Received July 8, 2020, accepted July 16, 2020, date of publication July 20, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010482

Exploring Different Paradigms to Extract Proper Implications From High Dimensional Formal Contexts

JULIO C. V. NEVES[1], PEDRO HENRIQUE BATISTA RUAS DA SILVEIRA[1], ROKIA MISSAOUI[2], SÉRGIO M. DIAS[1,3], LUIS E. ZÁRATE[1], AND MARK A. J. SONG[1]

[1]Department of Computer Science, Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte 30535-901, Brazil
[2]Department of Computer Science and Engineering, Université du Québec en Outaouais, Gatineau, QC J8X 3X7, Canada
[3]Federal Service of Data Processing (SERPRO), Belo Horizonte 31035-536, Brazil

Corresponding author: Julio C. V. Neves (juliocesar.neves@gmail.com)

The authors are grateful for the financial support of CAPES (Coordination for higher Education Development), FAPEMIG (Research Support Foundation of Minas Gerais state) and CNPq (Council for Scientific and Technological Development).

ABSTRACT Formal Concept Analysis (FCA) is an applied mathematical technique for data analysis, in which the relations between objects and attributes are identified. It introduces the notion of concepts and their hierarchical structure, from which we can obtain a set of implications between attributes that characterize a knowledge domain. The volume of information to be processed makes the use of FCA difficult in domains with a high number of dimensions, creating a demand for new solutions and algorithms for FCA applications. This article explores different approaches to extract proper implications from high dimensional contexts based on constraints to obtain the set of implications rules. We propose algorithms that use a data structure called Binary Decision Diagram (BDD) to represent the formal context, which reduces its size and, due to this, operates more efficiently. We also propose a heuristic to obtain proper implications by reducing the unnecessary generation of premises. In addition, we implemented a parallel computing model for generating and obtaining different implications. To analyze the proposed algorithms, we used different synthetic contexts with a varying number of objects, attributes, and density. The results obtained presented speed gains of up to 22 times when compared to the solutions proposed in the literature such as *Impec* and *PropIm*.

INDEX TERMS Formal concept analysis, proper implications, binary decision diagram.

I. INTRODUCTION

Extracting knowledge from large volumes of data collected and stored currently is unfeasible without the support of automation and techniques devised for this purpose. The requirements for knowledge discovery tools have increased greatly with this recent change in amount of stored data that needs to be interpreted in a timely manner.

Formal Analysis of Concepts (FCA) is a branch of applied mathematics, which is based on a formalization of concepts and their hierarchy [1], [2]. By applying FCA to a set of objects and its known properties, it is possible to organize and understand this set of objects through the formal concepts they form [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Huiling Chen.

Since it was first formalized, FCA has been applied in several applications, such as information retrieval [4], linguistics [5], security analysis [6], software engineering [7], social network analysis [8], web services [9], text mining [10], Topic Detection [11], Search engine [12], among others [13].

In general, FCA has been used in data analysis process and knowledge representation where associations and dependencies among instances are identified [14], [15]. Considering a number of objects (G) (instances), attributes (M) and an incidence binary relation (I) between objects and attributes, a formal context (G, M, I), can be created to represent that set of instances. From a formal context, we can then extract a set of implications $\mathcal{I}$, which are related to the attributes [16], as possibly novel knowledge. An implication $A \rightarrow B$, where $(A, B \subseteq M)$, means that *every object* containing the attributes belonging to the premise A also has the attributes from the conclusion B. Applying different operators to $\mathcal{I}$, we can generate different implication sets, with certain constraints according to specific requirements.

A given set of implications that can be obtained from $\mathcal{I}$ is the *proper implications* set, where for each implication the premise is minimal and the conclusion is a singleton (unit set) [17]. Implications from this set are useful because they provide a minimum data representation - especially for applications that require a minimum set of attributes to reach a specific conclusion. In [18], the authors used the proper implications set to identify relationships between professional skills of *LinkedIn* users. From the set of proper implications, the authors identified the minimum set of skills (premise) required for a certain professional competence (conclusion). Other examples of use for proper implication sets are minimal path problems, such as finding the best flight routes, and inferring functional dependencies in database design [19].

The most commonly used algorithm for obtaining and extracting proper implications from a formal context is the *Impec*, proposed by [16]. The *Impec* algorithm generates, from a set of M attributes, the set of all proper implications $A \rightarrow b$, where $A \subseteq M$ and $b \in M$, in which the left side (*premise*) is minimal and the right side (*conclusion*) has only one attribute. However, in some cases, only implications generated from some of the attributes are relevant, but the *Impec* would require the user to generate the full set of proper implications and then manually select those that are relevant to them.

In general, proper implications can be used to find the minimal premises that lead to a given conclusion. In [16] the authors presented three motivators where proper implications should be used: the first and second motivators state that the proper implications set should be small when compared to the full set of implications, and composed by useful implications. In other words, the set should be a small number of non-trivial implications. Finally, the authors state that it is necessary to have a way for the user to measure of how interesting the implications are. As an example in data mining area, association rules have relevancy measures. In this context, this could be a weight function compatible with the closure operator.

Another example of algorithm that generates a set of proper implications from a set of attributes M is the *PropIm* algorithm [18], which differs from the *Impec* mainly by only generating implications with a support value greater than 0. Although implications with support equal to 0 are valid implications, most of the time, analysts are more interested in implications with support greater than 0, as those indicate characteristics explicitly represented by the data.

The computational performance of the *Impec* and *PropIm* algorithms makes their use prohibitive for processing high-dimensional contexts, as both work in the complexity order of $O(M(\sum_{i=1}^{n} C_{n,i}))$ where n is the size of the premise and $|M|$ is the number of attributes. In this article, we propose a solution to efficiently extract proper implications on high-dimensional formal contexts. Considering a high dimensional context, with a large number of objects (up to 100,000), in which the main goal is to find proper implications with support greater than 0, and given a subset of desirable conclusions to describe a specific domain, we propose two algorithms based on the original proposal of [18]: I) the first algorithm, *ImplicP*, contains a heuristic to avoid the generation of unnecessary premise sets, evaluated based on the notion of monotonic constraints; II) the second algorithm, named *PImplicPBDD*, includes a Binary Decision Diagram (BDD) structure to represent and manipulate the formal context efficiently, and a parallel computing model to process several conclusions simultaneously. The modifications and approaches presented in this article were more efficient, when compared to the solutions present in the literature, for the following reasons:

a) In applications where the user desires a set of proper implications with only a subset of the attributes as viable conclusions, it is not necessary to generate all the implications, for all the attributes, in order to extract the desired conclusions;
b) it also becomes unnecessary to generate and evaluate all sets of premises;
c) the use of BDDs to represent and manipulate data from the formal context is useful for efficiently handling high-dimensional datasets.
d) the parallel algorithm processes multiple conclusions simultaneously, increasing the number of objects being manipulated. Furthermore, we show in this approach that we can work on multiple queries (possibly from different users) simultaneously, greatly decreasing the total run time.

The experiments showed that our proposal was the most efficient in scenarios where the objective is to find proper implications with constraints. Using our proposed modified algorithms, we were able to manipulate a high dimensional dataset (e.g., 100,000 objects and 23 attributes), which no one of the algorithms found in the literature was able to process. In our experiments, we used both synthetic and real-world datasets (LinkedIn dataset used in [18]), varying the number of attributes and context density, and compared the time taken by our approaches and the *ImplicP* and *PImplicPBDD* algorithms.

We also varied the number of objects (instances) used in the experiments from 1,000, 10,000 and 100,000, partially meeting the latent challenge stipulated by [20]. Regarding the density of the formal contexts, which is the amount of incidence between objects and attributes, we used datasets with a varying density of 30%, 50% and 70%, to analyze the performance of algorithms in sparse and dense contexts. The results showed that *PImplicPBDD* has a better performance – up to 22 times faster – than *PropIm*, regardless of the number of attributes and objects as well as the density values.

In the following sections, we will present each algorithm and its results. The paper is organized as follows: Section 2 presents basic concepts of the FCA and BDD. It also describes some work related to BDD and proper implications. Section 3 discusses the experimental methodology adopted in this article and presents the evaluation of all proposed algorithms (ImplicP, PImplicPBDD). Finally, in Section 4 we present our conclusion and discuss future work.

II. RELATED WORK

The searching for a concept lattice with an appropriate size and structure, which allows exploring the relevant aspects of a formal context, is one of the most important problems when using FCA. As a consequence, several techniques have been proposed, which have specific characteristics for concept lattice reduction and/or extraction of implication. In [21], a review about the concept lattice reduction was presented. In [22], the authors proposed the reduction of concept lattice using fuzzy k-means clustering. In [23], scalable algorithms to deal with large datasets using spark was proposed. In [24], a strategy based on fuzzy clustering, applied on the original data set, to reduce the formal context and to obtain association rules was discussed. There is a big challenge when the formal context has high dimensionality and it is necessary to extract implications.

