PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Pós-Graduação em Informática

Sebastião Mendes Neto

FORMAL CONCEPT ANALYSIS APPLIED IN SOCIAL NETWORKS WITH USE OF BDD FOR HIGH DIMENSIONAL FORMAL CONTEXTS

Belo Horizonte

2016

Sebastião Mendes Neto

FORMAL CONCEPT ANALYSIS APPLIED IN SOCIAL NETWORKS WITH USE OF BDD FOR HIGH DIMENSIONAL FORMAL CONTEXTS

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.

Supervisor: Mark Alan Junho Song
Co-orientador: Luís Enrique Zárate Gálvez

Belo Horizonte
2016

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

Mendes Neto, Sebastião
M538f Formal concept analysis applied in social networks with use of BDD for high dimensional formal contexts/ Sebastião Mendes Neto. Belo Horizonte, 2016.
106 f. : il.

Orientador: Mark Alan Junho Song
Coorientador: Luís Enrique Zárate Gálvez
Dissertação (Mestrado) - Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática.

1. Banco de dados. 2. Redes sociais. 3. Análise de redes (Planejamento). 4. Algoritmos de computador. I. Song, Mark Alan Junho. II. Gálvez, Luís Enrique Zárate. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática. IV. Título.

SIB PUC MINAS

CDU: 681.3.01

Sebastião Mendes Neto

FORMAL CONCEPT ANALYSIS APPLIED IN SOCIAL NETWORKS WITH USE OF BDD FOR HIGH DIMENSIONAL FORMAL CONTEXTS

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.

Mark Alan Junho Song (PUC Minas)
Orientador

Luís Enrique Zárate Gálvez (PUC Minas)
Membro interno

Alneu de Andrade Lopes (USP)
Membro externo

Adriano César Machado Pereira (UFMG)
Membro externo

Belo Horizonte, 20 de Abril de 2016

To my parents, for their endless love, support and encouragement

ACKNOWLEDGEMENTS

I would like first to thank my parents Sebastião and Tércia for the example and foundations that are essential to a happy life. I would also like to thank my sisters Nadejda and Cybelle who I admire so much. I am also grateful to my father in law Gilberto and mother in law Norma who adopted me and always supported me. I would like to thank my wife Yáskara for her understanding and love.

I also thank the support of my mentors, teachers, colleagues and staff in the IT department of PUC - Minas.

I acknowledge the financial support received from the Foundation for Research Support of Minas Gerais state, FAPEMIG; the National Council for Scientific and Technological Development, CNPQ; Coordination for the Improvement of Higher Education Personnel, CAPES.

"The joy is in the fight, trying, in suffering and not involved in the actual victory."
(Mahatma Gandhi)

RESUMO

Nos últimos anos, a crescente complexidade de um mundo hiper-conectado direcionou as atenções de muitas pesquisas, gerando a necessidade de novas técnicas multidisciplinares para análise de redes complexas. Nosso objetivo é encontrar e analisar padrões de acesso de usuários usando um tipo de regra de associação denominada implicações e viabilizar a manipulação de grandes bases de dados adaptando algoritmos da Análise Formal de Conceito (AFC). Os acessos a vinte e cinco sites mais visitados foram analisados e onze contextos formais foram criados representando uma variedade de dias e horários. Os dados foram convertidos para contextos formais clarificados e três algoritmos capazes de encontrar implicações foram utilizados. Interseções entre alguns conjuntos de implicações foram feitas à procura de padrões recorrentes. Uma cobertura canônica de implicações também foi extraída permitindo representação do conhecimento com redução do tamanho de até 99,90%. Além disso, extraímos conjuntos de implicações com padrões de fácil entendimento que revelam comportamento aditivo. Todas as implicações foram usadas para explorar e analisar subestruturas existentes em redes complexas do tipo *two-mode networks*. Como resultado, encontramos padrões de acesso que garantem que sempre que os sites da premissa são acessados, os sites da conclusão também o são. Tais resultados podem ajudar a criar políticas de segurança e configurações de rede para melhor prever e controlar acessos futuros. Porém, com a crescente quantidade de informação provenientes de redes complexas, tais como as redes sociais, recursos computacionais cada vez mais poderosos são necessários. Propomos a utilização do diagrama binário de decisão (BDD) para lidar com estas situações. Adaptamos o BDD aos *operadores de derivação* da FCA permitindo manipular uma quantidade de informação que sem esta técnica não seria viável. Para avaliar os benefícios do BDD implementamos os dois tipos de algoritmos existentes, incremental e lote, ambos mostraram ganhos de desempenho em contexto com alta dimensionalidade e densidade.

Palavras-chave: Análise Formal de Conceito, Redes Complexas, Redes Sociais, Regras de Associação, Implicações, Conceito Formal, Diagrama Binário de Decisão.

ABSTRACT

In recent years, the increasing complexity of a hyper-connected world has directed the attentions of many types of research and because of that, we need new multidisciplinary complex network analysis. Our goal is to find and analyze users access patterns using a type of association rules named implications and enable large database handling through adaptation of Formal Concept Analysis (FCA) algorithms. We analyzed users accesses to twenty-five most visited websites, and eleven formal contexts were created representing a variety of days and times. We converted the data to a clarified formal context, and we used three algorithms to extract implications sets. We crosschecked some implications sets in the search for recurring patterns. We also extracted a canonical coverage of implications able to represent the network knowledge with a reduction in the storage space of up to 99.90%. Besides, we obtained sets of implications with easy to understand patterns and representing additive behavior. The implications were used to explore and analyze existing substructures of two-mode networks. As a result, we found access patterns ensuring that whenever users access implications premise websites, they also access the websites in implications conclusion. Our results may help create security policies and network settings to better predict and control future users access. However, with the increasing amount of information from complex networks such as social networks, powerful computing resources are needed. We propose the use of binary decision diagram (BDD) to handle this situation. In this work, the BDD is adapted to FCA *derivation operators* allowing the handling of a quantity of information that without this technique would not be feasible. To evaluate BDD benefits, we used two types of algorithms, an incremental and a batch, both showed performance gains in formal context with high dimensionality and density.

Key-words: Formal Concept Analysis, Complex Networks, Social Networks, Association Rules, Implications, Formal Concept, Binary Decision Diagram.

LIST OF FIGURES

1 The Dissertation Work 22
2 User access network 30
3 Formal context example 31
4 The formula $(a \wedge b) \vee (c \wedge d)$ represented using decision tree and BDD 40
5 A two bit comparator BDD 41
6 Formal context of Table 1 as BDD 42
7 Approximate depiction of the network formed by implications 48
8 Network form by the implications extracted by Impec algorithm 53
9 Network formed by the implications extracted by Impec algorithm (only premises connected to one conclusion) 54
10 Network formed by implications 56
11 Formal context of Solar System planets and some of their characteristics 71
12 BDD representation of the Solar System formal context 72
13 Attribute of the context represented with a BDD. 72
14 Result of a conjunction between a context BDD (Figure 5) and an attribute BDD (Figure 6). 73
15 BDD of attributes shared by objects in Figure 14 BDD. 73
16 Time for extracting concepts of contexts with 25 attributes and objects between 1,000 and 1,000,000 79

LIST OF TABLES

1 Formal Context Example 42
2 Contexts and their characteristics 46
3 Some recurrent access patterns from intersection 49
4 Implications and intersections 50
5 Accesses according to implications - Recurring users 50
6 Some Impec Implications 52
7 Some implications extracted. 55
8 Influence of the DoublePrime function in the NextClosure algorithm 67
9 Minimum and Maximum densities of processed contexts 74
10 Comparisons Original NextClosure X BDD implementation 77
11 Comparisons of Original InClose2 X BDD implementation 80

LIST OF ALGORITHMS

1	Next Closure(A, M)	34
2	In-Close2(r, y)	36
3	Building a formal context using BDD	66
4	Extracting an intent from an assembled BDD	66
5	Extraction of formal concepts	68
6	DoublePrime Algorithm - Application of the $(.)''$ operator	69
7	Returns the set of objects containing the attributes in X	69
8	Returns a set of attributes shared by all objects of object set X'	70
9	Extent - Returns the set of objects containing the attributes in X	70
10	Returns a set of attributes shared by all objects in input parameter X'	74

SUMÁRIO

1 INTRODUCTION 21
1.1 Justification 22
1.2 Objectives 24
1.2.1 New Models to Analyze Two-Mode Networks 24
1.2.2 Handling High Dimensional Formal Contexts 26
1.3 Contributions 26
1.4 Outline 27

2 THEORETICAL FOUNDATIONS 29
2.1 Complex Networks 29
2.2 A Review about Formal Concept Analysis - FCA 31
2.2.1 Formal Context 31
2.2.2 Formal Concepts 33
2.2.3 Concept Lattice 33
2.2.4 FCA Algorithms to Extract Formal Concepts 33
2.2.4.1 NextClosure Algorithm 34
2.2.4.2 Algorithm In-Close2 35
2.2.5 Implications 35
2.2.6 FCA Algorithms to Extract Implications 37
2.2.6.1 FindImplications Algorithm 37
2.2.6.2 Impec Algorithm 38
2.2.6.3 NextClosure Algorithm 39
2.3 Binary Decision Diagram 40
2.3.1 Formal Context represented by a boolean formula 41

3 IMPLICATIONS FOR ANALYZING COMPLEX NETWORKS USING FCA 43
3.1 Related Works 43
3.2 Methodology : Materials and methods 45
3.2.1 Considered database 45
3.2.2 Social Networks Visualization Based on Implications 47
3.3 Experiments, Analysis and Results 49
3.3.1 The Identification of Conservative Behavior in Two-mode Networks Using All Implications 49
3.3.1.1 Visualization of the Network with Conservative Behavior 51

3.3.2 *Handling Additive Set of Behavior in Two-mode Networks with One-Item Conclusion Implications* 51
3.3.2.1 Visualization of the Network of Implications with Additive Set of Behavior 52
3.3.3 *Canonical Behavior Representation in Two-mode Networks Using a Canonical Set of Implications* 55
3.3.3.1 Visualization of the Network of Canonical Implications 56
3.4 Discussions 57
3.5 Conclusions 58

4 USING BINARY DECISION DIAGRAMS TO EXTRACT FORMAL CONCEPTS VIA FORMAL CONCEPT ANALYSIS FOR HIGH DIMENSIONAL FORMAL CONTEXTS 61
4.1 Related Works 61
4.2 Methodology: Materials and Methods 63
4.2.1 *Chosen BDD Library* 64
4.2.2 *Representing a formal context with BDD* 65
4.2.3 *Where to use BDD* 67
4.2.4 *NextClosure Algorithm Implementation* 68
4.2.5 *Using BDD* 70
4.3 Experiment, Analysis and Results 74
4.3.1 *Nexclosure-BDD* 75
4.3.2 *InClose2-BDD* 76
4.4 Conclusions 79

5 GENERAL CONCLUSIONS AND FUTURE WORKS 83

References 85

1 INTRODUCTION

In recent years, the increasing complexity of a hyper-connected world has attracted many researchers, as noted by Easley and Kleinberg (EASLEY; KLEINBERG, 2010). A variety of factors powered this connectivity, such as the Internet itself, telephone networks and the speed that information travels around the world. Motivated by the worldwide high-connectivity, research has emerged, and different research fields have been connected to contribute to new techniques and perspectives for the analysis of networks.

Currently, the discovery of relationship patterns which unite social actors is the focus of social network analysis (SNA). Such relations may occur between individuals, events or individuals and events, pinpoint properties such as importance, ranking and category. According to Getoor and Diehi (GETOOR; DIEHL, 2005), in some cases, some relationships are not observed. Then, it is relevant to discover the existence of potential relationships among actors, hidden substructures, and potential communities.

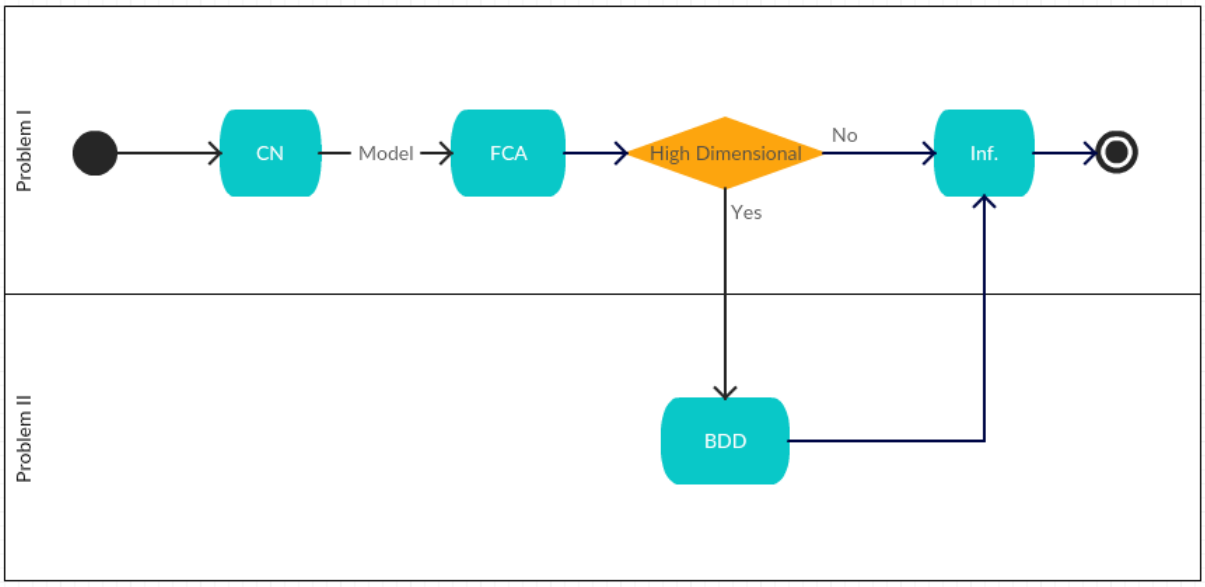
However, more often than not we need to deal with enormous amount of information coming from social and complex networks, which demand powerful computational resources.

Formal concept analysis (FCA) is considered an important mechanism for formalizing knowledge representation, enabling its analysis (WILLE, 2009).

Our work uses Binary Decision Diagrams (BDD), first introduced by Akers (AKERS, 1978) and further developed by Bryant (BRYANT, 1986), to represent information in a canonical and simplified way.

Figure 1 is a graph representation of our work process. We propose the discovery of new information such as relationship patterns in complex networks (CN) through new models using formal concept analysis (FCA). This process is represented in swimlane ”Problem I” in Figure 1 and we deal with it in Chapter 3. Constantly the network size can make inviable its handling and to help to address this issue we propose the use of BDD as shown in the swimlane ”Problem II” allowing us to handle high dimension information. We approach this second problem in Chapter 4.

Figure 1 – The Dissertation Work



1.1 Justification

The discovery of substructures is a social data mining field to which relationship mining in social networks is related. Therefore, the field seeks to find common substructures, which can be useful for example in the classification of networks and their structure. However, the identification of substructures requires new and improved methods to visualize efficiently and extract valuable information from such representations (FREEMAN, 2000; FREEMAN, 2005). Nowadays, social networks are undergoing a dramatic growth that makes the identification of patterns in relationships increasingly difficult. Moreover, it is important to investigate new computational methods that allow the representation, characterization, and analysis of social networks. Hence, rule-based computational models are very useful if the generated rule set contains relevant and unexpected knowledge.

A social network consists of actors (e.g., users, companies, etc.) and relationships among them. A graph is used to model such structure whose nodes are the actors, and the edges are links that express relationships between actors. When there is one type of nodes (e.g., web users), the structure is called a *one-mode data network*, *one-mode network* or interaction network (TAMASSIA, 2013). When the nodes are of two types, such as users and groups(BREIGER, 1997), we have a *two-mode data network*, *two-mode network* or an affiliation network expressed through a bipartite graph. Freeman and White presented a detailed study of the features, strengths, and limitations of techniques for two-mode data network analysis (FREEMAN, 2003).

Two-mode networks cannot be fully analyzed using only complex network techniques,

as mentioned by Freeman and White (FREEMAN; WHITE, 1993). Indeed, their work showed how difficult it is to find associations between nodes in two-mode networks and suggested the application of formal concept analysis (FCA) to discover central nodes and identify groups of actors and their properties. Freeman (FREEMAN, 2003)also provided a detailed survey (meta-analysis) of techniques used to analyze the Davis data about Southern Women and the attended events (DAVIS; GARDNER; GARDNER, 1941). In our work, we treated the problem of analyzing event type node relationships transforming network data into a formal context, extracting implications and finally processing premises and conclusions as nodes connected by an edge.

Moreover, FCA uses *concept lattice theory* to hierarchically organize *concepts* from a *formal context* consisting of objects, attributes and their incidences (or relationships). However, formal context with high dimensionality demands powerful computational resources. In (PRISS, 2006), some of FCA challenges were laid out, among them is the manipulation of large formal contexts, such as context containing 120,000 objects and 70,000 attributes to extract formal concept and consequently implications.

Several algorithms have been proposed to extract *formal concepts* (CHEIN, 1969), (NORRIS, 1978), (BORDAT, 1986),(GODIN; MISSAOUI; ALAOUI, 1991), (DOWLING, 1993), (KUZNETSOV, 1993), (GODIN; MISSAOUI; ALAOUI, 1995), (NGUIFO; NJIWOUA, 1998), (LINDIG, 1999), (NOURINE; RAYNAUD, 1999), (STUMME et al., 2000), (OBIEDKOV, 2003), (QIAO et al., 2003), (YEVTUSHENKO, 2004), (GANTER; STUMME; WILLE, 2005), (BERTET; GUILLAS; OGIER, 2007), (GANTER, 2010). These algorithms have an exponential complexity in the worst case, as previously explained in (KUZNETSOV, 2001). However, this behavior is not very frequent according to (GODIN; SAUNDERS; GECSEI, 1986). Even then, for high dimensionality contexts the computational cost become prohibitive. This problem motivated several studies seeking alternatives to extract formal contexts. Some of these studies found more efficient uses for algorithms, such as (GUÉNOCHE, 1990), (GODIN; MISSAOUI; ALAOUI, 1995), (KUZNETSOV; OBIEDKOV, 2002), (FU; NGUIFO, 2004). Among the aforementioned studies, Kuznetsov and Obiedkov's (KUZNETSOV; OBIEDKOV, 2002) sets itself apart by pointing out the lack of available information regarding testing of the algorithms and non-disclosure of source code used to assemble and manipulate data structures. To avoid such inconvenience, we sought to explain the tests we conducted as well as the necessary code to manipulate the structures used in the algorithms.

1.2 Objectives

FCA is a mathematical research field introduced in the beginning of the 80s by Rudolf Wille and has been used in different fields of research (WILLE, 1982; GANTER; WILLE, 1999). Due to its potential in knowledge representation, FCA can also be useful for complex network analysis, as discussed by Freeman (FREEMAN, 1996), Snášel et al. (SNÁŠEL; HORÁK; ABRAHAM, 2008) (SNASEL et al., 2009), Poelmans et al. (POELMANS et al., 2011), Freeman and White (FREEMAN; WHITE, 1993), Rome and Haralick (ROME; HARALICK, 2005) and Jaschke et al. (JÁSCHKE et al., 2006).

