Journal of Universal Computer Science, vol. 28, no. 6 (2022), 591-619
submitted: 27/4/2021, accepted: 4/4/2022, appeared: 28/6/2022 CC BY-ND 4.0

Extracting concepts from triadic contexts using Binary Decision Diagram

Julio Cesar Vale Neves
(Pontificia Universidade Catolica de Minas Gerais (PUC Minas), Belo Horizonte, MG, Brazil
https://orcid.org/0000-0002-0520-9976, juliocesar.neves@gmail.com)

Luiz Enrique Zarate
(Pontificia Universidade Catolica de Minas Gerais (PUC Minas), Belo Horizonte, MG, Brazil
https://orcid.org/0000-0002-3018-888X, zarate@pucminas.br)

Mark Alan Junho Song
(Pontificia Universidade Catolica de Minas Gerais (PUC Minas), Belo Horizonte, MG, Brazil
https://orcid.org/0000-0001-5053-5490, song@pucminas.br)

Abstract Due to the high complexity of real problems, a considerable amount of research that deals with high volumes of information has emerged. The literature has considered new applications of data analysis for high dimensional environments in order to manage the difficulty in extracting knowledge from a database, especially with the increase in social and professional networks. Triadic Concept Analysis (TCA) is a technique used in the applied mathematical area of data analysis. Its main purpose is to enable knowledge extraction from a context that contains objects, attributes, and conditions in a hierarchical and systematized representation. There are several algorithms that can extract concepts, but they are inefficient when applied to large datasets because the computational costs are exponential. The objective of this paper is to add a new data structure, binary decision diagrams (BDD), in the TRIAS algorithm and retrieve triadic concepts for high dimensional contexts. BDD was used to characterize formal contexts, objects, attributes, and conditions. Moreover, to reduce the computational resources needed to manipulate a high-volume of data, the usage of BDD was implemented to simplify and represent data. The results show that this method has a considerably better speedup when compared to the original algorithm. Also, our approach discovered concepts that were previously unachievable when addressing high dimensional contexts.

Keywords: Formal Concept Analysis, Binary Decision Diagram, Triadic Concept Analysis
Categories: E.1, G.0, H.3.1, H.3.2, H.3.3
DOI: 10.3897/jucs.67953

1 Introduction

Initially proposed by Rudolf Wille [Wille, 1982], Formal Concept Analysis (FCA) uses the lattice concept theory to design and analyze conceptual hierarchies from a set of objects and their properties [Davey and Priestley, 2001]. In FCA, concepts that are part of a conceptual hierarchy correspond to abstractions of the problem domain, which contain one or more attributes, describing one or more objects. FCA can be used in knowledge representation and organization, and to identify formal concepts and their dependencies.

Despite the benefit of FCA, which is to transform data into intelligible information (knowledge), it has some limitations, for example, the limitation to represent only the

relationship between attribute and objects. Although dyadic approaches have been successful in many applications [Tang et al., 2015] [Zhang et al., 2013], there have been situations suggesting an extension of formal concepts by a third component. This demand extends FCA, based on a formalization of the triadic relation, connecting formal objects, attributes and conditions. Similar to dyadic contexts, triadic contexts can be represented by three-dimensional cross tables. Triconcepts and trilattices have similar forms with formal concepts and concept lattices, respectively. Therefore, Triadic Concept Analysis (TCA) was proposed to deal with the third element that was added to the dataset (conditions) [Lehmann and Wille, 1995] [Wille, 1996]. TCA can provide an approach to solve existing problems among three-dimensional data, such as Folksonomy, in which users, tabs, and sources have a relation between them (three dimensions).

In 2006, an algorithm that works with the Triadic Formal Context to extract concepts was proposed. In [Jaschke et al., 2006], due to the problem of finding all the triadic concepts using a formal context, they proposed the algorithm known as TRIAS which performs dyadic projections to determine the dilemma. They changed the dyadic idea of mining all the item sets of the dyadic context to a triadic approach [Pei et al., 2000] .

The advance of technology has facilitated the process of collecting and storing data, leading to an increase in processing and storage requirements. This amount of data from various sources makes analysis impractical, especially when it has to generate intelligent information. Therefore, for triadic domains, techniques are needed to process, or assist in the acquisition of triadic information from data.

To improve performance in a high-dimensional context, different approaches that used Binary Decision Diagrams (BDD) were identified. BDDs are structures used to describe Boolean formulas canonically. Using these structures, it is possible to reduce space and simplify the number of operations, and consequently, reduce the number of computational resources utilized to manipulate data.

An algorithm to extract formal concepts using BDD was proposed by [Yevtushenko, 2002]. The BDD was used to express the list of concepts. Contexts with 900 objects x 50 attributes were used and it was proven to have a better runtime only in contexts with high density. Unlike [Yevtushenko, 2002], in [Rimsa et al., 2009] BDD was used to extract concepts, but the goal was to check different BDD libraries and ensure which library was the best option to be used in FCA. The goal was to obtain a set of intentions and its use of brute force methods, and BDDs were applied to represent the extensions. The algorithms that had the BDD structure proven to be more efficient, in terms of runtime, than the original version.

In [Santos et al., 2018], the author's modifications proposed to extract proper implications [Ganter et al., 2005], *ProperIm*, from dyadic formal context, adding BDDs in the data structure. The *ProperImplicBDD* algorithm, proposed by the authors, presented better speedup. They tested several contexts, varying the quantity of attributes and density for 120,000 defined objects. In all the existing literature, BDDs have thus been applied successfully to work as a more efficient data structure in all approaches.

The main objective of this paper is to improve the performance of the TRIAS algorithm using an alternative data structure based on Binary Decision Diagrams (BDD) to represent formal contexts. As a consequence of these performance gains with BDD, manipulating high dimensional formal contexts, in terms of object quantity, attributes and conditions, obtains triadic formal concepts efficiently. The TRIAS algorithm was chosen for this work because it projects, from a triadic context, dyadic contexts to retrieve triadic concepts. Once projected, it enables the usage of a binary structure (BDD) that can easily manipulate data and perform logical operations directly in memory efficiently. Therefore, it is possible to work with a large volume of data allowing us to deal with

high dimensional triadic contexts.

Additionally, we applied the efficient strategies of searching on BDDs in the TRIAS algorithm, specifically in the derivation operators (main operators of FCA and TCA theory) to extract formal concepts. Our approach allows the representation of triadic contexts using BDD in order to store and manipulate high-dimensional contexts efficiently [Akers, 1978].

The results from the TRIAS and the proposed algorithm TRIAS-BDD were compared. The results from the TRIAS-BDD were up to 56% faster than those without the BDD structure. Moreover, the TRIAS-BDD achieved results (concepts) that the original TRIAS was not able to achieve - considering high dimensional contexts (120,000 objects x 15 attributes x 5 conditions).

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the main definitions of FCA, TCA, BDD, TRIAS and SCGaz. The details of the proposed algorithm are described in Section 4. The experiments and results are presented in Section 5. Finally, Section 6 draws conclusions and proposes future work.

2 Related Work

In the literature, there are various works, some of them dating back to 1995 [Wille, 1995] [Lehmann and Wille, 1995], that have focused on triadic context analysis, concepts, diagrams and algorithms.

In [Trabelsi et al., 2012], the authors compared the results from three different triadic algorithms: TRICONS, TRIAS and DATA-PEELER. The same approach was used in [Ignatov et al., 2015]. The authors presented several definitions of optimal patterns for triadic data and the results of experimental comparisons of three triadic algorithms applied to both the real-world and synthetic datasets - including TRIAS.

There are many papers about applications of TCA, especially for analyzing data such as Folksonomy [Ignatov et al., 2011] [Trabelsi et al., 2012]. Recently, the idea of triadic decision contexts was proposed by combining triadic contexts and the rule acquisition method [Tang et al., 2016]. Analogously to FCA, many problems in TCA can be solved, such as acquisitions of triadic concepts, mining rules and the extraction of triadic association and implication rules.

In order to find common patterns in a substantial transactional database, BDDs were applied to record transaction logs as a truth table [Salleb et al., 2002]. Therefore, it was possible to load all the transactions into the main memory, eliminating the process for it to be stored on the hard disk, thereby presenting more efficient results.

In [Santos et al., 2018], alterations to extract implications *ProperIm* in a dyadic formal context were proposed, applying BDDs in the structure to extract and manipulate proper rules. The *ProperImplicBDD* solution had a better execution runtime. The tests changed the density percentage and quantity of attributes for 120,000 objects.

Other important research is to handle the complexity of large contexts in FCA. There are some approaches in the literature that deal with this problem, such as: using proper implications extracted from a reduced concept lattice to represent the behavior of the process being studied in a symbolic and qualitative form [Dias et al., 2020]

However, no triadic approaches have been found that use the efficient manipulation provided by BDDs. Therefore, this work presents an approach for triadic contexts by way of BDDs through dyadic projections.

Planets	Small	Medium	Big	Near	Far	Moon_Yes	Moon_No
Mercury	×			×			×
Venus	×			×			×
Earth	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturn			×		×	×	
Uranus		×			×	×	
Neptune		×			×	×	
Pluto	×				×	×	

Table 1: Formal Context - Planets

3 Background

3.1 Formal Context

Formally, a dyadic context is defined by the triple (G, M, I), where G is defined by the set of objects, M is defined through a set of attributes, and I is the relationship of the objects and their attributes - $I \subseteq G \times M$ [Ganter et al., 2005].

Table 1 exemplifies a formal context. In this example, objects correspond to planets, attributes are the characteristics, and the relationship of incidence represents whether or not the planet has that characteristic. A planet has that characteristic if and only if there is an "X" at the intersection between the row and the respective column.

