o_1, a_2)	(o_2, a_2)	(o_3, a_2)	(o_4, a_2)	(o_1, a_3)	(o_2, a_3)	(o_3, a_3)	(o_4, a_3)
c_1	×		×				×		×			×
c_2	×					×		×			×	×

2.2.2 Triadic Concept

One of the basic operations in FCA is the derivation of elements of the context through the derivation operators. The $\mathbb{K}^{(i)}$ contexts generated from a triadic context (Section 2.2.1), allow the definition of a *(i)-derivation* operator (LEHMANN; WILLE, 1995).

Let $\mathbb{K}: = (K_1, K_2, K_3, Y)$ be a triadic context and $\{i, j, k\} = \{1, 2, 3\}$ such that $i < j$. For $X_i \subseteq K_i$ e $(X_j, X_k) \subseteq K_j \times K_k$ the *(i)-derivation* operator is defined by :

$$\begin{aligned} X_i^{(i)} &:= \{(a_j, a_k) \in K_j \times K_k \mid a_i, a_j, a_k \in Y \text{ for all } a_i \in X\} \\ (X_j, X_k)^i &:= \{a_i \in K_i \mid a_i, a_j, a_k \in Y \text{ for all } (a_j, a_k) \in K_j \times K_k\} \end{aligned} \tag{2.6}$$

In words, when applied to a set X_i the *(i)-derivation* operator determines all the pairs (a_j, a_k) of the dimension j and k which are shared by all elements in X_i. On the other hand, when applied to a pair (X_j, X_k), the *(i)-derivation* operator determines all the elements a_i that share the elements of both dimensions j and k.

For example, take the context of Table 3 and $i = 1, j = 2$ and $k = 3$. By applying the *(i)-derivation* operator in a set of objects, for instance $\{o_1\}$, the result is a set of pairs of attributes and conditions. In this case, $\{o_1\}^{(1)} = \{(a_1, c_1c_2), (a_3, c_2)\}$. Furthermore, the operator can be applied in a pair of, for example, attributes and conditions, resulting in this case, in a set of objects, the inverse relation obtained before. Taking the exactly same pairs of the previously example, and applying the operator in each of these pairs, the result is the following: $\{(a_1, c_1c_2)\}^{(1)} = \{o_1\}$ and $(a_3, c_2)\}^{(1)} = \{o_3, o_4\}$.

In addition, another type of derivation operator, based on the triadic structure can be defined (LEHMANN; WILLE, 1995):

$$\begin{aligned} X_i \to X_i^{(i,j,A_k)} &:= \{a_j \in K_j \mid a_i, a_j, a_k \text{ are related by } Y \text{ for all } (a_i, a_k) \in X_i \times A_k\} \\ X_j \to X_j^{(i,j,A_k)} &:= \{a_i \in K_i \mid a_i, a_j, a_k \text{ are related by } Y \text{ for all } (a_j, a_k) \in X_j \times A_k\} \end{aligned} \tag{2.7}$$

Consider $X_3 = \{c_1, c_2\} \subseteq K_3$, by the Equation 2.7 we get the dyadic context $\mathbb{K}_{X_3}^{(12)} = (K_1, K_2, Y_{X_3}^{12})$ and from this definition it is possible to formalize triadic concepts as follows.

Concepts are understood as units of thought. This means that a concept tends to be homogeneous and closed. If a concept is viewed through the triadic paradigm then, for formalization, a concept should be seen as a combination of objects, attributes, and conditions which is homogeneous and closed. Homogeneity is attained if each object has each attribute under each condition within the concept. The closure is attained if the concept is maximal concerning this property (LEHMANN; WILLE, 1995). A triadic concept of a context $\mathbb{K} := (K_1, K_2, K_3, Y)$ is defined by a triple (A_1, A_2, A_3), such that A_i $\subseteq K_i$ for *i = {1,2,3}* and $A_i = (A_j \times A_k)^{(i)}$ for all {i,j,k} = {1,2,3} with $j < k$. The sets A_1, A_2, A_3 are called extent, intent and modus or conditions of the concept, respectively.

Unlike the dyadic approach where, given a context (G, M, B), concepts can be derived from a single set $X \subseteq G$ or $Y \subseteq M$ forming the pairs (X'', X') or (Y'', Y'), in TCA, two sets are required and the concepts can be constructed as follows:

For $X_i \subseteq K_i$ and $X_k \subseteq K_k$ with $\{i, j, k\} = \{1, 2, 3\}$, let $A_j = X_i^{(i,j,X_k)}$, $A_i = A_j^{(i,j,X_k)}$ e $A_k = (A_i \times A_j)^{(k)}$ (if $i < j$) or $A_k = (A_j \times A_i)^{(k)}$ (if $j < i$). Then, (A_1, A_2, A_3) is the triadic concept $b_{ik}(X_i, X_j)$ with the property that it has smallest k-th component among all triadic concepts (B_1, B_2, B_3), with the largest j-th component satisfying $X_i \subseteq B_i$ e $X_k \subseteq B_k$. In particular $b_{ik}(X_i, X_j) = (A_1, A_2, A_3)$ for each triadic concept (A_1, A_2, A_3) of $\mathbb{K}$ according to (WILLE, 1995).

For instance, take the triadic context of the Table 3. Consider $X_1 = \{o_1\} \subseteq K_1$, $X_3 = \{c_1, c_2\} \subseteq K_3$. According to Wille (1995), we have $A_2 = X_1^{(1,2,X_3)}$, $A_1 = A_2^{(1,2,X_3)}$ e $A_3 = (A_1 \times A_2)^{(3)}$. Through Table 3, we obtain $A_2 = \{o_1\}^{(1,2,X_3)} = \{a_1\}$, $A_1 = \{a_1\}^{(1,2,X_3)} = \{o_1\}$. Then, from Table 6, we have $A_3 = (\{o_1\} \times \{a_1\})^{(3)} = \{c_1, c_2\}$. So (o_1, a_1, c_1c_2) is a triadic concept (short: triconcept) of the Table 7, or $b_{i,k}(X_1, X_3) = (o_1, a_1, c_1c_2)$.

Table 7 – Dyadic context $\mathbb{K}^{12}_{X_3} = (K_1, K_2, Y^{12}_{X_3})$.

	a_1	a_2	a_3
o_1	×		
o_2			
o_3			
o_4			×

The set of all triadic concepts of a given context $\mathbb{K}$ is frequently denoted as $\mathfrak{T}(\mathbb{K})$. The Table 8 shows the set of all triconcepts extracted from the context represented by Table 3.

Table 8 – Triconcepts from Table 3.

Extent	Intent	Conditions
$\{\}$	$\{a_1, a_2, a_3\}$	$\{c_1, c_2\}$
$\{o_2, o_4\}$	$\{a_2\}$	$\{c_2\}$
$\{o_4\}$	$\{a_2, a_3\}$	$\{c_2\}$
$\{o_1\}$	$\{a_1\}$	$\{c_1, c_2\}$
$\{o_4\}$	$\{a_3\}$	$\{c_1, c_2\}$
$\{o_3, o_4\}$	$\{a_3\}$	$\{c_2\}$
$\{o_1, o_2, o_3, o_4\}$	$\{\}$	$\{c_1, c_2\}$
$\{o_3\}$	$\{a_1, a_2\}$	$\{c_1\}$
$\{o_1\}$	$\{a_1, a_3\}$	$\{c_1\}$
$\{o_1, o_3\}$	$\{a_1\}$	$\{c_1\}$
$\{o_1, o_4\}$	$\{a_3\}$	$\{c_1\}$
$\{o_1, o_2, o_3, o_4\}$	$\{a_1, a_2, a_3\}$	$\{\}$

The Table 9 shows the triconcepts (o_3, a_1a_2, c_1) in orange and (o_3o_4, a_3, c_2) in green. Note that the colored rectangles cannot be augmented without violating the triadic relation.

Table 9 – Triconcepts (o_3, a_1a_2, c_1), (o_3o_4, a_3, c_2).

	c_1			c_2		
	a_1	a_2	a_3	a_1	a_2	a_3
o_1	×		×	×		
o_2					×	
o_3	×	×				×
o_4			×		×	×

As shown by Lehmann e Wille (1995), every triadic context has at least three concepts, called trivial concepts. These concepts can be seen in the Table 8 in lines 1, 7 and 12, of the context represented by the Table 3. Each of these concepts has an empty dimension (extent, intent or condition). Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context,

the trivial concepts can be formally defined as follows (LEHMANN; WILLE, 1995):

$$\begin{aligned} o_1 &:= ((K_2 \times K_3)^{(1)}, K_2, K_3) \\ o_2 &:= (K_1, (K_1 \times K_3)^{(2)}, K_3) \\ o_3 &:= (K_1, K_2, (K_1 \times K_2)^{(3)}) \end{aligned} \tag{2.8}$$

2.2.3 Triadic Concept Lattice

The set of all triconcepts denoted by $\mathfrak{T}(\mathbb{K})$, of the context $\mathbb{K}$, can be partially ordered forming a complete lattice, also called trilattice (LEHMANN; WILLE, 1995). For $i \in \{1, 2, 3\}$, the relation $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3) \Leftrightarrow A_i \subseteq B_i$ constitutes a partial order whose equivalence class $\sim_i$ is given by $(A_1, A_2, A_3) \sim_i (B_1, B_2, B_3) \Leftrightarrow A_i = B_i$. These three quasi-orders satisfy the following antiordinal dependencies (WILLE, 1995): for $\{i, j, k\} = \{1, 2, 3\}$, $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3)$ e $(A_1, A_2, A_3) \lesssim_j (B_1, B_2, B_3)$ implies that $(A_1, A_2, A_3) \gtrsim_k (B_1, B_2, B_3)$ for all triconcepts (A_1, A_2, A_3) e (B_1, B_2, B_3).

In addition $[(A_1, A_2, A_3)]_i$ denotes the equivalence class $\sim_i$ containing the concept (A_1, A_2, A_3). The partial order $\lesssim_i$ induces an order relation $\leq_i$ in the factored set $\mathcal{T}(\mathbb{K})\backslash \sim_i$ of all classes of equivalences denoted by:

$$[(A_1, A_2, A_3)]_i \leq_i [(B_1, B_2, B_3)]_i \Leftrightarrow A_i \subseteq B_i \tag{2.9}$$

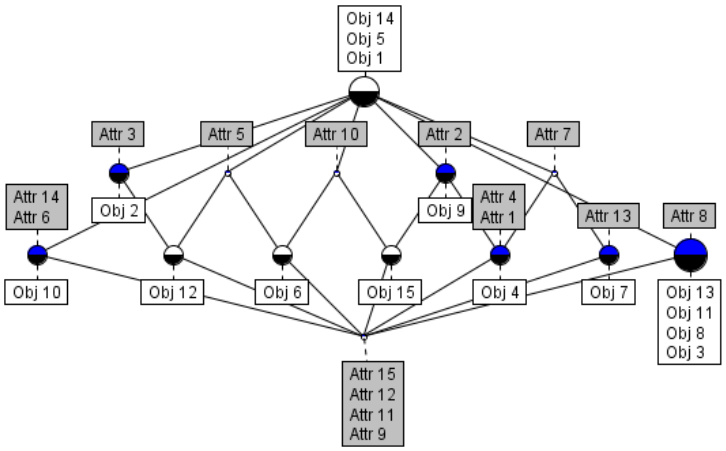
A structure analogous to the dyadic case is obtained by the triadic lattice or *trilattice* $\underline{\mathfrak{T}}(\mathbb{K}) := (\mathfrak{T}(\mathbb{K}), \lesssim_1, \lesssim_2, \lesssim_3)$ for the triadic relationship. All trilattices are complete, as proved by Wille (1995).

3 RELATED WORK

The classic FCA approach has several works in the literature related to data navigation and exploration. In Yevtushenko (2000) the authors proposed a tool for exploring concepts called ConExp.

ConExp is used to create and edit dyadic contexts. It is possible to analyze formal context, draw the corresponding conceptual structure and explore different dependencies between attributes. It provides tools for editing, building conceptual lattices from dyadic contexts, and extracting implication and association rules. The tool has an exploration technique such that some relations extracted from the implications are presented to the user in an iterative way. It is widely used by the FCA community due to the line diagram generated. The Figure 4 shows an example of a conceptual lattice of a dyadic context where the objects of each node are represented by a white box and the attributes are represented by a gray box. Exploration through ConExp is very simplistic, since only the association rules are displayed to the user in an iterative way. It is not possible to do any research in the structure or extract more information beyond the association rules extracted from the hierarchy.

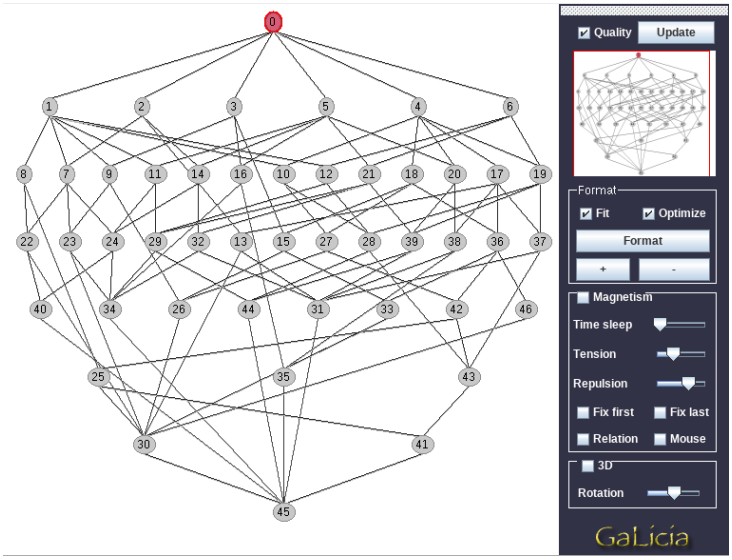
Figure 4 – Lattice diagram built by ConExp.



