information

Article

ImplicPBDD: A New Approach to Extract Proper Implications Set from High-Dimension Formal Contexts Using a Binary Decision Diagram †

Phillip G. Santos [1,‡], Pedro Henrique B. Ruas [1,*,‡], Julio C. V. Neves [1,‡], Paula R. Silva [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 31980-110, Brazil; phillipgs@gmail.com (P.G.S.); pedrohbruas@gmail.com (P.H.B.R.); juliocesar.neves@gmail.com (J.C.V.N.); sergiomariano@gmail.com (S.M.D.); zarate@pucminas.br (L.E.Z.); song@pucminas.br (M.A.J.S.)

2 Laboratory of Artificial Intelligence and Decision Support (LIAAD), INESC TEC, Porto 4200-465, Portugal; paula.csraissa@gmail.com

3 Federal Service of Data Processing (SERPRO), Belo Horizonte 31035-536, Brazil

* Correspondence: pedrohbruas@gmail.com

† This paper is an extended version of our paper published in Santos, P., Neves, J., Silva, P., Dias, S. M., Zárate, L., and Song, M. (2018). An Approach to Extract Proper Implications Set from High-dimension Formal Contexts using Binary Decision Diagram. In Proceedings of the 20th International Conference on Enterprise Information Systems (Vol. 1, pp. 50–57). Science and Technology Publications.

‡ These authors contributed equally to this work.

Received: 20 September 2018 ; Accepted: 22 October 2018; Published: 25 October 2018

Abstract: Formal concept analysis (FCA) is largely applied in different areas. However, in some FCA applications the volume of information that needs to be processed can become unfeasible. Thus, the demand for new approaches and algorithms that enable processing large amounts of information is increasing substantially. This article presents a new algorithm for extracting proper implications from high-dimensional contexts. The proposed algorithm, called ImplicPBDD, was based on the PropIm algorithm, and uses a data structure called binary decision diagram (BDD) to simplify the representation of the formal context and enhance the extraction of proper implications. In order to analyze the performance of the ImplicPBDD algorithm, we performed tests using synthetic contexts varying the number of objects, attributes and context density. The experiments show that ImplicPBDD has a better performance—up to 80% faster—than its original algorithm, regardless of the number of attributes, objects and densities

Keywords: formal concept analysis; proper implication; binary decision diagram

1. Introduction

With the advance of technology, the volume of data collected and stored to attend society's new requirements and services has increased significantly. Thus, it is practically unfeasible to analyze this large volume of data without the support of techniques of knowledge extraction and representation. One of the techniques that can be used to process and analyze information is the *Formal Concept Analysis* (FCA) [1].

The FCA is a field of mathematics applied for data analysis where associations and dependencies between instances and attributes are identified from a dataset [2,3]. It has been applied in different areas such as text mining [4], Information Retrieval [5], linguistics [6], security analysis [7], web services [8], social network analysis [9–11], software engineering [12], Topic Detection [13], Search engine [14], among others. In [15], a comprehensive revision of FCA applications and trends is given.

One way to represent a set of data composed of instances is through a formal context (G, M, I), which is an incidence table composed of objects (G), attributes (M) and the incidence relations (I) between objects (instances) and attributes. A possible knowledge that can be obtained from the formal context is the set of implications $\mathcal{I}$ among the attributes [16]. The implication $A \rightarrow B$, where $(A, B \subseteq M)$, reveals that *every object* containing the attributes belonging to set A has also the attributes from set B. The sets A and B are called, respectively, premise and conclusion.

Applying different properties to a $\mathcal{I}$ set, we can generate implication sets with certain constraints for specific domains. One of these specific sets is the *proper implications* set, in which for each implication the premise is minimal and the conclusion is a singleton (unit set) [17].

The proper implications set is useful because it provides a minimum representation of the data, for applications where the goal is to find the minimum set of attributes to achieve specific purposes. For example, the work developed in [18] aimed to identify relationships between professional skills of LinkedIn users. The proper implications set was applied to identify the minimum set of skills (premise) that imply in professional competence (conclusion). Minimal path problems, such as finding the best flight routes, can also be solved using proper implication sets. Finally, they can also be applied to infer functional dependencies from relations, since they can be rapidly converted into the structure and used in database design [19]. In addition, there are different types of rule coverage implemented by different algorithms. The minimum coverage—which contains the least number of rules describing the dataset— can be efficiently obtained, but it demands the use of axioms in order to discover new relations and implication rules. Conversely, the proper coverage makes the relationships more explicit, although it increases the number of rules describing the dataset. In several problems, the volume of data to be manipulated makes the extraction of implications unfeasible. In this article we propose a solution to extract these implications efficiently.