This work presents two primary objectives. The first is to use the FCA knowledge representation capability to create new models to help in two-mode networks analysis. The second objective is to enable high dimensional formal contexts analysis using BDD. The following sections will explain both objectives in detail.

1.2.1 New Models to Analyze Two-Mode Networks

In this FCA-based work, we show the potential of building computational models based on implications to represent and analyze two-mode networks. This new representation allows us to address issues such as:

a. Identification of conservative behavior in two-mode networks.

   The aim is to find user access patterns as well as recurring patterns from implications. To accomplish the task of looking for implication sets, similar contexts with different incidence relationships are compared. To extract all implications the *FindImplication* algorithm (CARPINETO; ROMANO, 2005) is used. After implication set extraction for each considered context, comparisons are done to look for implications that occur in all contexts.

b. Handling additive set of behavior in two-mode networks.

   Sometimes a behavior, represented by a premise, can lead to more than one action, denoted by a conclusion. If we consider two implications, both with the same premise and different conclusions such as $A \rightarrow B$ and $A \rightarrow C$ and applying the property of addition, we can

state that whenever A happens both B and C also occur. If we are looking for a premise that will be used to predict future conclusions, such implications may lead to erroneous behavior prediction in social network analysis.

Consequently, our goal is also to handle this kind of additive set of behavior, allowing its identification for future control. We then used the *Impec* algorithm (TAOUIL; BASTIDE, 2001) with its implications of one-item conclusion for that purpose.

c. Canonical behavior representation in two-mode networks.

The purpose is to increase our knowledge of finding a minimal implication set of user access patterns, a canonical base, to discover hidden substructures (relationships between different websites). The importance of a canonical base lies in its ability to represent the complete set of data with a minimum and irreducible set of implications and in the capacity to form the complete set of implications when expanded if necessary.

Furthermore, the amount of information is steadily increasing, and new strategies for storage must be considered. To solve this problem, we propose the use of the canonical set as mentioned earlier to store the knowledge derived from this information in a more optimized way. Hence, we choose the *NextClosure* algorithm (GANTER, 2010), based on closed sets and pseudo-intents, to extract a complete and non-redundant set of implications.

In this work, we consider *access pattern*, an expression used in items a and c, as the set of web sites accessed by a user in a given period. The implication $P \rightarrow Q$ between the patterns P and Q means that whenever users access the set P of websites they necessarily access the set Q of websites.

However, handling a large number of implications can be difficult. Thus, we propose in the three considered cases to transform the elements (premises and conclusions) of each generated implication on nodes of a graph and apply techniques of complex networks to find clusters to analyze further the relationships between premises and conclusions.

In our work, we treated the problem of analyzing event type node relationships transforming network data into a formal context, extracting implications and finally processing premises and conclusions as nodes connected by an edge. In this manner, our work differs from other studies because, in the search for substructures, we use association rules such as implications to extract more knowledge about the existing relationships between the event type nodes in two-mode networks. Also, after extracting the implications, we use techniques of complex networks to understand better the relationship between the implications.

1.2.2 Handling High Dimensional Formal Contexts

Unlike previous works in section 1.1 that propose novel algorithms to extract formal concepts, we aimed to improve the performance of formal concept analysis algorithms by using an alternate structure, *Binary Decision Diagrams* (BDD), to represent formal contexts. BDDs are structures used to represent canonically Boolean formulae (BRYANT, 1986). They can be used to describe formal contexts using smaller memory footprints (RIMSA; ZÁRATE; SONG, 2009). Since they are canonical structures, we also hope that BDDs can reduce the amount of resources used to manipulate data, as it has reduced in formal model analysis (CLARKE JR.; GRUMBERG; PELED, 1999). As a consequence of these expected performance gains, we hope to make the manipulation of concepts of higher dimensions viable.

Additionally, we also propose applying BDDs to FCA operators, called *derivation operators*. We first used the adapted operator to the NextClosure algorithm to extract formal concepts, and then we did the same to a version of In-Close2 (ANDREWS, 2011). The In-Close2 is known for its excellent performance in sparse contexts. Another goal was to assess the benefits brought by using BDD in situations for which the latter algorithm has poor performance.

1.3 Contributions

The two FCA techniques and complex networks were combined in other works, as it will be present, but the most commonly used elements are not implications. This work also presents innovates ways and possibilities of using association rules such as implications for complex networks analysis of two-mode networks.

With bigger and bigger databases, the use of BDD in FCA algorithms can allow researchers to manipulate formal contexts of previously unfeasible dimensions and densities.

As we propose to apply BDDs to FCA derivation operators, this can benefit almost all known formal concept and implication extraction algorithms used in several fields (POELMANS et al., 2013b).

1.4 Outline

This work presents an unusual structure because we address two problems closely related. First we propose using implication rules for knowledge extraction searching for hidden substructures in complex networks such as two-mode networks. To resolve this issue we dedicate Chapter 3.

When handling information related to these networks, we are faced with databases so big that prevents its use by current FCA's algorithms. Chapter 4 presents how we approach and solve this problem.

The theoretical foundations for solving the two problems have much in common and are addressed in Chapter 2.

We present conclusion in Chapter 3 and Chapter 4 with our proposals offer its conclusion, as well as a overall conclusion in Chapter 5 that includes proposals for future work.

2 THEORETICAL FOUNDATIONS

2.1 Complex Networks

Formally, $G = (N, E)$ represents a directed or undirected graph of nodes in $N \neq \emptyset$ and edges in E. The nodes of graph G are the elements in $N = \{n_1, n_2, \ldots, n_x\}$. The edges are the elements in $E = \{e_1, e_2, \ldots, e_y\}$.

As indicated before, one-mode data networks are those that contain nodes of a single type, such as nodes that represent users and their relationships while two-mode data networks contain nodes of two types, such as nodes that represent users and the websites that they visit. (WASSERMAN, 1994).

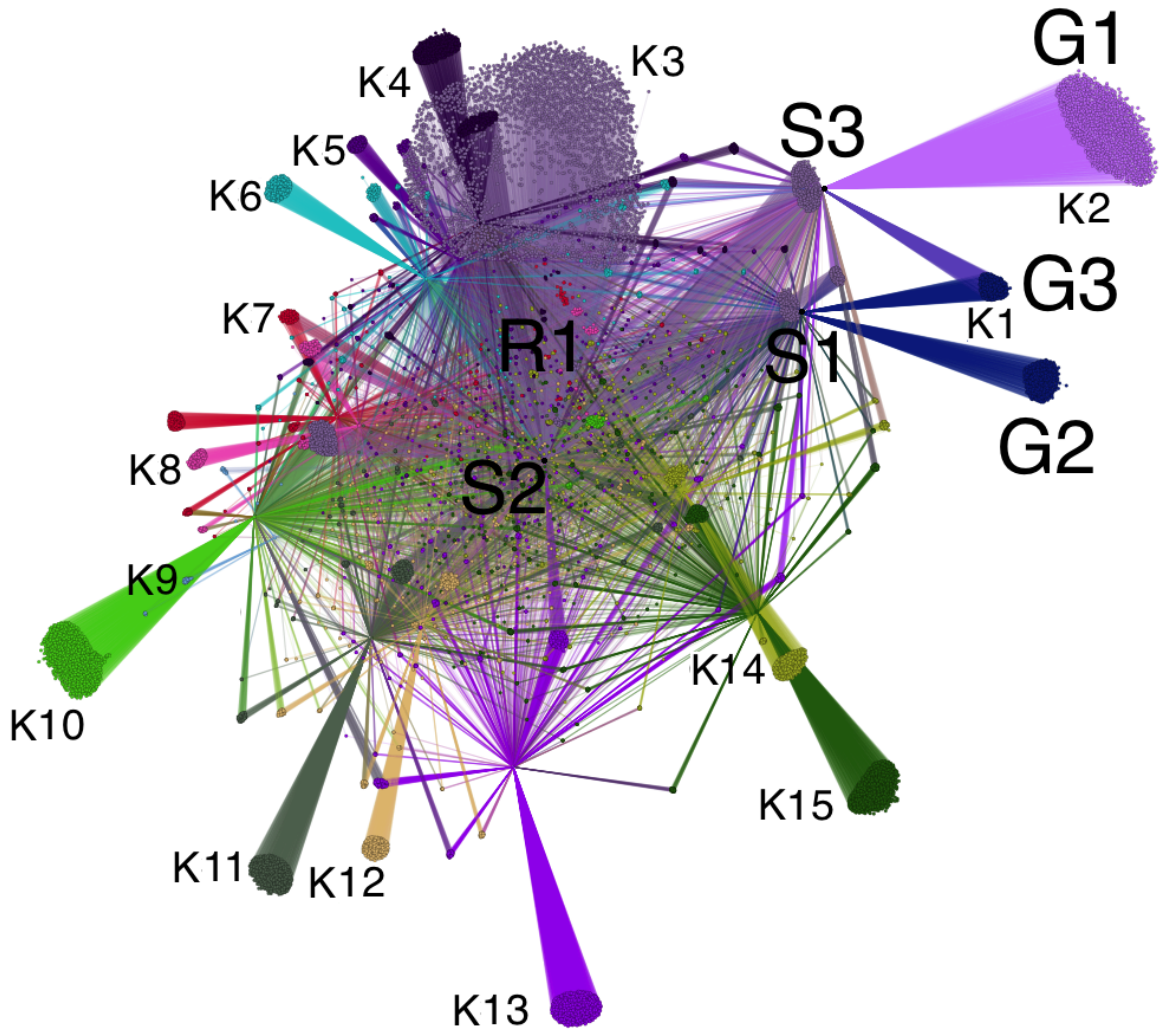
Figure 2 is our illustrative example of a network that represents connections of users to fifteen Internet websites, and the resulting graph is a bipartite directional graph whose readability and visualization was improved using the graph structure formatting algorithm *Force Atlas 3D* (JACOMY et al., 2014).

The algorithm for cluster generation called *Clustering Chinese Whispers* (BIEMANN, 2006) was applied to find fifteen clusters (K1-K15) of users with similar access patterns. The users inside a cluster have a greater probability of communicating and establishing a social network between them than with users outside the cluster (BREIGER, 1997) (FELD, 1981).

G1 and G2 in Figure 2 are two examples of groups. In such groups, the users access just one website. Thus, the attraction force of a single website makes the users stay very close to each other. Website S1 is responsible for attracting users from group G2. Users who access the two websites S1 and S3 create group G3. Located in the center of the graph is website S2 (Google). It is highlighted for its high degree centrality since it has numerous connections with users that simultaneously explore other websites.

Due to the several website attraction, region R1 forms a sparse set but not a cluster of users who access many websites. Note that one cannot easily observe existing relationships between websites and this raises the following question: How to determine which access to a set of websites implies access to another set of websites? Even with a sophisticated visualization tool, the answer may not be possible for large networks.

Figure 2 – Network generated from the accesses of 20.115 users in a day to 15 preselected websites.



The principle of *strong ties* and *weak ties* (GRANOVETTER, 1983) that are used to analyze the proximity and social contact can supplement the analysis of clusters based just on Figure 2. It states that if a node a is connected strongly to node b and node c, there must be a strong or weak connection between nodes b and c. Besides, clusters of users, such as K2 in Figure 2, are usually connected by *strong ties* while the connection between clusters K1 and K2, such as G3, normally called bridges, are formed by *weak ties* (GRANOVETTER, 1983).

In Figure 2, the relationships between users who access the same group of websites are instantly observable because they form clusters. User-website relationships are also recognizable because graph edges represent them. Nevertheless, merely observing user-website networks cannot help infer associations between access patterns. This limitation in the representation of

relevant patterns in the form of associations and clusters in two-mode networks was already pointed out by Freeman and White (FREEMAN; WHITE, 1993) who show that FCA can be helpful in analyzing such networks.

In our work, we show that implications from FCA along with complex network techniques can also help understand this kind of networks.

2.2 A Review about Formal Concept Analysis - FCA

FCA provides a formalization of a formal concept as a pair of extent and intent. The extent represents all objects that belong to the concept while all attributes shared by the objects as mentioned earlier are elements of the intent. Concepts in FCA (GANTER; WILLE, 1999) are groups of objects with a particular description/meaning organized in a lattice structure. FCA involves four fundamental elements: *formal contexts, formal concepts, concept lattices* and *implications*. (GANTER; STUMME; WILLE, 2005)

2.2.1 Formal Context

A table is usually used to represent a formal context in which rows denote objects and columns indicate attributes. When an object has an attribute, an incidence is identified and denoted by an "X" (GANTER; WILLE, 1999; GANTER; STUMME; WILLE, 2005). In Figure 3, a formal context is shown, where rows are objects and columns are websites accessed during a session.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	globo	orkut	youtube	uai	gstatic	oss-...	google	terra	rad...	ad...	vie...	uol	ad...	g...	yahoo
usu_1													X		X
usu_2							X								
usu_3		X					X		X			X			
usu_4		X					X								
usu_5															X
usu_6												X			
usu_7							X								
usu_8				X			X					X			

Figure 3 – Formal context example

A formal context has the notation (G, M, I), where G corresponds to a set of objects (rows), M is a set of attributes (columns) and I is a binary relation defined as $I \subseteq G \times M$. If an object $g \in G$ and an attribute $m \in M$ are in relationship I, their representation is defined as $(g, m) \in I$ or gIm, which reads as "object g has attribute m".

From a formal context (G, M, I) and for a subset of objects $A \subseteq G$, there is an attribute subset of M, common to every object of A. Similarly, for a set $B \subseteq M$, there is an object subset whose elements have all the attributes of B. The following derivation operations define these relationships (GANTER; WILLE, 1999):

$$A' := \{m \in M | gIm \forall g \in A\} \tag{2.1}$$

$$B' := \{g \in G | gIm \forall m \in B\} \tag{2.2}$$

Through clarification process, high dimensional contexts can be simplified and reduced (GANTER; WILLE, 1999). Clarification is performed by keeping only one instance of two identical objects or two corresponding attributes. Thus, the number of objects and attributes can be reduced while maintaining the shape of the lattice and consequently the implications set (GANTER; STUMME; WILLE, 2005).

The replacement of a set of objects with precisely the same attributes by a single object or the replacement of a set of attributes that occur in the same objects by a single attribute is a first possible simplification. More formally, if $g, h \in G$ and $g' = h'$ then g and h can be substituted by a single representative object. Dually, if $m, n \in M$ and $m' = n'$, then m and n can be substituted by a single representative attribute. In this work, we chose only the object (user) dimension for the clarification process. As there are no equivalent attributes (i.e., groups of websites visited by the same set of users), the clarification according to attributes was not needed.

However, as clarified contexts have no repeated objects and each object after the clarification process represents a set of objects from the original context, it is not possible to determine which objects were initially represented by the object in the clarified context. For example, in a business context where we need to understand the customer's profile to provide a better service or suggest consumption options, customer behavior storage in a clarified context is enough. However, in an audit system where security weaknesses in the Web evaluated, it would be ideal to handle a formal context with all objects, even the repeated ones, to be able to identify each element individually.

Figure 3 illustrates a formal context.

2.2.2 Formal Concepts

Using the *derivation operators* expressed by Eq.(2.1) and Eq.(2.2), a *formal concept* is defined as a pair $(A, B) = \{A \in G, B \in M\}$ such that $A' = B$ and $B' = A$, where A is called the *extent* and B the *intent* of the concept.

From the formal context shown in Figure 3 we can extract the formal concept $(\{usu_3, usu_4\}$, $\{orkut, google\})$, where $A = \{usu_3, usu_4\}$ represents the subset of users who access the websites *orkut* and $google$.

2.2.3 Concept Lattice

The set of formal concepts is ordered by the partial order $\preceq$ such that for any two formal concepts (A_1, B_1) and (A_2, B_2), $(A_1, B_1) \preceq (A_2, B_2)$ iff $A_1 \preceq A_2$ (equivalently, $B_2 \preceq B_1$). All the set of concepts ordered by $\preceq$ constitutes a complete lattice, the so-called concept lattice. The concept lattice obtained by a formal context (G, M, I) is denoted $\mathfrak{B}(G, M, I)$ (GANTER; STUMME; WILLE, 2005).

With all formal concepts sorted hierarchically by order of inclusion $\subseteq$ we can build the concept lattice. Sorting must be done so concept (A_1, B_1) is considered less than or equal to (A_2, B_2), if and only if, $A_1 \subseteq A_2$ (equivalent to $B_2 \subseteq B_1$). In this case, concept (A_1, B_1) is called subconcept and concept (A_2, B_2) superconcept (GANTER; STUMME; WILLE, 2005).

2.2.4 FCA Algorithms to Extract Formal Concepts

Two groups divide the formal concept extraction algorithms: batch and incremental. Batch algorithms compute the entire formal concept from scratch, taking the context into consideration. On the other hand, incremental algorithms build the concept lattice from the first object i provided in the ith step of the algorithm. To evaluate the effects of using BDDs in the

FCA derivation operator we chose two algorithms, each belonging to a group, to perform our adaptation and conduct tests. The selected algorithms were NextClosure (GANTER, 2010) and In-Close2 (ANDREWS, 2011).

2.2.4.1 NextClosure Algorithm

The NextClosure algorithm was proposed by Bernhard Ganter to generate closed attribute sets (GANTER, 2010). A set is closed in a given operation if the elements obtained by applying the operation to set elements also belong to the set. The algorithm works based on using closure operators to find closed attribute sets in lexicographical order (GANTER; WILLE, 2012).

Formally, when analyzing a formal context (G, M, I), the algorithm assumes a total, well-defined order between the elements of M. We can therefore assume that for a set M with n elements, we should stipulate an order such that: $M = \{m_1 < m_2 < ... < m_n\}$. Therefore, a set $A \subseteq M$ is lexicographically less than $B \subseteq M$, if $\exists b \in B \backslash A \; \forall a \in A(a < b \Rightarrow a \in B)$. That is, A is lexicographically less than B if the smaller element that differentiates both sets belongs to B.

Require: $A \subseteq M$
Ensure: Next closed set greater than A
  for For each $m \in M$ in descending order **do**
    if $m \in A$ **then**
      $A := A \setminus \{m\}$
    else
      $B := (A'' \cup \{m\})$
      if $B \setminus A$ has no element $< m$ **then**
        returns B
      end if
    end if
  end for

Algorithm 1: Next Closure(A, M)

The Algorithm 1 can iterate over all intents and pseudo intents. Below is a description of the algorithm based on the description found in (BAZHANOV; OBIEDKOV, 2011).

2.2.4.2 Algorithm In-Close2

The In-Close2 algorithm (ANDREWS, 2011) was developed using the canonicity test based on *backtracking* used by the In-Close algorithm (ANDREWS, 2009) and a combination of *depth-first search* and *breadth-first search* used in the FCbO algorithm (KRAJCA; OUTRATA; VYCHODIL, 2010) to enable attribute inheritance. In-Close2 incrementally creates closed sets of *parent concepts* and stores the new *child extents* during computations. Attributes stored during a closure are passed on to child extents to avoid retesting the attribute for inclusion during the closure of child extends.