Currently, the existing FCA algorithms to extract proper implications do not work efficiently for high-dimensional contexts. As mentioned in [16], the *Impec* - most popular algorithm to extract proper implications sets - was proposed in order to extract all proper implications from a formal context, but it does not perform well for larger datasets. A similar algorithm was proposed by [18], named *PropIm*, which is based on the *Impec* algorithm, but only identifies proper implications with support greater than zero, meaning all the implications found have at least one occurrence in the dataset. The algorithm was used to identify the relationships between professional skills of LinkedIn user profiles through proper implications. It showed the minimum set of skills that would be required to achieve certain job positions. Unlike these two algorithms, our proposed solution has as main objective to manipulate and process proper implication sets on high dimensional contexts.

In [25] the authors presented the concept of relationally approximable concepts that gives a generalized framework of FCA. They also introduced the idea of F-approximable mapping that provides as the morphism between relationally consistent F-augmented contexts.

Another important research was presented in [26]. It was introduced the concept of soft context that is produced using a soft set and not a binary relation of the formal context. Finally, it was demonstrated how they apply their method to get a set of formal concepts extracted from a formal context. Also, explained that their method is more effective than the traditional.

In [27], the authors proposed an algorithm to extract formal concepts using BDD. They used the data structure to represent the list of found concepts. However, the authors used contexts with 900 objects and 50 attributes, which proved to only be efficient for dense contexts.[1]

In [28] the authors used BDD for extracting formal concepts, but their focus was on using different BDD libraries to investigate which would be the best for an FCA application. The authors used brute force methods to obtain the set of intents, and the BDDs that represent the extents. In all experiments performed in the study, the algorithms with BDD were more efficient when compared to the original algorithms.

In [29], the BDD structure was applied to two FCA algorithms (*NextClosure* and *Inclose2*) that extract formal concepts from high dimensional datasets. Using BDD to represent the formal contexts reduced the required memory space to represent them, and simplified information manipulation operations. With this approach, the authors were able to manipulate larger datasets that previously were not feasible for the tested FCA algorithms. [30] explains that it is possible to work with larger volumes of data using BDD structures outside the context of FCA, which contextualizes these findings in [29].

Similarly to the studies in [27]–[29] and [30], we propose using BDD to process high dimensional contexts. The BDD structure is used to represent and manipulate the formal context in order to efficiently extract a set of proper implications from a given dataset.

III. BACKGROUND

This section presents basic FCA and BDD, related to our proposal. It also presents the algorithms used to compare with our approaches.

A. FORMAL CONCEPT ANALYSIS

As discussed previously, FCA is a field of mathematics that allows the identification of formal concepts and implications, which are extracted from a formal context [2]. In this section, we first recall the notions of formal contexts, formal concepts, and implications.

Definition 1 (Formal Context): It is formed by a triple $\mathcal{K} := (G, M, I)$, where G is a set of objects (rows), M is a set of attributes (columns) and I is defined as the binary relationship (incidence relation) between objects and their attributes where $I \subseteq G \times M$ [15].

An example of a formal context is given in Table 1. The rows depict instances (objects) in the dataset: *Paul*, *John*, *Ann*, *Peter* and *Susan*. The columns are the attributes, properties of the instances (in this scenario, use of social networks for each person). The incidences ("X") mark the relationships between objects and attributes.

Definition 2 (Formal Concepts): They are defined by a pair (A, B) where $A \subseteq G$ is called extent and $B \subseteq M$ is called intent. This pair must verify $A = B'$ and $B = A'$ [15]. The

[1]The BDD representation was shown to be more efficient when the binary table (formal context) is dense.

TABLE 1. An example of formal context.

	Facebook (a)	Instagram (b)	Twitter (c)	Yammer (d)
Paul	X	.	X	.
John	.	X	X	X
Ann	.	X	.	.
Peter	X	.	X	.
Susan	.	X	X	X

relation is defined by the derivation operator $(')$:

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

A formal concept identifies the set of attributes (intent) which delimit and characterize a set of objects (extent). Considering the formal context presented in Table 1, we get the following formal concepts:
({Paul, John, Ann, Peter, Susan}, { })
({John, Ann, Susan}, {Instagram})
({John, Susan}, {Instagram, Twitter, Yammer})
({Paul, John, Peter, Susan}, {Twitter})
({Paul, Peter}, {Facebook, Twitterl})
({ }, {Facebook, Instagram, Twitter, Yammer})

A partial order exists between two concepts $c_1 = (A_1, B_1) \leq$ and $c_2 = (A_2, B_2)$ if $A_1 \subseteq A_2 \iff B_1 \supseteq B_2$.

The set of all concepts together with the partial order of a given formal context $\mathcal{K}$ form a concept lattice denoted by $\mathfrak{B}(\mathcal{K})$.

From a formal context $\mathcal{K} := (G, M, I)$ or its concept lattice $\mathcal{B}(K)$ can be extracted *implications* [2].

Definition 3: Given a formal context $\mathcal{K} := (G, M, I)$, an implication is an expression $P \to Q$, where P and Q are sets of attributes, which express that $P' \subseteq Q'$.

An implication $P \to Q$, extracted from a formal context, or respective concept lattice, is such that $P' \subseteq Q'$. In other words, every object which has the attributes of P also has the attributes of Q. Note that, in $P \to Q$, P is the *left hand side* (lhs) and Q is the *right hand side* (rhs), where:

1) If X is a set of attributes, then X *implies* an implication $P \to Q$ if and only if (iff) $P \not\subseteq X$ or $Q \subseteq X$.
2) An implication $P \to Q$ *holds* in a set $\{X_1, \ldots, X_n\} \subseteq M$ iff each X_i implies $P \to Q$; and $P \to Q$ *is an implication of the context* (G, M, I) iff it holds in its set of object intents (an object intent is the set of its attributes).
3) An implication $P \to Q$ *follows from* a set of implications $\mathcal{I}$, iff for every set of attributes X if X implies $\mathcal{I}$, then it implies $P \to Q$.

As an example of an implication, we can consider the universe of users' accesses as described in Table 1 where, for example, every user that accesses Yammer also accesses Twitter. This type of relationship can be described as an implication. An example of implication is Yammer → Twitter, which is a way to describe that all Yammer users also use Twitter. Table 2 is an example of implications based on the formal context presented in Table 1, considering Definition 3 and the derivation operators described in Definition 2.

TABLE 2. Implications extracted from the formal context in the Table 1.

$A \to B$
Facebook → Twitter
Twitter, Yammer → Instagram
Yammer → Instagram, Twitter
Yammer → Instagram
Yammer → Twitter
Instagram, Twitter → Yammer

Definition 4 (Proper Implications): For some problems it is convenient to have each implication representing a minimum condition. For this, it is required that the set of implications $\mathcal{I}$ of a formal context (G, M, I) has the following characteristics:

- The right side of each implication is unitary: if $P \to m \in \mathcal{I}$, then $m \in M$;
- Trivial implications are not allowed: if $P \to m \in \mathcal{I}$, then $m \notin P$;
- Minimality is ensured, i.e. left side is minimal: if $P \to m \in \mathcal{I}$, then there is no $Q \to m \in \mathcal{I}$ such that $Q \subset P$.

The complete set of implications in (G, M, I) with these properties is named the set of *proper implications* [16] or *unary implication system* (UIS) [31]. The Table 3 shows the complete set of proper implications extracted from the formal context in the Table 1.

TABLE 3. Complete set of proper implications extracted from the formal context in the Table 1.

$A \to B$
Facebook → Twitter
Instagram, Twitter → Yammer
Yammer → Instagram
Yammer → Twitter
Facebook, Instagram → Yammer

The complete set of proper implication is still extensive. In order to make it more restrictive, it can be selected only proper implication with support above a previously defined threshold.

Definition 5: Given a proper implication $P \to m$, we can define a **support** as follow: $sup(P \to m) = \frac{|(P \cup \{m\})'|}{|G|}$.

The proper implications shown in Table 3 have support equal to 0.4; except the {Facebook, Instagram }→Yammer implication that has zero support. Note that implication with zero support are valid; i.e., they can be inferred from $\mathcal{I}$ through Armstrong inference axioms [32]. However, depending on the application these implications can be removed.

B. MONOTONE CONSTRAINTS

Since the introduction of association rules [33], researchers have been studying various problems related to pattern mining in large databases. These studies involve developing more efficient algorithms (both sequential and parallel) for finding associations, their quantitative variants, sequential patterns, and the use of clustering and sampling techniques.

Frequent Pattern Mining (FPM) is known to play an essential role in many important tasks in the data mining process. However, FPM activity often generates several frequent rules, so the user is required to post-process the output and select useful rules.

The authors in [34] highlights the importance of the constraint-based mining paradigm. This paradigm allows users add a focus to the mining process through constraint classes, allowing for a more targeted exploration and greater control over mining, and the generation of rules according to the user's need.

One type of restriction that can be added to the data mining process is the application of monotone constraints. Formally, a constraint C is *monotone* [35] iff and only if for all itemsets S and S':

$$S \supseteq S' \text{ and } S \text{ violates } C, \text{ then } S' \text{ violates } S.$$

Thus, if a rule S violates a monotone constraint C, it can be said that any subset of S will also violate it. Equivalently, if a rule S satisfies a constraint C, all of its supersets will satisfy it [36], [37].