The algorithm originally proposed by [16], known as *Impec*, is a well-known algorithm for obtaining and extracting proper implications from a formal context. *Impec* provides 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. From a set of M attributes the algorithm finds the basis of its proper implications. However, in some cases, it may be desired that only a few attributes be considered for computing an implication, which is not possible in the *Impec* algorithm without first generating all implications and then selecting those implications where the desired conclusion is on the right side of the rule.

Unlike *Impec* algorithm, which finds its proper implications considering the set of attributes M with support greater than or equal to 0, the *PropIm* algorithm [18] generates its proper implications (also considering the entire set M of attributes) but only with support greater than 0. Implications with support equal to 0 are valid implications (obtained from other implications, for example, Armstrong's axioms); however, in formal contexts, there is no occurrence of such implications.

Different challenges have been proposed by the FCA research community [20]. One of these challenges is the manipulation of high-dimensional formal contexts such as those with 120,000 instances and 70,000 attributes. High-dimensional formal contexts can be generated from big data problems and they are computationally difficult to handle—most algorithms do not perform adequately in these contexts, because they are usually unable to process large amounts of data.

Although the *Impec* and *PropIm* algorithms extract the proper implications, the computational performance of these algorithms processing high-dimensional contexts (in the number of instances) becomes impracticable. An algorithm called *ProperImplicBDD*, based on the *PropIm* [18], was proposed in [21]. The *ProperImplicBDD* algorithm uses a data structure known as Binary Decision Diagram (BDD) in order to manipulate high-dimensional formal contexts efficiently.

In this article, we introduce a new algorithm named *ImplicPBDD* (this being an evolution of the algorithm *ProperImplicBDD* [21]), which is capable of manipulating high-dimensional formal contexts.

Both algorithms used the Binary Decision Diagram (BDD) as a data structure. However, the algorithm *ImplicPBDD* presents a novel pruning heuristic (described in Section 4) to find the premises of the implications more efficiently, when compared to [18,21].

The main objectives of this study are:

- to use BDD as a data structure in order to extract proper implications from high-dimensional contexts in the number of instances;
- to process a greater volume of data compared to the algorithms proposed in [16,18];
- to handle datasets that partially fit the FCA challenge contexts [20], namely the context containing 120,000 instances.

In order to analyse the performance of the *ImplicPBDD* algorithm, we performed experiments using only synthetic contexts. We have decided to use synthetic contexts instead of real ones in order to control the experiments and produce concrete results regarding algorithm's performance in high-dimensional contexts, considering the number of instances, because we can control and vary the number of attributes and context density to evaluate our approach. The variations in attributes and densities were used to analyse the time of searches using both algorithms, *PropIm* and *ImplicPBDD*.

The number of objects (instances) used in the experiments ranged from 1000, 10,000 and 100,000, partially meeting the challenge stipulated in [20]. Regarding the densities of the formal contexts, which is the amount of incidence between objects and attributes, they were also varied to verify the impact on the algorithms' execution times.

Considering that approach, formal contexts with 30%, 50% and 70% density were generated for the experiments, in order to analyse the algorithms' performances in sparse and non-sparse contexts. The results showed that *ImplicPBDD* has a better performance—up to 80% faster—than *PropIm*, regardless of the number of attributes, objects and densities.

In addition, we presented the results obtained with the algorithm *ImplicPBDD* for a real context based on the social network LinkedIn. The main objective was to determine the minimum skills that a professional must have to reach certain job positions.

This article is organised as follows: Section 2 presents basic concepts of the FCA and BDD approaches. Section 3 describes the related works on BDD and implications. In Section 4, the proposed algorithm is described. Section 5 contains the experimental methodology, and the evaluation and discussion of the proposed algorithm *ImplicPBDD*. In Section 6, we present the real-world data case study, based on LinkedIn data. Finally, Section 7 presents our conclusions and future work suggestions.

2. Background

2.1. Formal Concept Analysis

Formal concept analysis (FCA) is a field of mathematics that allows the identification of a dataset's concepts and dependencies (implications), which are represented in a formal context [1]. Details about formal contexts, formal concepts, and implication rules, basic elements of FCA, are described below.

Definition 1. ***Formal Context***: *a formal context is formed by a triple (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 attributes where $I \subseteq G \times M$ [2].*
An example of a context is shown in Table 1. The objects (rows) are: Whale, Owl, Human, Shark and Penguin. The attributes (columns) are : Aquatic, Terrestrial, Irrational, Feathered. The incidences ("X") are the relationships between objects and attributes.