3.2 Formal Concepts

Let (G, M, I) be a formal context, $A \subseteq G$ a subset of objects and $B \subseteq M$ a subset of attributes. Formal concepts can be defined through the pair (A, B) where $A \subseteq G$ is known as an extension and $B \subseteq M$ is known as an intention. This pair must follow the conditions where $A = B'$ and $B = A'$ [Ganter et al., 2005]. The relation between them is defined through the derivation operator ($'$):

$$A' = \{\, m \in M \mid \forall\, g \in A, (g, m) \in I\}$$
$$B' = \{\, g \in G \mid \forall\, m \in B, (g, m) \in I\}$$

If A $\subseteq$ G, then A' is a set of attributes common to the objects of A. The derivation operator ($'$) can be reapplied in A' resulting in a set of objects again (A''). Intuitively, A'' returns the set of all objects that have the attributes of A' in common; note that $A \subseteq A''$. The operator is similarly defined for the attribute set. If $B \subseteq M$, then B' returns the set of objects that have the attributes of B in common. As a result, B'' returns the set of attributes common to all objects that have the attributes of B in common; consequently, $B \subseteq B''$. For example, the concepts described in Table 2 were extracted from Table 1.

3.3 Triadic Concept Analysis

Initially, TCA was defined by Lehmann and Wille [Lehmann and Wille, 1995] which extends FCA, but a new dimension was added [Wille, 1995].

Objects	Attributes
{Mercury,Venus,Earth,Mars,Jupiter,Saturn,Uranus,Neptune,Pluto}	{}
{Mercury,Venus,Earth,Mars,Pluto}	{Small}
{Earth,Mars,Jupiter,Saturn,Uranus,Neptune,Pluto}	{Moon_Yes}
{Mercury,Venus,Earth,Mars}	{Near,Small}
{Earth,Mars,Pluto}	{Small,Moon_Yes}
{Jupiter,Saturn,Uranus,Neptune,Pluto}	{Far,Moon_Yes}
{Mercury,Venus}	{Big,Near,Small}
{Earth,Mars}	{Small,Near,Moon_Yes}
{Pluto}	{Small,Far,Moon_Yes}
{Jupiter,Saturn}	{Big,Far,Moon_Yes}
{Uranus,Neptune}	{Medium,Far,Moon_Yes}
{}	{Small,Medium,Big,Near,Far,Moon_Yes,Moon_No}

Table 2: Concepts extracted from the Planets Formal Context - Table 1

3.4 Triadic Formal Context

Formally, a triadic context is given by the quadruple (K_1,K_2,K_3,Y), where K_1, K_2 and K_3 is defined by the sets and Y the relation of the K_1, K_2 and K_3, i.e., $Y \subseteq K_1$ x K_2 x K_3, the elements of K_1, K_2, and K_3 are called (formal) objects, attributes, and conditions, respectively, and $(g, m, b) \in Y$ is read: the object g has the attribute m under the condition b. An example of a triadic context is represented in Table 3. This example shows the dataset with 3 dimensions: Customers, Suppliers and Products. We have the Customers *{1,2,3,4,5}* as objects, Suppliers *{P,N,R}* as attributes and Products *{a,b,c,d}* as conditions.

$\mathcal{K}$	P	N	R
1	abd	abd	ac
2	ad	bcd	abd
3	abd	d	ab
4	abd	bd	ab
5	ad	ad	abd

$\equiv$

$\mathcal{K}$	P				N				R			
	a	b	c	d	a	b	c	d	a	b	c	d
1	X	X		X	X	X		X	X		X	
2	X			X		X	X	X	X	X		X
3	X	X		X				X	X	X		
4	X	X		X		X		X	X	X		
5	X			X	X			X	X	X		X

Table 3: Triadic formal context example - Customers {1,2,3,4,5} , Suppliers {P,N,R} and Products {a,b,c,d}

3.5 Triadic Concepts

Let $\mathcal{K}$ = *(K_1,K_2,K_3,Y)* be a triadic context, it gives rise to the following dyadic projected context:

– $\mathcal{T}$ = (K_1, K_2 x $K_3, Y^{(1)}$) where $gY^{(1)}(m, b) \Longleftrightarrow (g, m, b) \in Y^{(1)}$ - Table 4.

The definition described above yields the derivation operators of the dyadic projected context $\mathcal{T}$. Let $\mathcal{K}$ = *(K_1, K_2, K_3, Y)* be a triadic context, for $X_1 \subseteq K_1$ and (X_2,X_3) $\subseteq$ K_2 x K_3, the derivation operator ($'$) is defined by [Lehmann and Wille, 1995]:

$$X_1' = \{(a_2, a_3) \in K_2 \times K_3 \mid (a_1, a_2, a_3) \in Y \; \forall a_1 \in X_1\} \quad (1)$$

$$(X_2, X_3)' = \{a_1 \in K_1 \mid (a_1, a_2, a_3) \in Y \; \forall (a_2, a_3) \in X_2 \times X_3\} \quad (2)$$

$\mathcal{T}$	Pa	Pb	Pc	Pd	Na	Nb	Nc	Nd	Ra	Rb	Rc	Rd
1	X	X		X	X	X		X	X		X	
2	X			X		X	X	X	X	X		X
3	X	X		X				X	X	X		
4	X	X		X		X		X	X	X		
5	X			X	X			X	X	X		X

Table 4: Customers, Suppliers, Products context projected by Suppliers x Products {Pa,Pb,Pc,Pd,Na,Nb,Nc,Nd,Ra,Rb,Rc,Rd}

Triadic concepts can be defined using the derivation operators from Equations 1 and 2. Let $\mathcal{K}$ = *(*K_1,K_2,K_3,Y*)* be a triadic context, $A_1 \subseteq K_1$. If $B = A_1' = (A_2, A_3)$ for $A_2 \subseteq K_2$ and $A_3 \subseteq K_3$. If $B' = (A_2, A_3)' = A_1$, then (A_1, A_2, A_3) is called a triadic concept, where A_1, A_2 and A_3 are called the extent, the intent, and the modus, respectively.

The concept {5,PNR,ad} is an example from Table 3. First, it determines the set of all attributes which all objects of A_2 have under all conditions of X_3; second, A_1 is extended to the set of all objects having all those attributes under all conditions of X_3; and third, A_3 is extended to the set of all conditions under which each of the derived objects has each of the derived attributes. Table 5 presents all the concepts found.

3.6 Binary Decision Diagram

Initially presented by [Akers, 1978] and improved by [Bryant, 1986], BDD provides a representation in a canonical format for a more compressed Boolean formula than normal disjunctive and conjunctive forms. It is substantially more compact than these traditional structure forms and it can be manipulated much more efficiently [Bryant, 1986]. Thus, in general, it is more efficient in handling data.

In terms of structure, it is a directed acyclic graph, and it has two different nodes: nonterminal and terminal. The nonterminal nodes determine the variables of the Boolean formula, and the two terminal nodes are represented by the values 1 or 0. It occurs when the function has a false or true value. The dotted line represents a false transition, and the continuous line represents a positive transition.

Figure 1 represents a binary decision tree for Table 6, and Figure 2 shows an example of how BDD can be used to illustrate the binary decision tree presented in Figure 1. Note that solid lines illustrate that a specific object has that attribute, and dotted-lines illustrates that the object does not have that attribute.

As presented, it is feasible to characterize the same information through a more compact structure than the previous one. In the proposed approach, the goal is to define the formal context as BDD. As example, Equation 3 represents the Boolean formula that corresponds the formal context of Table 7. For a better understanding of the formal context, the object label was replaced by numbers and attributes changed to letters: *James* (1), *Greg* (2), *Catheryn* (3), *David* (4), *Shannon* (5) and *Google* (a), *YouTube* (b),

Object(s)	Attribute(s)	Condition(s)
{}	{P,N,R}	{a,b,c,d}
{5}	{P,N,R}	{a,d}
{4}	{P,N,R}	{b}
{3,4}	{P,R}	{a,b}
{2}	{N}	{b,c,d}
{2}	{N,R}	{b,d}
{2,5}	{R}	{a,b,d}
{2,5}	{P,R}	{a,d}
{2,5}	{P,N,R}	{d}
{2,4}	{N,R}	{b}
{2,3,4,5}	{R}	{a,c}
{1}	{R}	{a,c}
{1}	{P,N}	{a,b,d}
{1,5}	{P,N}	{a,d}
{1,5}	{P,N,R}	{a}
{1,4}	{P,N}	{b,d}
{1,3,4}	{P}	{a,b,d}
{1,2,4}	{N}	{b,d}
{1,2,3,4,5}	{P}	{ad}
{1,2,3,4,5}	{P,R}	{a}
{1,2,3,4,5}	{P,R}	{a}
{1,2,3,4,5}	{}	{a,b,c,d}
{1,2,3,4,5}	{P,N,R}	{}

Table 5: Triadic Concepts extracted from Table 3

X	**Y**	**Z**	**F (X, Y, Z)**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 6: Binary Decision Tree Table Example

Facebook(c), *Wikipedia* (d). As an example, object 1 (*James*) is represented by the path *Google*, $\overline{YouTube}$, *Facebook*, $\overline{Wikipedia}$.

$$f(a,b,c,d) = a\bar{b}c\bar{d} + \bar{a}bcd + \bar{a}b\bar{c}\bar{d} + a\bar{b}\bar{c}d + \bar{a}bcd \tag{3}$$

It must be noted that Equation 3 represents all the instances of the formal context (Table 7). The attribute a, without the slash over it, is true and it means that the part of the function is valid. In other words, this object has this attribute a, while $\bar{a}$ means the contrary. The $a\bar{b}c\bar{d}$ part was created to attest the $James$ and $David$ objects, $\bar{a}bcd$ was generated to attest the $Shannon$ and $Greg$ objects, and $\bar{a}b\bar{c}\bar{d}$ to confirm the $Catheryn$ object.