Suppose the following implication is proper: $\{abc\} \rightarrow \{C_1\}$. That is, the combined attributes a, b, and c yield a C_1 conclusion with minimal constraint. If $\{abc\} \rightarrow \{C_1\}$ is minimal, even if $\{abcd\} \rightarrow \{C_1\}$, this implication ($S$) violates the minimal restriction. Therefore, any other set $\{abcx_1..x_n\} \rightarrow \{C_1\}$ (S') would violate the same restriction.

As an example, consider Table 2. Note that $\{Yammer\} \rightarrow \{Instagram\}$ is a minimal implication of its own. Although $\{Yammer, Twitter\} \rightarrow \{Instagram\}$ (S) is an implication, this is not minimal, as it violates the restriction. Thus, any other set of $\{Yammer, Twitter, \ldots\} \rightarrow \{Instagram\}$ (S') also violates the minimal restriction.

C. BINARY DECISION DIAGRAM

The Binary Decision Diagram (BDD) is a form of representing canonical Boolean formulas. It is substantially more compact than the traditional structure forms (normal conjunctive and disjunctive form) used in FCA, and it can be efficiently manipulated [38], [39].

Figure 1 represents the Binary Decision Tree table presented in Table 4 in its tree form.

Figure 2 provides an example where a BDD is used to represent the binary decision tree described on the Figure 1. In this diagram, lines represent that the object has the attribute and the dotted-line represents the lack of that attribute.

Note that it was possible to represent the same information using a structure considerably more compact than the original. In our approach, we represented the formal context as a BDD. Consider equation 1 representing a Boolean formula correspondent to Table 1. For a better view, attributes names have been replaced by letters as follow: Facebook (a), Instagram (b), Twitter (c), Yammer (d).

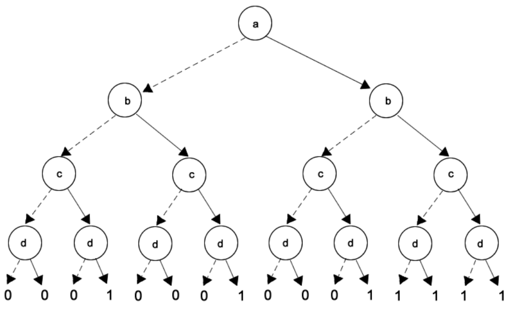

FIGURE 1. An example of binary decision tree table.

TABLE 4. An example of binary decision tree table.

A	B	C	D	F (A, B, C, D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

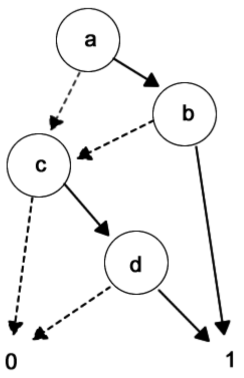

FIGURE 2. Example of a BDD from Figure 1.

$$\mathfrak{f}(a, b, c, d) = a\overline{b}c\overline{d} + \overline{a}bcd + \overline{a}b\overline{c}\overline{d} + a\overline{b}c\overline{d} + \overline{a}bcd \quad (1)$$

The attribute a means that, in this part of the function, the attribute is true (this object has this attribute), whereas $\overline{a}$ means the opposite. The part $a\overline{b}c\overline{d}$ of the equation was created to validate the *Paul* and *Peter* objects, $\overline{a}bcd$ was created to validate the *Susan* and *John* objects and the last part $\overline{a}b\overline{c}\overline{d}$ validates the Ann object.

The generated BDD corresponding to the formal context presented in Table 1 (and described by Equation 1) can be seen in Figure 3. Object names have been replaced by numbers and attributes replaced by letters: *Paul* (1), *John* (2), *Ann* (3), *Peter* (4), *Susan* (5) and *Facebook* (a), *Instagram*

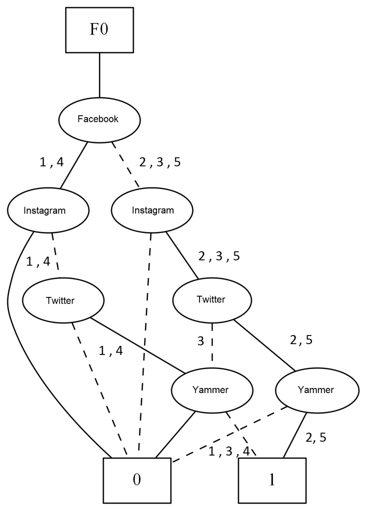

FIGURE 3. **The BDD representing the formal context of the Table 1.**

(*b*), *Twitter*(*c*), *Yammer* (*d*). Take as example, the object 1 (*Paul*) which is represented by the path *Facebook*, *Instagram*, *Twitter*, *Yammer*.

We used the BDD library CUDD (Colorado University Decision Diagram) to construct our BDD structures. The library provides function packages to work with BDDs, and has been recently updated. The CUDD also has functions for Algebraic Decision Diagrams (ADDs) and Zero-suppressed Binary Decision Diagrams (ZDDs) which could also be used o represent formal contexts.

D. BASE ALGORITHM DESCRIPTIONS

In this section, we describe the *PropIm* algorithm, used as the basis for our proposal (*PImplicPBDD*) and the synthetic formal context generator (*generateBDDFormalContext*).

1) THE PropIm ALGORITHM

The algorithm proposed by [18] finds the set of implications with support greater than 0 from a formal context. The set of implications is done in two steps: first, all sets of attribute premises are generated for a conclusion attribute, and then this set is pruned to find the minimum set of attributes for the premise.

Algorithm 1 receives as input a formal context *(G, M, I)*, and generates as an output a set of proper implications. The loop (lines 2-17) analyzes each attribute in *M*. Initially, each attribute *m* can be a conclusion for a set of premises. For each *m*, the algorithm calculates the premises *P*1.

In line 3, *P* records all the attributes that contains the same objects of *m*. The counter, named *size*, determines the size of each premise, and is initialized as 1 because the smallest possible size is 1 (an implication of type $x \to z$) (Line 4).

In line 5 *Pa* is initialized as empty once it stores a set of auxiliary premises that can generate an implication using *m* as a conclusion. From lines 6-16, the set of minimum premises is found, limited by $|P|$. In Line 7, the set *C* contains all sets of *size* from elements in *P*. In Line 8, the set of

Algorithm 1 PropIm

```
Input : Formal context $(G,M,I)$
Output: Set of implication with support greater than 0 $(imp)$
1  $imp = \emptyset$
2  forall the $m \in M$ do
3      $P = m''$
4      $size = 1$
5      $Pa = \emptyset$
6      while $size < |P|$ do
7          $C = \binom{P}{size}$
8          $Pc = getCandidate(C,Pa)$
9          forall the $P1 \subset Pc$ do
10             if $P1' \neq \emptyset$ and $P1' \subset m'$ then
11                 $Pa = Pa \cup \{P1\}$
12                 $imp = imp \cup \{P1 \to m\}$
13             end
14         end
15         size++
16     end
17 end
18 return $imp$
```

candidate premises is formed through the *Algorithm* 2, called `GetCandidate`. It obtains all subsets that do not contain an attribute that belongs to the *Pa* premise. It receives, as a parameter, the sets *C* and *Pa* and returns a set *D* of premises.

Algorithm 2 GetCandidate

```
Input : Premise Candidate (C), Set of Premises (P)
Output: Premise (D)
1  Function getCandidateProp($C$,$Pa$):
2      $D = \emptyset$
3      foreach $a \in A|A \subset Pa$ do
4          foreach $B \subset C$ do
5              if $a \notin B$ then
6                  $D = Pc/B$
7          end
8      end
9      return D
```

Each candidate premise $P1 \subset PC$ is checked to ensure that the premise *P*1 and the conclusion *m* result in a valid proper implication. In case $P1' \neq \emptyset$ and $P1' \subset m'$, the premise *P*1 is added to the set of auxiliary premises *Pa*, and also the implication $\{P1 \to m\}$ is added to the list of implications $imp = imp \cup \{P1 \to m\}$.

Table 5 presents a formal context used for a better understanding of the generation of premise sets in PropIm.

In the first iteration, the algorithm generates 4 sets of size 1 for all attributes, i.e., *unitary premises* {a}, {b}, {c} and {d} (Table 6). After the generation, the algorithm checks whether there is already an implication for the current conclusion that

TABLE 5. Formal context example.

	a	**b**	**c**	**d**	C_1
obj1	X	.	.	.	X
obj2	.	.	.	X	.
obj3	.	.	X	.	.
obj4	.	X	.	.	.
obj5	.	X	X	X	X

Attributes: { a, b, c, d };
Conclusion: $\{C_1\}$
Proper Implications: $\{a\} \to \{C_1\}$,
$\{bcd\} \to \{C_1\}$

TABLE 6. Generated premises.

Set	Size
{ {a}, {b}, {c} , {d} }	1
Total: 4 sets	

contains any attributes of the generated premises. If there is an implication with any of the attributes, the premises that would be generated referencing this attribute will be discarded, so we generate only premises that could become valid implications. Table 7 describes the first step of running the algorithm. As a result, the first implication **{*a*}** $\to$ **{**C_1**}** is found.

TABLE 7. Candidate premises.