Table 1. A simple formal context.

	Aquatic (a)	**Terrestrial** (b)	**Irrational** (c)	**Feathered** (d)
Whale	X	.	X	.
Owl	.	X	X	X
Human	.	X	.	.
Shark	X	.	X	.
Penguin	.	X	X	X

Definition 2. ***Formal Concept***: *a formal concept is defined by a pair (A, B) where $A \subseteq G$ is called extension and $B \subseteq M$ is called intention. This pair must follow the conditions where* A = B′ *and* B = A′ *[2]. The relation is defined by the derivation operator (′):*

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

A formal concept seeks to identify the set of attributes (intention) that delimit and characterise an object (extension). Using the formal context presented in Table 1 as an example, the generated concepts would be:

({Whale, Owl, Human, Shark, Penguin}, {})
({Owl, Human, Penguin}, {Terrestrial})
({Owl, Penguin}, {Terrestrial, Irrational, Feathered})
({Whale, Owl, Shark, Penguin}, {Irrational})
({Whale, Shark}, {Aquatic, Irrational})
({}, {Aquatic, Terrestrial, Irrational, Feathered})

Definition 3. ***Implications from Formal Context***: *implications are dependencies between elements of a set of attributes obtained from a formal context. Using a formal context (G, M, I) an implication rule would be $A \to B$ where $A, B \subseteq M$ (A and B are defined, respectively, as the rule's premise and conclusion). An implication rule $A \to B$ is considered valid for the context (G, M, I) if, and only if, every object that has the attributes of A also has the attributes of B. Formally $\forall g \in G[\forall a \in A\ gIa \to \forall b \in B\ gIb]$.*

As an example of an implication rule, we can consider the universe of animals as described in Table 1. In this table, every feathered animal is irrational. This type of relationship can be described as an implication. The implication "Feathered implies Irrational" (Feathered → Irrational) is a way to describe that any animal that has feathers is also irrational. Table 2 is an example of implication rules based on the formal context presented in Table 1.

Implication rules can be obtained through formal concepts [22], formal contexts [17] and concept lattice [23]. In our study, the implication rules extraction is based on formal contexts.

Table 2. Implication rules extracted from the formal context in Table 1.

$A \to B$
Terrestrial → Irrational, Feathered
Irrational → Terrestrial, Feathered
Feathered → Terrestrial, Irrational
Aquatic → Irrational
Irrational → Aquatic

Definition 4. ***Proper Implications***: *an implications set is called as the proper implications set [16], or unary implication system (UIS) [23], when, for each implication, the right side (conclusion) contains only one attribute and the left side (premise) is reduced: if $A \to m \in \mathcal{I}$ then there is not any $Q \to m \in \mathcal{I}$ such that*

$Q \subset A$. The set of proper implications *$\mathcal{I}$ for (G, M, I) is defined formally as: $\{A \to m \in \mathcal{I} \mid A \subseteq M$ and $m \in M \setminus A$ and $\forall Z \subset A \colon Z \to m \notin \mathcal{I}\}$.*

2.2. 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) and it can be manipulated efficiently [24,25].

Figure 1 represents the formal context presented in Table 1 as a binary decision tree. For a better view, attributes names have been replaced by letters: Aquatic (a), Terrestrial (b), Irrational (c), Feathered (d).

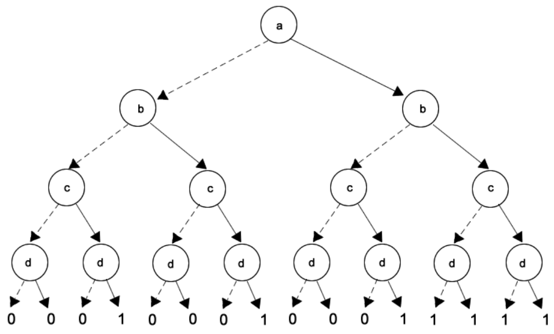


Figure 1. An example of Binary Decision Tree.

Figure 2 provides a simple example which the BDD is used to represent a binary decision tree described on the Figure 1 (where lines represents 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 used the formal context in order to create the BDD. Equation (1) represents a boolean formula correspondent to Table 1.

$$f(a,b,c,d) = a\bar{b}c\bar{d} + \bar{a}bcd + \bar{a}b\bar{c}\bar{d} \tag{1}$$

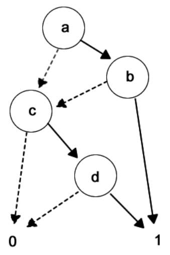


Figure 2. Example of a BDD from Figure 1.