According to its creator, the algorithm can be described as in Algorithm 2. Matrix I represents the formal context. Matrices A and B make up a formal concept where A represents the extent, B represents the intent and the variable r represents the index of a concept being accessed. r_{new} is the index for the next *candidate concept*.

Starting from y the algorithm iterates over the context, creating intersections between the current and the next extents. If the extent obtained by combining the attribute and the current extent is equal to the current extent, the attribute is added to the current extent. If the combination results in a new extent, a canonicity test is conducted on it. If a new extent is obtained, it is added as a child of the current concept to be processed later. The current intent is inherited by the child through the recursive function call.

The function $IsCanonical()$ goal is to verify whether the generated extension has been previously created using the *backtracking* technique mentioned above. But we will not describe the function here, for more details regarding In-Close2, please refer to (ANDREWS, 2011).

2.2.5 Implications

The knowledge embodied by a formal context (G, M, I) or its concept lattice $\mathfrak{B}(G, M, I)$ can be used to derive implications $P \rightarrow Q$, where P and Q are sets of attributes, which express that $P' \subseteq Q'$, or in other words, if an object has all the attributes in P , then it has all attributes in Q (equivalently, $P \rightarrow P''$) (GANTER; STUMME; WILLE, 2005).

Require: y : Initial attribute,
r : Current concept index
$jchildren \leftarrow \emptyset$
$rchildren \leftarrow \emptyset$
for $j \leftarrow y$ upto n - 1 **do**
  if $j \notin B[r]$ **then**
    $A[r_{new}] \leftarrow \emptyset$
    for $i \leftarrow 1$ to $i = A[r].size()$ **do**
      if $I[i][j]$ **then**
        $A[r_{new}] \leftarrow A[r_{new}] \cup i$
      end if
    end for
    if $A[r_{new}] = A[r]$ **then**
      $B[r] \leftarrow B[r] \cup j$
    else
      if $IsCanonical(r, r_{new}, j)$ **then**
        $jchildren \leftarrow jchildren \cup j$
        $rchildren \leftarrow rchildren \cup r_{new}$
        $B[r_{new}] \leftarrow B[r] \cup j$
        $r_{new} \leftarrow r_{new} + 1$
      end if
    end if
  end if
end for
for $K \leftarrow 0$ upto $\mid jchildren \mid -1$ **do**
  $InClosure2(rchildren[k], jchildren[k] + 1)$
end for

Algorithm 2: In-Close2(r, y)

$X \subseteq M$ *respects* an implication $P \rightarrow Q$ if $P \subseteq X$ implies $Q \subseteq X$. An implication $P \rightarrow Q$ *holds* in a set $\{X_1, ..., X_n\} \subseteq P(M)$ iff each X_i respects $P \rightarrow Q$; and $P \rightarrow Q$ *is an implication of the context* (G, M, I) iff it holds in the system of object intents. $P \rightarrow Q$ *follows* from a set of implications $\mathfrak{I}$ iff for every set of attributes X if X respects $\mathfrak{I}$, then it respects $P \rightarrow Q$. A set of implications $\mathfrak{I}$ is said to be *complete* iff every implication of (G, M, I) follows from $\mathfrak{I}$. Of particular importance are non-redundant sets of implications. A set of implications $\mathfrak{I}$ is said to be *redundant* iff it contains an implication $P \rightarrow Q$ that follows from $\mathfrak{I} \setminus \{P \rightarrow Q\}$.

It is important to note that association rules refer to rules with confidence between 0% and 100% (AGRAWAL; SRIKANT et al., 1994). In this paper, we focus on implications (i.e., rules with a confidence of 100%) because they reveal that whenever objects have the premise attribute set, they will also have the conclusion attribute set.

As mentioned earlier, a pattern can be used in many ways, such as establishing criteria to track users who access an implication premise which will result in improper access to a website present in the conclusion.

Several algorithms have been created to extract implications. We chose to use three: *FindImplications*, *Impec* and *NextClosure*. The next section presents a brief description of each algorithm.

It is important to note that the algorithms, when performed, require large computational effort mainly due to their complexity. The time complexity of *FindImplications* (CARPINETO; ROMANO, 2005) is at most proportional to $\Theta(\mid C \mid k^2 \mid M \mid q)$, where C is the number of concepts, k is the largest size of attributes in the premises, $\mid M \mid$ is the number of attributes and q is the largest number of parents per concept. The time complexity of *Impec* (TAOUIL; BASTIDE, 2001) is $\Theta(\mid M \mid 2^{|M|}(\tau_\varphi + 2^{|M|} \mid M \mid))$, where τ_φ is the maximum cost of calling the closure operator. *NextClosure* has a time complexity of is $\Theta(\mid 2^G \mid G \mid M \mid)$ (CARPINETO; ROMANO, 2005). However, the time needed to extract implications from the contexts considered were not prohibitive. For high-dimensional formal contexts, it is important to design or use a parallel or distributed version of the algorithms.

2.2.6 FCA Algorithms to Extract Implications

2.2.6.1 FindImplications Algorithm

The *FindImplications* algorithm proposed by Carpineto and Romano (CARPINETO; ROMANO, 2005) extracts an implication coverage, using the concept lattice. It is known for its completeness, being able to extract all implications in a formal context. It can find a coverage that although reduced can include redundancy.

For every formal concept (X, Y), the goal is to find the obtainable implications of (X, Y), which are implications $A \rightarrow B$, $A \subseteq Y$, $B = Y \setminus A$, such that $A \rightarrow B$ cannot be obtained from a concept (W, Z) which is a superconcept of (X, Y), which means it cannot be obtained from a concept $(W, Z) \geq (X, Y)$.

2.2.6.2 Impec Algorithm

The *Impec* algorithm (TAOUIL; BASTIDE, 2001) provides the minimum set of proper implications, known for having the lowest possible number of attributes on the left-hand side and only one element on the right-hand side. The algorithm makes use of the set of attributes M and a closure operator of M to find the implications. In other words, the algorithm finds the implications of the type $A \rightarrow b$, wherein $A \subseteq M$, $b \in M$ and there is no valid implications so that $D \rightarrow b$ where $D \subset A$.

The development of the right-hand side of the implication is obtained through the following proposition:

Let $A \subseteq M$ and $(.)''$ is a closure operator defined on M. The attribute *rightSide*(A) = $\{b \in M | A \rightarrow b\}$ is obtained with the proposition:

$$rightSide(A) = A'' - (A \cup \bigcup \{rightSide(B) | B \subset A\}) \tag{2.3}$$

The development of the left-hand side of the implication is obtained with the proposition:

Let $A, B \subseteq M, A \subset B$ and $(.)''$ is a closure operator defined on M, we have:

$$B'' = A'' \Leftrightarrow B - A \subseteq A'' - A \tag{2.4}$$

The algorithm obtains the implications for each set of attributes in M using its previously found implications. In the algorithm, the $\emptyset \rightarrow \emptyset''$ is the first implication used. After that, the implications of each attribute in M are verified. Existing implications are specialized using Equations 2.3 and 2.4 to get new implications. The propositions are applied at each specialization.

According to the authors (TAOUIL; BASTIDE, 2001), such implications are easy to understand, select and can be projected efficiently.

2.2.6.3 NextClosure Algorithm

The NextClosure algorithm was proposed by Bernhard Ganter to extract closed sets of attributes (GANTER, 2010). There is a closed set if an operation on members of the set always yields members of the same set. Afterward, Guigues (GUIGUES; DUQUENNE, 1986) proved that a minimal and complete cover of implications can be extracted from a formal context using the algorithm called Duquenne-Guigues or stem or canonical basis. With this smaller set of implications, it is possible to derive all the implications that represent the formal context. The algorithm is based on the use of closure operators to find closed attribute sets in lexicographic order (CARPINETO; ROMANO, 2005). Each implication has the form $P \rightarrow P''$ where P is called a pseudo-intent.

The importance of this implication basis lies in its minimum cardinality. However, the extraction of a canonical basis is a difficult problem that is at least coNP-complete (BABIN; KUZNETSOV, 2010) (DISTEL; SERTKAYA, 2011) (KUZNETSOV; OBIEDKOV, 2008).

The construction of a canonical base has been proposed by a few known algorithms. In (BAZHANOV; OBIEDKOV, 2011), *NextClosure* (GUIGUES; DUQUENNE, 1986), *LinClosure* (BEERI; BERNSTEIN, 1979) and Wild's Closure (BAZHANOV; OBIEDKOV, 2011) were compared, and in most cases, NextClosure outperformed the other algorithms.

This set of implications is of particular interest because it can represent all the relationships between attributes of a formal context using a reduced form. Applying the axioms of Armstrong it is possible to infer all the existing set of implications from this minimum basis of implications (ARMSTRONG, 1974).

Considering implication set $\mathfrak{I}$, it is possible to derive the complete implication set $\mathfrak{I}^+$ from the application of a complete set of axioms.

This question was deeply studied by Maier (MAIER, 1983) and the complete list of axioms is:

- Reflexivity : $Q \rightarrow Q$
- Augmentation : $Q \rightarrow R$ implies $Q \cup Z \rightarrow R$
- Additivity : $Q \rightarrow R$ and $Q \rightarrow Z$ implies $Q \rightarrow R \cup Z$
- Projectivity : $Q \rightarrow R \cup Z$ implies $Q \cup Z \rightarrow R$

- Transitivity : $Q \rightarrow R$ and $R \rightarrow Z$ implies $Q \rightarrow Z$

- Pseudo-transitivity: $Q \rightarrow R$ and $R \cup Z \rightarrow W$ implies $Q \cup Z \rightarrow W$

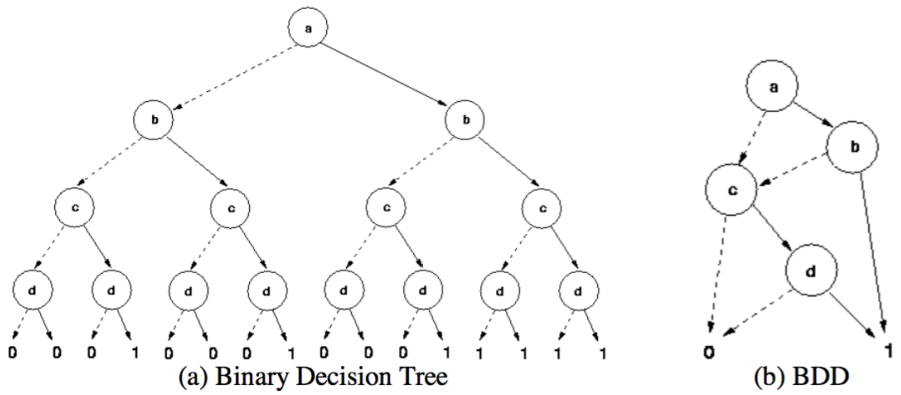
2.3 Binary Decision Diagram

BDDs are a data structure used to represent canonically binary functions (BRYANT, 1986), having a notoriously more compact structure than traditional structures (conjunctive and disjunctive normal forms) while also being easy to manipulate (CLARKE JR.; GRUMBERG; PELED, 1999). This property simplifies some operations, like checking the equivalence of two boolean formulas.

Making use of a binary decision tree it is possible to obtain a BDD. One can start merging identical subtrees present in the decision tree and then, eliminating nodes with identical left and right siblings. After this processes, the result is more similar to a graph than a tree in which nodes are eliminated and substructures are shared.

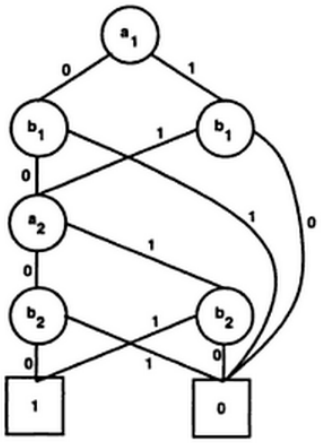
Formally, a BDD is a directed acyclic graph with two types of nodes: non-terminal and terminal. Each non-terminal node is a distinct variable of the corresponding boolean formula. Also, each node has two outgoing directed edges toward two children, which one represent the case if the variable is 0 (left) and the other represent the case if the variable is 1 (right). The truth-value of the formula false and true are represented by two terminal nodes labeled by 0 and 1 in a BDD. There is a corresponding path from the top of the BDD to a terminal node for each truth assignment to the boolean variables of the formula (BRYANT, 1986).

Figure 4 – The formula $(a \wedge b) \vee (c \wedge d)$ represented using decision tree and BDD



(a) Binary Decision Tree (b) BDD

Figure 5 – A two bit comparator BDD



Source: Clarke, Edmund M., Orna Grumberg and Doron Peled. Model checking. MIT press, 1999.

Figure 4 shows how the boolean formula $(a \wedge b) \vee (c \wedge d)$ is represented using a BDD and a Binary Decision Tree.

However, it has limitations which must be taken into account during implementation, since BDD's space complexity depends on the order in which the structure receives the variables.

As exemplified in (CLARKE JR.; GRUMBERG; PELED, 1999), consider a two bit comparator using variables a_1, a_2, b_1 and b_2. Values are identical if a_1 equals b_1 and a_2 equals b_2, otherwise they are different. By stipulating an order $a_1 < a_2 < b_1 < b_2$ the resulting BDD would have 11 nodes. However, if we kept related variables close to each other, in order $a1 < b1 < a2 < b2$ the resulting BDD would have 8 nodes (Figure 5). Thus, space complexity can range from $T(n) = 3n + 2$, in the best case, to $T(n) = 3 \cdot 2^n - 1$, that is, from a linear behavior $\Theta(n)$ to an exponential behavior $\Theta(2^n)$.

2.3.1 Formal Context represented by a boolean formula

First of all, a formal context must be converted into an equivalent boolean formula that can be used to create the BDD representation of the context. In the following, we present the same example used by Rimsa (RIMSA; ZÁRATE; SONG, 2009). Table 1 shows an illustration of a formal context and equation 2.5 shows a possible representation of this formal context as a logic function.

Table 1 – Formal Context Example

	a1	a2	a3
o1	**X**		**X**
o2	**X**	**X**	
o3		**X**	

$$f(a_1, a_2, a_3) = a_1\overline{a_2}a_3 + a_1a_2\overline{a_3} + \overline{a_1}a_2\overline{a_3} \tag{2.5}$$

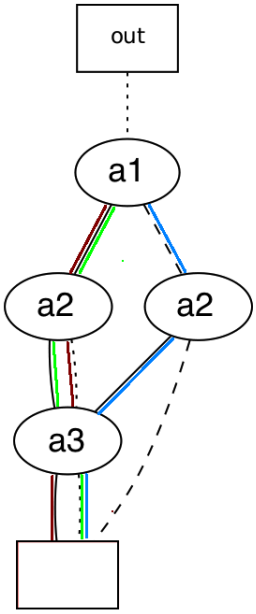


Figure 6 – Formal context of Table 1 as BDD

Note that a logic equation is used to represent every object, according to the incidence or not of attributes. If an object is present in the context the function $f(a_1, a_2, a_3)$ results is a positive state (1). Hence, if an object is not present in the context the function returns a negative state (0). Consequently, a logic function can be used to represent all formal contexts.

Figure 6 is a BDD formed by function 2.5. Each node represent an attribute. If a object has a attribute a line is used to link to the next node (attribute), otherwise a dotted line is used. Following the red line the object o1 can be found. The red line is showing us that there is a object in the BDD that has the attribute a1, does not have the attribute a2 and has the attribute a3. In the same way, object o2 is represented by the green line and object o3 by the blue line.

3 IMPLICATIONS FOR ANALYZING COMPLEX NETWORKS USING FCA

Our goal in this chapter is address the problem one, which consist of discovery hidden substructures of relationship in two-mode networks using FCA. The proposal is to handle tree issues:

a. Identification of conservative behavior.

b. Handling additive set of behavior.

c. Canonical behavior representation.

In an attempt to better understand the relationships formed by the nodes of a network, several studies have associated FCA and complex networks techniques. We discuss some major studies below.

3.1 Related Works

In the first research that used FCA for community detection in SNA, Freeman (FREEMAN, 1996) first transforms a one-mode data network into a formal context where nodes (objects) are described by cliques (attributes) they are involved. Then, *bridging cliques* and *bridging edges* are identified and exploited to get communities.

Freeman also presents in (FREEMAN, 2003) an extensive meta-analysis of techniques to analyze two-mode data networks to determine social groups as well as the core and peripheral members. In his work, twenty-one different methods are examined, and the benefits of FCA for social network analysis are highlighted.

Web communities are explored by Rome and Haralick (ROME; HARALICK, 2005). In their work the web pages and their links are defined as objects and attributes respectively in a formal context. The extent and intent of formal concepts are mapped respectively to the notions of authorities (pages considered important because they have numerous links to them) and hubs (pages that link to authorities pages), providing a way to gather similar communities and investigate the relationship between them.

In his work Jáschke (JÁSCHKE et al., 2006) develops an algorithm to mine tri-concepts that, represents FCA concepts in three dimensions. The algorithm is used to extract folksonomies data structures, in social resource sharing systems such as photos (Flickr), bookmarks (del.icio.us) and the system BibSonomy for managing scientific publications and bookmarks.

Experimentally FCA is applied to identify communities of researchers in (KIRALY, 2008). In their work, it is pointed out the inability of the FCA to handle too sparse relations. The problem is solved using a new concept, *power concept analysis*, allowing grouping of equivalent domains classes. Then, clusters of similar employees are found by complex networks analyze and tools.

Cuvelier and Aufaure (CUVELIER; AUFAURE, 2011) also carry out a study unifying FCA and complex network theories. In their study tweets about a specific subject are analyzed using FCA, allowing the establishment of relationships between messages. The final result is obtained by using the lattice to represent the network of messages and a data filtering criteria to assemble a topographical network graph to visualize efficiently the information gathered.

A socially-oriented work is presented by Poelmans et al. (POELMANS et al., 2011). It is proposed a semi-automatic process to expose a network of criminal organizations and their members. A lattice of suspected drug dealer is build using a police report database and a lattice containing suspect profiles. Criminal Networks are identified using the created lattices.

Aufaure and Le Grand (AUFAURE; GRAND, 2013) explore the FCA applicability in social network analysis. The authors describe the lattice expressiveness, especially when associated with ontologies. The paper is composed of a set of case studies. They observe that overlapping groups could be found by the lattices deterministically.

In a more recent work (CORDERO et al., 2015), FCA is used as formalization tool to create a logic (Simplification Logic) to identify user's influence in a social network. Structure containing the complete set of influences among users is build using implications as a core for an automated framework. Similarly to our work, his work also uses the implications to raise awareness of social networks.

3.2 Methodology : Materials and methods

As mentioned earlier, the objectives of this work are the identification of conservative behavior, handling the additive set of behavior and canonical behavior representation. In this section, we describe the database considered and the methodology followed starting from a data pre-analysis, data selection, formal context creation, implications extraction and finally the representation of these implications as a complex network.

It is possible to build a network composed of sets of websites (premises and conclusions) connected by edges (implications). In this way, we turn a two-mode network into a one-mode network. Hence, we expose existing relationships between event type nodes, which is not feasible in the original network.