Valtchev et al. (2003) the authors introduced the tool for manipulating, building and editing formal lattices and their contexts, called Galicia. The tool allows 2D and 3D visualization of dyadic lattices as well as the application of basic FCA algorithms

to reduce and transform multi-valued contexts. Galicia provides different algorithms for creating conceptual lattices (VALTCHEV; MISSAOUI, 2001; VALTCHEV et al., 2002). In addition, the tool supports manipulation of lattices through operations such as *assembly*, merging, splitting and decomposition according to (VALTCHEV; MISSAOUI, 2001). Figure 5 shows the line diagram of a dyadic lattice compute by the Galicia tool.

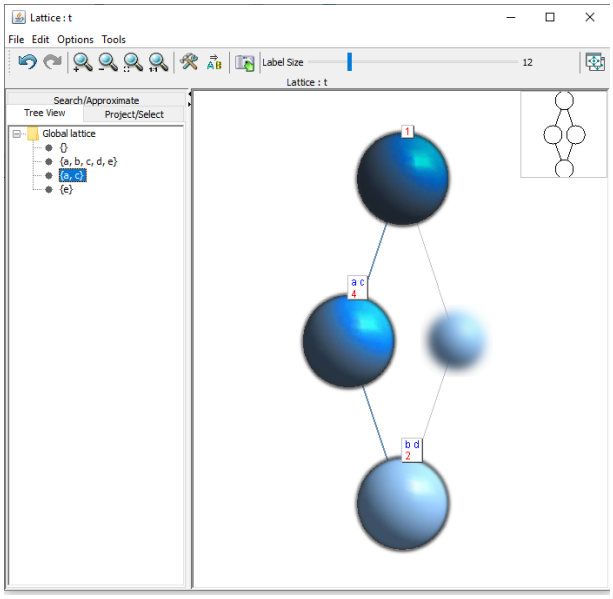
Figure 5 – Lattice diagram visualization using Galicia tool.



In Lahcen e Kwuida (2010) the authors proposed a tool called Lattice Miner, used for construction, visualization and manipulation of formal dyadic concepts. The tool allows the generation of formal concepts, association rules, as well as the transformation of formal contexts via apposition, sub-position, reduction, generalization of objects and attributes and the manipulation of conceptual lattices via approximation, projection and selection. With a focus on visualization, Lattice Miner implements techniques to improve the user experience, such as reducing edge collision, symmetry between the lattice nodes, as well as tools to make some basic operations easier. Through the tool it is possible to apply zoom-in and zoom-out to specific objects of the lattice that may be of interest to the user. It is possible to obtain details about the concepts related to the lattice as well as the number of objects and attributes, etc. It can display the entire hierarchy of a given concept (successors and predecessors) while the rest of the lattice is ignored. Figure 6 shows a dyadic lattice created using the tool with some concepts highlighted.

Lattice Miner has numerous techniques for manipulating contexts and dyadic concepts and is widely used in the literature. However, for triadic approach, only operations such as calculating concepts and implication rules are implemented. It is not possible to visualize or explore, even through projections, a triadic context.

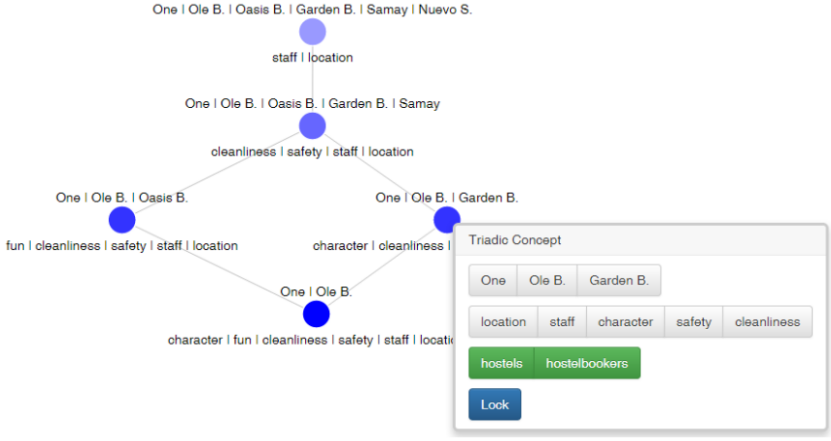
Figure 6 – Lattice diagram visualization using Lattice Miner.



Kis, Sacarea e Troanca (2016) facing the problem of manipulating and extracting information from triadic lattices, represented as 3-nets diagrams (LEHMANN; WILLE, 1995), proposed a tool for local exploration and navigation in triadic contexts through projections and perspectives. The tool uses the TRIAS algorithm (JASCHKE et al., 2006) to compute all the concepts of a given triadic context that are displayed graphically to the user. The user can choose a triadic concept and define as a perspective one of his three dimensions, i.e., objects, attributes or conditions.

Given the user-defined perspective, a dyadic concept is generated and its conceptual lattice is computed and presented through a Hasse diagram. Based on this diagram, the classical dyadic visualization is used to explore similar triadic concepts and therefore achievable from the chosen perspective. From the dyadic lattice, a new triadic concept can be chosen by the user, which can be a starting point for the next visualization and navigation using the tool.

Figure 7 – Conceptual lattice of a projected triadic concept.



Given a triconcept $T = (A_1, A_2, A_3)$ and a perspective k, a triadic context $\mathbb{K}_{A_k}^{(ij)}$ is generated and its lattice $\mathfrak{B}(\mathbb{K}_{A_k}^{(ij)})$ is built. The authors prove that given a triadic concept $(A, B, C) \in \mathfrak{T}(\mathbb{K})$ then, $(A, B) \in \mathfrak{B}(\mathbb{K}_{A_k}^{(ij)})$ and $(A, B, C) \in \mathfrak{T}(\mathbb{K})$, $(A_1, A_2) \in \mathfrak{B}(\mathbb{K}_{A_k}^{(ij)})$ then $(A_1, A_2, (A_1 \times A_2)^{(3)}) \in \mathfrak{T}(\mathbb{K})$. Based on these properties, the concept of *directly reachable concepts* is presented and also used for navigation and clustering (CARPINETO; ROMANO, 1993) of triadic concepts.

Based on the properties of reachable concepts, the authors identified that not all concepts can be directly achieved through an initial concept, so the idea of a group of mutually reachable concepts (*clusters*) is defined and explored. A set of directly reachable concepts is computed and using an initial triconcept T it is possible to navigate between these clusters. This is due to the fact that a partial order is induced over these groups of concepts. The properties of these groups are then combined in a tool allowing the exploration of information in a given triadic context.

In Rudolph, Săcărea e Troancă (2015) the authors combined the triadic exploration technique proposed in (KIS; SACAREA; TROANCA, 2016) with other resources for manipulating dyadic and triadic contexts. Among these functionalities is possible to apply basic scale operations (GANTER; WILLE, 1989) and context projections, as well as calculation of dyadic and triadic concepts, creation, and manipulation of dyadic lattices. The Figure 7 shows use of the tool.

Although there are works that allow the visualization and exploration of concepts, few have explored the triadic approach. In Rudolph, Săcărea e Troancă (2015) the authors proposed a tool for viewing and navigating triadic lattices using projections made from concepts selected by the user. Information is obtained through the exploration of dyadic lattices, which introduce the idea of directly reachable concepts, allowing the user to navigate through all lattices generated from an initial triadic concept.

The proposed approach in this dissertation differs from the works previously analyzed, for the best of our knowledge, it's the first approach that allows the user to explore the triadic diagram through queries. The queries allow the user to find the corresponding concepts being searched (identical match) or the concepts closest to the defined pattern. Also, this work uses the ordering and the graphical representation of triadic lattices proposed by Missaoui et al. (2020), which allows grouping the triadic concepts into a more concise and simplified hierarchical structure. The queries of up to three dimensions find two sets of concepts that can be viewed in the hierarchy as upper and lower bounds in the static diagram provided by T-iPred.

4 TRIADIC CONCEPT APPROXIMATION

Due to the complexity of the ternary relation, the conceptual lattice it is not easy to read or manipulate, mainly when considering a large triadic context. The complexity of using these diagrams as a meaningful tool for data exploration is even more problematic because, to the best of our knowledge, there is no tool in the literature capable of automatically generating Hasse diagrams for triadic contexts, once the diagram must be built manually.

In recent work, Missaoui et al. (2020) shows how to produce the diagram of the dyadic lattice for triadic concept analysis. This diagram is produced by an adaptation of the dyadic algorithm called iPred (BAIXERIES et al., 2009). The diagram produced by this new algorithm has the same elegance and expressiveness as the diagrams used in classical FCA. Despite the abstraction and expressiveness of this representation for the triadic lattice, the necessity to extract information in the triadic context is still a challenge for the TCA community. Working with large databases can produce large lattice diagrams, even in the dyadic case. In TCA, this problem is more complex due to the number of concepts produced by triadic data (since we can have more than one dimension associated with the same extent). There are efforts in the literature (RUDOLPH; SĂCĂREA; TROANCĂ, 2015) to abstract the complexity of triadic data through projections and mechanisms to explore the data as previously mentioned.

In this work, we propose a new approach to query and extract information from triadic contexts, through concept approximation to visually explore triadic concepts.

The concept approximation aims to find the exact triadic concept given by a query, if such exists, or to approximate a set of elements from that context. That is, given a set (objects, attributes, conditions), the approximation allows us to find the concepts that have exactly these elements or that are closer to them, considering the conceptual hierarchy.

4.1 Query

In this section, we define the query structure used in this work. Essentially, the triadic concept analysis is an extension of the classic FCA (LEHMANN; WILLE, 1995). One of the main differences, that gives rise to the whole field of study called TCA, is the third dimension frequently called conditions or modus. So, by definition, the Triadic FCA

has three dimensions known as objects, attributes, and conditions.

In this work, we propose an approach to approximate triadic concepts from elements of one of these dimensions or any combination of them. By combination, we say that the query can contain one dimension (objects, attributes, or conditions), two dimensions (objects/attributes, attributes/conditions, objects/conditions), and finally, all the three dimensions specified.

Given a triadic context $\mathbb{K} = (K_1, K_2, K_3, Y)$, a query is defined as a triple (A_1, A_2, A_3) where $A_1 \subseteq K_1$, $A_2 \subseteq K_2$ and $A_3 \subseteq K_3$, and the result is a subset of triadic concepts of $\mathfrak{T}(\mathbb{K})$, known as the closest concepts of (A_1, A_2, A_3). The triple representing a query may contain at least one of the three dimensions defined in the triple. The symbol *?* is used to represent the absence of the information related to the dimension represented by the component of the triple (A_1, A_2 or A_3). The number of missing dimensions in a query defines the type of the query. For example, a query with two missing dimensions is called a one-dimension query because the information used to approximate concepts is related to only one dimension.

To approximate concepts using one, two, and three dimensions we define three types of queries respectively called one-dimensional, two-dimensional, and three-dimensional queries. All the possible triples can be seen below:

- One-dimensional
  - $(A_1, ?, ?)$
  - $(?, A_2, ?)$
  - $(?, ?, A_3)$
- Two-dimensional
  - $(A_1, A_2, ?)$
  - $(A_1, ?, A_3)$
  - $(?, A_2, A_3)$
- Three-dimensional
  - (A_1, A_2, A_3)

4.2 T-iPred

Faced with the difficulty of reading and navigating the original structure of the triadic diagram as mentioned before and pointed out in (RUDOLPH; SĂCĂREA; TROANCĂ, 2015), in Missaoui et al. (2020) one of the main contributions is an adaptation of

the iPred algorithm, originally proposed by Baixeries et al. (2009) to calculate precedence links among triadic concepts. Due to the complexity of the original graphic diagram, proposed in Lehmann e Wille (1995), exploring triadic data using visual tools is still an object of investigation. Therefore, in order to visualize and navigate triadic concepts, together with its quasi-order in relation to the extents, the authors proposed a Hasse diagram in which each node represents a set of triadic concepts with the same extent.

Such an adaptation, called T-iPred, considers concepts according to an increasing order in the size of the extent instead of the size of the intent and, therefore, creates the diagram in a bottom up way. Since the *poset*[*] obtained is not closed under the intersection of the extent, intent or modus, the authors modified the initial iPred algorithm to discard extent intersections that do not represent actual extents of existing triadic concepts.