Note that Equation (1) represents all the formal context in Table 1. The attribute a means that in this part of the function the attribute is true (this object has this attribute), whereas $\bar{a}$ means the opposite. The part $a\bar{b}c\bar{d}$ of the equation was created to validate the Whale and Shark objects, $\bar{a}bcd$ was created to validate the Penguin and Owl objects and the last part $\bar{a}b\bar{c}\bar{d}$ validates the Human object.

The generated BDD corresponding to the formal context presented in Table 1 (and described by Equation (1)) can be seen in Figure 3. For a better view of each object in the BDD, object names have been replaced by numbers and attributes replaced by letters: Whale (1), Owl (2), Human (3), Shark (4), Penguin (5) and Aquatic (a), Terrestrial (b), Irrational(c), Feathered (d). Take as example, the object 1 (Whale) which is represented by the path Aquatic, $\overline{Terrestrial}$, Irrational, $\overline{Feathered}$.

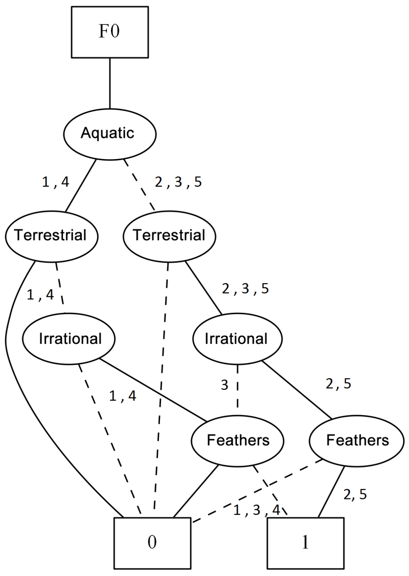


Figure 3. BDD that represents the context in the Table 1.

The most relevant BDD libraries were evaluated in [26] and among them the CUDD— Colorado University Decision Diagram was chosen for providing function packages to work with Binary Decision Diagrams (BDDs) besides having more recent updates. The CUDD also has functions for Algebraic Decision Diagrams (ADDs) and Zero-suppressed Binary Decision Diagrams (ZDDs) that can be used to represent formal contexts.

2.3. Representing a Formal Context with the BDD

In order to use BDDs to represent a the formal context, we employed a similar algorithm to the one proposed by [26].

The Algorithm 1 receives as input a file in Burmeister format [27], traverses all attributes of the object and, if the object contains the attribute (incidence), the BDD variable indicated by index "*i*" is included in the object's temporary BDD with the true indication. Otherwise, the variable is included with the false indication. Finally, the algorithm adds the BDD representing the current object (BDDTemp) to the BDD that represents the formal context (BDDContext). Thus, by doing AND operations to insert the attributes and OR for the objects, the formal context is built as a BDD.

Algorithm 1: generateBDDFormalContext.

```
Input  : Formal Context$(G, M, I)$
Output : Formal Context using BDD
1  $Arquivo$ =File.readLine() while $File.End()$ do
2      $BDDTemp$ =NewBDD for $i \leftarrow 0$ to $NumberOfAttributes$ do
3          if $File[i]$ ==$'X'$ then
4              $BDDTemp\&$ =noTrue(i)
5          end
6          else $BDDTemp\&$ =noFalse(i)
7      end
8      BDDContext $||$ = BDDTemp
9  end
10 return BDDContext
```

2.4. PropIm Algorithm

For a better analysis of our algorithm (ImplicPBDD), we first present and describe the *PropIm* algorithm proposed by [18].

The Algorithm 2 receives as input a formal context *(G, M, I)*, and outputs a set of proper implications. Line 1 initializes the *imp* set as empty. The following loop (lines 2–17) analyzes each attribute present in M. Initially, each attribute m can be a conclusion for a set of premises. For each m, the algorithm calculates the premises $P1$.

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

Pa stores a set of auxiliary premises that can generate an implication using m as a conclusion (in line 5 *Pa* is initialized as empty).

From lines 6–16, the set of minimum premises is found, limited by $|P|$. In Line 7, the set C obtains all combinations of *size* from elements in P. In Line 8, the set of candidate premises is formed through the Algorithm 3, called `GetCandidate`.