Premises	Size
{ {a}, {b}, {c} , {d} }	1
Total: 4 premises	

In the second iteration of the algorithm, 10 sets of premises will be generated, 4 of size 1 and 6 of size 2, as described in Table 8. After the generation, the algorithm checks whether there is already an implication for the current conclusions, containing any attribute of the generated premises of size 1 and 2. If there is an implication with any of the used attributes, the premises that refer to this attribute are discarded, generating only premises that could become valid implications. Table 9 describes the result of the algorithm's execution. Note that premises {a}, {ab}, {ac}, {ad} are removed, as there is already an implication that uses the attribute ***a*** (**{*a*}** $\to$ **{**C_1**}**).

TABLE 8. Generated premises.

Set	Size
{ {a}, {b}, {c} , {d} }	1
{ {ab}, {ac}, {ad}, {bc}, {bd}, {cd} }	2
Total: 10 sets	

TABLE 9. Candidate premises.

Premises	Size
{{b}, {c}, {d}}	1
{{bc}, {bd}, {cd}}	2
Total: 6 premises	

In the third iteration 14 sets of premises will be generated, being 4 of size 1, 6 of size 2 and 4 of size 3, as shown in Table 10. As mentioned, the algorithm then checks whether

TABLE 10. Generated premises.

Set	Size
{{a}, {b}, {c}, {d}}	1
{{ab}, {ac}, {ad}, {bc}, {bd}, {cd}}	2
{{abc}, {abd}, {acd}, {bcd}}	3
Total: 14 sets	

there is already an implication to the current conclusion that contains any attribute of the generated premises, now of sizes 1, 2 and 3, and unnecessary premises are discarded.

Table 11 describes the result of the execution of the algorithm that discards premises {a}, {ab}, {ac}, {ad}, {abc}, {abd}, {acd} since there is already an implication using the attribute ***a*** (**{*a*}**$\to$**{**C_1**}**). A new implication **{*bcd*}** $\to$ **{**C_1**}** was found. At this point, as all premise sets have been generated, the algorithm moves on to the next conclusion.

TABLE 11. Candidate premises.

Premises	Size
{{b}, {c}, {d}}	1
{{bc}, {bd}, {cd}}	2
{bcd}	3
Total: 7 premises	

2) THE *generateBDDFormalContext* FUNCTION

The creation of the BDD structure for a formal context make use of the *generateBDDFormalContext* function (*Algorithm* 3), which receives as input a file in the Burmeister format [40], built through BDD operations with the current context

Algorithm 3 generateBDDFormalContext

```
Input  : Formal context $(G, M, I)$
Output: Formal context structured as a BDD
        (BDDContext)
1  $BDDTemp = \emptyset$
2  $BDDContext = \emptyset$
3  forall the $g \in G$ do
4      forall the $m \in M$ do
5          if $(g, m) \in I$ then
6              $BDDTemp = BDDTemp.nodeTrue$
7          else
8              $BDDTemp = BDDTemp.nodeFalse$ ;
9          end
10         BDDContext = BDDContext+BDDTemp
11     end
12     return BDDContext
13 end
```

Basically the algorithm runs through all attributes of an object and, if it contains a given incidence, the BDD variable is included as true in the temporary BDD representing that object (line 5), otherwise it is included as false (line 6).

Finally, the algorithm adds the object's BDD (BDDTemp) to the formal context (BDDContext) (line 8).

Given the above, there is a clear combinatorial problem in generating sets of premises of the *PropIm* algorithm, which is difficult to handle. To illustrate, if we have a context with 15 attributes, then we will have 45,045 sets, for a context with 20 attribute, that number increases to 310,080 and, with 25, to 1,328,250 sets. Note that, because this is a combinatorial problem, a small increase in the number of attributes can result in an exorbitant number of premise sets.

Thus, we inserted into our algorithm *ImplicP* a pruning heuristic which was able to eliminates unnecessary premise sets, before generating their implications. Another important factor is that with the use of the list data structure, the processing of this algorithm tends to occupy a large memory space, making it impractical to use in some cases. Changing the data structure to BDDs has made storage in main memory and data manipulation much easier and more dynamic.

IV. PROPOSED ALGORITHMS

A. *ImplicP*

We developed the Algorithm 4 to improve the performance of *PropIm*. It is considered a heuristic (*Algorithm* 5) to eliminate unnecessary generation of premise sets to be evaluated. This proposed heuristic is based on the theory of monotonic constraints [35], where if a S rule violates a C constraint, any superset of S will also violate such a constraint (section III-B).

Algorithm 4 ImplicP

```
Input  : Number of Conclusions (Nc), Formal Context (G,M,I)
Output: Set of implications with support greater than 0 (Imp)
1  $Imp = \emptyset$
2  $Conclu = getC(\text{Nc}, M)$
3  foreach $m \in M \backslash Conclu$ do
4      $P = m'' \backslash Conclu$
5      $Size = 1$
6      $Pa = \emptyset$
7      $ResultImpAtr = \emptyset$
8      while $Size < |P|$ do
9          $C = getP(P, Size, 0, ResultImpAtr, Pa)$
10         $Pc = getCP(C, Pa)$
11         foreach $P_1 \subset Pc$ do
12             if $P_1' \neq \emptyset$ and $P_1' \subset m'$ then
13                 $Pa = Pa \cup \{P_1\}$
14                 $Imp = Imp \cup \{P_1 \rightarrow m\}$
15         end
16         $ResultImpAtr = Imp$
17         $Size{+}{+}$
18     end
19 end
20 return $Imp$
```

Algorithm 5 getC

```
Input  : Total Number of Conclusions (NumC), Attribute list (Pa)
Output: List of attributes that will be used as a conclusions (C)
1  Function getC(NumC, Pa):
2      $C = \emptyset$
3      for $i = 1$ to $i <= NumC$ do
4          $C = C \cup Pa[i]$
5      end
6      return C
```

Therefore, if a rule that satisfies both constraints is found (support greater than 0, and a minimum set of attributes that implies a conclusion), we will no longer compute and generate new combinations of rules that have such attributes, avoiding unnecessary processing.

Basically, the algorithm receives as input a formal context *(G,M,I)*, and the output is a set of proper implication *Imp* with support greater than 0. Initially, we determine the attributes that will be used as conclusions (line 2). Following (lines 3-18), the m attributes of M are analyzed, considering that each could be a conclusion to a set of premises.

In line 4, P records all attributes that contain the same objects of attribute m, excluding the conclusions. The *Size* variable determines the size of each premise, initially 1 indicating the smallest possible match for an implication of type $x \rightarrow z$ (line 5). *Pa* stores a set of attributes that represent the auxiliary premises that can generate an implication using m as a conclusion. *ResultImpAtr* stores all the implications that were generated for the current conclusion (line 7).

On lines 8-17, the minimum premise set, limited by $|P|$, is computed. At line 9, *getP* receives all size combinations *Size* from elements in P. The set of candidate premises is obtained through *getCP*. The *getP* function uses the proposed heuristic to reduce the number of calculated premise sets (line 9). Each *candidate premise* $P_1 \subset Pc$ is checked to ensure that *premise* P_1 and *conclusion* m result in a valid implication. If $P_1' \neq \emptyset$ and $P_1' \subset m'$, then the premise P_1 is added to the set of *auxiliary premises Pa*, and the implication $\{P_1 \rightarrow m\}$ is added to list of implications *Imp* (line 14).

The Algorithm 5, *getC*, takes as a parameter the total number of conclusions and the *Pa* attribute list from the formal context, and returns the list of attributes that will be used as conclusions. Attributes selected as conclusions will not be evaluated again as possible premises for a new conclusion.

Algorithm 6, *getCP*, receives all the subsets that do not contain an attribute belonging to the *premise Pa*. It takes as parameter the sets C and *Pa*, and outputs a set of *premises D*.

Algorithm 7, *getP*, receives the sets of size *Size* from elements in P. If the *premise* has already been created (Len=0), it is added to the result list E, and the recursion is terminated. The repeat loop on lines 5-9 generates the next premise and verifies that it has not been used in previous implications,

Algorithm 6 getCP

```
Input  : All the subsets that do not contain an attribute
         belonging to the premise Pa (set C, set Pa)
Output: Set of Candidate Premises (D)
1  Function getCP (C,Pa):
2      $D = \emptyset$
3      foreach $B \subset C$ do
4          if $Pa = \emptyset$ then
5              $D = D \subset B$
6          else foreach $a \in A | A \subset Pa$ do
7              if $a \notin B$ then
8                  $D = D \subset B$
9          end
10         ;
11     end
12     return D
```

Algorithm 7 getP

```
Input  : List of Attributes (Atr), Size of the attribute
         (Len), StartPosition (SP), Result (R), Premises
         (Pa)
Output: Set of Reduced Premises (E)
1  Function getP (Atr, Len, SP, R, Pa):
2      $E = \emptyset$
3      if $Len = 0$ then
4          $E = E \cup \{R\}$
5          return
6      for $i = SP$ to $i <= |Atr| - Len$ do
7          $R[|R| - Len] = Atr[i]$
8          if $R \subset Pa$ then
9              continue
10         $getP(Atr, Len - 1, i + 1, R)$
11     end
12     return E
```

discarding it if it has. The output is a list of premises with the size of the variable *Size*.