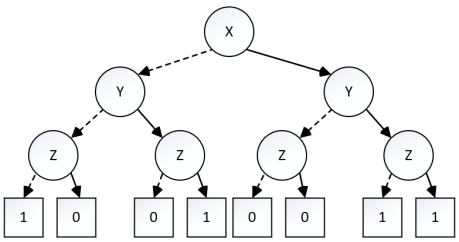

Figure 1: Binary Decision Tree example

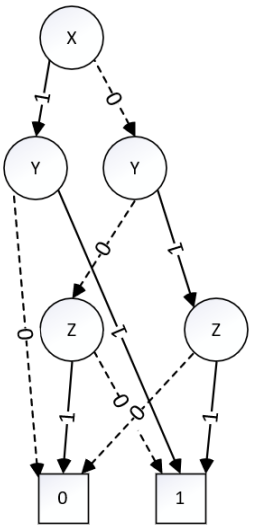

Figure 2: BDD Example from Figure 1

	Google (a)	**Youtube** (b)	**Facebook** (c)	**Wikipedia** (d)
James (1)	X	.	X	.
Greg (2)	.	X	X	X
Catheryn (3)	.	X	.	.
David (4)	X	.	X	.
Shanon (5)	.	X	X	X

Table 7: An example of the formal context

BDD which corresponds to the context presented in Table 7 - described by Equation 3 - is illustrated in Figure 3. Although BDD is a substantially more compact structure than traditional forms, some limitations should be considered during its use since spatial complexity depends on the order in which its variables are added. In [Bryant, 1986] the author uses, as an example, two variables (a and b) that work as a two-bit comparator. The variable a is composed by bits a_1 and a_2 and the variable b by the bits b_1 and b_2.

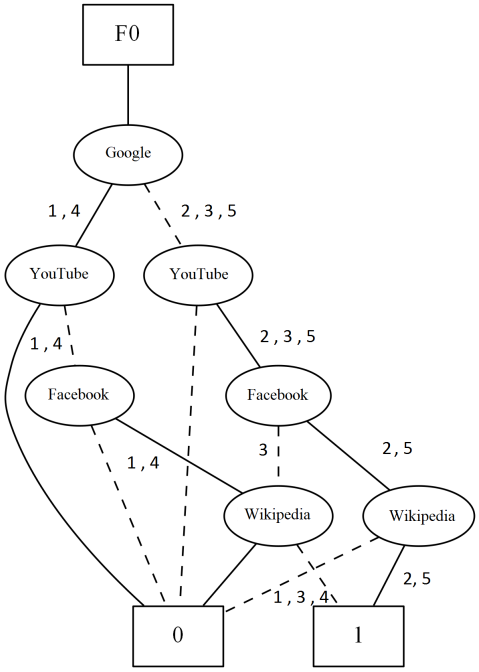

Figure 3: BDD that represents Table 7

The value of a is equal to the value of b, when $a1 = b1$ and $a2 = b2$, otherwise they are different. If the BDD is built using the order $a_1 < a_2 < b_1 < b_2$ the result has 11 nodes. However, if we kept the related variables closer, in the order $a_1 < b_1 < a_2 < b_2$ the resulting BDD would have 8 nodes. In other words, according to [Bryant, 1986], depending on the order in which variables are added to the BDD, the complexity can become exponential. In our work, we used the BDD package which decides the best variable ordering by itself.

In order to build the BDD structure for a formal context, the BDD library CUDD (Colorado University Decision Diagram) was chosen for providing function packages to work with Binary Decision Diagrams (BDDs) besides having more recent updates [Javabdd, 2019].

3.7 TRIAS Algorithm

In [Jaschke et al., 2006], the authors defined the mining problem of all triadic concepts from a formal context. Finally, the TRIAS algorithm was proposed, which aims to resolve the problem through dyadic projections. They assumed the dyadic idea approach of mining all the item sets of the formal context, defined in [Pei et al., 2000] for a triadic context. They also introduced the TRIAS algorithm to compute all the frequent triadic concepts of a *folksonomy* formal context. Given $\mathcal{K} = (K_1, K_2, K_3, Y)$ a triadic context, the TRIAS algorithm builds a dyadic context $\mathcal{T} = (K_1, K_2 \times K_3, Y)$ where columns are related to a tuple (pairs of elements) that belongs to K_2 and K_3 and through projection. Then, the process to extract all concepts.

The TRIAS algorithm was developed using *NextClosure* generating concepts [Jaschke et al., 2006]. Thus, it was decided to use the same approach. However, the original implementation, defined in [Ganter, 2010] uses a bit structure to store objects and their attributes and conditions. The proposed approach was implemented using BDD that will be explained in the following section.

3.8 SCGaz - Synthetic Context Generator

Due to the high complexity to obtain a database from real scenarios, the usage of a synthetic dataset to generate triadic formal contexts is an interesting approach. The real database requires pre-processing, and if not performed correctly, may impact the results directly. Therefore, it is important to use tools that can simulate real data. It is also useful to compare and analyze results among algorithms, as realized in [de Moraes et al., 2016] [Santos et al., 2018].

The *SCGaz* is a synthetic random generator for dyadic formal contexts [Rimsa et al., 2013]. It is feasible to determine the quantity of objects, attributes and density for a formal context. The density values may vary according to the dimensions or can be determined in advance for a given context. The context created is irreducible. In other words, at least one attribute shares one object and vice-versa. In FCA, objects that share the same attributes are considered redundant. In this case, they will not be added to the generated context.

4 Proposed Algorithm

4.1 Applying Binary Decision Diagram in TCA

Given a formal triadic context *(K_1, K_2, K_3, Y)* where K_1 is a set of objects, K_2 a set of attributes, K_3 a set of the conditions, and *Y* the relation of K_1, K_2 and K_3, the projection approach is applied in a triadic context (Table 8), and as a result, a dyadic context is defined *(K_1, $K_2 \times K_3$, Y)* (Table 9).

K_1/K_2-K_3	c_1		c_2		c_3	
	a_1	a_2	a_1	a_2	a_1	a_2
o_1	×			×	×	
o_2		×	×			×
o_3	×		×		×	×

Table 8: Triadic Context (K_1, K_2, K_3, Y)

The proposed projection is the result of a combination of attributes and conditions. Each attribute is labeled according to the condition it belongs to. The retrieval and manipulation of attributes and conditions have to be done by the label defined to each attribute. In the context shown in Table 9, the dyadic incidence is defined by the tuple (o_1, a_1c_1), and it is the same as the triadic incidence given by the triple (o_1, a_1, c_1) of the context shown in Table 8.

After the triadic context is projected to a dyadic context, it can be expressed by a BDD, converting the context from its normal structure to Boolean formulas that will be

$K_1/K_2 \times K_3$	a_1c_1	a_2c_1	a_1c_2	a_2c_2	a_1c_3	a_2c_3
o_1	×			×	×	
o_2		×	×			×
o_3	×		×		×	×

Table 9: Projecting to a Dyadic Context (K_1, $K_2 \times K_3$, Y)

used to generate the analogous BDD. Table 9 presents the triadic context projected to a dyadic context. Equation 4 shows it through disjunctive and conjunctive operations between objects and attributes. The symbols with a slash over the letter represent the attribute as false.

$$f(a_1c_1, a_2c_1, a_1c_2, a_2c_2, a_1c_3, a_2c_3) = (a_1c_1 \cdot a_2\bar{c}_1 \cdot a_1\bar{c}_2 \cdot a_2c_2 \cdot a_1c_3 \cdot a_2\bar{c}_3) \\ +(a_1\bar{c}_1 \cdot a_2c_1 \cdot a_1c_2 \cdot a_2\bar{c}_2 \cdot a_1\bar{c}_3 \cdot a_2c_3) + (a_1c_1 \cdot a_2\bar{c}_1 \cdot a_1c_2 \cdot a_2\bar{c}_2 \cdot a_1c_3 \cdot a_2c_3) \quad (4)$$

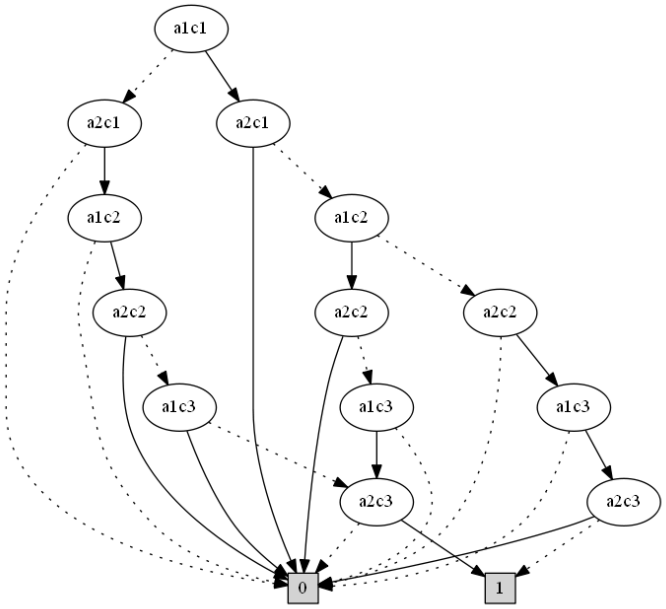

Figure 4: A Formal Context represented by a BDD.

Figure 4 illustrates the triadic context through the dyadic projection that was defined by Expression 4. This representation allows manipulating triadic contexts using a BDD, providing efficient manipulation and storage [Bryant, 1986]. Given a triadic context projected and represented by a BDD, it is possible to provide strategies for recovering objects, attributes and conditions, since any algorithm that uses this representation requires retrieving and changing these elements.