Each candidate premise $P1 \subset PC$ is checked to ensure that the premise $P1$ and the conclusion m result in a valid proper implication. Case $P1' \neq 0$ and $P1' \subset m'$, the premise $P1$ is added to the set of auxiliary premises Pa, and also the implication *{P1 →m}* is added to the list of implications $imp = imp \cup \{P1 \rightarrow m\}$.

Algorithm 2: PropIm.

```
Input  : Formal context ($G$,$M$,$I$)
Output : set of implication $imp$ with support greater than 0
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                 $imp = imp \cup \{P1 \rightarrow m\}$
13             end
14         end
15         size++
16     end
17 end
18 return $imp$
```

2.4.1. getCandidate Algorithm

The Algorithm 3 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 3: getCandidate.

```
Function getCandidateProp(C,Pa):
    $D = \emptyset$;
    foreach $a \in A | A \subset Pa$ do
        foreach $B \subset C$ do
            if $a \notin B$ then
                $D =$ Pc/B
        end
    end
    return D;
```

3. Related Works

Currently, there are several FCA applications to extract proper implications. However, none of the existing algorithms works efficiently in high-dimensional contexts. We found studies that used BDD in different applications [28–30]. Other researches included FCA, but did not consider the use of BDD [31,32].

As mentioned in [16], the *Impec*—the state-of-the-art algorithm to extract proper implications sets—was proposed in order to extract all proper implications with some support from a formal context, but the algorithm does not perform well in high-dimensional contexts.

Similarly to [16,18] proposed the *PropIm* algorithm to extract proper implications based on the *Impec* algorithm, but *PropIm* extracts proper implications with support greater than zero, meaning all the implications found have at least one occurrence in the context. The algorithm was used to identify the relationships between professional skills of LinkedIn user profiles through appropriate implications. The results presented the minimum sets of skills that would be required to achieve certain job positions. Unlike other articles referenced here, our proposed solution has (as the main goal) to manipulate and process proper implication sets on high-dimensional contexts, as the algorithm proposed by [16], did not obtain a satisfactory performance.

An algorithm to extract formal concepts using BDD was proposed by [33]. The BDD was used to represent the list of found concepts. However, the authors used contexts with 900 objects and 50 attributes, which proved to be efficient only for dense contexts.

Different from [33,34] proposed to use BDD in two FCA algorithms (*NextClosure* and *Inclose2*) that extract formal concepts to be able to manipulate high-dimensional contexts. The authors proposed to use BDD to represent the formal contexts, reducing the required memory space used to represent the formal context and simplifying information manipulation operations. Their results showed that this approach allows the manipulation of higher dimensional contexts in the quantities of objects that were previously not feasible for the original algorithms. Similarly to [34,35] explained that it is possible to work with larger volumes of data using BDD outside the context of FCA. The authors used BDD to mine transactions in large transactional databases.

In [26], the authors used BDD for extracting formal concepts, but their focus was on using different BDD libraries to check which library best could be used in FCA. [26] used brute force methods to obtain the set of intentions, and consequently the BDDs that represented the extensions. In all experiments performed in the study, the algorithms with BDD were more efficient when compared to the original algorithms. Similarly to the works of [26,33–35], our study proposes to use BDD to process high-dimensional contexts, but BDD is used to represent and manipulate the formal context to extract implications.

4. The Proposed Algorithm ImplicPBDD

In order to achieve a better performance than the *PropIm* algorithm [18] when processing larger volumes of data, all *PropIm* algorithm methods were previously analyzed. With this analysis, we verified that the function that returned the objects in common from a set of attributes was the most costly part of the algorithm.

In order to optimize this function, we employed BDD to represent the formal context in a more compressed form, and to obtain a better performance in context operations with attributes and objects.

The order of complexity for the extraction of proper rules task is $O(|M||imp|(|G||M| + |imp||M|))$. The order of complexity of the $ImplicPBDD$ algorithm is exponential. However, an heuristic was implemented to reduce the combinations of attributes in the premises.

The $ImplicPBDD$ Algorithm uses a heuristic to reduce the combinations of required assumptions to obtain the rules (presented in Algorithm 4) and also the BDD data structure to simplify the representation of a formal context and increase the performance in manipulating objects.

We also created a function—`primeAtrSetBDD`—to improve the calculation of the similarity among attributes and objects (Algorithm 5). parameters, and returns a BDD containing all objects of M.

A pre-processing step is performed to transform a formal context into its corresponding BDD. For the creation of the BDD representing the formal context, we used the $generateBDDFormalContext$ algorithm, proposed by [26], as presented and described in Algorithm 1.