Note that, by utilizing the *getP* function, we generate a smaller volume of premise sets. Only if there is no implication for the current conclusion using the premise attribute will the entire sequence be generated. In contrast, the original *PropIm* algorithm always generates all combinations of all premises for all sizes of *Size*. For a better understanding of the premise generation process of *ImplicP* algorithm, we show an example of a formal context that will be used for extracting premises on Table 12.

In 4's first iteration (lines 8-16), using *c1* as conclusion, the algorithm generates the subsets of candidate premises shown in Table 13. As a result, the implication **{*a*} → {*c1*}** is found.

In the second iteration, we generate all subsets of size 1 and 2 that do not contain any of the attributes that were already used in valid premises for implications (Table 14).

TABLE 12. Formal context example.

	a	b	c	d	c1
obj1	X	.	.	.	X
obj2	.	.	.	X	.
obj3	.	.	X	.	.
obj4	.	X	.	.	.
obj5	.	X	X	X	X

Attributes: {a, b, c, d}; Conclusion: {c1}
Proper implications: $\{a\} \to \{c1\}$, $\{bcd\} \to \{c1\}$

TABLE 13. Candidate premises.

Premises	Size
{{a}, {b}, {c}, {d}}	1
Total: 4 premises	

TABLE 14. Candidate premises.

Premises	Size
{{b}, {c}, {d}}	1
{{bc}, {bd}, {cd}}	2
Total: 6 premises	

The premises {ab}, {ac} and {ad} are not generated because there is already an implication using the attribute ***a*** in **{*a*} → {*c1*}**.

For the third iteration, the subsets of size 1, 2 and 3 are generated (Table 15). Again, the premises $\{abc\}$, $\{abd\}$ and $\{acd\}$ are discarded as attribute ***a*** is already used in **{*a*} → {*c1*}**. In this iteration, a new implication **{*bcd*} → {*c1*}** is found, and as all attributes were used as premises the algorithm moves on to the next conclusion.

TABLE 15. Candidate premises.

Premises	Size
{{b}, {c}, {d}}	1
{{bc}, {bd}, {cd}}	2
{bcd}	3
Total: 7 premises	

It is important to highlight that, with the inclusion of *Algorithm 7 in Algorithm 4*, we only calculate the candidate (viable) premises. In most cases, the implications will be found before testing all premise sizes, increasing the runtime economy brought by the *ImplicP* heuristic. In our experiments, we observed a reduction of up to 43% in the number of premise sets being calculated, when compared to the original algorithm, *PropIm*.

For the complexity analysis of the *ImplicP* algorithm, consider N the size of the *premise* that will be generated and, $|M|$ the number of attributes that will be used as a conclusion. In the best case, all implications are generated with size 1, that is, we have $O(N * |M|)$. In the worst case, it would be necessary to generate all combinations of all *premises* for all conclusion attributes, with $O(M(\sum_{i=1}^{n} C_{n,i}))$, where n is the size of the *premise* that will be generated and $|M|$ the number of attributes that will be used as conclusion.

Algorithm 8 PImplicPBDD

```
Input  : Formal context (G,M,I)
Output: The set of proper implications with support
        greater than 0, Imp
1 $(G_bdd, M_bdd, I_bdd) = GenerateBDDContext$(G,M,I)
2 $Imp = \emptyset$
3 foreach $m \in M$ do
4     $Imp = Imp \cup AtrImpPBDD(m, (G_bdd, M_bdd, I_bdd))$
5 end
6 return Imp
```

Algorithm 9 AtrImpPBDD

```
Input  : Attribute m, formal context (G,M,I)
Output: The set of proper implications of m with
        support greater than 0, ImpM
1 $ImpM = \emptyset$
2 $BddC = primeAtrSetBDD$(m) $P$ = m"
3 $Size = 1$
4 $Pa = \emptyset$
5 $ResultImpAtr = \emptyset$
6 while $Size < |P|$ do
7     $C = getP(P, Size, 0, ResultImpAtr, Pa)$
8     $Pc = getCPBDD$(C,Pa)
9     foreach $P_1 \subset Pc$ do
10        $BddP = primeAtrSetPBDD$(P1)
11        if $BddC \neq 0$ and $BddP \neq 0$ and $BddC ==$
          $BddP$ then
12            $Pa = Pa \cup \{P1\}$
13            $ImpM = ImpM \cup \{P1 \rightarrow m\}$
14    end
15    Size++
16 end
17 return ImpM
```

Algorithm 10 primeAtrSetPBDD

```
1 Function primeAtrSetPBDD(ListAttributes):
2     $BddNewExt$ = BddCxt;
3     foreach $it \in ListAttributes$ do
4         $BddNewExt\& =$ BddNewExt.and(it);
5     end
6     return bddNewExt;
```

Algorithm 11 getPPBDD

```
1 Function
  getPPBDD(Atr, Len, StartPosition, Result, Pa):
2     if $Len = 0$ then
3         $D = D \cup$ {Result};
4         return;
5     for $i = StartPosition$ to $i <= |Atr| - Len$ do
6         $Result[|Result| - Len] = Atr[i]$;
7         if $Result \subset Pa$ then
8             continue;
9         $getP(Atr, Len - 1, i + 1, Result)$;
10    end
11    return D;
```

B. THE PImplicPBDD ALGORITMHM

The *PImplicPBDD* (*Algorithm* 8) uses a heuristic to reduce the sets of premises required to obtain the rules of implication and also the BDD framework to simplify the representation of a formal context, making object manipulation more efficient. The objective is to distribute the processing load of different conclusions across distinct cores. For each available processor core, a thread is created to process a distinct conclusion.

The algorithm receives as input a formal context (G, M, I), and its output is a set of proper implications with support greater than 0. The algorithm makes use of the *AtrImpPBDD* function, which receives as parameters an attribute m and the formal context, and outputs the set of proper implications that has m as a conclusion.

The *AtrImpPBDD* calls the function *primeAtrSetPBDD*, responsible for creating the BDD containing all objeccts in the list of attributes received as a parameter. The function outputs a BDD of only the objects that contain those attributes.

The function *getP* calculates all premise sets of size *Size* through elements in P. If the premise has already been created, it is included in the result list and the process is terminated. Otherwise, the next premise of the combination is generated and if already used in the previous implications is discarded. As output, a list of premises with the size of *Size* is generated.

Initially, each attribute m in M that can be a conclusion to a premise is analyzed. For each m, a set of premises P_1 is determined. Through the *primeAtrSetPBDD* function, the *BddC* stores the BDD containing all objects with attribute m (line 2). This BDD is used to verify the premise P_1 and the conclusion m.

Pa stores a set of auxiliary premises that can generate an implication using m as their conclusion. In lines 7-17 of the pseudo-code, we find the set of minimal premises, limited by $|P|$. The premise set C is a combination of size *Size* of elements in P (line 8). In line 9, the set of candidate premises is created with the *getCPBDD* function and, for each candidate premise $P_1 \subset P_C$, a BDD is stored (*bddP*) with all objects of the premise. In line 12, the algorithm verifies *BddC* and *BddP* to ensure that the P_1 premise and its conclusion m result in a valid implication *BddC* = *BddP*). Considering that it is valid, P_1 is added to the set *Pa* and the implication $\{P_1 \rightarrow m\}$ is added to *Imp* (line 14).

Figures 4, 5a, 5b, 5c and 5d show the process of generating an implication using BDD, in the *PImplicPBDD* algorithm.

Figure 5a presents the BDD generated by the conclusion *Instagram*. The BDD generated with the premise *Yammer* is shown in 5b. Figure 5d presents the intersection between the BDD with conclusion *Instagram* and the one

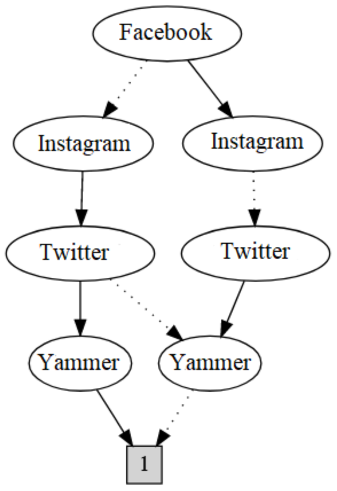

FIGURE 4. BDD generated from the formal context in Table 1.