Given the context shown in Table 9, retrieving objects can be performed, as an example, from logical operations OR or AND under Equation 4. If it is necessary to obtain all objects that have the attribute a_1c_2, it has to create a BDD that contains the attribute and perform a logical operation AND between BDDs. Figure 5 shows this operation, which returns a new BDD with objects o_2 and o_3 since both are sharing the same attribute a_1c_2.

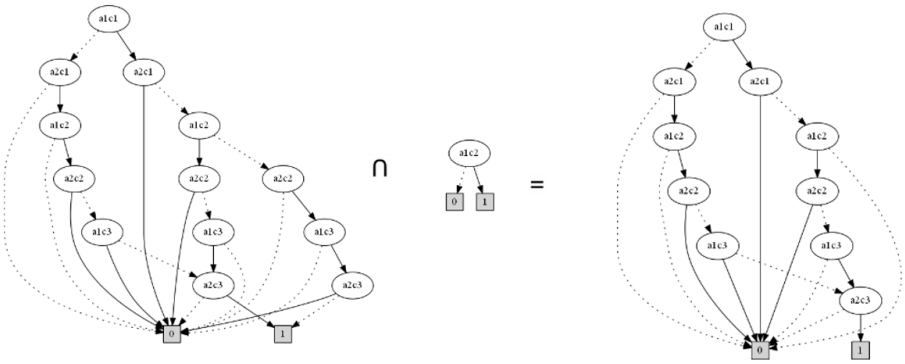

Figure 5: Logical operation - attribute a_1c_2 x BDD Context.

If it is necessary to retrieve all objects that have, as an example, attributes a_1c_1 and a_1c_3, the operation demonstrated previously can be used (Figure 6).

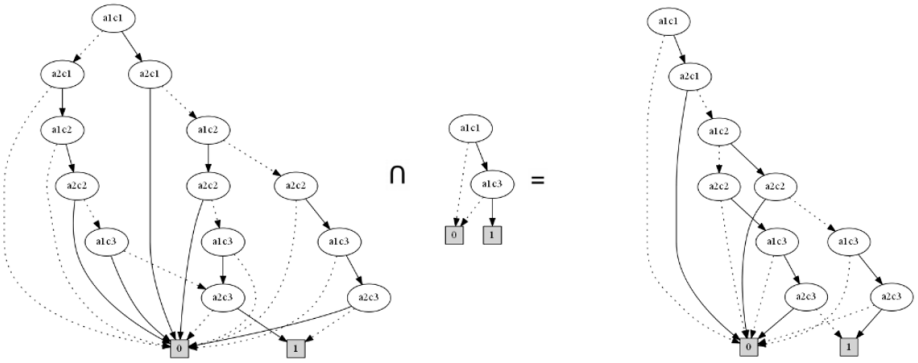

Figure 6: Logical operation - attribute a_1c_1 and a_1c_3 x BDD Context.

As presented in our experimental results (Table 13), TRIAS did not return any concept when submitted to process high-dimensionality contexts - as an example, it did not provide a concept from a context with 120,000 objects, 15 attributes and 5 conditions within 14 days. The proposed algorithm in this work includes the BDD structure which is used to represent the triadic context. **Algorithm 1** gets the input file where the lines

are the incidences that represent a formal triadic context. Additionally, each line defines if that object possesses that attribute x condition. The algorithm runs through all the objects, attributes x conditions and, if an object possesses the attribute x condition, the BDD variable indicated by $(g, m, b) \in I$ is included in the temporary BDD (node is true). Differently, a false node will be added. Lastly, it joins the BDD which represents the objects (BDDTemp) to the BDD which corresponds to the context (ContextBDD).

Algorithm 1: LoadTCAContext() - Function to create the triadic context using Binary Decision Diagram

```
   Input  : Formal Triadic Context (G, M, B, I)
   Output : Formal Triadic Context structured as a BDD (ContextBDD)
 1 BDDTemp = ∅
 2 ContextBDD = ∅
 3 forall g ∈ G do
 4     forall m ∈ M do
 5         forall b ∈ B do
 6             if (g, m, b) ∈ I then
 7                 BDDTemp = BDDTemp.nodeTrue
 8                 else
 9                 BDDTemp = BDDTemp.nodeFalse ;
10             end
11             ContextBDD = ContextBDD ∪ BDDTemp
12         end
13     end
14 end
15 return ContextBDD
```

After the BDD which represents the formal context is created, it is simple to execute some operations. Then, using the advantage of the inherited optimization to the structure shown in **Algorithm 4**. The algorithm provides a BDD containing all the objects that have the attributes and conditions defined. In our example, Table 9, there are conditions and attributes such as c_1a_1, c_1a_2, c_1a_3, c_2a_1, c_2a_2, c_2a_3, c_3a_C1, c_3a_2 and c_3a_3. Note that it requires only a logical AND operation with BDD. As a result, the *"Objects"* variable will contain a BDD that represents objects that shares attributes and conditions c_1a_1, c_1a_2, c_1a_3, c_2a_1, c_2a_2, c_2a_3, c_3a_1, c_3a_2 and c_3a_3.

4.2 Extracting Triadic Concepts through BDD

As explained in subsection 3.7, TRIAS was implemented using the *NextClosure* structure [Jaschke et al., 2006]. The basic logic of Algorithm *NextClosure* was not modified. However, modifications have been implemented to this Algorithm to support the new structure (BDD) - it has been denominated *NextClosureBDD*. The difference is the data type used to represent the formal context, objects, attributes and conditions. All of these structures are represented using BDDs.

The *Concepts Extraction()* function described in Algorithm **2** is the one responsible to identify and extract the formal concepts.

The module responsible for loading the formal context, *LoadTCAContext()*, receives as an input the file in TXT format, which describes the formal context that needs to

Algorithm 2: Concepts Extraction

Require : Formal triadic context
Ensure : Set of all formal concepts
1 $LoadTCAContext(TXTFile, Context)$
2 $X = \theta$
3 **while** $X.size() < Context.NumAttributesConditions.size()$ **do**
4 $\quad X = NextclosureBDD(X, ExtentBDD, IntentBDD)$
5 $\quad ConceptList.add(ExtentBDD, IntentBDD)$
6 **end**

be processed. A bit matrix stores the data. When an object has a certain attribute and condition, the corresponding attribute and condition bit is set to a *true* value. All other operations performed in the context also handle data in this format. It initializes the attribute and condition set (variable X, in **Algorithm 2**) with an empty set. The variable increases in lexicographical order until it has all the attributes and conditions of the context. The Algorithm calls *NextClosureBDD* while the attributes and conditions set is not complete. Internally, Algorithm *NextClosure* calls the function *DoublePrime* (**Algorithm 3**) which is responsible for the double derivation. The computational complexity of the *NextClosure* algorithm is θ ($|2^G \mid G \mid M \mid B \mid$) [Carpineto and Romano, 2005].

Algorithm 3 returns a set of attributes x conditions derived from the provided attribute set. Basically, the algorithm uses two operations. In the first one, the Extent of provided attributes and conditions are computed. In other words, all objects sharing the attributes and conditions provided as parameters are extracted from the formal context. The second function (Intent) computes the final derivation, which returns the intents of the object set returned by the first derivation. The function identifies all attributes x conditions that were shared by the objects obtained from the function extent. This attribute set represents the final derivation of the original attribute set.

Algorithm 3: DoublePrime Algorithm - ($''$) operator

Require : Attribute and Condition set $X \subset (M, B)$
Ensure : X'' = Attribute and Condition set derived from X
1 $X' = ExtentBDD(FormalContext, X)$
2 $X'' = IntentBDD(FormalContext, X')$
3 $returns\ X''$

In order to obtain the extent, **Algorithm 4** was implemented to manipulate BDDs. For each searched attribute and condition, the algorithm process through all structure, eliminating objects which do not have the specified attribute x condition. Therefore, for each attribute and condition, the BDD is reduced to the incidence for the same attribute and condition of the formal context. The computational complexity of the algorithm is $\theta(|G|$ x $|X|)$. Regarding the extraction of the intent, **Algorithm 5** also manipulates attributes and conditions in BDD structure. The algorithm checks the attributes and conditions in the set of objects passed as a parameter, and only attributes and conditions present in an object are passed forward to be tested in the next object.

The function responsible for processing the conjunctions of the two BDDs and

Algorithm 4: ExtentBDD - Retrieves a set of objects that contains the attributes x conditions of the X

Require : Set of attributes x conditions $X \subset (M, B)$ and Formal Context (G,M,B,Y)
Ensure : Set of objects that contains the attributes x conditions in X

$X' = BDDAnd(ContextBDD, BDDAttributesCondition)'$
$returns\ X'$

returning the resulting BDD is *BDDAnd()*. The first BDD is the representation of the formal context. The second BDD is the set of attributes x conditions that will be shared by the context of the object. As described in **Algorithm 1**, the context BDD is formed by a sequence of BDDs representing objects and linked by OR operators, whose attributes and conditions are nodes of that BDD.

Algorithm 5: IntentBDD - Retrieves the set of attributes x conditions shared for all the objects in X′

Require : The objects set $X' \subseteq G$ (BDD)
Ensure : Set of attributes x conditions that were shared by the objects in X′

for *(all* $m \in M$*) & (all* $b \in B$*)* **do**
  $BDDAttributeCondition = (m, b)$
  $BDDtmp = BDDAnd(X', (m, b))$
  if $BDDtmp = X'$ **then**
    $AttributeConditionList = (m, b) \cup AttributeConditionList$
  end
end
for $all\ attributecondition \in AttributeConditionList$ **do**
  $X'' = BDDAnd(X'', attributecondition)$
end
$returns X''$

Algorithm 5 receives the BDD which represents the set of objects. The algorithm loops through all attributes and conditions of the formal context, assembling BDDs for each one. If the concomitance of the attribute x condition BDD and the object BDD results is the same BDD as the original object BDD, it implies that all the represented objects by that object BDD have the attribute x condition by that BDD (attribute x condition). This attribute x condition is added into a separate list (attribute x condition). Afterward, it creates a BDD that contains all the attributes x conditions shared by the objects. This idea is the same used by the original algorithm; the difference is that the solution approach does not test every object individually. The computational complexity of the algorithm is $\theta(|M|\text{x}|B|\text{x}|X'|)$.