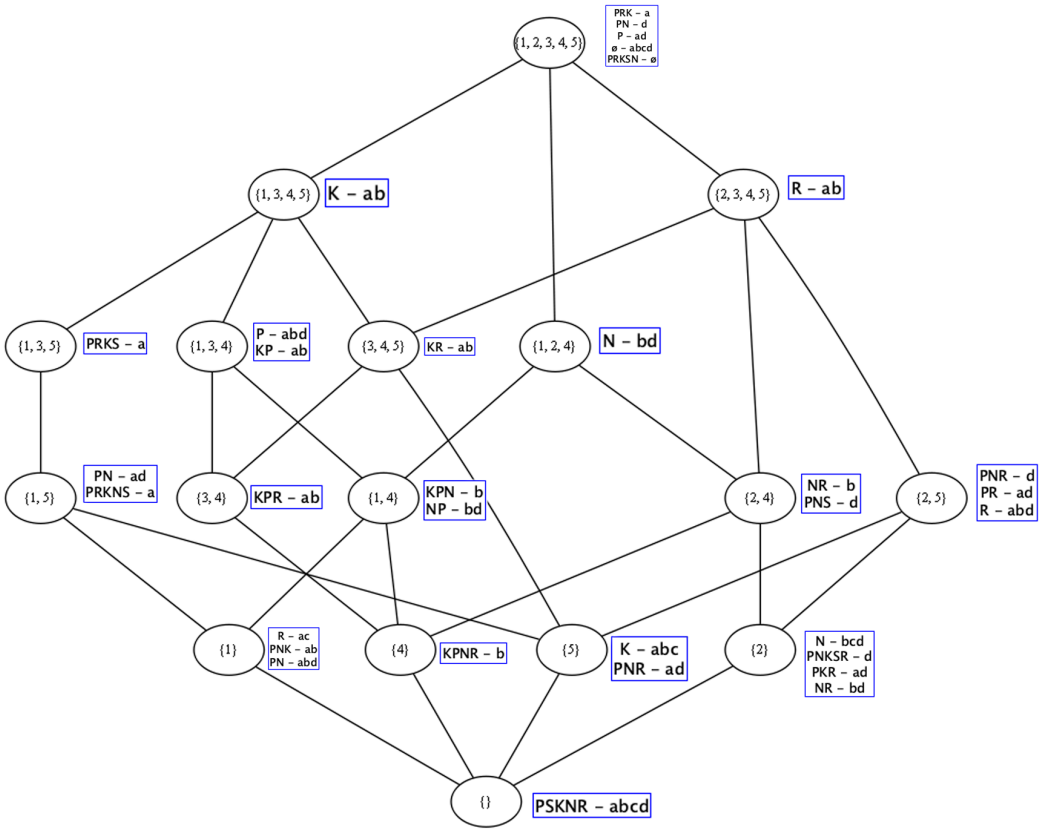
Consider the triadic context $\mathbb{K} := (K_1, K_2, K_3, Y)$ presented in Missaoui e Kwuida (2011) representing orders from customers in $K_1 = \{1, 2, 3, 4, 5\}$, from suppliers $K_2 = \{\mathbf{P}eter, \mathbf{N}elson, \mathbf{R}ick, \mathbf{K}evin, \mathbf{S}imon\}$ of the products $K_3 = \{$***a****ccessories,* ***b****ooks,* ***c****omputers,* ***d****igital cameras* $\}$ and the incidence relationship Y is showed in Table 10. The set of all triadic concepts $\mathfrak{T}(\mathbb{K})$, ordered by the T-iPred algorithm (Missaoui et al. (2020)) produces the line diagram of Figure 8.

Table 10 – Triadic context of customers.

$\mathbb{K}^{(1)}$	P				N				R				K				S			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
1	×	×		×	×	×		×	×		×		×	×		×	×			
2	×			×		×	×	×	×	×		×	×							×
3	×	×		×				×	×	×			×	×			×			
4	×	×		×		×		×	×	×			×	×						×
5	×			×	×			×	×	×		×	×	×	×		×			

[*]partially ordered set

Figure 8 – Trilattice from PNRKS context.



The diagram produced by T-iPred algorithm is constructed based on the extent of each triadic concept, therefore, each node of the lattice represents a group of concepts associated with the same extent. In Figure 8 the rounded nodes represent the extent of a group of concepts and the boxes on the right side of each node, represent the intent and conditions respectively, of each concept attached to the node. For example, the node with extent $\{1, 4\}$ has two concepts associated with it, they are: $(14, KPN, b)$ and $(14, NP, bd)$.

4.2.1 Upper and lower covers

The T-iPred algorithm uses the concepts of border and face defined in (BALCÁZAR; TÎRNĂUCĂ, 2011) and for that, the notion of *upper covers* and *lower covers* is also used.

According to (BALCÁZAR; TÎRNĂUCĂ, 2011), let $(P, \leq)$ be a partial order $\leq$ under the set P. If $x \leq y$ (resp. $x < y$) x is said to be below (strictly below) y. If $x < y$ and there is no element between x and y, x is considered a *lower cover* of y, meaning x is an immediate predecessor of y, and y, a *upper cover* of x, respectively, y is an immediate successor to x, and this relationship is denoted by $x \prec y$. The *upper cover* for x can be

written as $uc(x) = \{y \mid x \prec y\}$, and the *lower cover* $lc(x) = \{x \mid y \prec x\}$.

Let $\mathfrak{L} = (C, \leq_1)$ be a partial order where each node represents all the triadic concepts associated with the same extent, and $\leq_1$ the relation of order induced by *quasi-order* $\lesssim_1$. Such an order is not a complete lattice since the intersection under the nodes (in this case the extents) does not necessarily result in an extent that belongs to a triadic concept. For each node x, there is a border $B(x)$ that consists of the maximum of elements with respect to the order $\leq_1$. That is, $y \in B(x) \Leftrightarrow ext(x) > ext(y)$ (BALCÁZAR; TÎRNĂUCĂ, 2011).

The concept hierarchy produced by T-iPred is used to approximate concepts, as will be described in the following sections, and the term triadic lattice will be used from now on to refer to the partial lattice obtained by T-iPred. The set of concepts found by the query is divided into two sets of upper and lower covers in the triadic lattice. The answer to an approximation is visualized in the line diagram obtained by T-iPred to explore the concepts related to the elements found by the query and provide a visual mechanism for exploring the triadic context.

4.3 Concept Approximation

The concept approximation is done by using a triple (A_1, A_2, A_3) where each element represents a subset of objects, attributes, and conditions respectively, from a triadic context.

The result of a query can be divided into two sets called *upper covers* and *lower covers* or upper and lower bounds. These two sets can be displayed in the triadic lattice as the result of an approximation. The upper bound indicates the immediate successors of the query, and the lower bound indicates the immediate predecessors. Together, the two sets locate the query result at some level in the hierarchy. The upper coverage set is obtained by applying the derivation operators (Section 2.2.2) to the elements of the query, and then the concepts present in this set are used to obtain the lower bound.

The Table 10 represents the triadic context where customers are denoted by objects (1, 2, 3, 4, 5), buying products denoted by conditions (**a**cessories, **b**ooks, **c**omputers, **d**igital cameras), from suppliers (**P**eter, **N**elson, **R**ick, **K**evin, **S**imon) (which in turn denote the attributes of the context). An incidence in the triadic context, such as $(1, P, a)$, indicates that the consumer **1** bought the object **a** from the supplier **P**.

A user may be interested in finding out who are the suppliers and which products customers *1* and *3* bought together. This query can be expressed by the triple $(13, ?, ?)$ which represents a one-dimensional query where only the object dimension is presented. Note that the attributes and conditions are not specified in this example, as the search

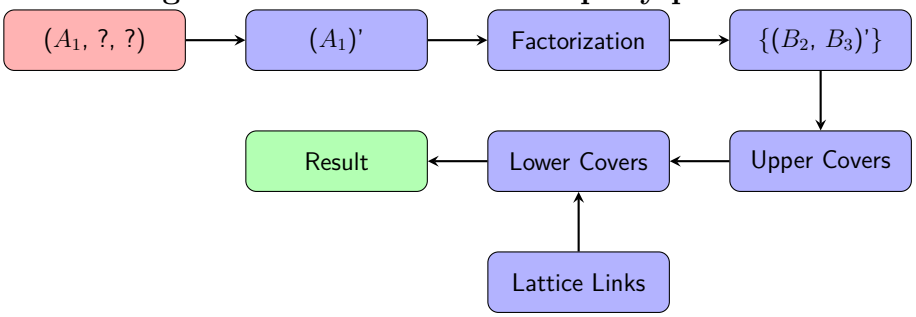
constraint is to find patterns related to objects *1* and *3* together. It is possible to further restrict the query to specify the pattern sought. The triple $(13, P, ?)$ indicates that it must be found in which concepts the objects *1* and *3* are relate to the attribute P. The concepts related to this triple can be found using a two-dimensional query, where the two dimensions related to the desired behavior are fixed. Similarly, a three-dimensional query can be defined by a triple where all three dimensions are present, such as $(13, P, a)$. In the three-dimensional case, if the pattern sought is exactly a triconcept, then this is the answer to the query.

4.4 One-dimensional Query

The one-dimensional query performs the concept approximation of one of the three triadic dimensions. Given a triple in which only one dimension is known, the one-dimensional query process consists of finding the triadic concepts closest to the elements provided by the query. Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context, the set of concepts denoted by $\mathfrak{T}(\mathbb{K})$ and an order defined by the trilattice $\mathfrak{L} = (\mathfrak{T}(\mathbb{K}), \leq_1)$ computed by T-iPred algorithm, the query can be formally defined as a triple, where only a dimension is specified. As the query for a dimension can be done by using the objects, attributes or conditions, three different triples are defined, one for the extent $(A_1, ?, ?)$, intent $(?, A_2, ?)$ and conditions $(?, ?, A_3)$.

Figure 9 shows the process of the one-dimensional query using objects. The first step of the concept approximation is to apply the derivation operator to the given set of elements, in this case, the set of objects A_1, which then must be factored. Factoring is the process of grouping the pairs resulting from the derivation process (previous step) into maximum pairs so that the next derivation is made in a fewer number of pairs. The factoring process is best described as follows: the concepts obtained from the second derivation of the pairs are filtered, keeping only the immediate successors of the approximate triple. The result of this filtering is the upper covers for the triple $(A_1, ?, ?)$, and then they are used to calculate the lower covers through the links produced by the T-iPred algorithm. The result set is produced and depicted in the line diagram - where it is possible to identify the upper and lower limits for the query.

Figure 9 – One-dimensional query process.



To visualize and explore the set of concepts closest to a given triple, we can separate this set into two sets, upper covers and lower covers, which can be hierarchically visualized in the Hasse diagram of the triadic lattice. That is, given an arbitrary query, the concepts closest to the defined triple can be viewed in the lattice to define an upper and lower bound for the query. These limits can bring relevant information such as position in the conceptual hierarchy, making the full view of the diagram unnecessary, since only the query result can be displayed.

The set of concepts closest to $(A_1, ?, ?)$ are those that constitute the upper limit of the triple in the lattice $\mathfrak{L}$. That is, the set of upper covers for a triple where only the objects are defined, will be those that immediately follow A_1 or the concepts that have all the elements of the approximate extent. The immediate successors of this extent have the approximated elements of A_1, if there is any concept in $\mathfrak{T}(\mathbb{K})$ whose extent is identical to A_1 this concept then sets the upper limit of the query. Otherwise, the approximation process identifies the extent that succeeds A_1, by the quasi-order $\leq_1$ in the conceptual hierarchy. Given a triple $(A_1, ?, ?)$ it is possible to identify the immediate successors (upper-covers) by applying the derivation operators as follows:

$$(A_1, ?, ?) \leq_1 (A_1'', B_2, B_3) \quad where \quad \left.\begin{array}{l} B_2 = Intent(F_i) \\ B_3 = Modus(F_i) \end{array}\right\} for\ each\ F_i \in A_1'$$

First A_1 is extended to the set of all attribute/condition pairs shared by all of its elements. Each pair $F_i \in A_1'$ constitutes the intent B_2 and the modus B_3 of each concept of the form (A_1'', B_2, B_3). Then, the extent is calculated by applying the derivation operator to each pair $(B_2, B_3) \in F_i$. According to Wille (1995) the extent $A_1 := (B_2 \times B_3)^{(1)}$ for each triple of the form of (A_1'', B_2, B_3) is a triadic concept and a successor to $(A_1, ?, ?)$.

The set of upper covers are exactly the concepts that immediately follow the approximate triple given the order defined by $\mathfrak{L}$, and are formally described by $(A_1, ?, ?) < (A_1'', B_2, B_3)$, i.e., there is no triadic concept between them, or they are exactly those that have all the elements of the approximate triple and are denoted $(A_1, ?, ?) \leq (A_1'', B_2, B_3)$.

However, the derivation A_1'' can produce extents of concepts of the form (X_1, B_2, B_3) and (Y_1, B_2, B_3) where $(B_2, B_3) \in A_1'$ and $X_1, Y_1 \in A_1''$ and $X_1 \subset Y_1$. The concept of the form (Y_1, B_2, B_3) is a successor of the triple $(A_1, ?, ?)$ but not an immediate one, because its size is greater than that of an immediately successor concept. In other words, given the order inferred by the trilattice $\mathfrak{L}$, the concepts can be put in the form $(A_1, ?, ?) < (X_1, B_2, B_3) < (Y_1, B_2, B_3)$ and then (Y_1, B_2, B_3) is not an immediate successor and therefore it is included in the set of upper covers and consequently in the query result.

Similarly we can define a lower bound for $(?, A_2, ?)$ and $(?, ?, A_3)$ as follows:

$$(?, A_2, ?) \leq_2 (B_1, A_2'', B_3) \quad where \quad \left.\begin{array}{l} B_1 = Extent(F_i) \\ B_3 = Modus(F_i) \end{array}\right\} for\ each\ F_i \in A_2',$$

$$(?, ?, A_3) \leq_3 (B_1, B_2, A_3'') \quad where \quad \left.\begin{array}{l} B_1 = Extent(F_i) \\ B_2 = Intent(F_i) \end{array}\right\} for\ each\ F_i \in A_3'$$

The same reasoning is applied to approximated triples through intent and modus but the application of the derivation operators is done in the set of attributes and conditions respectively. For $(?, A_2, ?)$, each pair (B_1, B_3) of object and attributes in F_i obtained by deriving the attributes provided in the query, constitute the extent and modus of the concepts that will then be derived to obtain the new intent. The same applies to $(?, ?, A_3)$ but the order of the sets is changed once the concepts are built from the conditions.