As shown in Figure 2 the data are usually not available in formal context structure for analysis. It is, therefore, necessary to define which are the objects and attributes to build the formal context to be used. The algorithms for extracting implications have high complexity related mainly to the number of attributes (DISTEL; SERTKAYA, 2011). Therefore, it is fundamental that we develop an appropriate amount of attributes to enable the extraction of implications promptly.

3.2.1 Considered database

A Brazilian cable Internet Service Provider, with completely anonymous access data, provided the considered database. Records refer to accesses done in March 2009, adding up to a total of 6,319,333 accesses. One access indicates that a user consumed resources (pages) from one website during some moment of their session.

In order to create our formal contexts, we performed a selection on the websites since it would not be feasible to analyze the entire access database due to computational time constraints when running these algorithms as we mentioned early. Therefore, the twenty-five most accessed websites in the month of March were selected. The user connections data were converted to a formal context with websites as attributes and users as objects. Websites from the same domain, for example ”www.globo.com”, ”ads.globo.com”and ”bbb.globo.com” were grouped together

for simplification sake, and categorized only as ”globo”. The number of attributes (websites visited by users) was reduced to fifteen after the grouping.

We created three data sets, which were used to construct all the formal contexts to explore each issue to be addressed.

a. The data set to the identification of conservative behavior.

We selected connections between 03/02/2009 and 03/09/2009 (one week) to create formal contexts to extract all the implications using the FindImplications. We counted 165,659 accesses (24,9% of the total) to the fifteen selected domains, done by 29,319 distinct users. Based on work done by (J. et al., 2015) which enabled identification of periods of greater access, we created nine formal contexts from specific days and times that represented periods of domestic and business activity. Table 2 shows contexts (K1-K9), the time and day in which accesses were logged, the number of users in the context (objects) and the number of profiles (user sets with identical accesses) in the clarified context.

Table 2 – Contexts and their characteristics

Context	Access Period	Users	Profiles
K1	Monday from 8 AM-6 PM	11,387	715
K2	Tuesday from 8 AM-6 PM	11,095	710
K3	Wednesday from 8 AM-6 PM	10,829	672
K4	Thursday from 8 AM-6 PM	10,860	700
K5	Friday from 8 AM-6 PM	10,633	683
K6	Friday from 10 PM-2 AM	5,445	315
K7	Saturday from 10 PM-2 AM	5,172	294
K8	Sunday from 10 PM-2 AM	5,482	309
K9	Wednesday from 10 PM-2 AM	5,487	296
K10	Saturday 10 PM - Sunday 10 PM	18,362	1,283
K11	March	45,426	6,405

b. The data set to handle the additive set of behavior.

The Impec algorithm received as input a formal context (K10) created with the accesses from Saturday 10 PM to Sunday 10 PM with connections of 18,362 users representing 1,283 profiles.

Figure 3 in section 2.2.1 shows part of the formal contexts obtained after processing the data.

c. The data set for canonical behavior representation.

We selected the user accesses of the entire month of March to create the formal context (K11) used to extract a canonical set of implications. We selected 1,470,450 accesses (23,27% of the total access done in March) to the fifteen selected domains, done by 45,426 distinct users. We built just one formal context from this data. The total number of objects was reduced to 6,405 after the clarification process. Each of these objects represents, after the clarification process, the behavior of one or more users defining a user profile.

3.2.2 Social Networks Visualization Based on Implications

It is possible to build social network visualization from the sets of premises and conclusions connected by edges (implications).

The following steps are performed to create nodes and edges from implications so they can be displayed as graphs:

1. Consider each premise and conclusions as a node. Each node is a set of attributes used for identification.

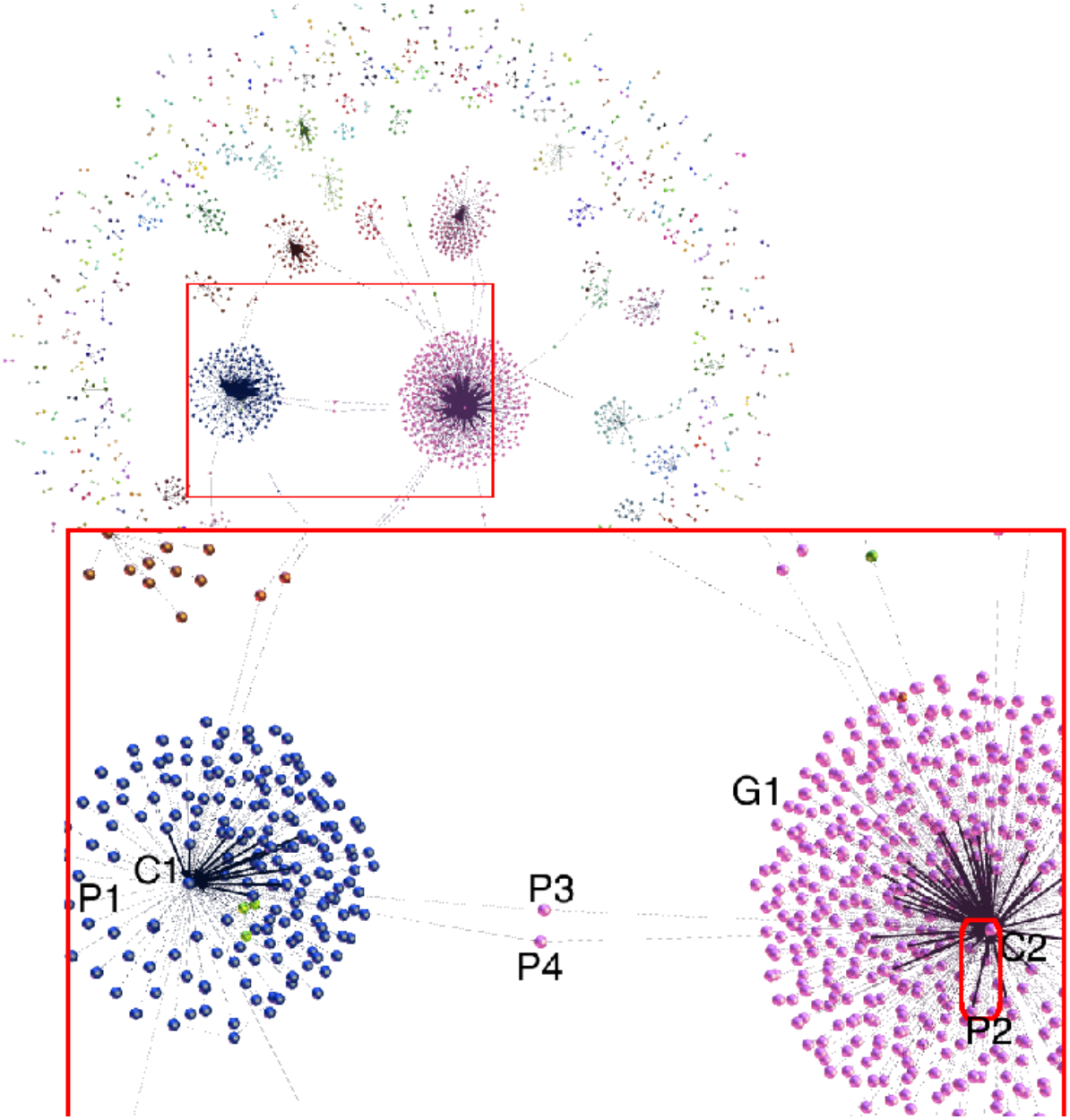
2. Create edges connecting the nodes. Each implication is converted to an edge connecting two nodes.

The network formed by the implications yields a directed graph, with edges going from the premise to the conclusion.

We propose a method to turn two-mode networks into one-mode networks. Consequently, we expose the existing relationships between event type nodes, which was not feasible in the original network. Hence, the difficulty to structurally analyze the bipartite network of heterogeneous nodes was overcome taking into account the implications.

The one-mode projection of two-mode networks is sometimes used to analyzes two-mode networks. A projection of event type nodes of a two-mode network will result in a one-mode network similar to the network of implication, where premises and conclusions are event nodes, however without the explicitness of dependence between the nodes that an implication network can reveal. Moreover, access patterns cannot be visualized in a one-mode projection.

Figure 7 – Approximate depiction of the network formed by implications



As mentioned before, we aim to find patterns between attribute relationships and propose the use of implications to creating a one-mode network. The result would yield every complex network technique applicable to this type of network. Therefore, using implications, it is possible to get a more meaningful representation, discover hidden substructures in two-mode networks and save space without loss of knowledge.

3.3 Experiments, Analysis and Results

We used the algorithms implementations found in (VIMIEIRO, 2007) and our analysis environment used is the framework proposal by Dias and Vieira (DIAS; VIERIA, 2011). (www.asdfg.com.br.)

3.3.1 The Identification of Conservative Behavior in Two-mode Networks Using All Implications

To identify conservative behavior in two-mode networks, we used the contexts from K1 to K9 defined in table 2. Thus, these formal contexts constructed in section 3.2.1 are provided as input to extract all implications.

We propose to look for subsets of intersections between implications. Consequently, enabling the gathering of recurrent access patterns. In table 3 are listed some examples of conservative behavior found between some formal context. In the first column is shown the formal contexts used and in the second column are examples of conservative behavior found represented as implications.

Table 3 – Some recurrent access patterns from intersection

Contexts	Implications (access patterns)
K1 ∩ K2	{doubleclick yahoo youtube orkut}→{google}
	{uol doubleclick uai orkut}→{google}
	{uai orkut gstatic yieldmanager}→{google yahoo}
	{uol yieldmanager gstatic google}→{yahoo}
K1 ∩ K5	{uol yahoo msn uai}→{google}
	{orkut yahoo msn terra}→{google}
	{youtube uol uai}→{google}
	{doubleclick terra gstatic google}→{yahoo}
K3 ∩ K7	{ceipmsn gstatic uai}→{google}
	{uai youtube globo}→{google}
	{youtube doubleclick uai}→{google}
	{atdmt securestudies}→{google}

The number of implications extracted and the number of common implications between contexts are shown in Table 4. In the first column are the formal contexts analyzed, the second column is filled with the total number of implications extracted. The remaining of the columns are the formal contexts used to create the intersection. Through an analysis from Table 4 one can see that the accesses performed on Wednesdays (K3) are more similar to the accesses performed

on Friday (K5) than any other intersections (sixty implications). Besides, Monday (K1) and Friday (K5) are in second in the number of conservative behaviors, fifty-one implications. It is also possible to note that there is no intersection during the weekend days (K7-K8). These results can be used to improve the bandwidth management and provide a better service.

Table 4 – Implications and intersections counts

		Daytime				Nightly			
Cxt	N.R	K2	K3	K4	K5	K6	K7	K8	K9
K1	1.246	50	35	20	51	0	2	0	0
K2	876		29	13	19	2	2	0	0
K3	955			37	60	0	5	0	1
K4	835				46	0	2	0	0
K5	919					1	0	0	1
K6	205						0	0	1
K7	123							0	3
K8	155								0
K9	149								

In addition to the intersections above, other intersections were found as the intersection between K1, K2, K3 and K4 where we found the following implication:

$$\{orkutgstatic, doubleclick, yahoo\} \rightarrow \{google\} \tag{3.1}$$

This implication (3.1) is valid for all four contexts. It can reveal us that every time users access the premise websites they also access the conclusion websites, with 100% of confidence. This assertion is possible thanks to the implication. In this case, we identify an access pattern that repeated itself in four days of the week, from Monday to Thursday.

In the implication (3.1), accessed websites are theoretically harmless. However, in the case of an implication which exposes improper behavior, with a premise website set that leads to dangerous conclusion websites, it can be used as a malicious behavior pattern and have alerts and exclusive controls associated with it whenever users access the websites present in the premise.

For non-clarified context K1 the implication (3.1) identify the behavior of thirteen users, which means that every time these thirteen users accessed the premise websites, conclusion websites were also accessed by them. For non-clarified context K2, there were nine users. Seven users were identified in the context K3. For context K4, the total of identified users was also seven. With this access pattern, we found a total of thirty-two distinct users, of which five followed the pattern in at least two days. Table 5 shows the user identification whenever he behaves according to the implication (3.1).

Table 5 – Accesses according to implications - Recurring users

Mon	Tue	Wed	Thu
1255	1255	1255	
2323	2323		
	3042		3042
8875	8875	8875	
		25109	25109

If we take implication (3.1) as suspicious user behavior, we can start a detailed control process, tailored to recurring users. As soon as one of them accesses a premise website, a security policy can be initiated.

3.3.1.1 Visualization of the Network with Conservative Behavior

Figure 7 shows an approximate depiction of a network form by implications from contexts K3 and K5. Both contexts have a higher number of intersections between implications (sixty implications as shown in table 4). The *Force Atlas* algorithm was used in the assembly of the layout, along with *Clustering Chinese Whispers* that identified 449 clusters (formed by implications with the same conclusion). Node P1 represents a premise linked to a conclusion, in this case, C1 (google, yahoo), located in the center of the cluster, on the left. Node P2, another premise, is linked to conclusion C2 (google), forming an implication that was recurrent in both days and because of this its edge has a different weight, making it darker (highlighted area). It is possible to determine that premises linked to the same conclusion, such as the G1 group, represent users who have similar behavior, making use of strong ties principles (GRANOVETTER, 1983). If the access to conclusion websites raises concern, such as increased bandwidth consumption, when one of the conclusion premises occurs, automatic configurations to avoid network bottlenecks can be triggered. Premises P3 and P4 are linked to more than one conclusion, indicating various possible actions from the premises. Premises with more than one conclusion only took place because implications from two days were used to create the graph. None of these observations can be done using the original data visualization and analysis.

3.3.2 Handling Additive Set of Behavior in Two-mode Networks with One-Item Conclusion Implications

In order to handle the additive set of behavior in two-mode networks, we used the formal contexts K10 in Table 2. So, the formal contexts formed by users accesses between Saturday and Sunday was given as input to Impec algorithm for extracting the implications.

The list of one-item conclusion implications obtained was extensive. The total amount

Table 6 – Some Impec Implications

	Implications list	Holds Vacuously
1	{globo securestudies rad.msn}→{atdmt}	no
2	{securestudies yieldmanager ceipmsn}→{atdmt}	no
3	{securestudies google rad.msn doubleclick}→{atdmt}	no
4	{youtube securestudies uol}→{atdmt}	no
5	{youtube securestudies ceipmsn}→{atdmt}	no
6	{globo gstatic oss-content.securestudies doubleclick}→{atdmt}	yes
7	{globo gstatic oss-content.securestudies yahoo}→{atdmt}	yes
8	{globo orkut oss-content.securestudies doubleclick}→{atdmt}	yes
9	{globo orkut oss-content.securestudies yieldmanager}→{atdmt}	yes
10	{globo oss-content.securestudies doubleclick yahoo}→{atdmt}	yes

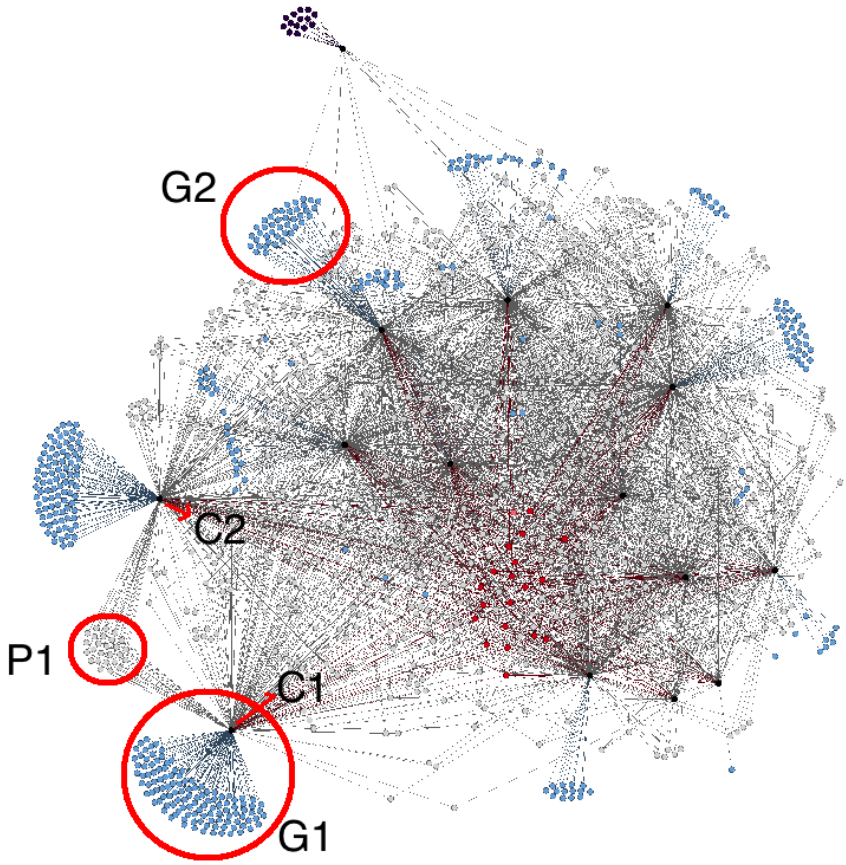
was 3,340 implications. All with a conclusion containing only one attribute (website). This kind of implication is ideal for those looking for dependence related specific attributes (websites) as premises or conclusions.

However, not all the implications have objects in the context that follow their rules, we say that they *hold vacuously* (CARPINETO; ROMANO, 2005). As an example, in Table 6, we list some implications that conclude with access to the website ATDMT. The total of implications was seventy-one but only five not hold vacuously. In Table 6 are listed ten implications with a conclusion to site ATDMT where five of these implications have objects that follow their rules. The total amount of implications that not hold vacuously was 2,473 (74,04%).