The Algorithm 4 receives as input a formal context *(G, M, I)*, and its output is a set of proper implications with support greater than 0.

Algorithm 4: ImplicPBDD.

```
Input  : Formal Context $\beta$(G,M,I)
Output : Set of implications $Imp$ with support greater than 0
1  $Imp = \emptyset$
2  foreach $m \in M$ do
3      $BddC = primeAtrSetBDD$(m)
4      $P$ =m"
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 $P1 \subset Pc$ do
12             $BddP = primeAtrSetBDD$(P1)
13             if $BddC \neq 0$ and $BddP \neq 0$ and $BddC = BddP$ then
14                 $Pa = Pa \cup \{P1\}$
15                 $Imp = Imp \cup \{P1 \to m\}$
16         end
17         $ResultImpAtr = Imp$
18         Size++
19     end
20 end
21 return Imp
```

Line 1 initializes the empty Imp set. The following *for* (lines 2–19) parses each attribute of the M set. Initially, each m attribute can be a conclusion for a set of premises. For each m, the algorithm calculates the premises $P1$.

The $BddC$ variable in the $primeAtrSetPBDD$ function receives and stores the BDD containing all objects of the m attribute (line 3). This BDD will be used to verify the equality between the premise BDD ($P1$) and the conclusion BDD (m). Algorithm 5 describes the primeAtrSetPBDD (ListAttributes) function, which is responsible for getting the BDD containing all objects from the list of attributes passed as parameters.

The P variable stores all the attributes that contain the same objects of m (line 4).

The *Size* variable determines the size of each premise. Because the smallest possible size is 1 (an implication of type $b \to a$), its value is initialized as 1 (Line 5).

A set of auxiliary assumptions that can generate an implication using m as conclusion is stored in Pa (in line 6, Pa is initialized as empty). In lines 8–16, the set of minimum assumptions is found and is limited by $|P|$. In line 9, the set C gets all combinations of size *Size* of elements in P. It receives, as a parameter, the elements in P, *Size*, start position of the combination and the list in which the result list of the combinations will be stored.

In line 9, the `getP()` function uses heuristics to reduce the total number of assumption combinations. Each candidate premise $P1 \subset PC$ is checked to ensure that premise $P1$ and conclusion m result in a valid correct implication. In case $P1' \neq \emptyset$ and $P1' \subset m'$, the premise $P1$ is added to the set of auxiliary premises Pa, and also the implication *{P1 →m}* is added to the list of implications Imp line 14.

In Line 10, the set of candidate premises is created through the `getCP()` function, which obtains all subsets that do not contain an attribute that belongs to the Pa premise. It receives, as a parameter, sets C and Pa and returns a set of D premises.

Note that using the solution proposed in the Algorithm `getP()`, it is possible to generate a smaller number of combinations for assumptions of a certain size of the *Size* variable. During the generation of a given set of assumptions, if a premise attribute to be generated has already been used as an implication for the current conclusion, the generation sequence is discarded improving the final performance of the algorithm. If there is no implication for the current conclusion using the premise attribute, the entire sequence will be generated. The original $PropIm$ algorithm always generates all combinations of all premises for all sizes of *Size*.

For each candidate premise $P1 \subset PC$, $BddP$ stores the BDD that contains all the objects of the premise. In line 12, a check is made between $BddC$ and $BddP$ to ensure that the premise *P1* and conclusion m result in a valid implication. If $BddC$ = $BddP$ the implication is valid. Considering that, the implication is valid, the premise $P1$ is added to the premises hypothesis Pa and also the implication *{P1→m}* is added to list of implications Imp line 15.

Algorithm 5 presents the *primeAtrSetBDD()* function, responsible for obtaining the BDD that contains all objects from the list of attributes informed as a parameter. The function computes the conjunction between the BDD that represents the entire formal context and the BDD of each attribute of the list of attributes. It outputs a BDD containing only the objects that contain all the attributes informed.

Algorithm 5: primeAtrSetPBDD.

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

5. Methodology of Experiments

The methodology we adopted in our study was to use randomly generated synthetic contexts with controlled densities and dimensions in the experiments. The considered contexts had 1000, 10,000 and 100,000 objects, combined with sets of 16 and 20 attributes (eight time more combinations than 16), with densities ranging from the minimum, 30%, 50% and 70% the maximum density for each generated context. For contexts creation, the SCGAz tool [26] was used, to generate random contexts within the limits of controlled densities and sizes. We also added a time-limit constraint for the time expent extracting the implication rules. This limit was set to up to 3 days, and it was necessary because some experiments performed with the *PropIm* algorithm did not return any result or finished execution, even after long periods of processing.