Once the set of upper covers of the query is obtained (for example, $(A_1, ?, ?)$), we can use it to define a lower bound. Let $(X_1, X_2, X_3) \geq_1 (A_1, ?, ?)$, then the lower bound will contain concepts of the form (Y_1, Y_2, Y_3) such that $(Y_1, Y_2, Y_3) \leq_1 (X_1, X_2, X_3)$ (immediate predecessors of (X_1, X_2, X_3)) and $Y_1 \cap A_1 \neq \emptyset$ - once the $\mathfrak{L}$ triadic lattice is constructed, the lower covers can be calculated by looking at each predecessor link of each concept in the upper bound set previously calculated and check if its intersects with the elements of the query. The same applies for queries using the attributes $(?, A_2, ?)$ and conditions $(?, ?, A_3)$.

4.4.1 Algorithms

Our implementation of the triadic contexts and the derivation operators uses *bitsets* as a data structure to store the incidences. For example, the first incidence $(1, P, a)$ of the customer context (Table 10) represents one bit in the context. Three triadic contexts $K^{(1)}, K^{(2)}$ and $K^{(3)}$ are generated for the derivation process, as defined in Equation 2.5. The derivation processes are made by using the *AND* logic operation in chunks of *bitsets*

in the context.

The Algorithm 1 computes the upper covers for a one-dimensional query. It receives as input only the set defined in the triple A_i such that $i = \{1, 2, 3\}$. Line 1 initializes the result set C as an empty set. In lines 2 and 3, the set A_i is derived according to the dimension i and the result is saved in the auxiliary variable P. The Algorithm 2 is used to factor the pairs in P and produce a set of factored pairs F, extracting the formal concepts from a dyadic context built from the pairs in P.

From lines 4 to 20, the external loop of the algorithm iterates over each pair *(b, c)* in F and calculates the immediate successor concepts for the set A_i of the query. Lines 5 and 6 derive each pair, producing a new set a of elements. A triple t is created using the set a and the pairs *(b, c)*. As the result of the derivation changes according to the dimension being approximated, the auxiliary procedure *makeTriple* organizes the elements of the triple so that the first set of the is an extent, the second intent, and conditions, respectively. Line 7 checks if it is the first iteration of the algorithm and adds t if C is empty.

From lines 8 to 17, the inner loop checks whether the i dimension of the triple t is in fact an immediate successor to A_i. The set C maintains only the concepts immediately successors to A_i in relation to i. This condition is verified from the tests on lines 9 and 10.

When a new t concept is calculated, the internal loop iterates over each previously calculated *t'* concept, checking whether the new concept can be maintained or not. That is, maintaining the constraint that $t[i] \subsetneq t'[i]$ or $t'[i] \subsetneq t[i]$ and $t[i] \neq t'[i]$ where $t[i]$ indicates the position i of the triple t. Line 9 checks the size of both sets and, if it is the same, the triple t inserted in C, as the sets will be subsets of each other, only when they are equal, they will soon be in the same level in the conceptual hierarchy, without prejudice to the condition of not being an immediate successor to the consultation. If the condition is not met, t and *t'* will be at different levels of the network and one of them is not an immediate successor. Line 10 calculates the size of the intersection between the two sets. If the intersection between $t'[i]$ and $t[i]$ is less than the set $t'[i]$, it implies that $t'[i] > t[i]$ and $t[i] \subseteq t'[i]$, indicating that *t'* is a successor to A_i, but not immediate, according to the order $\leq_i$. Lines 11 and 12 remove the non-successor concept and add the new one.

Algoritmo 1: QUERY1D - One-dimension query

Input : A set A_i of dimension i of the query elements.

Output: A Set C of closest concepts of A_i

```
1  C ← ∅
2  P ← Derive(A_i)
3  F ← factorization(P)
4  foreach (B, C) ∈ F do
5      a ← Derive(b, c)
6      t ← makeTriple(a, b, c)
7      if C ≠ ∅ then
8          foreach t' ∈ C do
9              if |t[i]| ≠ |t'[i]| then
10                 if |t[i] ∩ t'[i]| < |t'[i]| then
11                     C ← C \ t'
12                     C ← C ∪ t
13                 end
14             else
15                 C ← C ∪ t
16             end
17         end
18     else
19         C ← t
20     end
21 end
22 if A_i = K_i then
23     if K_j ∉ C then
24         C ← C ∪ makeTriple(K_i, K_j, ∅)
25     end
26     if K_k ∉ C then
27         C ← C ∪ makeTriple(K_i, K_k, ∅)
28     end
29 end
30 return C
```

Algoritmo 2: Factorization

Input : Set $\mathcal{L}$ of pairs

Output: Set B of factored pairs

```
1 G = createDyadicContext(L)
2 B = getConcepts(G)
3 return B
```

Finally, from lines 22 to 29, the algorithm checks whether A_i is the entire set of objects, attributes, or conditions in the context. In this case, it checks if the approximation has already found a related concept through A_i where the sets K_j and K_k are included, if it does not exist, it is included. For instance, when A_1 is equal to K_1, it means that the query is looking for a concept whose extent is the entire set of objects K_1 and the concepts $(K_1, \emptyset, K_3)$ and $(K_1, K_2, \emptyset)$ must be added to the result whenever C does not contain these concepts.

Algoritmo 3: Lower-Bound: Computing Lower bound of the one-dimensional query

Input : A set C of concepts where $C \geq_i$ of A_i.
A set A_i of dimension i of the query elements.

Output: A set C' of concepts where $C' \leq_i$ of A_i.

```
1  C' ← ∅
2  foreach c ∈ C do
3  |  links ← predecessors(c)
4  |  foreach link ∈ LINKS do
5  |  |  if link_i ∩ A_i ≠ ∅ then
6  |  |  |  C' ← C' ∪ link
7  |  |  end
8  |  end
9  end
10 return C'
```

The algorithm 3 computes the lower covers of the one-dimensional query. It receives the set C of the immediate successors, previously calculated, and the one-dimensional elements that were approximated. Line 1 creates an empty set C' to save the lower covers. From line 2 to 8, the algorithm iterates over all the approximated concepts. Line 3 uses the auxiliary functions that get the predecessors of a concept c i.e., $link < c$, in the lattice produced by the T-iPred. Then, for every concept *link* in the set *links* of predecessors, the algorithms checks if the i dimension intersects with the query elements. If so, the link is added to C'. Line 10 returns a set of lower covers for the A_i elements.

4.4.2 Example

The first step of the algorithm is to apply the derivation operator corresponding to the dimension queried (Line 2), the result of this process is a set of pairs. These pairs represent all elements that have an incidence relationship with the derived set. Consider the following example of a one-dimensional query $(13, ?, ?)$ to the context represented by Table 10. The result of the derivation of the elements of the query $(1, 3)'$ is the pairs of attributes and conditions listed below. In summary:

$$(1, 3)' = \{(P, a), (P, b), (P, d), (N, d), (R, a), (K, a), (K, b), (S, a)\}$$

The second step is to factor the pairs (Algorithm 2), to group them into maximum pairs. The factorization of the pairs resulting from the derivation of $(1, 3)$ are the pairs $\{(PN, d), (PK, ba), (PKRS, a), (P, abd)\}$.

Factoring is necessary since the next step, in the approximation process, is the application of the derivation operator over the set of pairs previously calculated. The set of pairs must be organized in such a way that the derivation takes place over the largest possible groups of pairs. Pairs, such as $(P, a), (P, b), (P, d)$, can be grouped into a single pair (P, abd) that expresses the same relation in a more concise way without violating the triadic relation. Otherwise, the derivation should be applied to each pair separately, and then the results combined to guarantee that the objects found in the attribute/condition derivation have the three incidences.

The factoring is done by generating a dyadic context $\mathcal{G} := (G, M, B)$ where the set of incidences is given by the pairs obtained from the initial derivation. The maximum grouping of these pairs is obtained by finding the formal concepts of this context, which represent the maximum rectangles of the two-dimensional table, i.e., the largest possible groupings. The set of objects G are all the attributes of the pairs obtained by the derivation and M the set of conditions of these pairs. The incidence relation defined by the set B is given exactly by each non-factored pair. Therefore, $G = \{P, N, R, K, S\}$, $M = \{a, b, d\}$ and $B =$\{(P, a), (P, b), (P, d), (N, d), (R, a), (K, a), (K, b), (S, a)\} where each pair represents an incidence between $g \in G$ and $m \in M$.

The Table 11 represents the dyadic context $\mathcal{G}$ generated by pairs derived from the set {1,3}. Note that the objects in the context are formed by the distinct elements on the left side of the pairs and the attributes by the distinct elements on the right side.

Table 11 – Dyadic context obtained by the derivation of the set {1,3}.

	a	b	d
P	×	×	×
N			×
R	×		
K	×	×	
S	×		

Then, the *Next-Closure* algorithm (GANTER, 2010) is used for compute all the dyadic concepts of $\mathcal{G}$. In total, five concepts are computed (Table 12). Note that, except the *supremum* (first concept) all the rest are exactly the factored pairs needed. Therefore, the result of the factoring process is the list of pairs: $\{(PN, d), (PK, ba), (PKRS, a), (P, abd)\}$.

Table 12 – Formal concepts from $\mathcal{G}$.

	Extent	**Intent**
1	{P,N,K,R,S}	{}
2	**{P,N}**	**{d}**
3	**{P,K}**	**{b,a}**
4	**{P,K,R,S}**	**{a}**
5	**{P}**	**{b,d,a}**

Besides reducing the number of pairs found, allowing better performance of the algorithm, the factoring process uses the strategy of finding maximum rectangles (dyadic concepts) in a dyadic context, preventing the pairs from being combined, decreasing the complexity of the algorithm.

The concepts found in the previous process are exactly the factored pairs obtained in the application of the *prime* operator. That is, they are the maximum attribute/condition pairs related to objects 1 and 3 derived initially. Once grouped, the derivation operator *double prime* can be applied to each pair (attribute / condition). Applying the *double prime* operator to each of the factored pairs we obtain the following result.

With factored pairs, the next step is to apply the derivation operator (Equation 2.7) to each pair (attribute/condition) that will result in a new set of objects. The result can be seen below:

$$(P, abd)' = (134)$$
$$(PN, d)' = (12345)$$
$$(PK, ab)' = (134)$$
$$(PRKS, a)' = (135)$$

The result of this derivation will be a set of objects that are related to the attributes and conditions found previously. Note that the objects found in this derivation are exactly sets of objects that constitute the extent of certain triadic concepts of the context. The result of each derivation is a set of objects that, together with the attribute/condition pairs, form triadic concepts. Thus, four triadic concepts are found $(134, PK, ab)$, $(134, P, abd)$, $(135, PRKS, a)$, $(12345, PN, d)$.

It is possible to notice that all the extents obtained by the approximation process intersect with the objects of the query $(13, ?, ?)$. However, the concept $(12345, PN, d)$ is not an immediate successor to the query, as its extent is a super-set of all other extents found, so this concept is removed from the result.

After removing the non-immediate successor concepts, the set of upper covers for the triple $(13, ?, ?)$ are the concepts $(134, PK, ab)$, $(134, P, abd)$, $(135, PRKS, a)$, considered the closest concepts to the objects $(1, 3)$.

Once the set of immediate successors for $(13, ?, ?)$ is calculated, the query can be incremented by the set of lower covers of the query. To find these concepts, we look for the lower links of each concept in the upper cover set and check if the link intersects with the elements 13}. As shown in Figure 8, the concept $(134, PK, ab)$ and $(134, PK, ab)$ have three predecessors: $(34, KPR, ab)$, $(14, KPN, b)$ and $(14, NP, bd)$. Finally, the concept $(135, PRKS, a)$ has two immediate predecessors, which are $(15, PN, ad)$ and $(15, PRKNS, a)$. At this moment, we have four possible lower covers for the query, so we check if the extent of these concepts intersects with any one of the elements in {1,3}. In this case, all the candidate's extents intersect with {13} and are then added to the query result set.

The Figure 10 displays the result of the one-dimensional query $(13, ?, ?)$ where the upper and lower bounds are highlighted in red.

If the query contains exactly the dimension of some triadic concept (e.g., extent, intent, or conditions), then the concept itself will be the upper cover for this query. For instance, take the one-dimensional query defined by $(?, PRKNS, ?)$ that searches for the closest concepts of the attributes $PRNKS$. The upper bound of the query are the concepts $(2, PRNKS, d)$, $(15, PRNKS, a)$, $(\emptyset, PRNKS, abcd)$ and $(12345, PRNKS, \emptyset)$.

Figure 10 – Result of one-dimensional query (13, ?, ?).