3.3.2.1 Visualization of the Network of Implications with Additive Set of Behavior

Figure 8 presents the network with one-item conclusion implications. Due to network characteristics, one significant component is shown. Some implications premises have more than one conclusion creating bridges between clusters, unlike other implications networks. Group P1 is an example of premises with more than one conclusion. In this example, the premises in P1 imply with the conclusion C1 (Google) and C2 (Orkut). Premises with only one conclusion create groups such as G1. The other groups such as G2 in blue too, have this characteristic. It is possible to use one item conclusion implications to expose premises that may imply different conclusions. So, since it is not feasible to establish just one conclusive behavior from the premise of some implications, these implications can be of no interest if you are interested in predict possible future behavior. Thus, the premises with more than one conclusion may be discarded. Furthermore, these nodes (premises) may be removed to allow the identification of groups easier (EASLEY; KLEINBERG, 2010). On the other hand, in the case of interest on the additive set of behavior, the premises with one conclusion can be removed from the graph exposing just

Figure 8 – Network form by the implications extracted by Impec algorithm



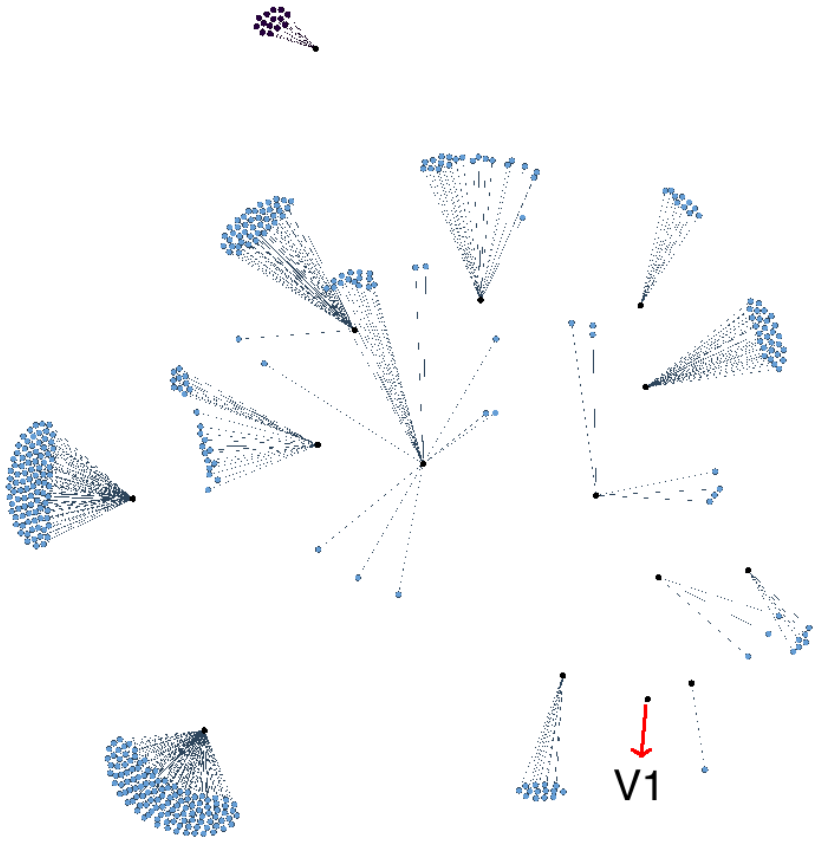
premises with more than one conclusion.

Therefore, such type of analysis can hardly be carrying out just by observing the listing of implications. The conclusions of the implications are the darkest regions of Figure 8 and they are fifteen each one corresponding to an attribute (website). The algorithm *Fruchterman Reingold* (FRUCHTERMAN; REINGOLD, 1991) and the clustering algorithm *Chinese Whispers* (BIEMANN, 2006) were used and only two clusters were found. Besides, some premises have many different conclusions, for example, of all premises with more than one conclusion, twenty were connected to twelve different conclusions.

However, if you ignore the implications premises linked to more than one conclusion and also obvious implications such as X implies X, there are 1,076 implications left of a total of 3,440 which together form the Network of Figure 9. Each component in the graph presents a set of users access profiles that conclude with specific user access. The premises are nodes in light gray, and the black nodes are the conclusions.

Therefore, if we are looking for users who accessed a website present at the implication conclusion, the implications with the additive set of behavior (premises with more than one

Figure 9 – Network formed by the implications extracted by Impec algorithm (only premises connected to one conclusion)



conclusion) need to be evaluated carefully. If you a looking for a particular accesses to a website (conclusion) as a result of a set of other accesses (premises) that characterize a crime a classification error may occur once the websites set accessed on the premise might imply access to different conclusions (websites). Therefore, the possibility of excluding premises with more than one conclusion from the analysis is primordial to avoid mistakes. In Figure9 one can easily see that no implication premise has concluded only with access to the website yieldmanager (V1). There are no premises linked to the node V1. Thus, as this figure does not show premises linked to more the one conclusion we can deduce that, all the premises that conclude with yieldmanager also concludes with another website. Also, Figure9 is also an example of bridging nodes exclusion as it is formed by the same data present in Figure 8 without the bridging nodes.

3.3.3 Canonical Behavior Representation in Two-mode Networks Using a Canonical Set of Implications

In order to address this issue, we used the formal context K11 in table 2, section 3.2.1. Hence, the formal context formed by more than a million users connections was provided as input for extraction of a canonical set of implications. Part of this set is illustrated in Table 7.

The amount of canonical implications extracted was 216.

Table 7 – Some of the implications extracted.

	Implications list
1	{youtube uai doubleclick uol yahoo}→{google}
2	{globo orkut youtube uai ceipmsn}→{google}
3	{orkut uai gstatic atdmt yieldmanager yahoo}→{google}
4	{youtube uai terra uol yahoo}→{google}
5	{globo orkut oss-content ceipmsn yahoo}→{atdmt}

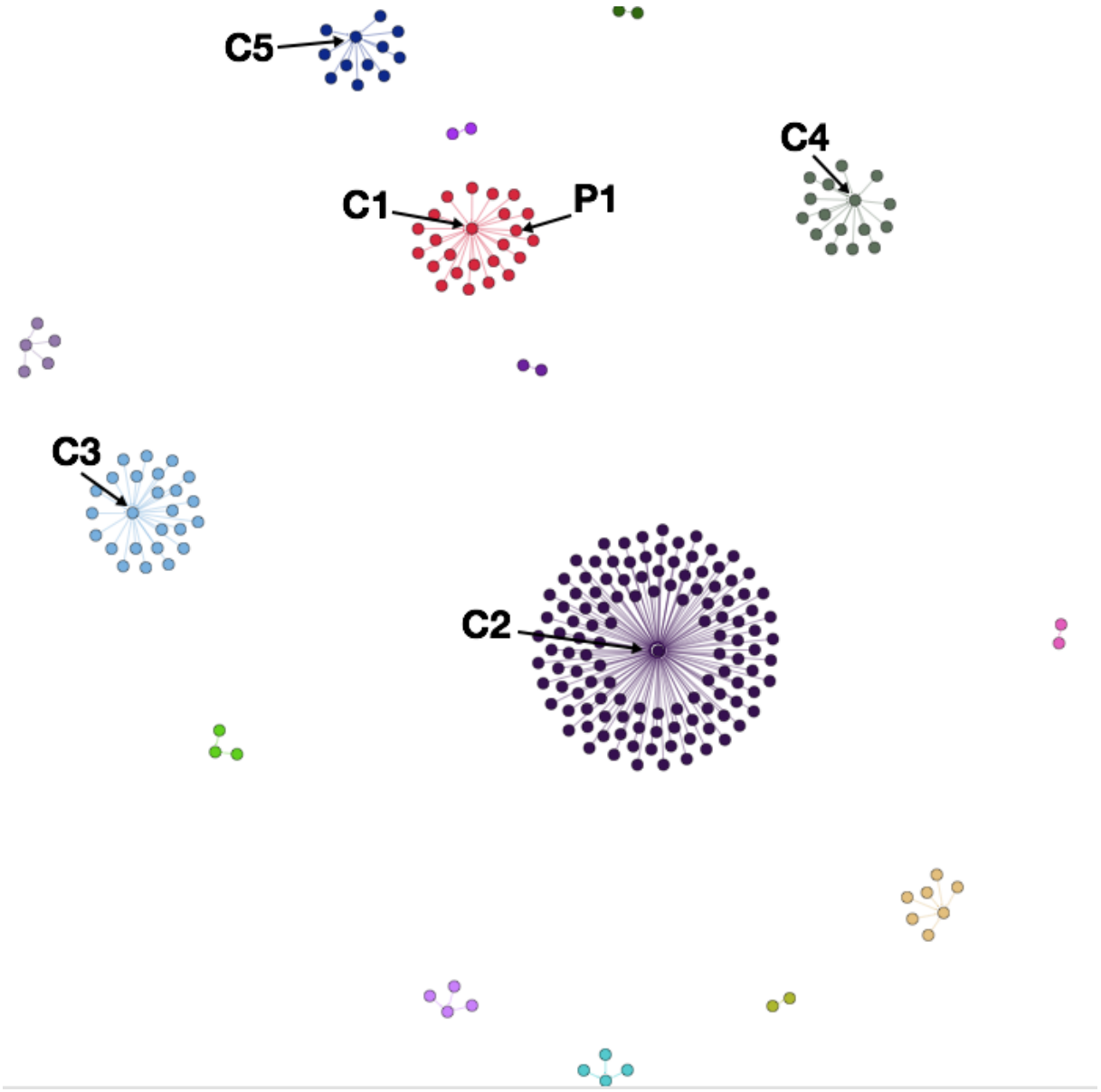
The canonical set of implications can be used to generate the whole set of implication or a particular set of interest.

Besides, the data relating to the accesses used for canonical representation in the database as a log file containing the user code and the name of the website needs twenty-two Megabyte of space. Our proposal for canonical representation reduces expressively the space required for storing such information.

The formal context is similar to a technique already used to represent such network. Tamassia (TAMASSIA, 2013) highlight the use of a matrix where each coordinate of the array is a type of node. With this transformation of the data, there is a great saving of memory. The connections, represented as a no clarified formal context need 988 Kilobytes of space, i.e. a reduction of 95,52 % of the space required. However, this representation would not let us retrieve how many times a user has accessed one specific website. After the clarification process, the context requires 136 Kilobytes. The total reduction was of 99.38%, keeping all the user's behavior.

Furthermore, we start from a list of 1,470,450 connections that are represented in a formal context with 45.436 objects, which after implications extraction could be represented with only 216 implications requiring a storage space of twenty-one Kilobytes. With the original space required to store the users accesses equal to 22 Megabyte, the total reduction of storage space was equal to 99.90 %.

Figure 10 – Network formed by implications



3.3.3.1 Visualization of the Network of Canonical Implications

Using implications premises and conclusions as nodes allow us to identify the main groups and the most common conclusion of users accesses quickly. As one can note in Figure 10 Google (C2) is the conclusion in the middle of the largest component in the graph. One can also easily visualize small clusters formed by a few numbers of nodes. It is not possible to visualize such clusters in the network form by the original data.

Each central node located at the center of the clusters in the graph of Figure 10, represents an implication conclusion form by one or more websites. The node C1 corresponds to the

conclusion with the website {atdmt} and all the premises related to it are access patterns that always occur in combination with the conclusion. The ATDMT website is a cookie tracker, used to track user access behavior. This component grouped all sets of websites linked to this tracker. One can conclude that the websites that form the premises used the website tracker's resources.

The premise P1 corresponds to websites {orkut uai oss-content.securestudies ceipmsn }. The first two websites are respectively one social networking website and a Brazilian news portal. The other two websites are cookies trackers as the ATDMT website (conclusion C1), and they are not intentionally accessed by the user. Scripts in pages loaded by users call these trackers. The implication formed by premises P1 and conclusion C1 shows that users who access the premise websites always have their cookies tracked.

The algorithm *Force Atlas* (JACOMY et al., 2014) assembled the layout of Figure 10. Making use of the principles of *strong ties* (GRANOVETTER, 1983) is possible to establish that all the premises related to the same conclusion can also be linked indicating the proximity between users behaviors. The other conclusions C2, C3, C4 and C5, which are the major components of the graph respectively, comprise websites Google, Orkut, yieldmanager (a cookie tracker also) and Terra (Brazilian news website).

3.4 Discussions

The implications depicted in Figure 10 contains all the user behavior during the month of March 2009. In addition to being a canonical vision without loss of knowledge related to the behavior of users accesses, this network, consisting of websites sets with dependencies between them, reveals unknown substructures of the original two-mode networks. However, more than one non-redundant minimum set may exist (CARPINETO; ROMANO, 2005) (CORDERO et al., 2012). In the light of this, it is not advisable to make comparisons across canonical implications extracted from different contexts.

Furthermore, if an implication unveils an improper behavior, with premise websites that lead to dangerous conclusion websites, it can be used as a malicious behavior pattern and have alerts and exclusive controls associated with it that can be triggered as soon as users access the premise websites.

For economic reasons, an application example of such information would be identifying premises linked to a conclusion with websites that demand more bandwidth. When some

premises of this conclusion occur, automatic safeguarding measures can be triggered. If the implication unveils improper behavior, the Armstrong axioms (ARMSTRONG, 1974) can be used to derive a complete set of behaviors. A system can use this set as a malicious behavior pattern and have alerts and exclusive controls associated as said before.

In a generic view, implicit network substructures are exposed using implications that highlight the dependency relationships between attributes (event type nodes). Each element of an implication (premise or conclusion) has a set of one or more attributes. The attributes are grouped together because they are characteristic of one or more objects from the formal context. Thus, the implication shows a relationship between the premise and conclusion. Considering attributes as nodes, we can establish a relation between these nodes that is not visible without such treatment.

As the implications reveal a cause and effect relationship with 100% of confidence between objects that are identified by them, depending on what attributes represent, they can establish behavior patterns among user access when attributes represent websites. These patterns can be used to predict accesses that should receive some attention for configuration or safety reasons.

3.5 Conclusions

In order to resolve the first problem we associate FCA and complex networks through the analysis of implications sets. The result allows us to uncover access patterns, recurrent access, patterns and additive access patterns. Furthermore, through the use of implications sets we analyze the relationships between event type elements (websites) in two-mode networks.

We propose a method to identify conservative behavior in two-mode networks with all implications extracted using the FindImplications algorithm. Users accesses in different days of a week are analyzed. By implications intersections, patterns and recurring users are identified. We found recurrent access patterns and also users that follow this patterns on different days. The network formed by implications from two days helped to find premise groups with the same conclusion. To identify a suspicious user behavior, we could start a detailed control process, tailored to recurring users.

We also propose the use of implications extracted by the Impec algorithm to handle an additive set of behavior in two-mode networks. The one-item conclusion implications are easy

to understand (TAOUIL; BASTIDE, 2001) and can be more relevant to anyone looking for one particular website access pattern. Its implications allow analyzing of premises with more than one conclusion. Besides, ignore this premise sets can help avoid mistakes during a categorization of user behavior. We identified a website, *yieldmanager*, whose premises always connect to other conclusions indicating that the user access profile (premise) that visited this website always visited another website too (conclusion).

Besides, we also propose a method for canonical behavior representation in two-mode networks using a canonical set of implications (steam base), which present a minimal universe of data without loss of information. With the compressed knowledge provided by this minimum set of implications, it is possible to use the axioms of Armstrong (ARMSTRONG, 1974) to expand this set and get all the existing implications in the context. However, this expansion can be made focusing on a particular implication of high interest, either for economic or safety reasons.

Besides, one log file related to the users accesses in the database used for canonical representation required a space of twenty-two Megabyte. Our proposal for canonical representation expressively reduces the space required to store such information and after implications extraction, only 216 implications represented the users accesses behaviors requiring a storage space of twenty-one Kilobytes. The total reduction of storage space was 99.90%, thus creating a possibility of permanently storing information for later analysis, avoiding information disposal.

Moreover, some hidden relationships were found that expose the use of cookies trackers by websites, which was not visible by looking at the network of the users and websites. Cookies trackers can read and access browser cookies to mine information about the users. With the implications extracted in our experiments, we found relationships between a news website and a social network with a cookie tracking website.

4 USING BINARY DECISION DIAGRAMS TO EXTRACT FORMAL CONCEPTS VIA FORMAL CONCEPT ANALYSIS FOR HIGH DIMENSIONAL FORMAL CONTEXTS

In this chapter we present our proposal to address the problem two, which consist of apply the Binary Decision Diagram (BDD) in FCA algorithms to extract formal concept in highly dimensional formal context.

The remainder of this chapter is organized as follows. Related works are given in Section 4.1. The methodology used for representing formal context and formal concept is presented in Section 4.2. The empirical analysis is given in Section 4.3 while the rest of the chapter provides discussions and conclusion.

4.1 Related Works

FCA stemmed from the work of Gratzer (GRÄTZER, 2003), which introduced General lattice theory. Later, Wille reworked into lattice theory and concept hierarchy (WILLE, 2009).

As mentioned before, we will use the algorithm proposed in (GANTER, 2010), NextClosure, to extract closed attribute pseudo sets. The same algorithm is also currently used to obtain a canonical set of implications.

In-Close2, which we presented before, was considered to be a realization of the Close-by-One (CbO) algorithm by its author. Consequently, its performance was compared to the FCbO algorithm, a different implementation of CbO, which was until then considered to have the best performance (KRAJCA; OUTRATA; VYCHODIL, 2010). However, In-Close2 was shown to be more efficient for more realistic, lower-density contexts. For this reason, we will implement an adaptation of this algorithm using BDD, hoping to achieve further performance gains in situations where the algorithm has poor performance.

When looking for related works using BDD, we found an article by Salleb (SALLEB; MAAZOUZI; VRAIN, 2002) proposing a strategy to mine the maximum frequent item set in a database transaction base. Each transaction, composed by items, was considered to be a bit vector, indicating whether or not the item is a part of the transaction. Since the authors

represented transactions using BDD and each maximum set obtained was also represented using BDD when treating transactions as objects, the representation used by the authors is similar to what we used in our work.

We also came across a work by Yevtuchenko, who developed an algorithm for formal concept extraction using BDD (YEVTUSHENKO, 2002). However, his approach used BDD to represent the set of identified concepts. It is possibly the first work to use BDD in the formal concept extraction process.

In addition to the aforementioned paper, Yevtushenko proposed two additional algorithms (YEVTUSHENKO, 2004): one for generating all formal concepts of a context (Grail), based on depth-first search; and another to visualize conceptual lattices. Furthermore, he wrote a very thorough comparison of the NextClosure (GANTER, 2010), Nourine-Raynaud (NOURINE; RAYNAUD, 1999), Norris (NORRIS, 1978), Norris-Godin (KUZNETSOV; OBIEDKOV, 2002), (OBIEDKOV, 2003), Charm(ZAKI; HSIAO et al., 1999), SimpleDFS(YEVTUSHENKO, 2004) algorithms and his own Grail (YEVTUSHENKO, 2004).

Like us, he chose to use synthetic contexts. However, he also performed tests on real databases, because one of his goals was making comparisons with more realistic and, therefore, bigger databases.

It is worth noting, though, that his work evaluated only contexts with up to 900 objects and 50 attributes. Our tests reached as many as 1,000,000 objects and 25 attributes. In his solution, BDD is presented as an implicit representation structure to calculate all intents. Nonetheless, BDD is used to represent the intent lattice and was shown to be efficient only for highly dense contexts.

We agree in part, because as we will present below, our use of BDD to represent the formal context got better results for contexts with varying densities using the NextClosure algorithm. When it comes to the InClose2BDD algorithm, the best performance is linked to more dense contexts.

We drew inspiration from the most relevant formal concept extraction algorithms, since they all use FCA operators in some way and can benefit from the performance gains. For more information, please refer to (BORDAT, 1986) (GODIN; MISSAOUI; ALAOUI, 1991) (KUZNETSOV, 1993) (GODIN; MISSAOUI; ALAOUI, 1995) (CARPINETO; ROMANO, 1996) (NOURINE; RAYNAUD, 1999) (ZAKI; HSIAO et al., 1999) (LINDIG, 2000) (VALTCHEV; MISSAOUI, 2001) (QIAO et al., 2003) (GANTER; STUMME; WILLE, 2005) (BERTET; GUILLAS; OGIER, 2007) (GANTER, 2010) (NGUIFO; NJIWOUA, 1998) (LINDIG, 1999) (STUMME et al., 2000) (OBIEDKOV, 2003) (YEVTUSHENKO, 2004) .