Note that the randomly generated incident table can vary between two contexts with the same dimensions and densities, giving rise to different implication sets. Therefore, a different performance was obtained for each context. For each combination between the number of objects (1000, 10,000 and 100,000), attributes (16 and 20) and density (30%, 50% and 70%), two formal contexts were generated. Since the contexts were randomly generated, initially there was no a set of conclusions. Therefore, each attribute of the context was considered as conclusion in our experiments.

Thus, for the formal context with 16 attributes, for example, we have 16 executions using each attribute as the desired conclusion and the remaining 15 as premises, totaling 32 executions for each combination set.

The main objective of the experiments was to compare the performances of our *ImplicPBDD* algorithm and the *PropIm* algorithm, when generating the proper implications sets. As mentioned, the comparisons were made using synthetic contexts where the number of objects, attributes and density were varied.

The algorithms were implemented in Java and executed on an IBM Server (OS Ubuntu 16.04) with an Intel Xeon processor (3.1 GHz) with 4 cores, 32 GB of RAM and 30 GB of SSD storage. We kept the compatibility with the language in which the *PropIm* algorithm was written (JAVA). It was used a library that provides support for the BDD in JAVA. The library chosen was JavaBDD [36] which provides JNI for relevant BDD libraries which are CUDD, BuDDy and CAL [26].

The experiments with the synthetic contexts allowed for a better control regarding the generation of contexts for analysis. In every experiment, we evaluated the performance of *PropIm* and *ImplicPBDD* running the same context with a maximum run-time of 3 days to obtain the proper implications set. Our algorithm obtained significant gains in execution time when compared to *PropIm* (see Table 3, Figures 4 and 5).

Table 3. Comparison of speed increase between PropIm and ImplicPBDD algorithms.

∣G∣	∣M∣	Den (%)	PropIm (min)	ImplicPBDD (min)	Gain
1000	16	30	6.11	2.38	61.0%
1000	16	50	6.3	3.07	51.3%
1000	16	70	6.38	3.49	45.3%
10,000	16	30	6.49	3.13	51.8%
10,000	16	50	6.57	4.12	37.3%
10,000	16	70	7.2	4.58	36.4%
1000	20	30	2657	520	80.4%
1000	20	50	2925	1145	60.9%
1000	20	70	2081	1414	32.1%
10,000	20	30	2433	581	76.1%
10,000	20	50	3078	1532	50.2%
10,000	20	70	3198	1684	47.3%
100,000	20	30	3460	1406	59.4%
100,000	20	50	3068	1595	48.0%
100,000	20	70	2601	1664	36.0%

A significant improvement in relation to the execution time is noticeable, when comparing our proposed algorithm to the *PropIm* algorithm. The results pointed out a run-time gain ranging from 32.1% (1000 objects, 20 attributes and 70% density) to 80.4% (1000 objects, 20 attributes and 30% density), with an average performance gain of 51.6%. In general, the algorithm *ImplicPBDD* has a better performance for contexts with a low range of density.

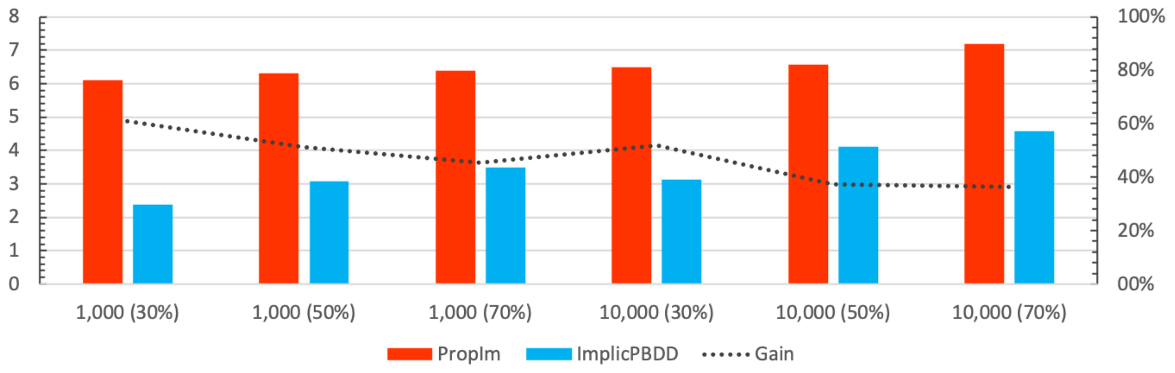


Figure 4. Execution time comparison between the PropIm algorithm and ImplicPBDD for the formal contexts with 16 attributes.