5 Experiments and Results Analysis

Our experiments considered two types of triadic contexts - synthetic (subsection 5.1) and real contexts (subsection 5.2). The results (time execution) were compared for both TRIAS and TRIAS BDD Algorithms.

The experiments were run on an Intel Core i7-4790 3.60GHz with 8 cores, 16 threads, 32GB RAM and Ubuntu 18.04 LTS operating system. Our approach was developed using sequential processing.

5.1 Results for the synthetic triadic contexts

The random synthetic generator *SCGaz* was used to generate several contexts. The main objective was to evaluate the performance of TRIAS and TRIAS BDD in the concept extraction. Initially, synthetic triadic contexts with an arbitrary number of objects, dimensions, conditions and densities were generated to be used in both algorithms. In this work, contexts with 500, 1,500, 3,000, 5,000 and 10,000 objects were created with 10, 15 and 20 attributes and 5 conditions. The density was fixed at 30%, 50% and 70% for all contexts.

In order to perform an adequate statistical analysis, 10 different contexts were generated for each testing scenario randomly.

The results presented in Table 10, Table 11, Table 12 and Table 13 correspond to the time execution average of the 10 contexts. In Table 10, Table 11 and Table 12, cells with the "-" symbol show that the algorithm failed to complete the concepts extraction processing within 7 days.

The proposed algorithm (TRIAS BDD) consumed less memory and allowed the algorithm to extract concepts in larger and denser contexts. The cells in bold present the fastest times.

First, the number of objects of each context was increased while maintaining the number of attributes, conditions and density. Second, the density of each context was varied. Lastly, the number of attributes were increased to make our contexts more complex in order to test both algorithms and observe their behaviors in high-dimensional scenarios.

As presented in Table 10, which considers 10 attributes and 5 conditions, it can be seen that the TRIAS BDD algorithm showed a higher speedup than TRIAS, varying between 28% and 36%, for 30% density and between 12% and 22% for 50% density.

Considering the density of 30%, there was no relationship of proportionality between the increase in the number of objects and the percentage variation of speedup. However, when the 50% density is observed, a reduction in speedup is directly proportional to the number of objects, up to 5,000 objects. For 10,000 objects, TRIAS BDD's speedup percentage gain compared to TRIAS was exactly the same achieved for 500 objects (22%).

Analyzing the density of 70%, it can be observed that TRIAS BDD obtained a speedup greater than TRIAS for 500, 1,500 and 5,000 objects, presenting a variation between 6% and 19%. However, for 3,000 objects, TRIAS showed a speedup greater than TRIAS BDD by 5%.

It is important to note that TRIAS did not obtain results for 10,000 objects and a density of 70% in the same context of attributes and conditions, which did not allow the comparison between the algorithms.

As can be seen in Figure 7, considering the context of 10 attributes, 5 conditions and 30% density, for processing context with 500 objects, TRIAS spent 4.55 minutes and TRIAS BDD spent 3.57 minutes, a difference of 0.98 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, TRIAS spent 19.36 minutes and the proposed algorithm spent 13.16 minutes, which is a difference of 6.2 minutes less. For processing context with 3,000 objects, the original algorithm spent 30 minutes and the proposed algorithm spent 21.70 minutes. This difference corresponds to 8.3 minutes in favor of

Context (GxMxB)	Density (%)	Incidences	TRIAS (Minutes)	TRIAS BDD (Minutes)	SpeedUp (%)
500x10x5	30	7,500	4.55	**3.57**	22%
1,500x10x5	30	22,500	19.36	**13.16**	32%
3,000x10x5	30	45,000	30.00	**21.70**	28%
5,000x10x5	30	75,000	55.18	**35.32**	36%
10,000x10x5	30	150,000	119.53	**82.48**	31%
500x10x5	50	12,500	7.09	**5.53**	22%
1,500x10x5	50	37,500	31.62	**25.61**	19%
3,000x10x5	50	75,000	54.85	**46.62**	15%
5,000x10x5	50	125,000	130.37	**114.72**	12%
10,000x10x5	50	250,000	292.19	**227.91**	22%
500x10x5	70	17,500	9.09	**7.37**	19%
1,500x10x5	70	52,500	39.52	**33.20**	16%
3,000x10x5	70	105,000	**62.22**	65.33	-5%
5,000x10x5	70	175,000	128.76	**121.03**	6%
10,000x10x5	70	350,000	-	**269.61**	-

Table 10: Results for TRIAS x TRIAS BDD Algorithms - 10 attributes and 5 conditions

TRIAS BDD. For processing context with 5,000 objects, TRIAS spent 55.18 minutes and TRIAS BDD spent 35.32 minutes, which corresponds to a difference of 19.86 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, TRIAS spent 119.53 minutes while TRIAS BDD spent 82.48 minutes, corresponding to a difference of 37.05 minutes.

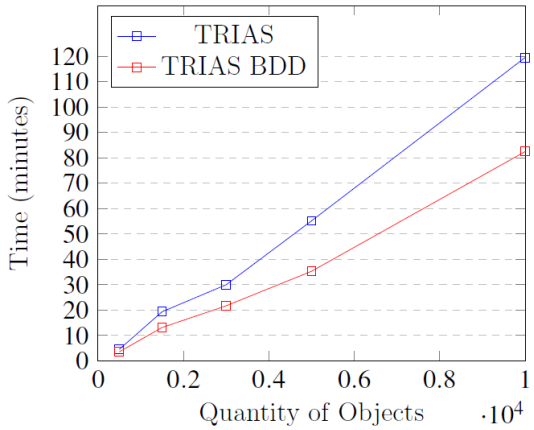

Figure 7: Contexts with 10 attributes, 5 conditions and 30% density

As shown in Figure 8, considering the context of 10 attributes, 5 conditions and 50% density, for processing context with 500 objects TRIAS spent 7.09 minutes and TRIAS BDD spent 5.53 minutes, a difference of 1.56 minutes in favor of the proposed

algorithm. For processing context with 1,500 objects, TRIAS spent 31.62 minutes and TRIAS BDD spent 25.61 minutes, which corresponds to a difference of 6.01 minutes less for the proposed algorithm. For processing context with 3,000 objects, TRIAS spent 54.85 minutes and TRIAS BDD spent 46.62 minutes. This difference corresponds to 8.23 minutes in favor of TRIAS BDD. For processing context with 5,000 objects, the original algorithm spent 130.37 minutes and the proposed algorithm spent 114.72, which corresponds to a difference of 15.65 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, TRIAS spent 292.19 minutes while TRIAS BDD spent 227.91 minutes, corresponding to a difference of 64.28 minutes.

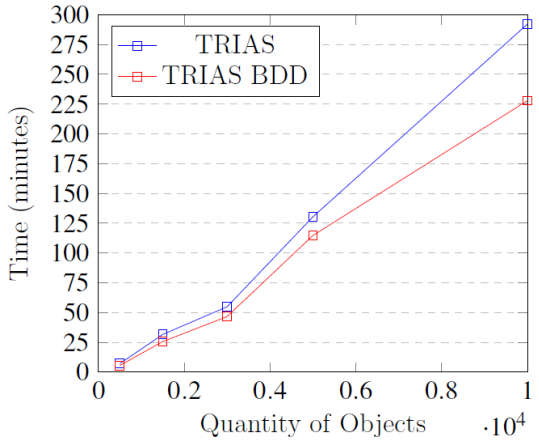

Figure 8: Contexts with 10 attributes, 5 conditions and 50% density

As can be seen in Figure 9, considering the context of 10 attributes, 5 conditions and 70% density, for processing context with 500 objects, TRIAS spent 9.09 minutes and TRIAS BDD spent 7.37 minutes, a difference of 1.72 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, the original algorithm spent 39.52 minutes and the proposed algorithm spent 33.20 minutes, which corresponds to a difference of 6.32 minutes less. For processing context with 3,000 objects, TRIAS spent 62.22 minutes and TRIAS BDD spent 65.33 minutes. This difference corresponds to 3.11 minutes in favor of TRIAS. For processing context with 5,000 objects, the original algorithm spent 128.76 minutes and the proposed algorithm spent 121.03, which corresponds to a difference of 7.73 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, the TRIAS BDD spent 269.61 minutes. TRIAS did not generate results within 7 days. Thus, it was not possible to compare the algorithm's results.

As presented in Table 11, it can be seen that the TRIAS BDD algorithm presented a higher speedup than TRIAS, with variations between 19% and 22%, for 30% density and between 5% and 18% for 50% density. It is important to point out that TRIAS did not show results, at densities - 30% and 50%, for 10,000 objects, considering 15 attributes and 5 conditions. Therefore, it was not possible to compare the algorithm's results.

Analyzing the density of 70%, it is observed that the TRIAS BDD obtained results for the five contexts with the quantity of the following objects: 500, 1,500, 3,000, 5,000 and 10,000. However, TRIAS only presented results for 500 and 1,500 objects. For

these values, TRIAS BDD showed a higher speedup than TRIAS, by 22% and 19%, respectively.