Our work extends a work previously developed by Rimsa (RIMSA; ZÁRATE; SONG, 2009), which also extracted formal FCA concepts using BDD. However, the work was focused on evaluating BDD libraries to find the best one to use in FCA. In his work, Rimsa uses brute force methods to obtain the sets of all possible intents and consequentially the corresponding extent BDDs. By doing operations in the formal context represented as a BDD, concepts are validated and selected.

Results obtained with the algorithm with and without BDD were compared, and all versions using BDD were faster than versions without it. As is stated in the work, for the final tests with multivalued contexts of 70 attributes and 60,000 objects, execution times of the algorithms using BDD were, on average, 73,33% faster. With multivalued contexts of 70 attributes and 120,000 objects, the BDD implementations had an edge of 76,16% over other implementations.

Contrary to Rimsa, we propose an adaptation to the FCA derivation operator to use BDD resources, and incorporate this adaptation in the NextClosure and In-Close2 algorithms to analyze the results. Since both algorithms belong to different algorithm classes, batch (NextClosure) and incremental (In-Close2), we sought to meet expectations concerning the use of BDD in these two algorithm classes.

Poelmans (POELMANS et al., 2013a) (POELMANS et al., 2013b) evaluated 1072 articles focused on models, technique and applications of FCA and did not cite any articles using BDD. As mentioned before, however, we were able to find works by Yevtushenko (YEVTUSHENKO, 2004) and Salleb (SALLEB; MAAZOUZI; VRAIN, 2002).

4.2 Methodology: Materials and Methods

FCA still lacks well-defined methodologies to conduct performance tests in algorithms. Due to this reason, we followed the approach proposed by Moraes (MORAES; ZÁRATE; FREITAS, 2010), which uses randomly generated synthetic formal contexts with controlled densities and dimensions.

To generate synthetic formal contexts, we used the tool described in (RIMSA et al., 2013) because it is capable of generating irreducible random contexts, that is, with no duplicate objects and within the boundaries of controlled densities and size. Tests with such contexts allow for better exploration of algorithm scalability as input sizes vary.

In his approach, Moraes (MORAES; ZÁRATE; FREITAS, 2010) used contexts with up to 15, 20 and 25 attributes. Since we are following his method, we decided to do the same. Attributes were matched to sets of 1,000, 5,000, 10,000, 50,000, 100,000, 250,000, 500,000 and 1,000,000 objects and densities varied from the bare minimum allowed for the context, 30%, 50%, 70% and maximum permitted density.

Randomly generated incidence tables can vary even for contexts with the same dimensions and densities, which results in a different amount of concepts and therefore varying performances for each context.

We decided to set sample sizes to 10 formal contexts for small contexts, with execution times in the order of seconds. For bigger contexts, which can take from minutes to hours to analyzes, we reduced sample sizes to 5. These numbers aim to obtain average execution times for contexts of the same size and density, but with different attributes x objects incidence tables. Evidently, the same contexts were used to feed the original algorithms as well as the BDD-enhanced versions for comparison.

We compared execution times for all implementations. We also conducted additional tests to validate both approaches. This way, results and concepts generated by the algorithms, with and without BDD, were compared to assure the results are the same.

4.2.1 Chosen BDD Library

In (RIMSA; ZÁRATE; SONG, 2009) several BDD libraries, considered the most relevant, were evaluated. Among them, we selected CUDD - Colorado University Decision Diagram. Despite not being the best performer, it is kept more up to date and its code is maintained more frequently than other libraries.

Fabio Somenzi (SOMENZI, 2009) developed the library that it is currently in version 2.5.0. In the library, many function packages are exposed to manipulate Binary Decision Diagrams (BDDs), Algebraic Decision Diagrams (ADDs) and Zero-suppressed Binary Decision Diagrams (ZDDs). However, in our work, we used just the BDD manipulation functions.

There are three ways to use the CUDD library. The first is like a black box, where programmers use functions that are exposed to manipulate their BDDs without worrying about their inner workings. The second is like a white box, and programmers can change the library to

adapt to their problems, a better option when performance is the end goal. The third and final way is through an interface, which relieves the programmers from the job of controlling memory management processes. We used the first and third options in this work.

The basic structures of the library are the nodes, manager, and cache. Nodes have several members, but the ones used in a black box implementation are the variable index, reference counter, and node value. The remaining members are pointers linking nodes to one another, which are used to implement the *unique table*. The variable index is a permanent attribute which reflects creation order, and an *unsigned short int* variable limits its maximum possible value, whose maximum value is platform-dependent.

The manager - DdManager - is tasked with maintaining the unique table structure, assuring that each node is unique, as well as similar structures. Due to this, the application must initialize the manager, by calling the appropriate function.

Due to how it is structured, BDDs can have their size altered according to variable order. Consequently, how fast one can access information contained in the diagram can also vary. CUDD provides a vast set of sorting algorithms, which are variations of already developed technical solutions: (RUDELL, 1993), (DRECHSLER; BECKER; GÖCKEL, 1996), (BOLLIG; LÖBBING; WEGENER, 1995), (ISHIURA; SAWADA; YAJIMA, 1991), (F., 1993), (JEONG; KIM et al., 1993), or developed for the library: (PANDA; SOMENZI; PLESSIER, 1994), (PANDA; SOMENZI, 1995).

4.2.2 Representing a formal context with BDD

To create the BDD that would represent the formal context, we used an algorithm similar to the one proposed in (RIMSA; ZÁRATE; SONG, 2009).

Algorithm 3 describes the process of building a formal context using BDD. It takes as input a file whose lines are objects that may or may not have certain attributes according to a particular indicator ('X'). It is therefore perceivable that the BDD assembly is related to variables (attributes) of the context. Consequently, the algorithm loops through all object attributes and, if the object possesses that attribute, the BDD variable indicated by index "i" is included in the object's temporary BDD as a node with the true value. Otherwise, the variable is added as a node with the false value. Finally, the algorithm adds the BDD representing the object to the BDD representing the context. This way, one can build the context BDD executing AND operations to

```
Require: File
Ensure: Context_Format_BDD
  FileLine = File.GetLine()
  while !File.EOF() do
    TemporaryBDD = BDD.New()
    for i = 0 to NumAttributes do
      if FileLine[i] == 'X' then
        TemporaryBDD &= InsertTrueNode(i)
      else
        TemporaryBDD &= InsertFalseNode(i)
      end if
      BDD_Context |= TemporaryBDD
    end for
  end while
```

Algorithm 3: Building a formal context using BDD

insert variables and OR for objects.

Once we create the BDD representing the formal context, some operations can be done quickly, taking advantage of an optimization inherent in the structure, as demonstrated in Algorithm 4. The algorithm returns a BDD with the objects that have the desired attributes, which in our example are attributes 4, 22, 33 and 44. To do this procedure, all that is needed is a logical AND operation with the BDD representing the context and the BDD variables that describe the attributes.

```
Require: BDD Context
Ensure: BDD Objects
  Objects = BDD.New()
  Objects = Context & bdd_ithvar(4) & bdd_ithvar(22) & bdd_ithvar(33) & bdd_ithvar(44)
```

Algorithm 4: Extracting an intent from an assembled BDD

Note that in Algorithm 4 each iteration calls the function bdd_ithvar() which returns objects that have the attribute passed as a parameter. The & operator allows us to get the intersection of the values returned by the functions. This way, the line containing calls to this function attributes to the "Objects" variable the objects that are in the intersection between the original context and objects returned by each function. In the end, the "Objects" variable will contain a BDD representing objects that share attributes {4,22,33,44}.

4.2.3 Where to use BDD

The choice to use the FCA derivation operator to insert the BDD was made based on a study of the most computationally expensive parts of the NextClosure algorithm, which scrutinized its major procedures. We took into account the total execution time, the time spent by the DoublePrime function and time spent by the NextClosure function.

After running some tests, we realized that the DoublePrime function has an influence of up to 99.96% compared to the total execution time, as shown in Table 8. Moreover, this value increases as contexts become denser and have higher dimensions. Table 8 contains the average of collected execution times, in seconds, for the tested contexts.

Table 8 – Influence of the DoublePrime function in the NextClosure algorithm

Dimensions	Density	Time	DoublePrime	NextClosure	Influence
15x1000	min	1.30s	0.32s	0.95s	25.16%
15x1000	30%	4.38s	0.97s	3.17s	22.23%
15x1000	50%	13.13s	5.59s	7.46s	42.59%
15x1000	70%	10.28s	10.02s	0.22s	97.43%
15x1000	max	10.59s	10.32s	0.19s	97.43%
15x5000	min	24.52s	9.52s	14.76s	38.84%
15x5000	30%	21.75s	9.45s	12.47s	43.46%
15x5000	50%	45.67s	42.68s	2.89s	93.47%
15x5000	70%	48.74s	48.47s	0.18s	99.44%
15x5000	max	48.71s	48.44s	0.18s	99.44%
20x1000	min	4.36s	0.44s	3.73s	10.31%
20x1000	30%	130.77s	5.68s	124.81s	4.35%
20x1000	50%	2734.66s	72.38s	2658.66s	2.65%
20x1000	70%	1260.96s	379.55s	878.78s	30.10%
20x1000	max	462.64s	452.96s	7.45s	97.91%
20x10000	min	223.68s	47.69s	175.08s	21.32%
20x10000	30%	1159.23s	239.13s	917.94s	20.63%
20x10000	50%	9813.27s	2601.70s	7201.81s	26.51%
20x10000	70%	4108.29s	4098.00s	7.49s	99.75%
20x10000	max	4146.57s	4136.93s	7.21s	99.77%
20x30000	min	1555.06s	437.90s	1112.45s	28.16%
20x30000	30%	4454.54s	1213.45s	3230.19s	27.24%
20x30000	70%	12250.36s	12240.24s	7.34s	99.92%
20x30000	max	12214.43s	12204.27s	7.31s	99.92%
20x70000	min	7314.46s	2485.38s	4804.43s	33.98%
20x70000	30%	6406.35s	2880.93s	3513.07s	44.97%
20x70000	70%	28366.03s	28355.61s	7.54s	99.96%
20x70000	max	28303.93s	28293.34s	7.63s	99.96%

Drawing from the above observation, the function DoublePrime, composed of the sequential call to the FCA derivation operators, was chosen as the subject to evaluate the use of the structure, because performance gains from including BDD in this function would have the most impact on total execution time.

4.2.4 NextClosure Algorithm Implementation

The original implementation described in Algorithm 5 uses a bit structure to store objects and their attributes. In our BDD implementation, we created a similar structure, but instead of using a bit structure we replaced it with BDD.

Require: Formal context in cxt format
Ensure: Set containing all formal concepts in the provided context.
$LoadFormalContext(CxtFile, Context)$
$X = \varnothing$
while $X.size() < Context.NumAttributes()$ **do**
$X = Nextclosure(X, extent, intent)$
$conceptList.add(extent, intent)$
end while

Algorithm 5: Extraction of formal concepts

The main structure of Algorithm 5 was not altered, remaining for the most part exactly as described in Algorithm 1 in subsection 2.2.4.1. The difference is in the data type used to represent the formal context and its attribute set. Both were now represented using a BDD structure.

The module responsible for loading the formal context, loadFormalContext(), takes as input a file in the Burmeister format (BURMEISTER, 2003), representing the formal context. The program store the information in a bit matrix. When an object has a certain attribute, the corresponding attribute bit is set to a '*true*' value. All other operations performed in the context also handle data in this format.

Since the attribute set – variable X in Algorithm 5 – is initialized with the smallest possible size and grows in lexicographical order until it has all the attributes existent in the context, the algorithm calls the functions DoublePrime and NextClosure while the maximum size is not reached.

The NextClosure function was explained previously, and thus more details are not needed. However, the alterations to include BDD structure were done in the DoublePrime function, which is why we go into more detail about it below.

Algorithm 6 contains pseudo code for the logic behind the DoublePrime function, which returns a set of attributes derived from the provided attribute set. To accomplish this, the algorithm makes use of two operations. In the first one, the extent of provided attributes is computed. In other words, all objects sharing the attributes provided as parameters are extracted

from the formal context. The second function calculates the final derivation, which returns the intents of the object set returned by the first derivation. To do that all attributes shared by the objects obtained from the extent are identified. This attribute set represents the final derivation of the original attribute set, which corresponds to the use of the $(.)''$ FCA operator, present in the equations in section 2.2.1.

Require: Attribute set $X \subset M$
Ensure: X'' = Attribute set derived from X.
$X' = Extent(FormalContext, X)$
$X'' = Intent(FormalContext, X')$
$returns\ X''$
Algorithm 6: DoublePrime Algorithm - Application of the $(.)''$ operator

Algorithm 7 describes the steps in function $Extent(FormalContext, X)$, which receives a formal context (G, M, I) and an attribute set $X \subseteq M$ as parameters. From these parameters, it returns a set of objects $X' = \{g \in G \mid \forall m \in X : gIm\}$ containing these attributes. The algorithm loops through all objects in the formal context and checks whether or not they have the attributes in the provided attribute set. The computational complexity of the algorithm is $\Theta(\mid G \mid \times \mid X \mid)$.

Require: Formal context (G, M, I) and an attribute set $X \subseteq M$
Ensure: Set of objects containing the attributes in X
for $allg \in G$ **do**
$HasAttributes = true$
for $all\ m \in X$ **do**
if $!gIm$ **then**
$HasAttributes = false$
end if
end for
if $HasAttributes$ **then**
$X' = X' \cup \{g\}$
end if
end for
$returns X'$
Algorithm 7: Returns the set of objects containing the attributes in X

Algorithm 8 describes the logic behind the $Intent(FormalContext, X')$ function. The function parameters are the formal context (G, M, I) and the set of objects $X' \subseteq G$, which is returned from function $Extent(FormalContext, X)$. The function returns a set of attributes shared by all objects of the provided object set $X'' = \{m \in M \mid \forall g \in X' : gIm\}$. It loops through all context attributes and checks whether all objects of the object set provided have that attribute. If all objects have the attribute, it is selected to be a part of the returned set. The computational complexity of the algorithm is $\Theta(\mid M \mid \times \mid X' \mid)$.

Require: Formal context (G, M, I) and a set of objects $X' \subseteq G$
Ensure: Set of attributes shared by all objects of X'
for $allm \in M$ **do**
$HasAttribute = true$
for $all\ g \in X'$ **do**
if $!gIm$ **then**
$HasAttribute = false$
end if
end for
if $HasAttribute$ **then**
$X'' = X'' \cup \{m\}$
end if
end for
$returns X''$
Algorithm 8: Returns a set of attributes shared by all objects of object set X'

We have implemented an optimization in both functions. When it comes to obtaining the extent, Algorithm 7, we propose manipulating a dynamic list of objects. For each searched attribute, the algorithm would walk through the list, eliminating objects that do not have the attribute. So, for each attribute examined, the list would be reduced proportionally to the incidence of that attribute in the formal context. Regarding the extraction of the intent, the algorithm would manipulate a dynamic list of attributes. The algorithm checks the set of objects passed as a parameter, and only attributes shared by the object are passed forward to be tested for the next object. In our tests conducted after the optimization, the improved NextClosure algorithm was up to 4.5 or 10 times faster than the original version.

4.2.5 Using BDD

Algorithms 9 present the new version of the $Extent()$ function after the adaptations made to take advantage of BDD features.

Require: Formal context (G, M, I) and a set of attributes $X \subseteq M$
Ensure: Set of objects containing the attributes in X
$X' = bddAnd(BddContext, BddAttributes)'$
$returns X'$
Algorithm 9: Extent - Returns the set of objects containing the attributes in X

The function $bddAnd()$ does all the work which was previously described in Algorithm 7. Function $bddAnd()$ computes the conjunction of two BDDs and returns a pointer to the resulting

BDD if it is successful. In this case, one of the BDDs passed as a parameter is a formal context benefiting from structural optimizations due to being represented by a BDD. The other BDD represents the set of attributes that must be shared by the context objects that will be returned. As described above, the context BDD is formed by a sequence of BDDs representing objects and linked by OR operators, whose attributes are nodes of that BDD.

To better visualize these structural changes, imagine the formal context as used by Romano and Carpineto (CARPINETO; ROMANO, 2004), formed by planets of the Solar System and some of their characteristics. Figure 11 illustrates such a context.

Figure 11 – Formal context of Solar System planets and some of their characteristics

Once data from this context are loaded, as described in section 4.2.2, the resulting BDD can be represented as shown in Figure 12. Thus, the nodes are attributes, and the context objects are formed by combining these attributes, represented by the dotted and solid lines.

For instance, take the object representing Earth, which is small, has a short distance from the sun, has a moon and short days. It is represented by the combination of the nodes linked by the blue line. Each time the Earth has an attribute a solid line links the node that represents the attribute to the following node (attribute), otherwise a dotted line is used. Another example is the path to form the Jupiter object, which nodes are highlight by a red line. These two planets (objects) share the last three nodes. These common nodes are an example of a possible simplification.

Let us take the attribute long distance from the sun, *distance far*, as an element belonging to the set of attributes to be passed to the function $Extent()$, along with the formal context. This attribute, in the BDD representation, could be described according to what is shown in Figure 13.

When these two BDDs undergo a conjunction operation, the resulting BDD only has paths to attributes (nodes) which satisfy the previous BDD as an attribute set. In our example, all large planets, medium planets, and tiny Pluto. The Earth object is no longer a part of this BDD.

Algorithms 10 present the new version of the $Intent()$ function after the adaptations to take advantage of BDD features were made.