Algorithm 12 getCPBDD

```
1  Function getCPBDD(C,Pa):
2      $D = \emptyset$;
3      foreach $a \in A | A \subset Pa$ do
4          foreach $B \subset C$ do
5              if $a \notin B$ then
6                  $D = \text{Pc/B}$;
7          end
8      end
9      return D;
```

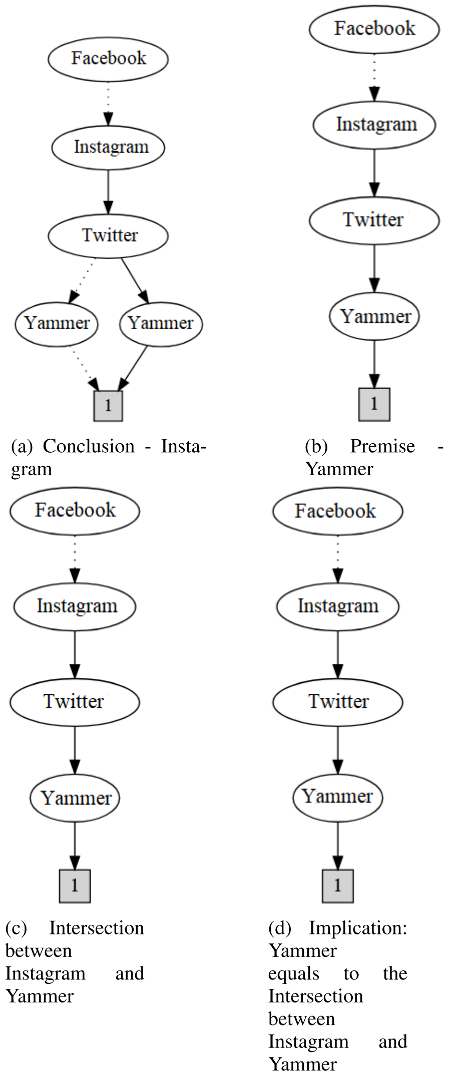
Facebook
Instagram
Twitter
Yammer
Yammer
1
(a) Conclusion - Instagram
Facebook
Instagram
Twitter
Yammer
1
(b) Premise - Yammer
Facebook
Instagram
Twitter
Yammer
1
(c) Intersection between Instagram and Yammer
Facebook
Instagram
Twitter
Yammer
1
(d) Implication: Yammer equals to the Intersection between Instagram and Yammer

FIGURE 5. Internal BDD process for obtaining an implication.

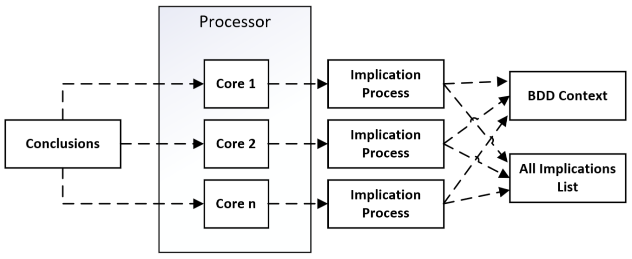

FIGURE 6. Parallelism model used in the *PImplicPBDD* algorithm. Independent cores process independent conclusions.

with premise *Yammer*. If the intersection between the BDD *Instagram* and premise *Yammer* (Figure 5c) is the same as the premise *Yammer*, a valid implication is created. Note that Figure 5b is identical to Figure 5d. This means that all objects that share the premise *Yammer* also share the conclusion *Instagram*.

The order of complexity of the *PImplicPBDD* algorithm's extraction of the set of proper implications is $O(|M||imp|(|G||M|+|imp||M|))$. This is an exponential complexity, as $|imp|$ is combinatory in the worst case scenario. However, we implemented a heuristic that reduces the combinations of attributes to be tested, and the number of premises generated for each conclusion.

The *getCPBDD* algorithm receives all subsets that does not contain an attribute which belongs to the *premise Pa*. It receives the C and Pa sets as a parameter and returns a D set of *premises*.

Figure 6 presents the parallelism model used in the *PImplicPBDD* algorithm (Algorithm 8). In order to determine which parts of the algorithm would be parallelized, the library-based software programming model was used [41]. The *PImplicPBDD* algorithm uses the shared memory parallel computation model, designed using domain partitioning and decomposition that can be divided by conclusions, sharing the same formal context and listing all the implications in common memory. With these changes, it was possible to calculate the premise sets for different conclusions simultaneously.

V. EXPERIMENTS

Our experimental methodology includes randomly generated synthetic contexts with controlled densities and dimensions in all experiments. The random contexts were generated by the SCGAz (Synthetic Context Generator) tool within preset limits of controlled densities and sizes.

The SCGaz tool randomly fills synthetic formal contexts ensuring the absence of redundancies among objects and attributes which could lead to erroneous comparison analysis. At the same time, the tool allowing users to select any density in the bounds of the minimum and maximum permitted for a type of context. According to the authors [42], the tool allows a more controllable and reliable simulation environment. The created formal contexts had 1,000, 10,000 and 100,000 objects, each with sets of 20 and 23 attributes and densities ranging from 30%, 50% and 70% (the maximum density for each generated context). We added a time-limit constraint of 20 days for the time spent extracting the proper implications set, necessary some of the experimental setups would not execute in a viable time on the *PropIm* algorithm.

Note that the randomly generated incident table can vary among two contexts with the same dimensions and densities, resulting in different implication sets. Therefore, a different performance was obtained for each context. For each combination of objects (1,000, 10,000 and 100,000), attributes (20 and 23) and density (30%, 50% and 70%), two formal contexts were generated. Since the contexts were randomly generated, every attribute could be considered as a conclusion in our experiments. Thus, for the formal context with 20 attributes, for example, we have 20 executions using each attribute as the desired conclusion and the remaining 19 as premises, totaling 40 executions for each experimental setup.

The algorithms have been implemented in Java and the experiments were performed using an IBM server machine (OS Ubuntu 16.04), equipped with an Intel Xeon (3.1GHz) 4-core processor, 32GB of RAM memory and 30GB of SSD storage.[2] The parallel algorithm used the multicore architecture to process 4 conclusions simultaneously, one in each of the cores. In the parallel experiments, the goal was to partition the data an simultaneously process different parts of the problem. For all performance tests, we compared different algorithms running the same contexts.

The experiments performed with the *ImplicP* and *PropIm* algorithms, using synthetic contexts, showed that the pruning heuristic proposed in this article caused a reduction in the time required to extract the proper implications set. Table 16 shows the results obtained by both algorithms in experiments varying the number of objects, attributes, and context density. For each scenario, we measured the run times for both algorithms. However, for the scenarios with 23 attributes, the *PropIm* algorithm was not able to execute within 20 days, and had its run terminated.

As shown in Table 16, the speed up obtained with the *ImplicP* algorithm compared to *PropIm* in 20 attribute contexts was of up to 5 times. This gain is based on the effectiveness of the heuristic for reducing the number of assumption combinations required to find an implication. The heuristic

[2]Files available at http://github.com

TABLE 16. Runtime minutes taken to generate the proper implications set.

|G|	|M|	Den(%)	PropIm	ImplicP	Speed Up	PImplicPBDD
1,000	20	30	2,657	529	5.02	118
1,000	20	50	2,925	1,158	2.53	264
1,000	20	70	2,081	1,430	1.46	328
10,000	20	30	2,433	933	2.61	144
10,000	20	50	3,078	1,738	1.77	343
10,000	20	70	3,198	1,706	1.87	355
100,000	20	30	3,460	1,425	2.43	269
100,000	20	50	3,068	2,067	1.48	371
100,000	20	70	2,601	1,869	1.39	517
1,000	23	30	-	7,501	-	1,875
1,000	23	50	-	14,361	-	3,799
1,000	23	70	-	15,587	-	3,897
10,000	23	30	-	7,393	-	1,946
10,000	23	50	-	22,886	-	4,975
10,000	23	70	-	24,660	-	5,138
100,000	23	30	-	15,875	-	3,113
100,000	23	50	-	28,142	-	6,864
100,000	23	70	-	25,082	-	7,166

ensures that, if an implication in the form $\{a, b\} \rightarrow \{C1\}$ has already been formed, new combinations will not test the attributes a or b for the conclusion $C1$. In its next iteration, the algorithm would try to generate combinations of three premises, but it would not use $\{a, b, *\} \rightarrow \{C1\}$ combinations, reducing the number of tests performed by the algorithm. Regarding the scenarios where the contexts had 23 attributes, the heuristic was once again more efficient, as it was able to run before the established 20 day timeout. In most cases, bigger densities have reduced the total speedup, as in those cases the volume of incidences is greatly increased, which causes the algorithm to perform more attribute combinations, consequently taking much longer to find the proper implications set.

From our experimental results, it is noticeable that the execution times of the evaluated algorithms tend to increase when the number of attributes and, mainly, the density (incidence) in the formal context is increases. We investigated the number of attributes in the premises to verify the reason for this increase in computational time. For this, we established a confidence interval of $\alpha = 0.05$ to determine the amount of attributes found in each premise.

Table 17 displays the minimum (Int. Min. Premises) and maximum (Int. Max. Premises) values for the confidence interval for the premises found in each scenario. In addition, we found the most frequent premises (mode), the percentage of that the mode represents considering the total of rules, and finally, the number of proper implications. Importantly, regardless of the algorithms (*PropIm*, *ImplicP* and *PImplicPBDD*), the set of proper implications is the same.