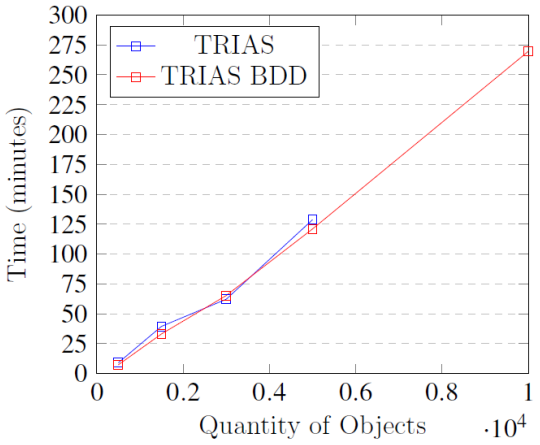

Figure 9: Contexts with 10 attributes, 5 conditions and 70% density

Context (GxMxB)	Density (%)	Incidences	TRIAS (Minutes)	TRIAS BDD (Minutes)	SpeedUp (%)
500x15x5	30	11,250	42.68	**34.39**	19%
1,500x15x5	30	33,750	212.48	**166.60**	22%
3,000x15x5	30	67,500	376.20	**297.49**	21%
5,000x15x5	30	112,500	768.8	**595.97**	22%
10,000x15x5	30	225,000	-	**1,348.12**	-
500x15x5	50	18,750	53.20	**50.28**	5%
1,500x15x5	50	56,250	284.56	**256.36**	10%
3,000x15x5	50	112,500	575.94	**488.09**	15%
5,000x15x5	50	187,500	1,564.42	**1,282.31**	18%
10,000x15x5	50	375,000	-	**3,155.67**	-
500x15x5	70	26,250	122.35	**95.48**	22%
1,500x15x5	70	78,750	583.78	**474.26**	19%
3,000x15x5	70	157,500	-	**839.92**	-
5,000x15x5	70	262,500	-	**1,738.24**	-
10,000x15x5	70	525,000	-	**4,183.66**	-

Table 11: TRIAS x TRIAS BDD Algorithms Results 15 attributes and 5 conditions

In all contexts with 15 attributes, the TRIAS BDD was more efficient in all five object variations with speedup up to 22%. However, in contexts with 10 attributes and 5 conditions, the BDD implementation was faster for contexts with 30% and 50% densities. For density equal 70%, 3,000 objects, 10 attributes and 5 conditions the original algorithm

was 5% faster. The main benefit provided by the BDD was to manipulate a more complex data structure.

As can be seen in Figure 10, considering the context of 15 attributes, 5 conditions and 30% density, for processing context with 500 objects TRIAS spent 42.68 minutes and TRIAS BDD spent 34.39 minutes, a difference of 8.29 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, the original algorithm spent 212.48 minutes and the proposed algorithm spent 166.60 minutes, which corresponds to a difference of 45.88 minutes less. For processing context with 3,000 objects, TRIAS spent 376.20 minutes and TRIAS BDD spent 297.49 minutes. The difference corresponds to 78.71 minutes in favor of TRIAS BDD. For processing context with 5,000 objects, the original algorithm spent 768.8 minutes and the proposed algorithm spent 595.97, which corresponds to a difference of 172.83 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, the TRIAS BDD spent 1,348.12 minutes. TRIAS did not generate results within 7 days. Thus, it was not possible to compare the algorithm's results.

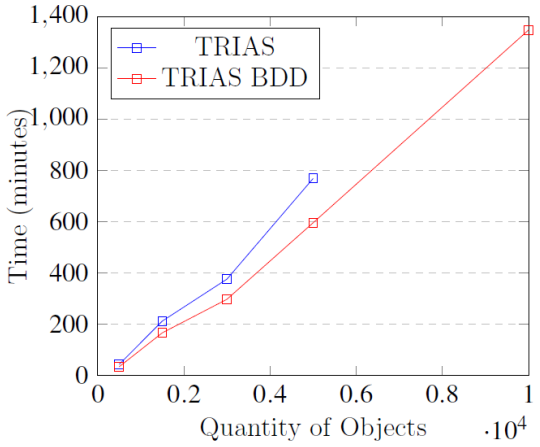

Figure 10: Contexts with 15 attributes, 5 conditions and 30% density

As presented in Figure 11, considering the context of 15 attributes, 5 conditions and 50% density, for processing context with 500 objects TRIAS spent 53.20 minutes and TRIAS BDD spent 50.28 minutes, a difference of 2.92 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, the original algorithm spent 284.56 minutes and the proposed algorithm spent 256.36 minutes, which is a difference of 28.2 minutes less. For processing context with 3,000 objects, TRIAS spent 575.94 minutes and TRIAS BDD spent 488.09 minutes. The difference corresponds to 87.85 minutes in favor of TRIAS BDD. For processing context with 5,000 objects, the original algorithm spent 1,564.42 minutes and the proposed algorithm spent 1,282.31, which corresponds to a difference of 282.11 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, the TRIAS BDD took 3,155.67 minutes. TRIAS did not generate results within 7 days. Thus, it was not possible to compare the algorithm's results.

As can be observed in Figure 12, considering the context of 15 attributes, 5 conditions

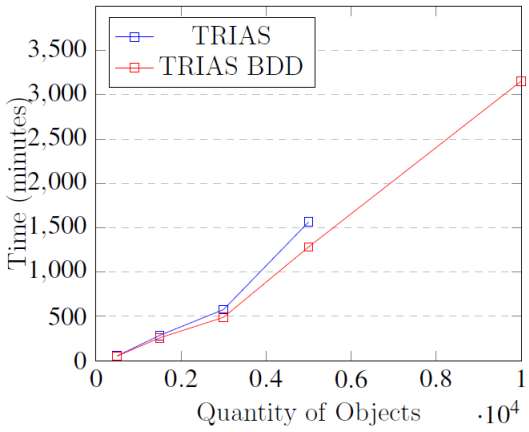

Figure 11: Contexts with 15 attributes, 5 conditions and 50% density

and 70% density, for processing context with 500 objects TRIAS spent 122.35 minutes and TRIAS BDD spent 95.48 minutes, a difference of 26.87 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, the original algorithm spent 583.78 minutes and the proposed algorithm spent 474.26 minutes, which corresponds to a difference of 109.52 minutes less. For processing context with 3,000, 5,000 and 10,000 objects, TRIAS BDD spent 839.92, 1,738.24 and 4,183.66 minutes, respectively. TRIAS did not generate results within 7 days for 3,000, 5,000 and 10,000 objects. Thus, it was not possible to compare the algorithm's results.

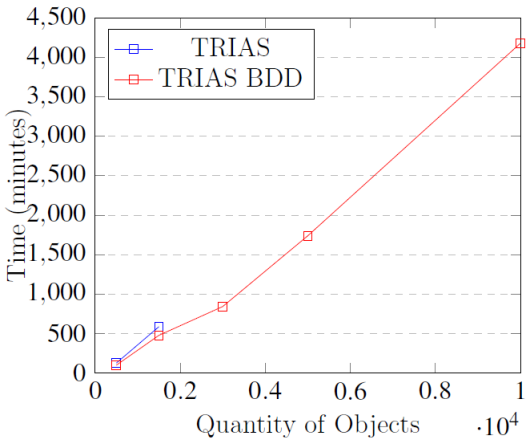

Figure 12: Contexts with 15 attributes, 5 conditions and 70% density

As presented in Table 12, it can be seen that the TRIAS BDD algorithm showed a higher speedup than the TRIAS for contexts with 500, 3,000 and 5,000 objects, corre-

sponding 6%, 22% and 30%, for density equal to 30%. For context with 1,500 objects, TRIAS had a speedup 4% higher than TRIAS BDD. TRIAS did not obtain results for the processing context with 10,000 objects, with 30% density for 20 attributes and 5 conditions. Thus, it was not possible to compare the algorithm's results.

Analyzing the results for 50% density, it is observed that the proposed algorithm presented a percentage of speedup higher than the original algorithm in 9% for context with 500, 16% for context with 1,500, 19 % for context with 3,000 and 56 % to context with 5,000 objects. It was not possible to perform comparative analysis for 10,000 objects in view of the fact that the original algorithm did not present any result.

Analyzing the same results for context with 70% density, it is observed that the TRIAS algorithm presented processing results only for contexts with 500 and 1,500 objects. For these results, TRIAS BDD showed a higher speedup than TRIAS, at 34% and 36%, respectively.

The concept extractions performed in contexts that have more than 250,000 incidences, TRIAS BDD was able to process contexts that the original implementation was not able to handle. Thus, the use of the BDD was mandatory in these cases to enable concept extraction. For instance, contexts with 5,000 objects, 20 attributes, 5 conditions and 70% density - only TRIAS BDD was able to extract the concepts.

As a summary, the execution time for TRIAS became faster than TRIAS BDD in only two contexts (3,000 objects x 10 attributes x 5 conditions x 70% density and 1,500 objects x 20 attributes x 5 conditions x 30% density). However, the speedup in both cases was 5% and 4%, respectively, when compared to TRIAS BDD. On the other hand, TRIAS BDD was faster in all remaining contexts (42 of 45 in total).