In order to run, Algorithm 10 takes as input a BDD representing a set of objects, like

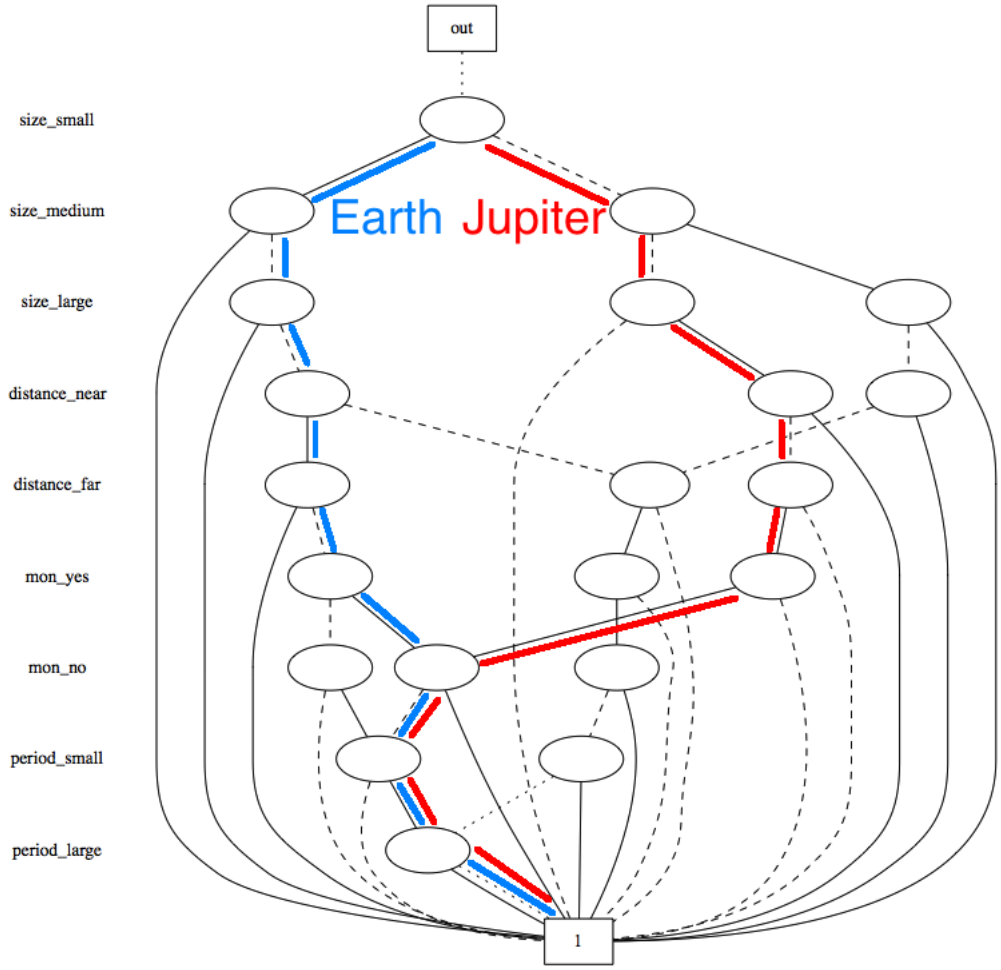


Figure 12 – BDD representation of the Solar System formal context

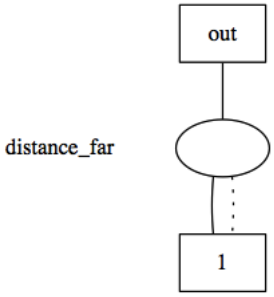


Figure 13 – Attribute of the context represented with a BDD.

the objects illustrated in Figure 14. The algorithm loops through all attributes of the formal context, assembling BDDs for each one. If the conjunction of an attribute BDD and an object BDD results in a BDD identical to the original object BDD, it means that all objects represented by that object BDD have the attribute represented by that attribute BDD. This attribute is then inserted into a separate attribute list so that later a BDD containing all attributes shared by the objects can be created. This logic is similar to the original algorithm, with the exception that objects do not have to be tested to check whether or not an attribute is shared by all objects. The

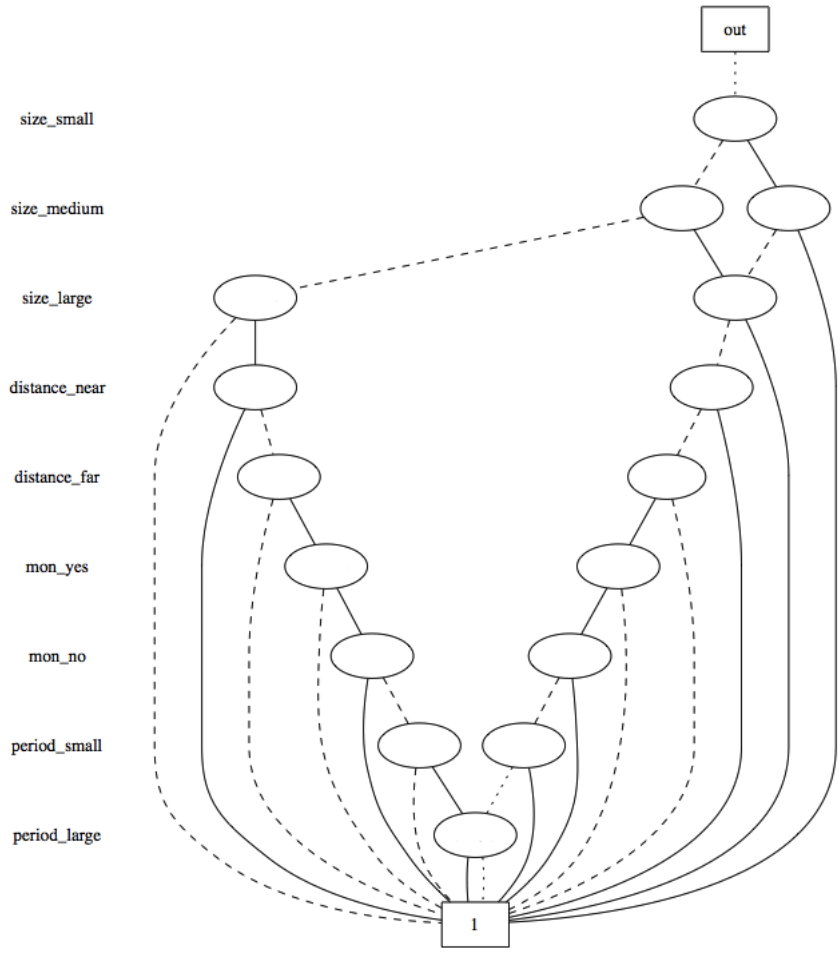


Figure 14 – Result of a conjunction between a context BDD (Figure 5) and an attribute BDD (Figure 6).

result of the algorithm is displayed in Figure 15, for an input similar to what is shown in Figure 14.

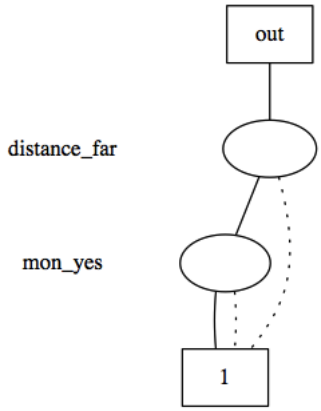


Figure 15 – BDD of attributes shared by objects in Figure 14 BDD.

Adaptations done to the In-Close2 algorithm are similar to the adjustments explained above. BDD operations replaced the derivation operation. However, when using BDD, the row

Require: A set of objects $X' \subseteq G$ in BDD structure
Ensure: Set of attributes shared by objects in X', in BDD structure.
for $all\ m \in M$ **do**
$BddAttribute = m$
$Bddtmp = bddAnd(X', m)$
if $Bddtmp = X'$ **then**
$AttributeList = m$
end if
end for
for $all\ attribute \in AttributeList$ **do**
$X'' = BddAnd(X'', attribute)$
end for
$returns\ X''$
Algorithm 10: Returns a set of attributes shared by all objects in input parameter X'

Table 9 – Minimum and Maximum densities of processed contexts

Attributes	15		20		25	
Objects	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
1.000	21.94%	78.06%	13.85%	86.15%	10.60%	89.40%
5.000	29.88%	70.12%	18.42%	81.58%	13.62%	86.38%
10.000	34.98%	65.02%	21.12%	78.88%	14.81%	84.19%
50.000	#	#	27.06%	72.94%	18.54%	81.46%
100.000	#	#	30.51%	69.49%	20.54%	49.46%
250.000	#	#	?	?	22.69%	77.31%
500.000	#	#	?	?	25.35%	74.65%
1.000.000	#	#	?	?	25.35%	74.65%

sorting stage had no benefits and was therefore eliminated. We made significant changes to the function that checks the canonicity of generated extents. Each BDD is unique and also has a unique identifier. In the BDD In-Close2 implementation, the extent is represented by a BDD and stored in a *HashMap* which enables checking of previously computed extents in $\theta(1)$ time.

4.3 Experiment, Analysis and Results

Minimum and maximum densities vary for each tested context. Table 9 presents all minimum and maximum densities of our tests. Please note that it is not possible to achieve densities of 30% and 70% for some contexts. For these cases, only the minimum, maximum, and an average value of 50% were considered for comparison.

Since the contexts have no duplicate objects, it is impossible to create certain contexts with the number of attributes we used. If we consider 15 attributes, it would be possible to create

a context with up to $2^{15} = 32,768 - 2$ objects (empty objects and objects containing all attributes are not generated). When the number of attributes did not allow for a possible combination to produce the desired number of objects, we fill the cells in tables 10 and 11 with the "#" symbol. Some contexts have little variation between minimum, average, and maximum density values, and were therefore not tested, since variations in execution times would be too small and would contribute very little to our tendency analysis. In these cases, cells were filled with "?" instead of the density.

The results obtained by running both the original algorithms and the BDD adaptations are below.

4.3.1 Nexclosure-BDD

Results for the BDD NextClosure algorithm, presented in Table 10, were obtained using a workstation with 32 GB of RAM, Lubuntu 15.04 64-bit operating system, and a 2.4 Ghz processor. Table cells filled with the "*" symbol represent executions that failed to complete due to memory overflow problems, in which the algorithm consumed all of the RAM memory (32 GB) as well as swap memory (32 GB), in a total of approximately 64 GB of memory consumption. The adapted algorithm, using BDD, consumed less memory and allowed for the execution of the algorithm with bigger, denser contexts. All the execution times are in seconds.

In Table 10, one can observe that as we increased the number of attributes, execution times for the original algorithm, without BDD, become faster than the fitting algorithm. Let us take as example the contexts with 1,000 objects shown in Table 10. The BDD implementation was more efficient than the original version in all five density variations, with speedups of up to 3.7 considering contexts with 15 attributes. However, when we look at contexts with 20 attributes and 1,000 objects, the BDD implementation was faster for contexts with minimum densities as well as densities of 30%. For densities of 50%, 70% and maximum densities, the original algorithm enhanced with our suggested optimizations was faster, with speedups of up to 1.6. Sadly, the benefits brought by the BDD implementation come with the cost of manipulating a more complex data structure. Since the BDD implementation improved mainly the manipulation of objects by replacing the derivation operators, it becomes a more interesting option as the number of objects increases. For example, in contexts with 10,000 objects, the BDD algorithm was better in all cases, with speedups of up to 23.55 in contexts with 15 attributes and maximum density.

In concept extractions performed with 20 and 25 attributes, the BDD implementation was able to process contexts with densities and sizes that the original implementation was not able to handle. Therefore, the use of the BDD implementation was mandatory in these cases to enable concept extraction in high-dimension, high-density contexts. As an example, we can take the contexts with 20 attributes and 50,000 objects, with densities of 70% and maximum. For these contexts, only the BDD implementation was able to extract the concepts, as can be seen in Table 10.

The NextClosure algorithm belongs to the class of algorithms that extract concepts in batch. The introduction of BDD in these algorithms brought benefits to contexts of varied dimensions and densities. BDD is a better option for both low and high-density contexts, but it becomes a bad choice when the number of attributes increases significantly. As demonstrated, attributes are represented as BDD nodes. This representation has a high cost. The proportion is of one attribute for each node into the BDD. That is why, when the number of attributes increases, the BDD implementation starts losing performance. Objects, however, are represented as BDDs and the algorithm manipulate them in groups. This group manipulation is what gives the algorithm a better performance when the number of objects is increased.

The BDD implementation of NextClosure was able to run up to 23 times faster than the original implementation, in addition to allowing processing of formal contexts with bigger dimensions and higher densities, impossible to be handled by the original implementation, by using computational resources more efficiently.

4.3.2 InClose2-BDD

Results for the In-Close2 algorithm, found in Table 11, were obtained using cloud equipment, provided by the Microsoft Azure service, with 112 GB of RAM, Ubuntu 15.04 64-bit operating system, and nine processor cores. Table cells filled with an "*" represent scenarios that we did not run. All the execution times are in seconds. Due to the enormous amount of time necessary for the algorithm to finish we did not process these scenarios. The synthetic context with 25 attributes, 1,000,000 objects, and 25% density, for example, took 20 hours to be generated. In addition to this fact, we decided not to run the algorithm with greater densities (70% and more). Since the goal was to find the intersection of the execution time curves (contexts for which the BDD implementation is faster than the original) and the execution times for the context with 1,000,000 objects and 25 attributes were already faster for the BDD implementation.

Table 10 – Comparisons Original NextClosure X BDD implementation

	Attributes	15					20					25				
Objects	Density	Min	30%	50%	70%	Max	Min	30%	50%	70%	Max	Min	30%	50%	70%	Max
1,000	Orig.	0.042	0.104	0.88	2.11	2.848	0.06	0.747	9.368	45.628	79.623	0.056	1.996	51.924	*	*
	BDD	0.024	0.066	0.35	0.768	0.77	0.049	0.697	11.092	72.733	109.866	0.082	6.678	174.372	2558.602	*
5,000	Orig.	0.78	0.766	5.152	7.392	7.706	1.245	7.527	98.86	234.452	323.257	1.16	40.88	*	*	*
	BDD	0.082	0.09	0.904	0.808	0.81	0.389	4.737	69.234	260.655	332.459	0.6	42.114	1118.114	*	*
10,000	Orig.	3.048	#	11.54	#	16.768	4.805	22.431	235.574	454.803	554.973	4.41	142.396	*	*	*
	BDD	0.15	#	1.156	#	0.712	0.553	6.034	92.808	353.034	298.037	1.134	126.608	2480.63	*	*
50,000	Orig.	#	#	#	#	#	81.563	190.643	*	*	*	*	*	*	*	*
	BDD	#	#	#	#	#	4.173	13.81	*	634.813	456.632	*	*	*	*	*

The In-Close2 algorithm is capable of extracting formal concepts more efficiently than the NextClosure algorithm. Consequently, we conducted tests with a greater number of objects to try and find spots where the InClose2BDD algorithm can outperform In-Close2. However, we had to use equipment with more memory capabilities.

As we can see in Table 11 it is perceivable that as the density increases, execution times for the BDD implementation of the algorithm become better than the original one. Let us take as example the context with 1,000 objects presented in Table 11. Considering contexts with 15 attributes the BDD implementation was more efficient for all dimensions and densities of 70% or maximum. Speedups reached up to 2.22 for contexts with maximum densities and 5,000 objects.

When analyzing contexts with 20 attributes, the same logic applies. InClose2BDD is faster for all contexts with maximum densities, being surpassed only for the smallest context (1,000 objects) with 70% density. With 100,000 objects, InClose2BDD was faster for contexts with 50% density. It is noticeable that as the number of objects increases, InClose2BDD requires lower densities to outperform In-Close2. As one can see, InClose2BDD was faster only for contexts with maximum density and 1,000 objects. With 5,000, 10,000 and 50,000 objects, the BDD implementation was faster for densities of 70% and above. However, with 100,000 objects and densities starting from 50%, InClose2BDD becomes the better option. Consequently, for the context with 20 attributes and 100,000 objects and maximum density, Inclose2BDD was 6.67 times faster than the original implementation. As previously mentioned, objects and their incidences are represented as BDDs. As the density increases, BDDs allow for a more compact representation of the density due to other optimizations, clearing the path to performance gains when manipulating the structure. Depending on how dimensions change, BDD performance is better at increasingly lower densities.

When analyzing execution times for contexts with 25 attributes, the same pattern emerges. As we increase densities or the number of objects, InClose2BDD becomes a better option.

Overall, in some of the tested contexts with 25 attributes, InClose2BDD was the only reliable option to extract concepts, because it was the only one able to finish its execution. It is important to highlight that only the memory optimizations brought by the BDD implementation allowed the computation of some of the results. Even when we used equipment with more available RAM memory, In-Close2 was not able to compute formal concepts from formal concepts of maximum density with 10,000 or more objects, and 70% density with 50,000 or more objects. The execution times for these contexts, presented in Table 11, correspond to the execution time accumulated until a “segmentation fault” occurred, by a failure attempting to allocate memory in unreachable areas due to lack of free memory. Therefore, to better differentiate between these times and times for executions that successfully finished, faulty times are in italic. We concluded that if we had more memory available, execution times would be

Figure 16 – Time for extracting concepts of contexts with 25 attributes and objects between 1,000 and 1,000,000

greater than what we were able to measure.

The In-Close2 algorithm belongs to the incremental concept extracting algorithm class. The introduction of BDD in this algorithm has been particularly beneficial for high-density contexts. Since BDD nodes represent attribute incidences, when they are very abundant their storage is optimized by BDD features to save space and accelerate access to nodes when applying FCA derivation operations. Therefore, BDD is a real option for high-density contexts and is not a good one for sparse contexts. As we demonstrated, however, when the number of objects increases, the required density for InClose2BDD to become a better option than the original becomes lower and lower. Nonetheless, its use can make viable the utilization of contexts previously thought to be impossible to manipulate.

4.4 Conclusions

The NextClosure algorithm is a batch algorithm, due to its characteristic of generating concepts from scratch considering the entire context. We expected that BDD would prove to

Table 11 – Comparisons of Original InClose2 X BDD implementation

	Attributes	15					20					25				
Objects	Density	Min	30%	50%	70%	Max	Min	30%	50%	70%	Max	Min	30%	50%	70%	Max
1000	Orig.	0	0	0,01	0,054	0,074	0	0,002	0,164	1,254	3,708	0	0,03	1,154	24,856	147,362
	BDD	0	0,01	0,022	0,04	0,04	0,008	0,04	0,342	1,848	2,062	0,02	0,12	2,88	46,698	84,962
5000	Orig.	0	0	0,06	0,154	0,16	0	0,04	0,66	4,434	9,414	0	0,17	6,042	77,274	404,854
	BDD	0,04	0,04	0,082	0,072	0,072	0,074	0,196	1,646	3,138	2,946	0,118	0,864	18,236	108,498	124,842
10000	Orig.	0,02	#	0,116	#	0,196	0,016	0,066	1,114	8,054	13,47	0,01	0,318	11,062	141,062	*535,876*
	BDD	0,086	#	0,14	#	0,118	0,176	0,364	2,474	3,732	3,422	0,248	1,84	33,156	140,864	135,59
50000	Orig.	#	#	#	#	#	0,136	0,24	4,584	28,094	30,148	0,118	1,236	41,134	*623,48*	*566,804*
	BDD	#	#	#	#	#	0,948	1,222	5,11	5,266	4,66	1,378	7,346	104,2	206,664	177,398
100000	Orig.	#	#	#	#	#	0,402	#	8,978	#	39,452	0,316	2,25	74,506	*678,384*	*568,828*
	BDD	#	#	#	#	#	1,938	#	6,502	#	5,91	2,93	11,946	145,334	240,93	202,812
250000	Orig.	#	#	#	#	#	#	#	#	#	#	0,97	4,816	171,678	*657,652*	*552,504*
	BDD	#	#	#	#	#	#	#	#	#	#	7,408	22,678	213,14	285,172	229,878
500000	Orig.	#	#	#	#	#	#	#	#	#	#	2,775	7,755	330,49	*635,34*	*571,212*
	BDD	#	#	#	#	#	#	#	#	#	#	16,88	35,59	262,474	305,037	240,597
1000000	Orig.	#	#	#	#	#	#	#	#	#	#	7,58	13,89	669,87	*	*
	BDD	#	#	#	#	#	#	#	#	#	#	37,54	54,51	342,5	*	*

be a better option for all contexts with high density. However, for contexts with 1,000 objects, as the number of attributes and density increased, the BDD implementation did not perform better than the original, due to the high cost of managing its structure. As we increased the number of objects, however, BDD quickly becomes more efficient for all variations of contexts tested with the NextClosure algorithm. It is worth highlighting that for the tests conducted using the original implementation of NextClosure without our dynamic list optimization, the BDD implementation was faster in all cases. We, therefore, conclude that the BDD implementation was faster in most instances, and it is a better option for contexts with a high number of objects and high density. The BDD enables the execution of the algorithm with contexts that would be impossible to analyze. In some cases, the algorithm can be up to 22 times faster than the original (context with 15 attributes and 10,000 objects, maximum density).