The scenarios with 30% density (incidence of attributes in relation to objects) presented the smallest amount of attributes in the premises, which resulted in the shortest time to find the set of proper implications. Scenarios with lower density are favorable for the proposed heuristic, since after finding a minimum rule, for example, $\{a \rightarrow b\}$, no new rules will be generated using the attribute $\{a\}$ on the left side of the implication. In contrast, for the scenarios with 70% density, in general, there were more number of attributes in

TABLE 17. Quantity of attributes in the premises.

|G|	|M|	Den(%)	Int. Min.Premises	Int. Max.Premises	Mode	Mode % in dataset	# proper implications
1,000	20	30	5.11	5.12	5	85.00%	24,576
1,000	20	50	7.71	7.72	8	65.25%	80,695
1,000	20	70	11.61	11.67	12	43.30%	3,270
10,000	20	30	6.60	6.60	7	59.93%	50,390
10,000	20	50	10.51	10.52	11	51.25%	78,021
10,000	20	70	16.74	17.08	17	90.91%	11
100,000	20	30	8.86	8.86	9	85.97%	135,921
100,000	20	50	13.37	13.39	13	59.97%	13,409
100,000	20	70	18.22	18.24	18	72.28%	8,765
1,000	23	30	5.09	5.09	5	87.66%	58,582
1,000	23	50	7.83	7.84	8	71.37%	389,572
1,000	23	70	11.69	11.70	12	48.52%	69,091
10,000	23	30	6.82	6.83	7	82.11%	244,704
10,000	23	50	10.69	10.69	11	65.38%	717,603
10,000	23	70	17.05	17.15	17	57.99%	695
100,000	23	30	7.87	7.87	8	87.02%	30,256
100,000	23	50	12.98	12.98	13	97.87%	55,078
100,000	23	70	20.76	20.78	20	69.87%	43,257

TABLE 18. Example of proper implication extracted from the formal context of Linkedin.

$A \rightarrow b$
{algorithms, digital marketing} $\rightarrow$ [data scientist}
{computer architecture, office applications, user interface] $\rightarrow$ [data scientist}
{computer architecture, user experience, user interface] $\rightarrow$ [data scientist}

the premises, which resulted in greater computational effort required to find the set of proper implications.

In order to test the *PImplicPBDD* algorithm on a real world problem, we used a formal context based on data from the Linkedin social network. The context uses as conclusion 10 professions and 24 skills as premises. The dataset has a total of 1,269 objects, sampled from Linkedin.

For this real world formal context, the *PImplicPBDD* algorithm processed found the proper implications set in 8 days, while *PropIm* had returned no results after 14 days. Note that, while the number of objects is relatively small, Linkedin's context contains 24 attributes, so it requires a greater generation of combinations and candidate premises for each conclusion, which led to the *PropIm* algorithm being unable to run in a viable time. The *PImplicPBDD* algorithm, which uses both solutions proposed in this article, was able to process the entire context. Table 18 displays three example implications, drawn from the Linkedin formal context with the conclusion of Data Scientist.

VI. CONCLUSION AND FUTURE WORK

Some aspects of this study proved challenging, such as concluding all the experiments on high dimensional contexts in viable time. We compared the performance of several approaches/algorithms, on different contexts, and were able to successfully find the proper implication sets for some contexts that existing algorithms were not able to.

The experiments with the *PropIm* algorithm showed that it was not able to run some of the context with high dimensionality, and that it is less efficient when compared to the approaches proposed by our study. On the other hand, the *PImplicPBDD* algorithm was more efficient in finding the set of proper implications from formal contexts. We also explored the use of parallel algorithms to reduce processing time by computing the implications simultaneously.

The number of implications in the proper implications set is an important aspect of this discussion. When the user needs to find only rules with the minimal number of premises (attributes) that imply in a conclusion, the algorithm will certainly find some redundancy in the rules it generates when exploring that.

As an example, consider the following proper implications: $a \rightarrow C_1$, $a \rightarrow C_2$ and $a \rightarrow C_3$. These rules are redundant, and could be written simply as $a \rightarrow C_1, C_2, C_3$, which would no longer be a proper implication. Proper implications have more interpretability, so the knowledge they represent is easier to understand both by domain experts and common users, but this knowledge will likely be represented a greater number of rules.

Considering this, the algorithms proposed in this article enable a user to, in the rule extraction process, define a subset of attributes to be used as desirable conclusions. Traditional approaches in the literature (*Impec* and *PropIm*) do not have that characteristic, and required a post-processing by the user to select rules they are interested in (i.e., those with the desirable conclusions and attributes). Consequently, many of the rules generated are discarded, and their computation time is wasted.

As future work, we suggest the use of other techniques to parallelize the code, such as using *GPUs* (Graphics Processing Units) and/or *Xeon Phi*, or even distributing the computation of this problem through a cluster of computers.

Other potential work, we suggest using temporal logic (important application in formal verification) to obtain a verification model [43], [44] [45] from a formal context, defining a set of properties, written in temporal logic, that could be viewed as proper implication. For example, the implication a, b $\rightarrow$ c could be represented by the specification AG ((a & b) $\rightarrow$ AG (c)) in this new model. All valid specifications (true) could be viewed as possible implications. The focus of the study would be to obtain the initial model that describes the behavior of the system from the formal context, since it contains only the description of the objects and their attributes. Once this model was built, the verification time could be compared with the response time from other approaches that also obtain proper implications.

ACKNOWLEDGMENT

The authors would also like to warmly thank PUC-MG researchers for raising important questions and making relevant suggestions that helped improve the quality of the article.

REFERENCES

[1] R. Wille, "Restructuring lattice theory: An approach based on hierarchies of concepts," in *Ordered Sets*. Banff, AB, Canada: Springer, 1982, pp. 445–470.

[2] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Berlin, Germany: Springer-Verlag, 1999.

[3] F. Lehmann and R. Wille, "A triadic approach to formal concept analysis," *Conceptual Structures: Applications, Implementation and Theory*. Berlin, Germany: Springer, 1995, pp. 32–43.

[4] C. A. Kumar, M. Radvansky, and J. Annapurna, "Analysis of a vector space model, latent semantic indexing and formal concept analysis for information retrieval," *Cybern. Inf. Technol.*, vol. 12, no. 1, pp. 34–48, 2012.

[5] M. Croitoru, O. Nir, M. Simon, and L. Michael, "Graphical norms via conceptual graphs," *Knowl.-Based Syst.*, vol. 29, pp. 31–43, May 2012. [Online]. Available: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00763678