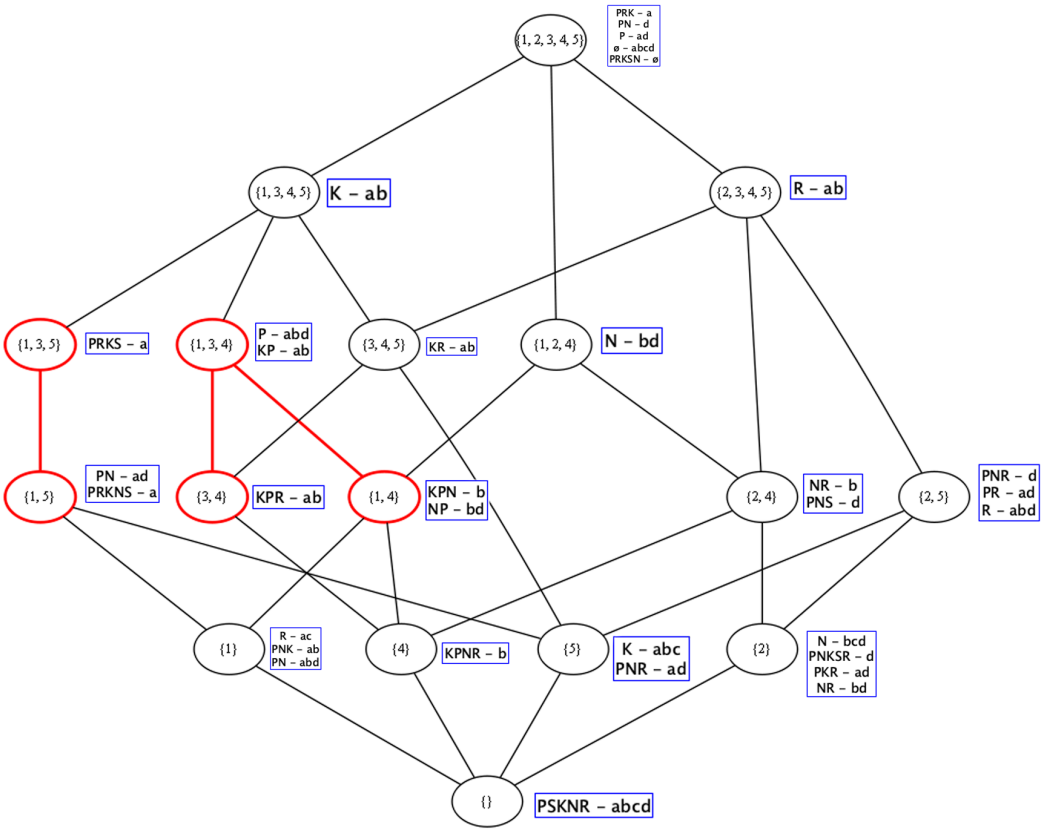


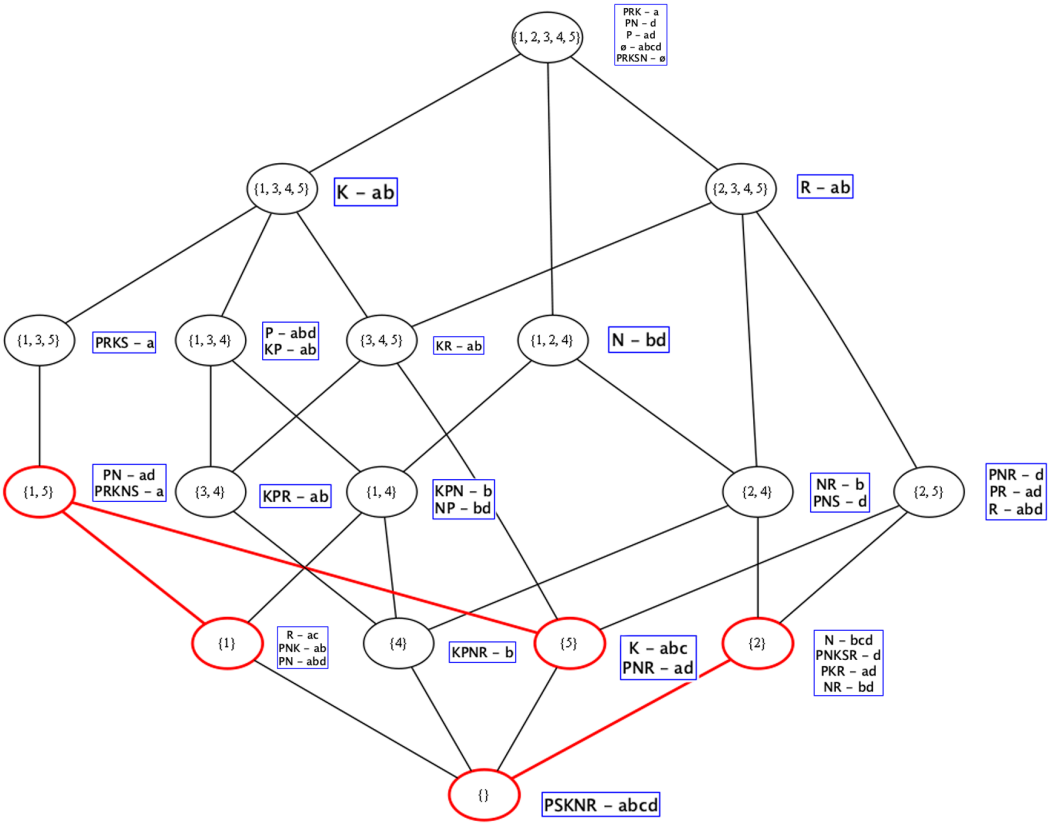
Figure 11 shows the upper and lower covers for the $(?, PRKNS, ?)$ one-dimensional query.

4.4.3 One-dimensional query complexity analysis

As previously described, the query process approximate concepts using one dimension, and it consists of a series of steps. The factorization of pairs in line 3 of Algorithm 1 uses the algorithm *next-closure* (GANTER, 2010) which has computational cost in the worst case $\mathcal{O}(C * (|G| * |M|))$ where C is the number of concepts in the context, $|G|$ the number of objects and $|M|$ the number of attributes. The main loop of the algorithm (Lines 4-21) iterates over the pairs factored and for each pair, the derivation operator is applied. The computational cost in the worst case of the loop is $\mathcal{O}(C*(k*(|X|*|Y|))$ where C is the number of concepts (number of factored pairs - *conceptsPairs*), k is the number of different elements in each pair, $|X|$ and $|Y|$ is the size of the projected dimensions in the $\mathbb{K}^i$ context.

The algorithm 3 uses upper covers found by Algorithm 1 to find the set of immediate predecessors for the one-dimensional query. The algorithm iterates over all the concepts in the upper covers set and for each concept, it checks whether all of its immedi-

Figure 11 – Result of one-dimensional query $(?, \boldsymbol{PRKNS}, ?)$.



ate predecessors intersect with the elements approximated by the query. The complexity of the algorithm related to the number of intersections performed, in the worst case is $\mathcal{O}(n * m)$ where n is the number of upper covers found and m the number of immediate predecessors of each concept. In the worst case, m can contain $|\mathfrak{L}| - 1$ such that $|\mathfrak{L}|$ is the number of concepts present in the trilattice.

4.5 Two-dimensional Query

The two-dimensional query allows the approximation of concepts through two dimensions. The purpose of the query is to find information associated with concepts that are closest to any two triadic dimensions. The two-dimensional query allows a more restricted search than the one-dimensional query defined in Section 4.4. The results are concepts closest to the elements of both approximated dimensions.

The triadic dimensions can be combined according to the interest and nature of the query made by the user. Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context and $\mathfrak{L} = (\mathfrak{T}(\mathbb{K}), \leq_1)$ the triadic lattice whose order is defined by $\leq_1$, the two-dimensional query can be formally defined by a triple where two dimensions will be approximated. According

to the three triadic dimensions, three possible variations of this triplet can be observed such as: $(A_1, A_2, ?)$, $(A_1, ?, A_3)$ and $(?, A_2, A_3)$ where A_1, A_2, A_3 are, respectively, subsets of the objects ($A_1 \subseteq K_1$), attributes ($\mathrm{A}_2 \subseteq K_2$) and conditions ($\mathrm{A}_3 \subseteq K_3$).

The two-dimensional query can find the closest concepts according to: for $X_i \subseteq K_i$ and $X_k \subseteq K_k$ with {i, j, k} = {1, 2 ,3}, let $A_j := X_i^{(i,j,X_k)}$, $A_i := A_j^{(i,j,X_k)}$ and $A_k := (A_i \times A_j)^{(k)}$ (if $i < j$) or $A_k := (A_j \times A_i)^{(k)}$ (if $j < i$). So (A_1, A_2, A_3) is the triadic concept with the property of having the smallest *k-th* component among all triadic concepts (B_1, B_2, B_3) and the largest *j-th* component satisfying the constraint $X_i \subseteq B_i$ and $X_k \subseteq B_k$ according to Wille (1995).

Similarly to the one-dimensional query, the two-dimensional query can be divided into two sets that define an upper and lower bound according to the trilattice $\mathfrak{L}$.

The set of upper covers for a two-dimensional query for objects and context attributes whose triple is given by $(X_1, X_2, ?)$ can be defined as follows:

$$(X_1, X_2, ?) \lesssim (A_1, A_2, A_3) \quad where \begin{cases} A_3 = (X_1 \times X_2)^{(3)} \\ A_1 = (X_2 \times A_3)^{(1)} \\ A_2 = (A_1 \times A_3)^{(2)} \end{cases}$$

That is, given a triple $(X_1, X_2, ?)$, the sets X_1 and X_2 are used to build a concept from the elements of both dimensions. First, the set of objects and attributes is used to find the set A_3 of conditions that share these elements. This new set will constitute the conditions of the approximate concept and is obtained through the operation $(X_1 \times X_2)^{(3)}$. Once the set A_3 is computed, it can be combined with the objects and attributes X_1, X_2 specified in the query to find the extent and intent of a concept that shares all elements of the query.

Then, A_3 is used to find the concept extent set through the operation $(X_2 \times A_3)$ that gives us the set A_1 of objects that have the attributes of the approximate triple and the conditions extended initially. At this point, we have two sets A_1, A_3 generated from the elements of the query. Intuitively the set of attributes is obtained by deriving this pair generating the final set A_2 of attributes that share both the objects in A_1 and the conditions in A_3. Finally, (A_1, A_2, A_3) forms a triadic concept generated from the approximated pair. Note that the order of the second and third derivation can vary. The set A_2 could have been generated first instead of A_1. In this case, the operators would produce the elements of the concept in different orders but producing the same result.

Similarly, it is possible to use the same reasoning to approximate a triple $(?, X_2, X_3)$ from the attributes and conditions as follows:

$$(?, X_2, X_3) \lesssim (A_1, A_2, A_3) \quad where \begin{cases} A_1 = (X_2 \times X_3)^{(1)} \\ A_2 = (A_1 \times X_3)^{(2)} \\ A_3 = (A_1 \times A_2)^{(3)} \end{cases}$$

In this case, we can obtain the concepts starting from a set of attributes and conditions to generate an extent A_1. This set of objects is derived along with the conditions in X_3 to produce a new intent A_2 which ultimately gives rise to the set of conditions A_3 by deriving A_1 and A_2.

The last possible two-dimensional query is defined by the triple $(X_1, ?, X_3)$ that makes the approximation through objects and conditions. The triadic concept can be obtained by applying the derivation operators, as shown below:

$$(X_1, ?, X_3) \lesssim (A_1, A_2, A_3) \quad where \begin{cases} A_2 = (X_1 \times X_3)^{(2)} \\ A_1 = (A_2 \times X_3)^{(1)} \\ A_3 = (A_1 \times A_2)^{(3)} \end{cases}$$

This approach limits the set of immediate successors to an arbitrary triple to just one concept. Once this concept has been calculated, it is possible to identify the immediate predecessor's concepts to constitute a lower cover for the two-dimensional using the $\mathfrak{L}$ triadic lattice.

Let $(A_1, A_2, A_3) \geq_1 (X_1, X_2, ?)$, then the lower bound will contain the concepts of the form (Y_1, Y_2, Y_3) such that $(Y_1, Y_2, Y_3) \leq_1 (A_1, A_2, A_3)$ (immediate predecessors of (A_1, A_2, A_3)) where $Y_1 \cap X_1 \neq \emptyset$ or $Y_2 \cap X_2 \neq \emptyset$. The concepts of the form (Y_1, Y_2, Y_3) immediately precede the successors of the two-dimensional query $(X_1, X_2, ?)$ and that intersect with at least one of the dimensions of the query. Similarly, lower covers can be obtained for triples $(?, X_2, X_3)$ and $(X_1, ?, X_3)$.

4.5.1 Algorithm

The Algorithm 4 receives two sets A_i, A_j where {i, j} = {1,2,3} ($i < j$) representing the two sets of the triple being approximated. Line 1 initializes the output c as the empty set. Lines 2 through 4 apply the derivation operator and use the result to calculate the next set. Line 2 derives the pair (A_i, A_j) and stores the result in A_k. The elements in A_k are related to the elements received as input and their size may vary according to the values of i and j.

For example, if $i = 1$ and $j = 2$, then $k = 3$, in other words, A_3 will be a set

of conditions resulting from the derivation of the objects and attributes received as A_1 and A_2. Then, in line 3 the set A_1 is replaced to the set of objects related to the initial attributes and conditions found in Line 2. The operation $Derive(A_2, A_3)$ produces a new extent from the concept being generated. Line 4 uses the new extent and modus (A_1, A_3) to find the new set of attributes by deriving them. As the result of each derivation can vary based on the values of i, j, and k, line 5 uses the auxiliary function which takes three sets as parameters and returns a triple, where the three elements are a subset of objects, attributes, and conditions, respectively and return the concept c.

Algoritmo 4: Two-dimension query

Input : Two sets A_i, A_j of elements where $A_i \subseteq K_i$ and $A_j \subseteq K_j$.

Output: A concept c immediate successor of A_i and A_j.

1 $c \leftarrow \emptyset \quad A_k \leftarrow \emptyset$

2 $A_k \leftarrow Derive(A_i, A_j)$

3 $A_i \leftarrow Derive(A_j, A_k)$

4 $A_j \leftarrow Derive(A_i, A_k)$

5 $c \leftarrow makeTriple(A_i, A_j, A_k)$

6 **return** c

The Algorithm 5 calculates the lower covers for the two-dimensional query based on the upper covers. The process of approximation through two dimensions proposed in the section 4.5 allows to find only one successor concept to the approximate triple, i.e., the set of immediate successors to a triple where two dimensions are known has only one concept, therefore, only a concept is received as a parameter for calculating the lower covers. Also, two sets A_i and A_j corresponding to the elements approximated by the query are received as parameters, where *i,j* are the respective dimensions.