The In-Close2 algorithm belongs to the incremental class, characterized by generating the contexts in small increments starting from a target object. The BDD implementation of this algorithm reaped benefits for contexts of high densities and an enormous number of objects. Nevertheless, as context sizes increased due to the addition of more objects, InClose2BDD becomes faster than the original implementation in smaller densities as well. This characteristic can be easily visualized in Figure 16. In the chart, the vertical axis represents the time, in seconds, spent extracting formal concepts from contexts represented in the horizontal axis. All contexts have 25 attributes and 1,000, 5,000, 10,000, 50,000, 100,000, 250,000, 500,000 and 1,000,000 objects. In the chart, the exponential growth in execution time of the In-Close2 algorithm and the linear growth of the InClose2BDD algorithm are evident. InClose2BDD is the best choice when it comes to manipulating high-dimension contexts because its execution time increases linearly with the addition of objects.

5 GENERAL CONCLUSIONS AND FUTURE WORKS

The application of FCA with the aim to represent social networks (two-mode networks) has been used in other works but not through implication rules (extracted by NextClosure, FindImplications and Inpec algorithms). In this work, we proposed new techniques to explore the knowledge hidden in substructures of social networks. With the increasing amount of social networks created, a large number of users, and the Internet of things becoming a reality (WHITMORE; AGARWAL; XU, 2015) new models will be needed to understand better the relations among network elements. We think that our work can fill at least part of this gap.

Facebook, one of the largest social networks, had a total of 1.44 billion users in 2015 (SUH; HARGITTAI, 2015). Analyzing a huge network like this can be sometimes unfeasible. We propose a change in the FCA algorithms to make these analyses possible in many cases. Not just enabling the analysis of bigger networks, but also doing the task in less time. The BDD structure has already been used in other areas like model checking with great success and this work has showed its importance in FCA algorithms.

In this work, we improve the In-Close2 and NextClosure algorithms to deal with a context of high-dimensionality in the number of objects. We are representing the formal context in OBDD and adapting the algorithms to manipulate this new representation based on OBDD. Our approach and algorithm are preferable in dealing with a database of social networks of high-dimensionality and high density. As present in Chapter 4, we had a significant improvement adopting this approach.

In our future research, we plan to add another dimension, time, to collected data for access prediction analysis based on a profile containing accesses sequences. For highly dimensional formal contexts, execution times of the algorithms become prohibitive, so we plan to use concept lattice reduction techniques (DIAS; VIEIRA, 2015), optimize some algorithms using parallelism and the BDD approach. We also propose to study the impact of BDDs in other relevant FCA algorithms, to extract formal concepts as well as generate implications and investigate new mechanisms to increase the number of attributes.

Both algorithms In-Close2BDD and NextClosureBDD can be improved if one use the chosen BDD library with more direct function calls and even by implementing functions that can more quickly compute attributes shared by objects, the formal concept intents.

REFERENCES

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.: s.n.], 1994. v. 1215, p. 487–499.

AKERS, S. B. Binary decision diagrams. *Computers, IEEE Transactions on*, IEEE, v. 100, n. 6, p. 509–516, 1978.

ANDREWS, S. In-close, a fast algorithm for computing formal concepts. 2009.

ANDREWS, S. In-close2, a high performance formal concept miner. In: *Conceptual Structures for Discovering Knowledge*. [S.l.]: Springer, 2011. p. 50–62.

ARMSTRONG, W. W. Dependency structures of data base relationships. In: GENEVA, SWITZERLAND. *IFIP congress*. [S.l.], 1974. v. 74, p. 580–583.

AUFAURE, M.-A.; GRAND, B. L. Advances in fca-based applications for social networks analysis. *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*, IGI Global, v. 1, n. 1, p. 73–89, 2013.

BABIN, M. A.; KUZNETSOV, S. O. Recognizing pseudo-intents is conp-complete. In: CITESEER. *CLA*. [S.l.], 2010. p. 294–301.

BAZHANOV, K.; OBIEDKOV, S. A. Comparing performance of algorithms for generating the duquenne-guigues basis. In: CITESEER. *CLA*. [S.l.], 2011. p. 43–57.

BEERI, C.; BERNSTEIN, P. A. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems (TODS)*, ACM, v. 4, n. 1, p. 30–59, 1979.

BERTET, K.; GUILLAS, S.; OGIER, J.-M. Extensions of bordat's algorithm for attributes. In: *CLA*. [S.l.: s.n.], 2007.

BIEMANN, C. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the first workshop on graph based methods for natural language processing*. [S.l.], 2006. p. 73–80.

BOLLIG, B.; LÖBBING, M.; WEGENER, I. Simulated annealing to improve variable orderings for obdds. In: CITESEER. *In Int'l Workshop on Logic Synth*. [S.l.], 1995.

BORDAT, J. P. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, v. 96, p. 31–47, 1986.

BREIGER, R. L. The duality of persons and groups. *CONTEMPORARY STUDIES IN SOCIOLOGY*, JAI PRESS INC., v. 15, p. 83–98, 1997.

BRYANT, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, IEEE, v. 100, n. 8, p. 677–691, 1986.

BURMEISTER, P. *Formal concept analysis with ConImp: Introduction to the basic features*. [S.l.]: Fachbereich Mathematik, Technische Universität Darmstadt, 2003.

CARPINETO, C.; ROMANO, G. A lattice conceptual clustering system and its application to browsing retrieval. *Machine learning*, Springer, v. 24, n. 2, p. 95–122, 1996.

CARPINETO, C.; ROMANO, G. *Concept data analysis: Theory and applications*. [S.l.]: John Wiley & Sons, 2004.

CARPINETO, C.; ROMANO, G. *Concept data analysis: Theory and applications*. [S.l.]: John Wiley & Sons, 2005.

CHEIN, M. Algorithme de recherche des sous-matrices premieres d'une matrice. *Bull. Math. Soc. Sci. Math. RS Roumanie*, v. 13, n. 61, p. 21–25, 1969.

CLARKE JR., E. M.; GRUMBERG, O.; PELED, D. A. *Model Checking*. Cambridge, MA, USA: MIT Press, 1999. ISBN 0-262-03270-8.

CORDERO, P. et al. Computing minimal generators from implications: a logic-guided approach. In: CITESEER. *CLA*. [S.l.], 2012. v. 2012, p. 187–198.

CORDERO, P. et al. Knowledge discovery in social networks by using a logic-based treatment of implications. *Knowledge-Based Systems*, Elsevier, v. 87, p. 16–25, 2015.

CUVELIER, E.; AUFAURE, M.-A. A buzz and e-reputation monitoring tool for twitter based on galois lattices. In: *Conceptual Structures for Discovering Knowledge*. [S.l.]: Springer, 2011. p. 91–103.

DAVIS, A.; GARDNER, B. B.; GARDNER, M. R. *Deep South*. [S.l.]: The University of Chicago Press, 1941.

DIAS, S. M.; VIEIRA, N. J. Concept lattices reduction: Definition, analysis and classification. *Expert Systems with Applications*, Elsevier, 2015.

DIAS, S. M.; VIERIA, N. J. Um arcabouço para desenvolvimento de algoritmos da análise formal de conceitos. *Revista de Informática Teórica e Aplicada*, v. 18, n. 1, p. 31–57, 2011.

DISTEL, F.; SERTKAYA, B. On the complexity of enumerating pseudo-intents. *Discrete Applied Mathematics*, Elsevier, v. 159, n. 6, p. 450–466, 2011.

DOWLING, C. E. On the irredundant generation of knowledge spaces. *Journal of Mathematical Psychology*, Elsevier, v. 37, n. 1, p. 49–62, 1993.

DRECHSLER, R.; BECKER, B.; GÖCKEL, N. Genetic algorithm for variable ordering of obdds. *IEE Proceedings-Computers and Digital Techniques*, IET, v. 143, n. 6, p. 364–368, 1996.

EASLEY, D.; KLEINBERG, J. *Networks, crowds, and markets: Reasoning about a highly connected world*. [S.l.]: Cambridge University Press, 2010.

F., P. B. *A General Framework for Verification of Sequential Circuits*. Tese (Doutorado) — PhD thesis, University of Colorado at Boulder, Dept. of Electrical and Computer Engineering, 1993.

FELD, S. L. The focused organization of social ties. *American journal of sociology*, JSTOR, p. 1015–1035, 1981.

FREEMAN, L. C. Cliques, galois lattices, and the structure of human social groups. *Social Networks*, Elsevier, v. 18, n. 3, p. 173–187, 1996.

FREEMAN, L. C. Visualizing social networks. *Journal of social structure*, v. 1, n. 1, p. 4, 2000.

FREEMAN, L. C. Finding social groups: A meta-analysis of the southern women data. *Dynamic social network modeling and analysis*, The National Academies Press Washington, p. 39–97, 2003.

FREEMAN, L. C. Graphic techniques for exploring social network data. *Models and methods in social network analysis*, p. 248–269, 2005.

FREEMAN, L. C.; WHITE, D. R. Using galois lattices to represent network data. *Sociological methodology*, v. 23, n. 127, p. U146, 1993.

FRUCHTERMAN, T. M.; REINGOLD, E. M. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, v. 21, n. 11, p. 1129–1164, 1991.

FU, H.; NGUIFO, E. M. *A lattice algorithm for data mining*. [S.l.], 2004.

GANTER, B. *Two basic algorithms in concept analysis*. [S.l.]: Springer, 2010.

GANTER, B.; STUMME, G.; WILLE, R. *Formal concept analysis: foundations and applications*. [S.l.]: Springer Science & Business Media, 2005.

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

GANTER, B.; WILLE, R. *Formal concept analysis: mathematical foundations*. [S.l.]: Springer Science & Business Media, 2012.

GETOOR, L.; DIEHL, C. P. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, ACM, v. 7, n. 2, p. 3–12, 2005.

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

GODIN, R.; MISSAOUI, R.; ALAOUI, H. Incremental concept formation algorithms based on galois (concept) lattices. *Computational intelligence*, Wiley Online Library, v. 11, n. 2, p. 246–267, 1995.

GODIN, R.; SAUNDERS, E.; GECSEI, J. Lattice model of browsable data spaces. *Information Sciences*, Elsevier, v. 40, n. 2, p. 89–116, 1986.

GRANOVETTER, M. The strength of weak ties: A network theory revisited. *Sociological theory*, USA, v. 1, n. 1, p. 201–233, 1983.

GRÄTZER, G. *General lattice theory*. [S.l.]: Springer Science & Business Media, 2003.

GUÉNOCHE, A. Construction du treillis de galois d'une relation binaire. *Mathématiques et sciences humaines*, v. 109, p. 41–53, 1990.

GUIGUES, J.-L.; DUQUENNE, V. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences humaines*, Ecole Pratique des hautes études, Centre de mathématique sociale et de statistique, v. 95, p. 5–18, 1986.

ISHIURA, N.; SAWADA, H.; YAJIMA, S. Minimazation of binary decision diagrams based on exchanges of variables. In: *ICCAD*. [S.l.: s.n.], 1991. v. 91, p. 472–475.

J., G. R. et al. Canonical computational models based on formal concept analysis for social network analysis and representation. *ICWS*, v. 99, n. 9, p. 99–99, 2015.

JACOMY, M. et al. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, Public Library of Science, v. 9, n. 6, p. e98679, 2014.

JÁSCHKE, R. et al. Trias-an algorithm for mining iceberg tri-lattices. In: *ICDM*. [S.l.: s.n.], 2006. v. 6, p. 907–911.

JEONG, S. W.; KIM, T. S. et al. An efficient method for optimal bdd ordering computation. In: *ICVC: International Conference on VLSI and CAD*. [S.l.: s.n.], 1993. v. 3, p. 252–256.

KIRALY, B. D. *An experimental application of formal concept analysis to research communities*. Tese (Doutorado) — Case Western Reserve University, 2008.

KRAJCA, P.; OUTRATA, J.; VYCHODIL, V. Advances in algorithms based on cbo. In: *CLA*. [S.l.: s.n.], 2010. p. 325–337.

KUZNETSOV, S. O. A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automatic documentation and Mathematical linguistics*, v. 27, n. 5, p. 11–21, 1993.

KUZNETSOV, S. O. On computing the size of a lattice and related decision problems. *Order*, Springer, v. 18, n. 4, p. 313–321, 2001.

KUZNETSOV, S. O.; OBIEDKOV, S. Some decision and counting problems of the duquenne–guigues basis of implications. *Discrete Applied Mathematics*, Elsevier, v. 156, n. 11, p. 1994–2003, 2008.

KUZNETSOV, S. O.; OBIEDKOV, S. A. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, v. 14, n. 2-3, p. 189–216, 2002.

LINDIG, C. Algorithmen zur begriffsanalyse und ihre anwendung bei softwarebibliotheken. *Dr.-Ing.) Dissertation, Techn. Univ. Braunschweig*, 1999.

LINDIG, C. Fast concept analysis. *Working with Conceptual Structures-Contributions to ICCS*, Citeseer, v. 2000, p. 152–161, 2000.

MAIER, D. *The theory of relational databases*. [S.l.]: Computer science press Rockville, 1983.

MORAES, N. R. de; ZÁRATE, L. E.; FREITAS, H. C. A distributed algorithm for formal concepts processing based on search subspaces. In: *ICEIS (1)*. [S.l.: s.n.], 2010. p. 105–111.

NGUIFO, E. M.; NJIWOUA, P. Using lattice-based framework as a tool for feature extraction. In: *Feature Extraction, Construction and Selection*. [S.l.]: Springer, 1998. p. 205–218.

NORRIS, E. M. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, v. 23, n. 2, p. 243–250, 1978.

NOURINE, L.; RAYNAUD, O. A fast algorithm for building lattices. *Information processing letters*, Elsevier, v. 71, n. 5, p. 199–204, 1999.

OBIEDKOV, S. *Algorithms and methods of lattice theory and their application in machine learning*. Tese (Doutorado) — PhD thesis, Russian State University for the Humanities, 2003.

PANDA, S.; SOMENZI, F. Who are the variables in your neighbourhood. In: IEEE. *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*. [S.l.], 1995. p. 74–77.

PANDA, S.; SOMENZI, F.; PLESSIER, B. F. Symmetry detection and dynamic variable ordering of decision diagrams. In: IEEE COMPUTER SOCIETY PRESS. *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*. [S.l.], 1994. p. 628–631.

POELMANS, J. et al. A concept discovery approach for fighting human trafficking and forced prostitution. In: *Conceptual Structures for Discovering Knowledge*. [S.l.]: Springer, 2011. p. 201–214.

POELMANS, J. et al. Formal concept analysis in knowledge processing: A survey on applications. *Expert systems with applications*, Elsevier, v. 40, n. 16, p. 6538–6560, 2013.

POELMANS, J. et al. Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert systems with applications*, Elsevier, v. 40, n. 16, p. 6601–6623, 2013.

PRISS, U. Some open problems in formal concept analysis. In: *International Conference on Formal Concept Analysis, ICFCA*. [S.l.: s.n.], 2006.

QIAO, S.-Y. et al. A fast algorithm for building concept lattice. In: IEEE. *Machine Learning and Cybernetics, 2003 International Conference on*. [S.l.], 2003. v. 1, p. 163–167.

RIMSA, A. et al. Scgaz-a synthetic formal context generator with density control for test and evaluation of fca algorithms. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. [S.l.], 2013. p. 3464–3470.

RIMSA, A.; ZÁRATE, L. E.; SONG, M. A. Evaluation of different bdd libraries to extract concepts in fca–perspectives and limitations. In: *Computational Science–ICCS 2009*. [S.l.]: Springer, 2009. p. 367–376.

ROME, J. E.; HARALICK, R. M. Towards a formal concept analysis approach to exploring communities on the world wide web. In: *Formal Concept Analysis*. [S.l.]: Springer, 2005. p. 33–48.

RUDELL, R. Dynamic variable ordering for ordered binary decision diagrams. In: IEEE COMPUTER SOCIETY PRESS. *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*. [S.l.], 1993. p. 42–47.

SALLEB, A.; MAAZOUZI, Z.; VRAIN, C. Mining maximal frequent itemsets by a boolean based approach. In: *European Conf. on Artificial Intelligence, Lyon France (July 2002)*. [S.l.: s.n.], 2002. p. 285–289.

SNÁŠEL, V.; HORÁK, Z.; ABRAHAM, A. Understanding social networks using formal concept analysis. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*. [S.l.], 2008. p. 390–393.

SNASEL, V. et al. Analyzing social networks using fca: complexity aspects. In: IET. *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*. [S.l.], 2009. v. 3, p. 38–41.

SOMENZI, F. Cudd: Cu decision diagram package-release 2.4. 0. *University of Colorado at Boulder*, 2009.

STUMME, G. et al. Fast computation of concept lattices using data mining techniques. In: *KRDB*. [S.l.: s.n.], 2000. p. 129–139.

SUH, J. J.; HARGITTAI, E. Privacy management on facebook: Do device type and location of posting matter? *Social Media+ Society*, SAGE Publications, v. 1, n. 2, p. 2056305115612783, 2015.

TAMASSIA, R. *Handbook of graph drawing and visualization*. [S.l.]: CRC press, 2013.

TAOUIL, R.; BASTIDE, Y. Computing proper implications. In: *9th International Conference on Conceptual Structures: Broadening the Base-ICCS'2001*. [S.l.: s.n.], 2001. p. 13–p.

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

VIMIEIRO, R. *Um estudo de algoritmos para a extraçao de regras baseados em análise formal de conceitos*. Tese (Doutorado) — Master's thesis, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil, 2007.

WASSERMAN, S. *Social network analysis: Methods and applications*. [S.l.]: Cambridge university press, 1994.

WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. *Information Systems Frontiers*, Springer, v. 17, n. 2, p. 261–274, 2015.

WIKIPEDIA. *List of social networking websites — Wikipedia, The Free Encyclopedia*. 2016. [Online; accessed 23-February-2016]. Disponível em: <https://en.wikipedia.org/w/index-.php?title=List_of_social_networking_websites_oldid=701148283>.

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

WILLE, R. *Restructuring lattice theory: an approach based on hierarchies of concepts*. [S.l.]: Springer, 2009.

YEVTUSHENKO, S. Bdd-based algorithms for the construction of the set of all concepts. *Foundations and Applications of Conceptual Structures. Contributions to ICCS*, v. 2002, p. 61–73, 2002.

YEVTUSHENKO, S. *Computing and visualizing concept lattices*. Tese (Doutorado) — TU Darmstadt, 2004.

ZAKI, M. J.; HSIAO, C.-J. et al. *CHARM: An efficient algorithm for closed association rule mining*. [S.l.], 1999.