Context (GxMxB)	Density (%)	Incidences	TRIAS (Minutes)	TRIAS BDD (Minutes)	SpeedUp (%)
500x20x5	30	15,000	54.56	**51.29**	6%
1,500x20x5	30	45,000	**271.00**	281.84	-4%
3,000x20x5	30	90,000	479.95	**374.36**	22%
5,000x20x5	30	150,000	1,293.28	**903.89**	30%
10,000x20x5	30	300,000	-	**2,462.38**	-
500x20x5	50	25,000	70.93	**64.55**	9%
1,500x20x5	50	75,000	379.41	**318.70**	16%
3,000x20x5	50	150,000	767.93	**622.02**	19%
5,000x20x5	50	250,000	2,885.89	**1,279.88**	56%
10,000x20x5	50	500,000	-	**5,259.46**	-
500x20x5	70	35,000	223.56	**147.31**	34%
1,500x20x5	70	105,000	958.98	**614.45**	36%
3,000x20x5	70	210,000	-	**1,119.89**	-
5,000x20x5	70	350,000	-	**2,317.66**	-
10,000x20x5	70	700,000	-	**7,578.21**	-

Table 12: TRIAS x TRIAS BDD Algorithms Results 20 attributes and 5 conditions

As presented in Figure 13, considering the context of 20 attributes, 5 conditions and 30% density, for processing context with 500 objects, TRIAS spent 54.56 minutes and TRIAS BDD spent 51.29 minutes, a difference of 3.27 minutes in favor of TRIAS BDD. For processing context with 1,500 objects, TRIAS spent 271.00 minutes and TRIAS

BDD spent 281.84 minutes, which corresponds to a difference of 10.84 minutes less for TRIAS. For processing context with 3,000 objects, the original algorithm required 479.95 minutes and the proposed algorithm required 374.36 minutes. This difference corresponds to 105.59 minutes in favor of the proposed algorithm. For processing context with 5,000 objects, TRIAS spent 1,293.28 minutes and TRIAS BDD spent 903.89, which corresponds to a difference of 389.39 minutes in favor of TRIAS BDD. For processing context with 10,000 objects, TRIAS BDD spent 2,462.38 minutes. TRIAS did not generate results within 7 days. Thus, it was not possible to compare the algorithm's results.

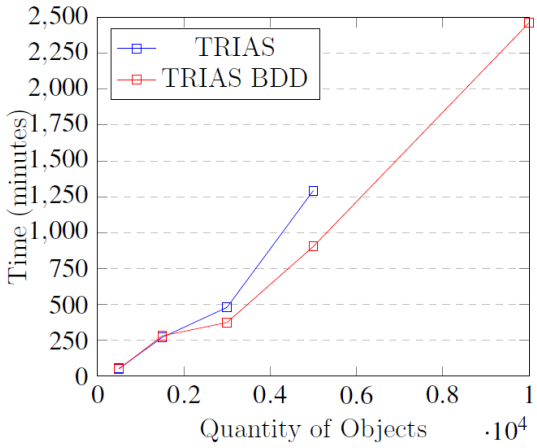

Figure 13: Contexts with 20 attributes, 5 conditions and 30% density

As can be seen in Figure 14, considering the context of 20 attributes, 5 conditions and 50% density, for processing the context with 500 objects TRIAS spent 70.93 minutes and TRIAS BDD spent 64.55 minutes, a difference of 6.38 minutes in favor of TRIAS BDD. For processing the context with 1,500 objects, the original algorithm spent 379.41 minutes and the proposed algorithm spent 318.70 minutes, which corresponds to a difference of 60.71 minutes less. For processing the context with 3,000 objects, TRIAS spent 767.93 minutes and TRIAS BDD spent 622.02 minutes. This difference corresponds to 145.91 minutes in favor of TRIAS BDD. For processing context with 5,000 objects, the original algorithm spent 2,885.89 minutes and the proposed algorithm spent 1,279.88, which corresponds to a difference of 1,606.01 minutes in favor of the proposed algorithm. For processing context with 10,000 objects, TRIAS BDD spent 5,259.46 minutes. TRIAS did not generate results within 7 days. Thus, it was not possible to compare the algorithm's results.

As presented in Figure 15, considering the context of 20 attributes, 5 conditions and 70% density, for processing the context with 500 objects TRIAS spent 223.56 minutes and TRIAS BDD spent 147.31 minutes, a difference of 76.25 minutes in favor of TRIAS BDD. For processing the context with 1,500 objects, the original algorithm spent 958.98 minutes and the proposed algorithm spent 614.45 minutes, which corresponds to a difference of 344.53 minutes less. For processing the contexts with 3,000, 5,000 and

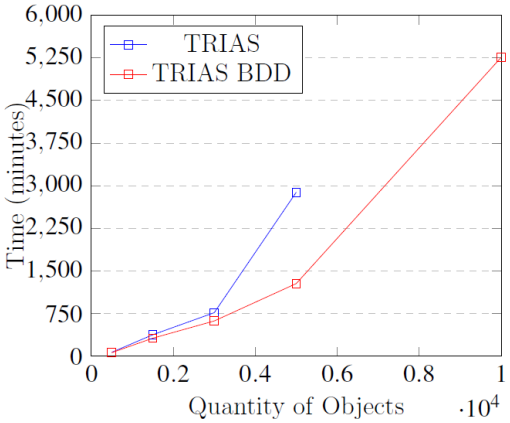

Figure 14: Contexts with 20 attributes, 5 conditions and 50% density

10,000 objects, TRIAS BDD spent 1,119.89, 2,317.66 and 7,578.21 minutes, respectively. TRIAS did not generate results within 7 days for 3,000, 5,000 and 10,000. Therefore, it was not possible to compare the algorithm's results.

In 2006, the ICFCA (International Conference on Formal Concept Analysis) at Desdren [Old and Priss, 2006] discussed the main challenges of formal analysis. Since that date, the requirements to deal with dense and high-dimensional formal contexts have been discussed, such as contexts with 120,000 objects and 70,000 attributes, which are considerably larger than the experiments that were performed here. In order to attend partially to the challenge, new synthetic triadic contexts were created using *SCGaz* (Table 13). Tests were performed and compared TRIAS and TRIAS BDD results for the contexts with 120,000 objects, 5, 10 and 15 attributes, 5 and 10 conditions and 30%, 50% and 70% density. First, the number of objects and density of each context were fixed at 120,000 and 30% while changing the number of attributes and conditions. Second, the density of each context was varied.

In Table 13, cells with the "-" symbol show that the algorithm failed to complete the concepts extraction processing within 14 days. It must be noted that in these scenarios the algorithms were dealing with contexts that have a high number of incidences, varying from 1,776,769 to 6,299,886.

As presented in Table 13, in none of the experiments was the TRIAS algorithm able to extract concepts within 14 days. The algorithm was not able to deal with those contexts called high-dimensional. However, TRIAS BDD retrieved concepts in view of the 14 days in all cases of contexts with 30% density. For instance, it completed the execution in 10.88 days for a context with a density of 30%, 120,000 objects, 10 attributes and 5 conditions. It was also able to extract concepts for contexts with a density of 50% considering the limit of 2,999,948 incidences. In these cases, with the use of BDDs optimizing the representation, it was confirmed to be an efficient solution. Thus, the use of BDD is an interesting approach in high-dimensional contexts.

Table 13 shows the results obtained for high-dimensional contexts. Considering 120,000 objects, for densities 30%, 50% and 70%, three different combinations between the number of attributes and conditions were tested, such as: 15 attributes and 5 conditions, 10 attributes and 5 conditions and 5 attributes and 10 conditions.

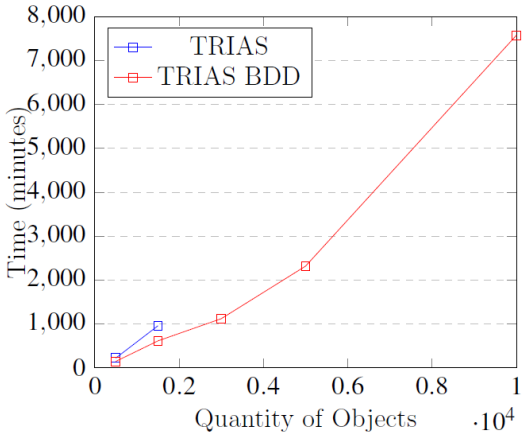

Figure 15: Contexts with 20 attributes, 5 conditions and 70% density

Context (GxMxB)	Density	Incidences	TRIAS (Days)	TRIAS BDD (Days)
120,000x15x5	30%	2,699,984	-	12.79
120,000x10x5	30%	1,776,769	-	10.88
120,000x5x10	30%	1,776,769	-	10.46
120,000x15x5	50%	4,498,986	-	-
120,000x10x5	50%	2,999,948	-	13.16
120,000x5x10	50%	2,999,948	-	13.54
120,000x15x5	70%	6,299,886	-	-
120,000x10x5	70%	4,199,988	-	-
120,000x5x10	70%	4,199,988	-	-

Table 13: Results for High-dimensional Contexts (TRIAS x TRIAS BDD)

According to Table 13, considering the density of 30%, it can be seen that TRIAS did not generate results for the three combinations of attributes and conditions. TRIAS BDD spent 12.79 days to process context with 120,000 objects, 15 attributes and 5 objects. The processing of the context with 120,000 objects, with 10 attributes and 5 conditions, TRIAS BDD spent 10.88 days. Also, for processing the context with 120,000 objects, 5 attributes and 10 conditions, TRIAS BDD spent 10.46 days. A variation of 3.86% is perceived between the time spent by both.

Moreover, it can be observed in Table 13, considering the density of 50%, TRIAS did not generate results for the three combinations of attributes and conditions. TRIAS BDD did not generate results for context with 120,000 objects, 15 attributes and 5 conditions. A variation of 2.88% is perceived between the time spent by TRIAS BDD for processing context with 120,000 objects, 10 attributes and 5 conditions (13.16 days) and the time spent by TRIAS BDD for processing context with 120,000 objects, 5 attributes and 10 conditions (13.54 days).

Also, according to Table 13, it is observed that, considering the density 70%, context with 120,000 objects and the three combinations of attributes and conditions (15 attributes x 5 conditions, 10 attributes x 5 conditions and 5 attributes x 10 conditions), neither

TRIAS nor TRIAS BDD were able to generate results.