Lines 1 and 2 start *C'* as the empty set and call the auxiliary function *predecessors* using the upper cover c as parameter. This function initializes the set *links* with the concepts predecessors of c in the conceptual lattice. From line 3 to 5, the algorithm iterates over all of the predecessor links of c and checks whether the dimensions i or j of each link intersect with A_i or A_j, if so the concept is added to *C'*. Finally, line 8 returns *C'*.

Algoritmo 5: Lower-Bound: Computing Lower bound of the two-dimensional query

Input : A concept c where $c \geq_i$ of A_i and $c \geq_j$ of A_j.
Two sets A_i, A_j of elements where $A_i \subseteq K_i$ and $A_j \subseteq K_j$.
Output: A set C' of concepts where $C' \leq_i$ of A_i and $C' \leq_j$ of A_j.

```
1 C' ← ∅
2 links ← predecessors(c)
3 foreach link ∈ LINKS do
4 |   if link_i ∩ A_i ≠ ∅ OR link_j ∩ A_j ≠ ∅ then
5 |   |   C' ← C' ∪ link
6 |   end
7 end
8 return C'
```

4.5.2 Example

Consider the triadic context of customers representing by Table 10. We can define a two-dimensional query through the triple $(35, PK, ?)$ that searches for concepts that relate customers *3* and *5* to suppliers P and K. This query, can be translated by the following question: *Are their concepts related to transactions carried out by customers 3 and 5, involving suppliers P and K?* If there is any concept with the extent {3,5} and the intent {PK}, this concept answers that question through the set of conditions linked to this possible concept, which will be found through the two-dimensional approximation.

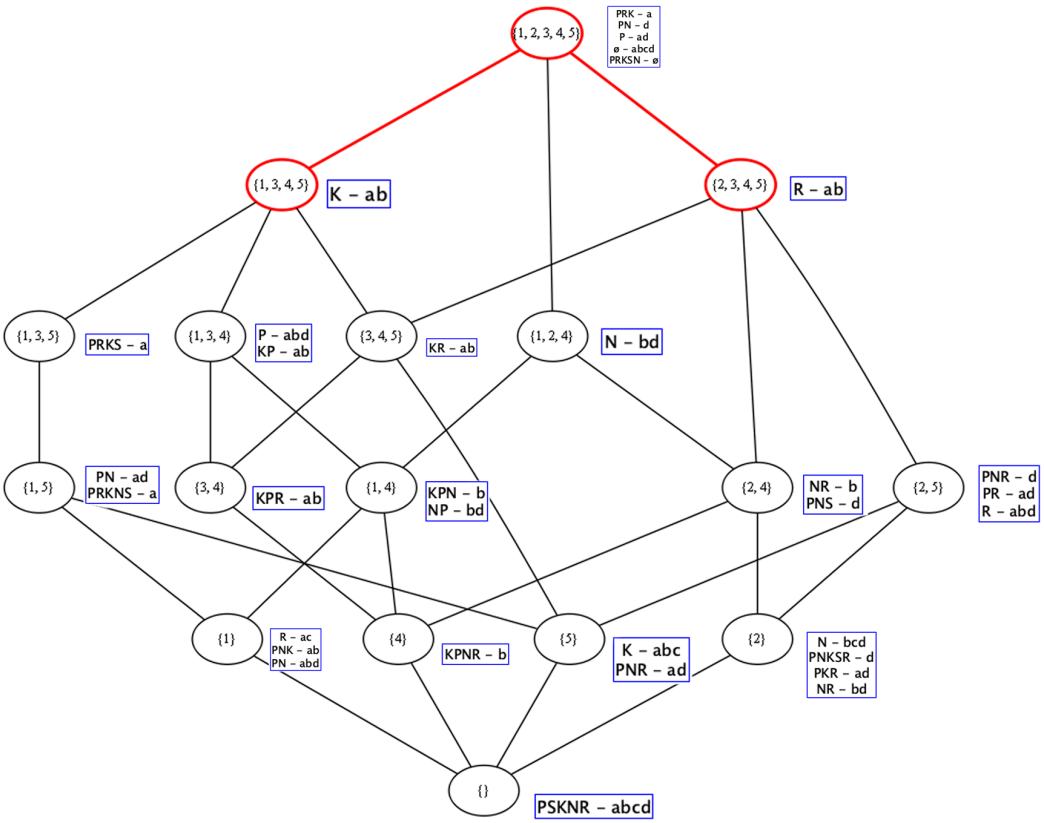
The approximate dimensions are of the objects (i = 1) and attributes (j = 2), so the sets received as parameters are $A_1 = \{35\}$ and $A_2 = \{$ PK$\}$. Line 2 applies the derivation operator over the set of objects and attributes resulting in a set of conditions shared by both A_1 and A_2. The result of the *Derive({35}, {PK})* operation is {a}, so $A_3 = \{a\}$. The set of conditions found will constitute the set of conditions in the concept and these will be used to approximate the other dimensions. Line 3 uses the query attributes and the conditions found previously to find a new set of objects, through the derivation, which is related to both, that is $A_1 = Derive\ (\{PK\}, \{a\})$. The new extent found is formed by the elements {12345}. Finally, in line 4 the extent and conditions are derived and the set of attributes found to form the intent of the approximate concept. A_2 receives {PRK} result of the operation *Derive ({12345}, {a})*. The auxiliary function creates a triple with the sets found and the resulting concept is $(12345, PRK, a)$. Note that the concept obeys the constraint defined in (WILLE, 1995), that is {35} ⊆ {12345} and {PK} ⊆ {PRK}. That is, there is no concept whose extent and intent are the elements {35} and {PK}, but the concept closest to these elements is $(12345, PRK, a)$. The result can be interpreted

according to the initial question. Consumers {35} did not buy from suppliers {PK} alone but each time they both bought from these suppliers, consumers {124} also bought and all transactions also include supplier R.

Once the immediate successors to the two-dimensional query have been calculated, we can use this set to find the lower bound, i.e., the lower covers for the query. As the two-dimensional query only approximates one concept, the calculation of lower covers can be done by looking only at the links of this concept (Algorithm 5). The concept closest to the triple $(35, PK, ?)$ is $(12345, PRK, a)$. It can be notice (Figure 8), that this concept has three lower links. Therefore, the *links* set has the concepts $(1345, K, ab)$, $(2345, R, ab)$ and $(124, N, bd)$. For any of these concepts to be part of the lower set, it is necessary to verify that $\{35\} \cap link_1 \neq \emptyset$ or $\{\text{PK}\} \cap link_2 \neq \emptyset$ for each *link* in the *link* set. In this example, the dimensions checked are {1,2} respectively. Of all three triadic concepts in the *links* sets, only two intersect with the query and will be add to the lower covers set. Therefore, C' has two concepts, which are: $(1345, K, ab)$, $(2345, R, ab)$.

Finally, the query results are graphically displayed in the conceptual lattice, where the upper and lower bounds are highlighted in red as showed in Figure 12.

Figure 12 – Result of the query (35, PK, ?)



4.5.3 Two-dimensional query complexity analysis

The two-dimensional query makes use of the triadic operators (Section 2) to build a triadic concept through the two-dimensional triple. The Algorithm 4 takes both sets and applies the derivation operator three times. The computational cost related to the derivation operator is constant, which in turn has a cost of $\mathcal{O}(n * (|M| * |B|))$ such that n is the number of derived pairs, $|M|$ and $|B|$ the size of the dimensions projected according to the derivation operator, so the cost of the algorithm for calculating the immediate successors is the same as the operator's cost.

As mentioned, a two-dimensional query uses a different approach than a one-dimensional query for the approximation of concepts. The set of successors of a triple with two defined dimensions approximates only one concept. Therefore, the Algorithm 5 has $\mathcal{O}(n)$ complexity in relation to the number of intersections performed between the predecessors of the c concept and the query elements, where n is the total number of c predecessors.

4.6 Three-dimensional Query

The last possible approximation of triadic concepts is made using all three dimensions, through a query using a subset of objects, attributes, and conditions. Unlike other previously defined queries, the approximate triple may constitute a formal triadic concept since it has the three dimensions present. In this case, there is no need to use the approximation process because the concept closest to the triple is the query itself. Therefore, every three-dimensional query checks whether the approximate triple belongs to the set of concepts or not. If it is a concept, the set of immediate successors to the triple is the triple itself, leaving only the calculation of predecessors, if necessary.

If the approximate triple is not a triadic concept, the approximation is made looking for the concepts closest to the query. Using three dimensions, the approximation is made using the two-dimensional query (Section 4.5). In other words, three queries are created based on the pair combination of the triadic dimensions. Each one of the new queries has an empty dimension and its result consists of the concepts closest to the combined pair. The result of the original three-dimensional query will be the joining of the three two-dimensional approximations.

Let $\mathbb{K}$ be a triadic context defined by (K_1, K_2, K_3, Y) where K_1, K_2, K_3 is a set of objects, attributes and conditions respectively, Y a incidence relation and $\mathcal{T}(\mathbb{K})$ the set of all triadic concepts in $\mathbb{K}$. Let q be is a three-dimensional query formally defined by the triple (A_1, A_2, A_3) that searches for the concepts closest to the three dimensions, the first step of the query is to check whether $q \subseteq \mathcal{T}(\mathbb{K})$. If q belongs to the set of concepts,

then this is the concept sought by the query. If q is not a concept, then three new triples are defined, $(A_1, A_2, ?)$, $(?, A_2, A_3)$ and $(A_1, ?, A_3)$, one for each two-dimensional query. Note that all triples generated by the original query have an empty dimension, indicated by the *?* symbol, as defined by the two-dimensional query. The first triple searches for the concepts closest to the object-attribute pair (A_1, A_2), and the concept found resulting from the approximation of this triple will be part of the result of the original query q. The same is done with the subsequent triples $(?, A_2, A_3)$ and $(A_1, ?, A_3)$ that search for the concepts closest to the attribute-condition pairs and object-condition respectively. Finally, the set of concepts closest to the three-dimensional query will be those found by the two-dimensional approximations.

Similarly, the set of immediate predecessors can be calculated for the three-dimensional query. Once the successors can be calculated, as previously described, the lower bound of the query can be computed by looking at the lower links of the successor concepts and identifying which ones intersect with one of the three triadic dimensions.

4.6.1 Algorithms

The three-dimensional query makes use of the two-dimensional query if necessary. The Algorithm 6 receives as input a triple (A_1, A_2, A_3) and a set of triadic concepts $\mathcal{T}(\mathbb{K})$ and outputs a set C' of triadic concepts closer to the query. Line 1 initializes the response set as empty. Line 2 checks whether the approximate triple is a triadic concept since the three dimensions consulted can form a concept. If the triplet belongs to the set of triadic concepts in the $\mathbb{K}$ context, then it is enough to return this concept, since the concept closest to the triplet is the triplet itself. Otherwise, between lines 6 and 8, the algorithm makes three consecutive calls to the procedure described in Algorithm 4. Line 6, makes a call to the function *QUERY2D* passing as parameter the objects and attributes of the queried triple. As shown in Section 4.5, the approximation of concepts through two dimensions produces a concept immediately successor to the approximate pair. Therefore, the set C receives the concept closest to the elements (A_1, A_2). The subsequent lines invoke the same procedure but changing the dimensions passed as parameters to produce all possible combinations. The C set is increased with the result of each two-dimensional approximation. Finally, line 10 returns the set of concepts closest to the triple (A_1, A_2, A_3).

Algoritmo 6: QUERY3D - Three-dimension query

Input : A triple (A_1, A_2, A_3) where $A_1 \subseteq K_1$, $A_2 \subseteq K_2$ and $A_3 \subseteq K_3$.
A set $\mathcal{T}(\mathbb{K})$ of triadic concepts.

Output: A set C of the closest concepts of (A_1, A_2, A_3).

```
1  C ← ∅
2  if (A1, A2, A3) ⊆ T(K) then
3  |   C ← (A1, A2, A3)
4  end
5  else
6  |   C ← C ∪ QUERY2D(A1, A2)
7  |   C ← C ∪ QUERY2D(A2, A3)
8  |   C ← C ∪ QUERY2D(A1, A3)
9  end
10 return C
```

The computation of the set lower covers for the three-dimensional query is similar to the computation performed for the queries of one and two dimensions except for the number of comparisons made to check whether the concept intersects with one of the three triadic dimensions. The Algorithm 7 receives the set C of successor concepts to the three-dimensional query and the query defined by the triple (A_1, A_2, A_3). Line 1 initializes the set C' that will be used as an output, containing the predecessor concepts. The external loop iterates over all the c concepts in C (from lines 2 to 9). For each c concept, the algorithm searches for all the predecessor links of this concept through the auxiliary function *predecessors*. The loop between lines 4 and 8 checks whether for each *link* found, any of the three dimensions intersect with the query in the respective dimension. If there is an intersection, *link* is inserted into the set *C'*. Finally, the algorithm returns the predecessor concepts.

Algoritmo 7: Lower-Bound: Computing Lower bound of the three-dimensional query

Input : A set C of concepts where $C \geq$ of (A_1, A_2, A_3).
The three-dimensional query (A_1, A_2, A_3).

Output: A set C' of concepts where $C' \leq$ of (A_1, A_2, A_3).