[6] J. Poelmans, P. Elzinga, and G. Dedene, ''Retrieval of criminal trajectories with an FCA-based approach,'' in *Proc. FCAIR Formal Concept Anal. Meets Inf. Retr. Workshop, Co-Located 35th Eur. Conf. Inf. Retr. (ECIR)*, vol. 977. Moscow, Russia, National Research Univ. Higher School Economics, 2013, pp. 83–94.
[7] M. Rouane-Hacene, M. Huchard, A. Napoli, and P. Valtchev, ''Relational concept analysis: Mining concept lattices from multi-relational data,'' *Ann. Math. Artif. Intell.*, vol. 67, no. 1, pp. 81–108, Jan. 2013. [Online]. Available: https://doi.org/10.1007/s10472-012-9329-3
[8] R. Missaoui, S. O. Kuznetsov, and S. A. Obiedkov, Eds., *Formal Concept Analysis of Social Networks* (Lecture Notes in Social Networks). Berlin, Germany: Springer, 2017, doi: 10.1007/978-3-319-64167-6.
[9] M. Watmough, *Discovering the Hidden Semantics in Enterprise Resource Planning Data Through Formal Concept Analysis*. Berlin, Germany: Springer, 2014, pp. 291–314, doi: 10.1007/978-3-642-35016-0_11.
[10] C. De Maio, G. Fenza, M. Gallo, V. Loia, and S. Senatore, ''Formal and relational concept analysis for fuzzy-based automatic semantic annotation,'' *Appl. Intell.*, vol. 40, no. 1, pp. 154–177, Jan. 2014, doi: 10.1007/s10489-013-0451-7.
[11] J. Cigarrán, A. Castellanos, and A. García-Serrano, ''A step forward for topic detection in Twitter: An FCA-based approach,'' *Expert Syst. Appl.*, vol. 57, pp. 21–36, Sep. 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0957417416301038
[12] E. Negm, S. A. Rahman, and R. Bahgat, ''PREFCA: A portal retrieval engine based on formal concept analysis,'' *Inf. Process. Manage.*, vol. 53, no. 1, pp. 203–222, 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0306457316303569
[13] P. K. Singh, C. A. Kumar, and A. Gani, ''A comprehensive survey on formal concept analysis, its research trends and applications,'' *Appl. Math. Comput. Sci.*, vol. 26, no. 2, pp. 495–516, 2016.
[14] T. H. Davenport and L. Prusak, *Working Knowledge: How organizations Manage What They Know*. New York, NY, USA: Harvard Business School Press, 1997.
[15] B. Ganter, G. Stumme, and R. Wille, *Formal Concept Analysis: Foundations and Applications*, vol. 3626. Darmstadt, Germany: Springer, 2005.
[16] R. Taouil and Y. Bastide, ''Computing proper implications,'' in *Proc. Int. Conf. Conceptual Struct. (ICCS)*, 2001, pp. 46–61.
[17] U. Ryssel, F. Distel, and D. Borchmann, ''Fast algorithms for implication bases and attribute exploration using proper premises,'' *Ann. Math. Artif. Intell.*, vol. 70, nos. 1–2, pp. 25–53, Feb. 2014, doi: 10.1007/s10472-013-9355-9.
[18] P. R. Silva, S. Dias, M. Song, and L. Zárate, ''Minimal implications base for social network analysis,'' *Int. J. Web Inf. Syst.*, vol. 14, no. 1, pp. 62–77, Aug. 2018, doi: 10.1108/IJWIS-04-2017-0028.
[19] H. Mannila and K.-J. Räihä, ''Algorithms for inferring functional dependencies from relations,'' *Data Knowl. Eng.*, vol. 12, no. 1, pp. 83–99, Feb. 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0169023X9490023X
[20] U. Priss, ''Some open problems in formal concept analysis,'' in *Proc. Int. Conf. Formal Concept Anal.*, 2006, pp. 1–6.
[21] S. M. Dias and N. J. Vieira, ''Concept lattices reduction: Definition, analysis and classification,'' *Expert Syst. Appl.*, vol. 42, no. 20, pp. 7084–7097, Nov. 2015.
[22] C. A. Kumar and S. Srinivas, ''Concept lattice reduction using fuzzy K-means clustering,'' *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2696–2704, Mar. 2010.
[23] R. K. Chunduri and A. K. Cherukuri, ''Scalable formal concept analysis algorithms for large datasets using spark,'' *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 11, pp. 4283–4303, Nov. 2019.
[24] C. A. Kumar, ''Fuzzy clustering-based formal concept analysis for association rules mining,'' *Appl. Artif. Intell.*, vol. 26, no. 3, pp. 274–301, Mar. 2012.
[25] W. K. Min and Y. K. Kim, ''Soft concept lattice for formal concept analysis based on soft sets: Theoretical foundations and applications,'' *Soft Comput.*, vol. 23, pp. 9657–9666, Sep. 2019.
[26] L. Guo, Q. Li, and G.-Q. Zhang, ''A representation of continuous domains via relationally approximable concepts in a generalized framework of formal concept analysis,'' *Int. J. Approx. Reasoning*, vol. 114, pp. 29–43, Nov. 2019.
[27] S. Yevtushenko, ''BDD-based algorithms for the construction of the set of all concepts,'' in *Foundations and Applications of Conceptual Structures. Contributions to ICCS 2002*. Berlin, Germany: Springer, 2002, p. 61–73.
[28] A. Rimsa, L. E. Zárate, and M. A. Song, ''Evaluation of different BDD libraries to extract concepts in FCA–perspectives and limitations,'' in *Proc. 9th Int. Conf. Comput. Sci.* Berlin, Germany: Springer-Verlag, 2009, pp. 367–376, doi: 10.1007/978-3-642-01970-8_36.
[29] S. M. Neto, L. E. Zárate, and M. A. J. Song, ''Handling high dimensionality contexts in formal concept analysis via binary decision diagrams,'' *Inf. Sci.*, vol. 429, pp. 361–376, Mar. 2018.
[30] A. Salleb, Z. Maazouzi, and C. Vrain, ''Mining maximal frequent itemsets by a Boolean based approach,'' in *Proc. 15th Eur. Conf. Artif. Intell.* Amsterdam, The Netherlands: IOS Press, 2002, pp. 385–389. [Online]. Available: http://dl.acm.org/citation.cfm?id=3000905.3000987
[31] K. Bertet and B. Monjardet, ''The multiple facets of the canonical direct unit implicational basis,'' *Theor. Comput. Sci.*, vol. 411, nos. 22–24, pp. 2155–2166, May 2010, doi: 10.1016/j.tcs.2009.12.021.
[32] W. W. Armstrong, ''Dependency structures of database relationships,'' *Inf. Process.* vol. 74, pp. 580–583, Aug. 1974.
[33] R. Agrawal, T. Imieliński, and A. Swami, ''Mining association rules between sets of items in large databases,'' *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993, doi: 10.1145/170036.170072.
[34] J. Pei, J. Han, and L. V. S. Lakshmanan, ''Mining frequent itemsets with convertible constraints,'' in *Proc. 17th Int. Conf. Data Eng.*, Apr. 2001, pp. 433–442.
[35] C. K.-S. Leung, *Monotone Constraints*. Boston, MA, USA: Springer, 2009, p. 1769, doi: 10.1007/978-0-387-39940-9_5048.
[36] S. Brin, R. Motwani, and C. Silverstein, ''Beyond market baskets: Generalizing association rules to correlations,'' *ACM SIGMOD Rec.*, vol. 26, no. 2, pp. 265–276, Jun. 1997, doi: 10.1145/253262.253327.
[37] J. Pei and J. Han, ''Can we push more constraints into frequent pattern mining?'' in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD*. New York, NY, USA: ACM, 2000, pp. 350–354, doi: 10.1145/347090.347166.
[38] R. E. Bryant, ''Graph-based algorithms for Boolean function manipulation,'' *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986, doi: 10.1109/TC.1986.1676819.
[39] Y. Mo, L. Xing, F. Zhong, Z. Pan, and Z. Chen, ''Choosing a heuristic and root node for edge ordering in BDD-based network reliability analysis,'' *Rel. Eng. Syst. Saf.*, vol. 131, pp. 83–93, Nov. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832014001550
[40] P. Burmeister, *Formal Concept Analysis With ConImp: Introduction to the Basic Features*. Darmstadt, Germany: Springer, 2003.
[41] R. Z. Khan and M. F. Ali, ''Current trends in parallel computing,'' *Int. J. Comput. Appl.*, vol. 59, no. 2, pp. 19–25, 2012.
[42] A. Rimsa, M. A. J. Song, and L. E. Zarate, ''SCGaz—A synthetic formal context generator with density control for test and evaluation of FCA algorithms,'' in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3464–3470.
[43] A. Souri and M. Norouzi, ''A new probable decision making approach for verification of probabilistic real-time systems,'' in *Proc. 6th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Sep. 2015, pp. 44–47.
[44] A. Souri, A. M. Rahmani, N. J. Navimipour, and R. Rezaei, ''Formal modeling and verification of a service composition approach in the social customer relationship management system,'' *Inf. Technol. People*, vol. 32, no. 6, pp. 1591–1607, Dec. 2019.
[45] A. Souri, A. M. Rahmani, N. J. Navimipour, and R. Rezaei, ''A symbolic model checking approach in formal verification of distributed systems,'' *Hum.-Centric Comput. Inf. Sci.*, vol. 9, no. 1, p. 4, Dec. 2019.

JULIO C. V. NEVES received the B.Sc. degree in data processing from UNA, in 2001, the master's degree in software engineering from PUC-MG, in 2003, the master's degree in strategic information management from UFMG, in 2004, the master's degree in computer science from PUC-MG, in 2009, and the M.B.A. degree in finance from UNA, in 2012. He is currently pursuing the Ph.D. degree in computer science with the Pontifical Catholic University of Minas Gerais. He is also a Professor with the Information Technology Management Institute (IGTI) and a Global IT Manager. His research interests include infrastructure and service desk and systems.

PEDRO HENRIQUE BATISTA RUAS DA SILVEIRA received the B.Sc. degree in computer information systems and the master's degree in computer science from the Pontifical Catholic University of Minas Gerais, in 2016, where he is currently pursuing the Ph.D. degree in computer science. He is also an Associate Professor with PUC Minas. His research interests include triadic concept analysis, formal concept analysis, data mining, and social network analysis.

ROKIA MISSAOUI received the Ph.D. degree in computer science from the University of Montreal, in 1988. She was a Professor, Canada, for more than 32 years. She was a Professor with the University of Quebec in Montreal (UQAM), for a period of 15 years. She is currently a Full Professor with the Department of Computer Science and Engineering, University of Quebec Outaouais (UQO). She also leads the LARIM Laboratory with UQO and a member with the LATECE Laboratory, UQAM. She was involved in several research projects funded by granting agencies and industrial partners. Her teaching activities and research interests include advanced databases, data mining and warehousing, machine learning, social network analysis, and big data analytics.

SÉRGIO M. DIAS received the B.S. degree in computer science from the Pontific Catholic University of Minas Gerais (PUC Minas), in 2007, and the M.Sc. and Ph.D. degrees in computer science from the Federal University of Minas Gerais (UFMG), in 2016 and 2010, respectively. He is currently a Data Scientist, a Professor, and a Researcher in computer science. He is also with the Federal Service of Data Processing (SERPRO) and PUC Minas.

LUIS E. ZÁRATE graduated in electrical engineering from URP, Peru, in 1981. He received the M.Sc. and Ph.D. degrees from the Federal University of Minas Gerais (UFMG), in 1992 and 1998, respectively. Since 1992, he has been in research and teaching various disciplines with the Pontifical Catholic University of Minas Gerais. He is currently a Coordinator with the Laboratory of Computational Intelligence, PUC-MG (LICAP). He is responsible for several researches that are supported by governmental organizations, Brazil. His research interests include data mining and big data, formal concept analysis, and soft computing.

MARK A. J. SONG received the B.Sc., M.Sc., and Ph.D. degrees from the Federal University of Minas Gerais, in 1991, 1996, and 2004, respectively. Since 1993, he has been a Researcher and a Professor with the Pontifical Catholic University of Minas Gerais. He is currently a coordinator with the Laboratory of Formal Methods, PUC-MG (LabMF). He is responsible for several researches funded by granting agencies and governmental organizations, Brazil. His research interests include formal verification, model verification, software testing, and formal concept analysis.