However, it must be noted that even this new solution is not efficient when the number of incidences exceeds 3,000,000 - both algorithms were not able to complete the concept extractions for all contexts with 70% density within 14 days. But, for the contexts presented in Table 13, only the TRIAS BDD was able to extract the concepts.

5.2 Real datasets contexts results

One possible and interesting strategy to evaluate algorithms is using synthetic datasets. However, understanding the algorithm's behavior in real scenarios is very important to evaluate its real efficiency. Considering that, TRIAS and TRIAS BDD algorithms were applied to an extensive dataset that contains movie rates called *MovieLens*[1]. The database has more than 6,000 users and approximately 4,000 movies. Also, it has more than 1,000,000 records that consist of a classification rate from 1 to 5 by the user per movie. This dataset is considered sparse because it is not possible to confirm that users rated the exact same movies. For instance, it is not possible to confirm if user *A* has rated the same movies as user *B*. Additionally, in the movies range (attributes) that is required to be evaluated, it is not possible to confirm that the user rated (conditions) all movies. In other words, the user may have evaluated only a single movie within the possible movies available in the context.

The contexts created from the database have users as a set of objects, movies as attributes and rates received as conditions. Then, the context is defined as $\mathcal{K} = (K_1, K_2, K_3, Y)$ where K_1 is the set of users, K_2 is the set of movies, K_3 is the set of rates and Y is the relation among users x movies and its rates.

In Table 14, cells with the "-" symbol show that the algorithm failed to complete the concepts extraction processing within 7 days. It shows the results of the algorithms applied to the real dataset *MovieLens*. We fixed the number of objects (users) and conditions (rates), utilizing the maximum quantity of objects available. We also varied the number of attributes (movies).

Context (*G,M,B*) (G = Users × M = Movies × B = Rates)	Density	Incidences	TRIAS (Days)	TRIAS BDD (Days)
6,000x100x5	17.3%	105,986	-	5.88
6,000x150x5	16.4%	158,029	-	6.12
6,000x200x5	14.3%	203,898	-	6.85

Table 14: TRIAS x TRIAS BDD Algorithms results for MovieLens dataset

As presented in Table 14, the TRIAS and TRIAS BDD algorithms were applied to the *MovieLens* dataset. Three different scenarios were considered, such as: 1) 6,000 users (objects), 100 movies (attributes) and rates (conditions) equal to 5; 2) 6,000 users, 150 movies and 5 rates; 3) 6,000 users, 200 movies and 5 rates. It is noticeable that, the number of users and the rate were maintained in all analyzes. Only the quantity of movies was varied between 100, 150 and 200. For the context of 100 movies, a density of 17.3% was considered. For 150 movies, a density of 16.4% was considered and, finally, for 200 movies, a density of 14.3% was considered.

[1] https://grouplens.org

It must be noted that the TRIAS Algorithm did not complete the computation of all formal concepts, during a period of seven days. TRIAS BDD obtained results for 100 and 150 movies, spending 5.88 days and 6.12 days, respectively. TRIAS BDD was not able to generate results for 600 users, 200 films and 5 rates, in seven days.

6 Conclusion and future work

The approach that used BDD has shown better results for almost all triadic contexts. The main reason is that BDD allows the algorithm to execute contexts that could be difficult to analyze. As presented, the algorithm proposed had a better speedup in almost all the contexts used. In some cases, it was up to 56% faster than TRIAS - see Table 10.

Also, for the synthetic contexts with 120,000 objects, the TRIAS BDD algorithm finalized the concepts extraction before 14 days - see Table 13. Nevertheless, TRIAS did not retrieve any results in 14 days.

In the real dataset, presented in Table 14, the TRIAS, within 7 days, was not able to extract concepts in the contexts that contain more than 100,000 incidences. However, using TRIAS BDD the algorithm retrieved triadic concepts before 7 days in two scenarios. In other words, it was able to find concepts that the original algorithm was not.

As planned future work, it is intended to reduce the processing time by distributing the workload, paralleling the generation of the BDD and/or the concurrent extraction of the concepts. Furthermore, we intend to parallelize the algorithms through threads and/or GPU (Graphics Processing Unit). The main idea is to distribute the algorithm execution on different computers.

Acknowledgements

The authors acknowledge the financial support of FAPEMIG, CNPq, and CAPES. Also, the authors thank PUC-MG students and professors for raising relevant suggestions that have improved the work considerably.

References

[Akers, 1978] Akers, S. “Binary decision diagrams”; IEEE Transactions on Computers, C-27(6):509–516 (1978).

[Bryant, 1986] Bryant, R. E. “Graph-based algorithms for boolean function manipulation”; IEEE Transactions on vol. 100, no. 8, pp. 677–691 (1986).

[Carpineto and Romano, 2005] Carpineto, C. and Romano, G. “Concept data analysis: Theory and applications”; John Wiley and Sons. England (2005).

[Davey and Priestley, 2001] Davey, B. A. and Priestley, H. A.; “Introduction to lattices and order”; Cambridge Mathematical Textbooks (2001).

[de Moraes et al., 2016] de Moraes, N. R., Dias, S. M., Freitas, H. C., and Zarate, L. E. “Parallelization of the next closure algorithm for generating the minimum set of implication rules”; Artificial Intelligence Research, 5(2):40 (2016).

[Dias et al., 2020] Dias, S., Zarate, L., Song, M., Vieira, N., and Cherukuri, A. K. “Extraction of qualitative behavior rules for industrial processes from reduced concept lattice”; Intelligent Data Analysis, pages 643–663 (2020).

[Ganter, 2010] Ganter, B. "Two basic algorithms in concept analysis", Springer (2010).

[Ganter et al., 2005] Ganter, B., Stumme, G., and Wille, R. "Formal concept analysis: foundations and applications"; Springer Science Business Media, vol. 3626 (2005b).

[Ignatov et al., 2015] Ignatov, D., Gnatyshak, D., Kuznetsov, S., and Mikin, B. "Triadic formal concept analysis and triclustering: searching for optimal patterns"; Machine Learning - Springer, pages 271–302 (2015).

[Ignatov et al., 2011] Ignatov, D., SKuznetsov, Magizov, R., and Zhukov, L. "From triconcepts to triclusters"; RSFDGrC: International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, pages 257–264 (2011).

[Jaschke et al., 2006] Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., and Stumme, G. "TRIAS–an algorithm for mining iceberg tri-lattices"; In Data Mining, 2006. ICDM'06. Sixth International Conference on, pages 907–911. IEEE (2006).

[Javabdd, 2019] "Javabdd". Javabdd algorithm (2019).

[Lehmann and Wille, 1995] Lehmann, F. and Wille, R. "A triadic approach to formal concept analysis"; Springer, Conceptual structures: applications, implementation and theory, pages 32–43 (1995a).

[Old and Priss, 2006] Old, J. and Priss, U. "Some open problems in formal concept analysis"; Problems presented at international conference on formal concept analysis (ICFCA) (2006).

[Pei et al., 2000] Pei, J., Han, J., Mao, R., et al. "Closet: An efficient algorithm for mining frequent closed item sets"; In ACM SIGMOD workshop on research issues in data mining and knowledge discovery, volume 4, pages 21–30 (2000).

[Rimsa et al., 2013] Rimsa, A., Song, M. A., and Zárate, L. E. "SCGaz-a synthetic formal context generator with density control for test and evaluation of FCA algorithms"; In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, pages 3464–3470. IEEE (2013).

[Rimsa et al., 2009] Rimsa, A., Zárate, L. E., and Song, M. A. "Evaluation of different BDD libraries to extract concepts in FCA–perspectives and limitations"; Computational Science–ICCS 2009, pp. 367–376 (2009).

[Salleb et al., 2002] Salleb, A., Maazouzi, Z., and Vrain, C. "Mining maximal frequent itemsets by a boolean based approach"; In European Conf. on Artificial Intelligence, Lyon France (July 2002), pages 285–289 (2002).

[Santos et al., 2018] Santos, P., Neves, J., Silva, P., Dias, S. M., Zárate, L., and Song, M. "An approach to extract proper implications set from high-dimension formal contexts using binary decision diagram"; Proceedings of the 20th International Conference on Enterprise Information Systems, 1:50–57 (2018).

[Tang et al., 2015] Tang, P., Huia, S., and Fong, C. "A lattice-based approach for chemical structural retrieval". Engineering Applications of Artificial Intelligence, pages 215–222 (2015).

[Tang et al., 2016] Tang, Y., Fan, M., and Li, J. "An information fusion technology for triadic decision contexts"; International Journal of Machine Learning and Cybernet, page 13–24 (2016).

[Trabelsi et al., 2012] Trabelsi, C., Jelassi, N., and Yahia, S. B. "Scalable mining of frequent tri-concepts from folksonomies"; In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 231–242. Springer (2012b).

[Wille, 1982] Wille, R. "Restructuring lattice theory: An approach based on hierarchies of concepts". pages 445–470 (1982).

[Wille, 1995] Wille, R. "The basic theorem of triadic concept analysis"; Springer, Order, 12(2):149–158 (1995).

[Wille, 1996] Wille, R. "Restructuring mathematical logic: an approach based on peirce's pragmatism"; Lecture notes in pure and applied mathematics, pages 267–282 (1996).

[Yevtushenko, 2002] Yevtushenko, S. "BDD-based algorithms for the construction of the set of all concepts. foundations and applications of conceptual structures"; In Foundations and Applications of Conceptual Structures. Contributions to ICCS, volume 2002, page 61–73 (2002).

[Zhang et al., 2013] Zhang, L., Zhang, H., Shen, X., and Yin, L. "A bottom-up algorithm of vertical assembling concept lattices"; International Journal of Data Mining and Bioinformatics, pages 229–244 (2013).