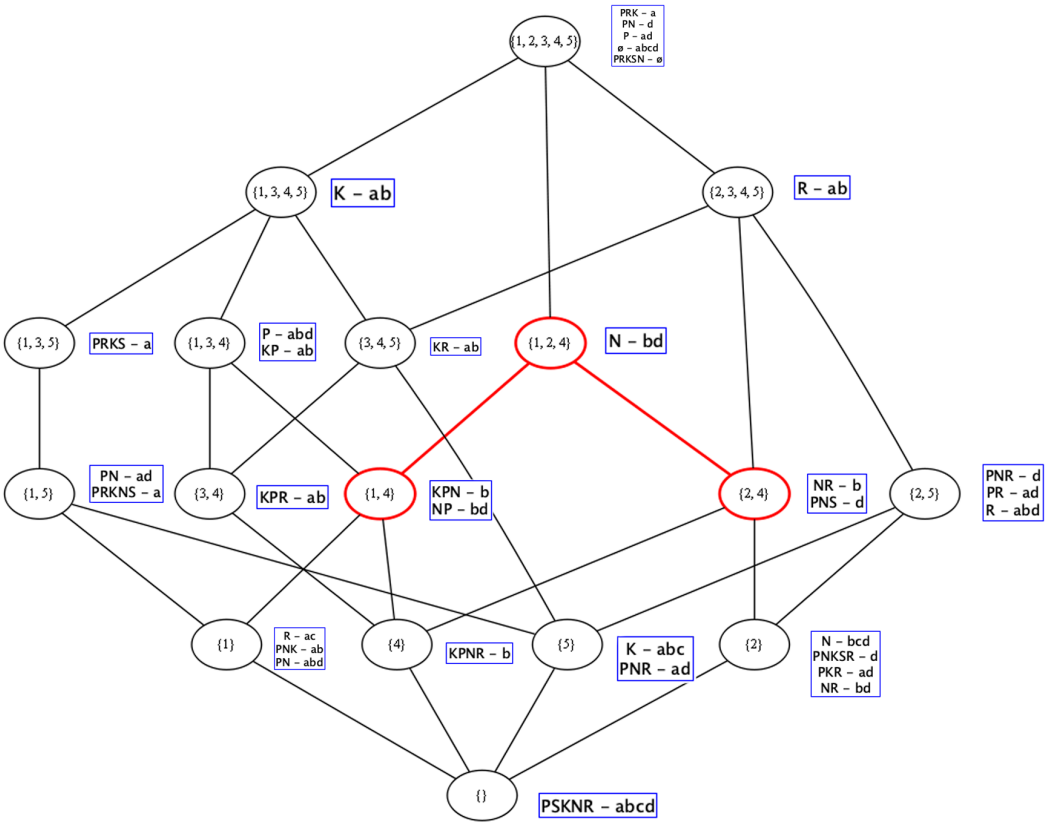
```
1  C' ← ∅
2  foreach c ∈ C do
3  |  links ← predecessors(c)
4  |  foreach link ∈ LINKS do
5  |  |  if link_1 ∩ A_1 ≠ ∅ OR link_2 ∩ A_2 ≠ ∅ OR link_3 ∩ A_3 ≠ ∅ then
6  |  |  |  C' ← C' ∪ link
7  |  |  end
8  |  end
9  end
10 return C'
```

4.6.2 Example

Using the context represented by the Table 10, consider the following three-dimensional queries, formally defined by the triples $(124, N, bd)$ and $(5, N, b)$. The first query looks for concepts that relate the objects {1,2,4}, attribute {N} and conditions {bd}. The query search for patterns related to purchases made by customers {1,2,4} of the products {bd} from the {N} supplier. The same logic can be extended for the second query that searches through the purchase records of the consumer {5} to the same supplier {N} for the product {b}.

As can be seen in the hierarchy (Figure 8), the triple $(124, N, bd)$ is a formal concept. So the concept closest to the query is the approximate triple itself. In this case, we just use the $(124, N, bd)$ concept to find the lower covers. The $(124, N, bd)$ concept has four predecessors. For each concept, we just check if any of the three dimensions intersect with the elements of the query. Of the four predecessor concepts, two share the extent {14} and the other two the extent {24} according to the triadic lattice. Therefore, both extents {14} and {24} intersect with the query objects {124}, so the four triadic concepts will be added to the lower covers. Figure 13 shows the concepts that make up the set of upper covers and lower covers for the three-dimensional query $(124, N, bd)$.

Figure 13 – Result of the three-dimensional query (124, N, bd)



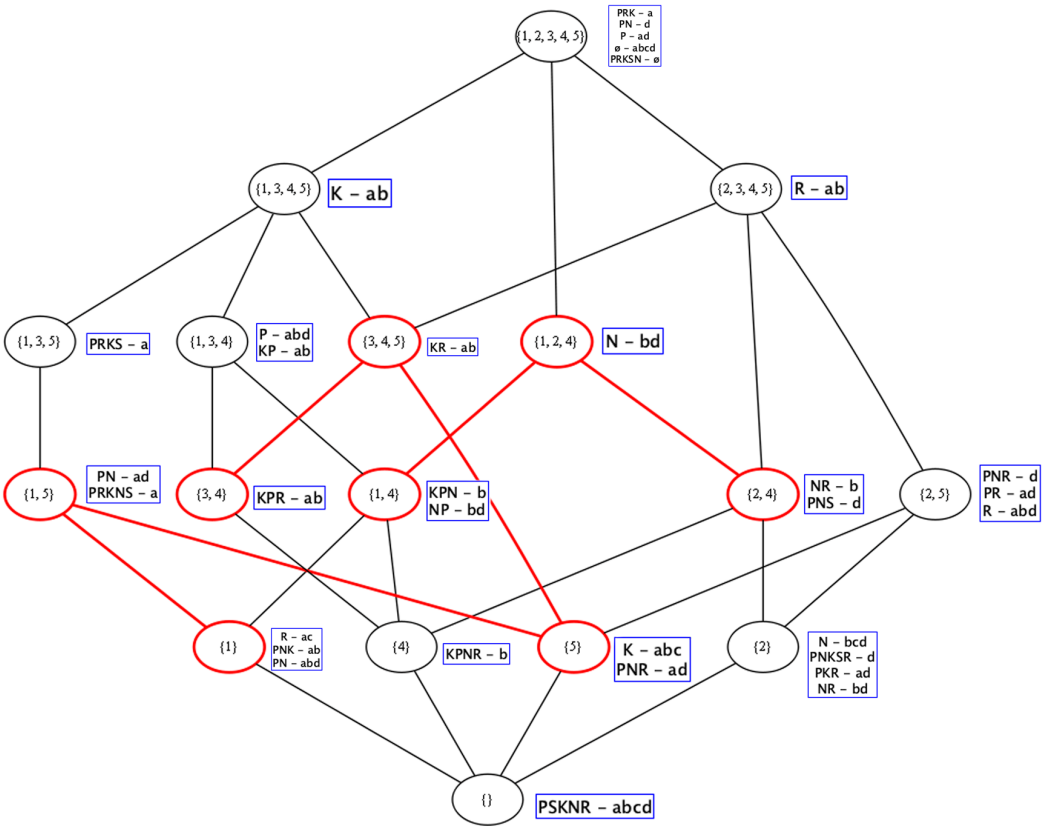
The second query approximated by the triple $(5, N, b)$ is not a triadic concept as can be seen in the conceptual hierarchy. There are no concepts whose extent is formed by the set $\{5\}$. Then, the process of finding the concepts closest to the triple is done using a two-dimensional query. All triadic dimensions are combined in two-dimensional queries so that the result of the original query is all the concepts closest to the combinations of the three dimensions, i.e., (extent, intent), (extent, modus) and (intent, modus).

Three two-dimensional triples are defined: $(5, N, ?)$, $(?, N, b)$ and $(5, ?, B)$. Table 13 shows the result of each query. Therefore, the immediate successors of $(5, N, b)$ are the triadic concepts $(15, PN, ad)$, $(124, N, bd)$ and $(345, RK, ab)$. Using the three concepts found in the set of upper covers it is possible to identify the immediate predecessors, checking all the concepts related to the successors and checking if each of the links intersects with the query. Figure 14 shows the result of the three-dimensional query $(5, N, b)$ highlighted in red.

Table 13 – Bi-dimensional queries produced by the three-dimensional query $(5, N, b)$.

$(5, N, b)$	
Query	Result
$(5, N, ?)$	$(15, PN, ad)$
$(?, N, b)$	$(124, N, bd)$
$(5, ?, b)$	$(345, RK, ab)$

Figure 14 – Result of the three-dimensional query (5, N, b)



4.6.3 Three-dimensional query complexity analysis

The three-dimensional query uses the two-dimensional query to find the closest concepts to three dimensions if the specified triple is not a concept itself. The Algorithm 6 receives the triple (A_1, A_2, A_3) as well as the set $\mathfrak{T}(\mathbb{K})$ of all triadic concepts in the context. From lines 2 to 4 the algorithm checks (constant time) whether the triple (A_1, A_2, A_3) belongs to the set $\mathfrak{T}(\mathbb{K})$. If the queried triple is a concept, then it is returned as a result. If the test fails, from lines 6 to 8 the algorithm makes three calls to the two-dimensional query (algorithm 4) combining the three triadic sets of the triple. Therefore, the complexity

of finding the immediate successors of a three-dimensional query, in the worst case, is equal to the calculation of the immediate successors of the two-dimensional query. Its complexity is denoted by $\mathcal{O}(n * (|M| * |B|))$ such that n is the number of derived pairs and $|M|$ and $|B|$ the size of the dimensions designed according to the derivation operator.

The Algorithm 7 used to find the lower covers of the three-dimensional query is similar to Algorithm 3 which finds the same concepts for a one-dimensional query. The main difference is the number of intersections performed (one for each approximate dimension) and the number of interactions of the algorithm. As the set of immediate successors of the three-dimensional query is restricted, that is, for any three-dimensional query, the set of upper covers may contain a concept, when the triple is the approximate concept itself, or three concepts constructed from the two-dimensional query. Therefore, the main loop of the algorithm (lines 2-9) in the worst case will be executed three times. Then, the three triadic dimensions are checked for one of the predecessors of these concepts. The asymptotic notation of the algorithm in the worst case can be expressed by $\mathcal{O}(m)$ where m is the number of immediate predecessors for each concept. In the worst case, m can contain $|\mathfrak{L}| - 1$ such that $|\mathfrak{L}|$ is the number of concepts present in the lattice.

4.7 Experimental Results

This section presents some experimental results obtained. The main objective is to show the performance of the algorithms applied to a popular data set adapted to the TCA framework by (MISSAOUI et al., 2020).

All tests were performed on an Intel Core i5-9400F 2.90GHz with 16GB of RAM using Java 8 with JDK 1.8. We use the *The Mushroom Data Set*[*] to run our experiments. We created two contexts to test the performance of the algorithms. The formal concepts were calculated using the Data-Peeler algorithm (CERF et al., 2008). The first context is a subset of the original data set containing 2104 objects, 16 attributes, and 8 conditions with 292 concepts. The second context uses the entire data set and has 8416 objects, 32 attributes, and 4 conditions with 1935 concepts. The tests were divided by the type of query. Each subset has a table with results for the same query, but with different numbers of elements being queried according to the maximum dimensions of each context. All elements of the query were randomly generated and all the results presented in *milliseconds* with a confidence level of 95%.

The Tables 14 and 15 show the results for one-dimensional query applied to 2104x16x8 and 8416x32x4 contexts respectively. Both tables were divided into three parts, each column represents one of the possible one-dimensional queries, using objects,

[*]Available at: https://archive.ics.uci.edu/ml/datasets/mushroom

attributes, or conditions. The first column shows run time for queries using objects dimension followed by attributes and conditions.

Table 14 – One-dimensional queries time in the 2104x16x8 context.

Objects	time	Attributes	time	Conditions	time
500	4.39 ± 0.50	2	1.81 ± 1.64	1	27.18 ± 3.90
1000	4.17 ± 0.21	4	0.02 ± 0.01	2	2.79 ± 1.31
1500	3.96 ± 0.20	8	0.01 ± 0.05	4	0.029 ± 0.07
2000	3.55 ± 0.06	16	0.45 ± 0.01	8	0.45 ± 0.01

Table 15 – One-dimensional queries time in the 8416x32x4 context.

Objects	time	Attributes	time	Conditions	time
1000	7.30 ± 0.60	4	0.08 ± 0.07	1	827.10 ± 49.62
2000	6.98 ± 0.05	8	0.03 ± 0.0009	2	131.00 ± 25.03
4000	6.87 ± 0.05	16	0.02 ± 0.002	3	15.71 ± 4.03
8000	6.88 ± 0.11	32	3.32 ± 0.07	4	9.20 ± 0.20

It's possible to notice that the worst empirical result for the one-dimensional query is obtained using only one condition in both contexts. For the 2104x16x8 context, this query took 27.18 ms with a confidence interval of 3.90 ms (Table 14). In the 8416x32x4 context, the same query using a single random condition took 827.10ms with a confidence interval of 49.62ms (Table 15). This scenario can be obtained when the number of pairs generated by the derivation of the query elements is very large, making the dyadic context generated in the factorization process (as explained in section 4.4) to be large and sparse, consequently taking more time to be factored. The derivation of just one element can create a large number of pairs since the only triadic constraint is to be related to a single element being derived.

The Tables 16, 17 show the result for all the possible two-dimensional queries for both contexts. All the results can be seen looking by the specific column and row. For example, the average worst time in the first context can be found in row 4, column 2 on Table 17 that represents the two-dimensional query with 1000 random objects and 4 random attributes (8.79 ms to run with ± 0.73 ms of confidence interval).

Table 16 – Two-dimensional queries time in the 2104x16x8 context.

	Attributes			
Objects	2	4	8	16
500	1.70 ± 0.40	1.39 ± 0.09	1.19 ± 0.06	1.13 ± 0.03
1000	1.11 ± 0.02	1.16 ± 0.03	1.12 ± 0.03	1.09 ± 0.01
1500	1.11 ± 0.04	1.27 ± 0.11	1.25 ± 0.04	1.09 ± 0.02
2000	1.04 ± 0.005	1.06 ± 0.04	0.99 ± 0.007	0.99 ± 0.04
	Conditions			
Objects	1	2	4	8
500	1.16 ± 0.02	1.15 ± 0.01	1.06 ± 0.03	1.07 ± 0.004
1000	1.11 ± 0.04	1.07 ± 0.006	1.10 ± 0.06	1.23 ± 0.03
1500	1.14 ± 0.02	1.13 ± 0.05	1.06 ± 0.007	1.09 ± 0.04
2000	1.10 ± 0.03	1.07 ± 0.03	1.00 ± 0.004	1.03 ± 0.002
	Conditions			
Attributes	1	2	4	8
2	0.13 ± 0.02	0.14 ± 0.006	0.25 ± 0.03	0.24 ± 0.06
4	0.02 ± 0.0003	0.03 ± 0.0002	0.04 ± 0.0002	0.06 ± 0.00009
8	0.02 ± 0.0009	0.03 ± 0.001	0.04 ± 0.0008	0.06 ± 0.0007
16	0.02 ± 0.0003	0.03 ± 0.0002	0.04 ± 0.0002	0.06 ± 0.00009

Table 17 – Two-dimensional queries time in the 8416x32x4 context.

	Attributes			
Objects	4	8	16	32
1000	8.73 ± 0.50	8.53 ± 0.18	8.47 ± 0.18	8.63 ± 0.21
2000	8.79 ± 0.73	8.51 ± 0.48	8.46 ± 0.52	8.24 ± 0.14
4000	8.08 ± 0.12	7.96 ± 0.08	8.35 ± 0.38	8.37 ± 0.17
8000	7.90 ± 0.21	7.64 ± 0.12	7.6 ± 0.07	7.39 ± 0.07
	Conditions			
Objects	1	2	3	4
1000	7.48 ± 0.21	7.19 ± 0.11	7.20 ± 0.06	7.28 ± 0.05
2000	7.91 ± 0.13	7.45 ± 0.04	7.61 ± 0.05	7.62 ± 0.07
4000	8.07 ± 0.10	8.04 ± 0.09	8.12 ± 0.07	8.09 ± 0.09
8000	8.17 ± 0.63	7.93 ± 0.11	7.88 ± 0.12	7.77 ± 0.06
	Conditions			
Attributes	1	2	3	4
4	0.46 ± 0.03	0.25 ± 0.11	0.15 ± 0.0015	0.17 ± 0.0011
8	0.09 ± 0.001	0.12 ± 0.002	0.15 ± 0.001	0.17 ± 0.003
16	0.10 ± 0.003	0.12 ± 0.002	0.15 ± 0.0005	0.17 ± 0.0006
32	0.10 ± 0.002	0.12 ± 0.002	0.15 ± 0.001	0.17 ± 0.0004

Finally, the Tables 18 and 19 show the result of the three-dimensional queries. Each table was split vertically to vary all the three dimensions. Table 18 shows the results for the queries performed on the first context. The number of random objects varies from 500 to 2000, the number of attributes varies from 2 to 8 and conditions from 1 to 8. In Table 19, the number of random objects varies from 1000 to 8000, the number of attributes varies from 8 to 32, and conditions from 1 to 4. The results obtained are quite satisfactory considering that the base used is a real base with a significant number of elements. The worst result obtained in this scenario takes less than a second to complete the concept approximation.

Table 18 – Three-dimensional queries time in the 2104x16x8 context.

	Attributes			
	2			
	Conditions			
Objects	1	2	4	8
500	3.48 ± 0.88	2.35 ± 0.05	2.38 ± 0.06	2.58 ± 0.11
1000	2.31 ± 0.01	2.28 ± 0.04	2.27 ± 0.03	2.34 ± 0.02
1500	2.31 ± 0.04	2.26 ± 0.03	2.43 ± 0.01	2.52 ± 0.03
2000	2.23 ± 0.06	2.14 ± 0.05	2.24 ± 0.01	2.35 ± 0.05
	Attributes			
	4			
	Conditions			
Objects	1	2	4	8
500	2.30 ± 0.07	2.33 ± 0.06	2.36 ± 0.05	2.48 ± 0.07
1000	2.35 ± 0.040	2.26 ± 0.028	2.25 ± 0.016	2.34 ± 0.035
1500	2.53 ± 0.04	2.39 ± 0.05	2.30 ± 0.02	2.31 ± 0.01
2000	2.21 ± 0.03	2.25 ± 0.03	2.25 ± 0.04	2.25 ± 0.02
	Attributes			
	8			
	Conditions			
Objects	1	2	4	8
500	2.34 ± 0.03	2.47 ± 0.08	2.39 ± 0.06	2.41 ± 0.07
1000	2.30 ± 0.02	2.30 ± 0.03	2.27 ± 0.02	2.32 ± 0.01
1500	2.26 ± 0.03	2.31 ± 0.04	2.42 ± 0.07	2.32 ± 0.03
2000	2.16 ± 0.02	2.18 ± 0.02	2.183 ± 0.02	2.28 ± 0.04

Table 19 – Three-dimensional queries time in the 8416x32x4 context.

	Attributes			
	8			
	Conditions			
Objects	1	2	3	4
1000	14.92 ± 0.10	15.00 ± 0.22	15.06 ± 0.25	14.80 ± 0.09
2000	15.55 ± 0.07	15.09 ± 0.12	14.97 ± 0.08	14.96 ± 0.06
4000	15.26 ± 0.06	15.36 ± 0.21	15.48 ± 0.13	15.47 ± 0.09
8000	14.49 ± 0.05	14.58 ± 0.10	14.58 ± 0.05	14.64 ± 0.08
	Attributes			
	16			
	Conditions			
Objects	1	2	3	4
1000	14.57 ± 0.09	14.52 ± 0.07	14.68 ± 0.08	14.69 ± 0.06
2000	15.10 ± 0.09	14.88 ± 0.08	14.95 ± 0.09	15.73 ± 0.53
4000	15.41 ± 0.08	15.34 ± 0.12	15.48 ± 0.07	15.40 ± 0.06
8000	4.42 ± 0.08	14.5 ± 0.09	4.49 ± 0.11	4.49 ± 0.08
	Attributes			
	32			
	Conditions			
Objects	1	2	3	4
1000	14.53 ± 0.09	14.64 ± 0.14	15.03 ± 0.25	14.90 ± 0.17
2000	15.43 ± 0.18	15.21 ± 0.10	15.20 ± 0.08	15.2 ± 0.07
4000	15.48 ± 0.10	15.32 ± 0.05	15.31 ± 0.06	15.38 ± 0.08
8000	14.58 ± 0.16	14.51 ± 0.11	14.51 ± 0.09	14.80 ± 0.19

Although the results showed to be satisfactory using the Mushroom Data set, a more deeply study can be made to stress the algorithms using different triadic contexts varying the densities and the number of concepts, to understand how scalable the technique is. Here, the main goal is to show that, for a triadic context with a similar configuration, i.e., the same number of objects, attributes, conditions, and density, the algorithm can perform very well.

5 CONCLUSIONS

In this dissertation, we propose an approach for the extraction and visualization of information from triadic contexts using queries and triadic lattice representation proposed in (MISSAOUI et al., 2020). These queries, defined as triple (A_1, A_2, A_3), aim to find the closest concepts to a set of objects, attributes or conditions, or any subset of them. The queries are useful to find patterns in triadic contexts especially in cases where the lattice diagram is difficult to use due to the number of concepts generated by the triadic relation. By using the queries, the user can retrieve concepts in the context and explore patterns through concept approximation. The concepts can be visualized in the formal hierarchy and the user can use the approximation to browse the structure.

The query to the triadic diagram can be made using a set of objects, attributes, conditions, or any subset of these and can produce as output two sets of concepts called *upper covers* and *lower covers*. The upper covers have the concepts immediately successors of the query given the conceptual hierarchy. The lower covers are the concepts that immediately precedes the query. Both sets are used to define a lower and upper bound for the query in the diagram and can be displayed graphically. Through these limits, it is possible to identify the concepts closer to the pattern sought and, also, in which part of the triadic diagram the query is located.

We show experimental results using a popular data set called Mushroom, through queries of one, two, and three dimensions applied to a real database with up to 8416 objects, 32 attributes, 8 conditions, and 1935 formal triadic concepts. The results showed to be satisfactory for this scenario since the queries take only a few milliseconds to be performed. The worst-case found remained below one second and is an expected situation since the algorithm used for factorization can be improved. Future work can make a deep analysis of the algorithms in different scenarios, using data sets with a rich number of densities in order to understand more how the technique performs in real scenarios. To improve the complexity of the algorithms, different structures such as *BDD's* (AKERS, 1978) can be used to represent the incidences of the context instead of the *bitsets* are used in this work.

As future work, we plan to create a tool that allows the construction, manipulation, and exploration of triadic contexts. The approximation algorithms proposed here can be used to browse and navigate through the lattice. The graphical visualization of the queries can be improved especially in contexts with a large number of concepts, allowing the user

to have a dynamic browse experience.

REFERENCES

AKERS, S. B. Binary decision diagrams. IEEE Transactions on computers, IEEE, n. 6, p. 509–516, 1978.

BAIXERIES, J. et al. Yet a faster algorithm for building the hasse diagram of a concept lattice. In: SPRINGER. International Conference on Formal Concept Analysis. [S.l.], 2009. p. 162–177.

BALCÁZAR, J. L.; TÎRNĂUCĂ, C. Border algorithms for computing hasse diagrams of arbitrary lattices. In: SPRINGER. International Conference on Formal Concept Analysis. [S.l.], 2011. p. 49–64.

BIEDERMANN, K. How triadic diagrams represent conceptual structures. In: SPRINGER. International Conference on Conceptual Structures. [S.l.], 1997. p. 304–317.

CARPINETO, C.; ROMANO, G. Galois: An order-theoretic approach to conceptual clustering. In: Proceedings of ICML. [S.l.: s.n.], 1993. v. 293, p. 33–40.

CERF, L. et al. Data-peeler: Constraint-based closed pattern mining in n-ary relations. In: SIAM. proceedings of the 2008 SIAM International conference on Data Mining. [S.l.], 2008. p. 37–48.

DAVEY, B. A.; PRIESTLEY, H. A. Introduction to lattices and order. [S.l.]: Cambridge university press, 2002.

GANTER, B. Two basic algorithms in concept analysis. In: SPRINGER. International conference on formal concept analysis. [S.l.], 2010. p. 312–340.

GANTER, B.; WILLE, R. Conceptual scaling. In: Applications of combinatorics and graph theory to the biological and social sciences. [S.l.]: Springer, 1989. p. 139–167.

GANTER, B.; WILLE, R. Formal concept analysis: mathematical foundations. [S.l.]: Springer, 1999.

GLODEANU, C. V. Tri-ordinal factor analysis. In: SPRINGER. International Conference on Formal Concept Analysis. [S.l.], 2013. p. 125–140.

GODIN, R.; MISSAOUI, R.; ALAOUI, H. Learning algorithms using a galois lattice structure. In: IEEE. [Proceedings] Third International Conference on Tools for Artificial Intelligence-TAI 91. [S.l.], 1991. p. 22–29.

JASCHKE, R. et al. Trias–an algorithm for mining iceberg tri-lattices. In: IEEE. Data Mining, 2006. ICDM'06. Sixth International Conference on. [S.l.], 2006. p. 907–911.

JAY, N.; KOHLER, F.; NAPOLI, A. Analysis of social communities with iceberg and stability-based concept lattices. Formal Concept Analysis, Springer, p. 258–272, 2008.

KIS, L. L.; SACAREA, C.; TROANCA, D. Fca tools bundle-a tool that enables dyadic and triadic conceptual navigation. In: FCA4AI@ ECAI. [S.l.: s.n.], 2016. p. 42–50.

LAHCEN, B.; KWUIDA, L. Lattice miner: a tool for concept lattice construction and exploration. In: Suplementary Proceeding of International Conference on Formal concept analysis (ICFCA'10). [S.l.: s.n.], 2010.

LEHMANN, F.; WILLE, R. A triadic approach to formal concept analysis. Conceptual structures: applications, implementation and theory, Springer, p. 32–43, 1995.

MISSAOUI, R.; KWUIDA, L. Mining triadic association rules from ternary relations. In: SPRINGER. International Conference on Formal Concept Analysis. [S.l.], 2011. p. 204–218.

MISSAOUI, R. et al. Pattern discovery in triadic contexts. In: SPRINGER. International Conference on Conceptual Structures. [S.l.], 2020. p. –.

RUDOLPH, S.; SĂCĂREA, C.; TROANCĂ, D. Towards a navigation paradigm for triadic concepts. In: SPRINGER. International Conference on Formal Concept Analysis. [S.l.], 2015. p. 252–267.

VALTCHEV, P. et al. Galicia: an open platform for lattices. In: Using Conceptual Structures: Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03). [S.l.: s.n.], 2003. p. 241–254.

VALTCHEV, P.; MISSAOUI, R. Building concept (galois) lattices from parts: generalizing the incremental methods. In: SPRINGER. International Conference on Conceptual Structures. [S.l.], 2001. p. 290–303.

VALTCHEV, P. et al. Generating frequent itemsets incrementally: two novel approaches based on galois lattice theory. Journal of Experimental & Theoretical Artificial Intelligence, Taylor & Francis, v. 14, n. 2-3, p. 115–142, 2002.

WILLE, R. Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered sets. [S.l.]: Springer, 1982. p. 445–470.

WILLE, R. The basic theorem of triadic concept analysis. Order, Springer, v. 12, n. 2, p. 149–158, 1995.

YEVTUSHENKO, S. A. System of data analysis"concept explorer". In: Proc. 7th National Conference on Artificial Intelligence (KII'00). [S.l.: s.n.], 2000. p. 127–134.