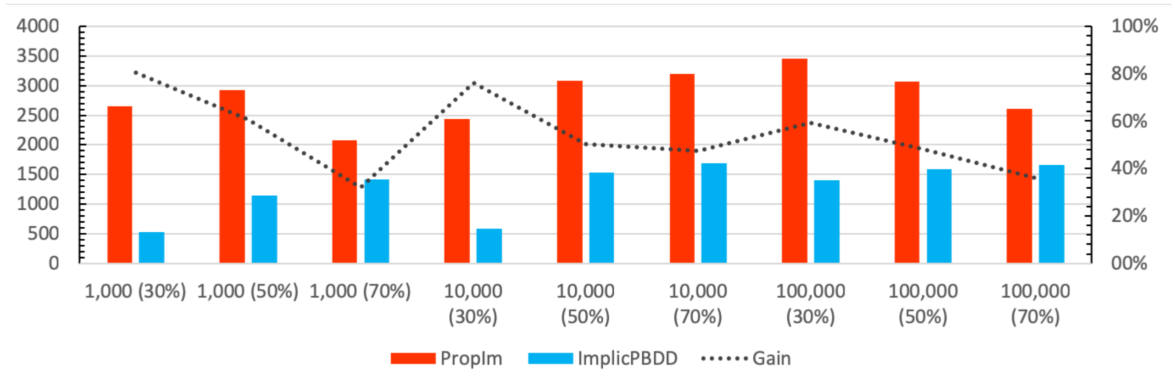


Figure 5. Execution time comparison between the PropIm algorithm and ImplicPBDD for the formal contexts with 20 attributes.

It is also possible to observe, based on the gain lines in Figures 4 and 5, that the variation in the number of objects did not generate a significant impact in the execution time of the evaluated algorithms, since the increase in the number of objects was not proportional to the increase in execution time. In other words, as we can see by the dotted-line (gain), when the amount of objects increased 100 times, the processing time only increased by less than 30%. However, the execution time of both of the evaluated algorithms tends to grow when we increase the quantity of attributes and, mainly, the density of incidence in the formal context.

The hypothesis raised to explain this observation is that the higher the incidence density of the formal contexts, the greater the quantity of attributes will be used as premises in the implications. To verify this hypothesis, we analyzed the impact of the number of assumptions in the execution time of the algorithms.

Scenarios with 30% of density (incidence of attributes in relation to objects) were the ones that presented the least amount of attributes in the premises, which resulted in the least time spent to find the proper implications set. This occurs because that is the best scenario for the heuristic used in the algorithms presented in this article, since when finding a minimum rule, for example, $\{a \rightarrow b\}$, no new rule will be generated anymore using the attribute $\{a\}$ on the left side of the implication.

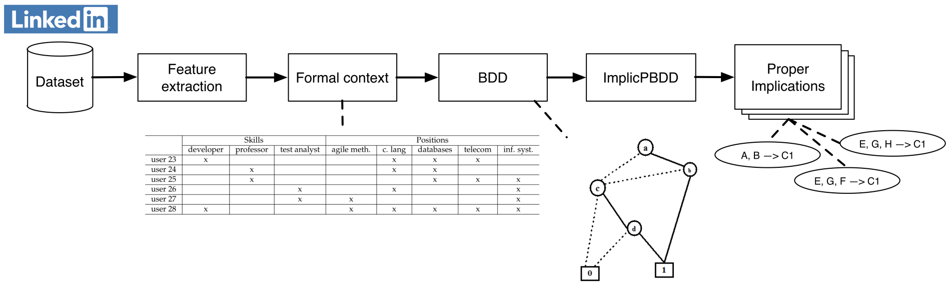
On the other hand, when we analyzed the scenarios with 70% of density, in general, we found a greater amount of attributes in the premises of the implications, which demands a greater computational effort to find the set of rules of implication.

6. Case Study: LinkedIn

The job market is becoming increasingly competitive, because companies are advancing and demanding highly prepared professionals to fill job positions. Thus, people who are looking for a job opportunity may benefit from help to become potential candidates. Such guidance could be build from a knowledge base composed by the set of competences that people have developed in their actual jobs. One type of information source to build the knowledge base are the on-line professional social networks, where the people made their resume available. The LinkedIn is the largest and most popular on-line professional social network with over 500 millions users [37]. The network provides a source of information which can be used to build the knowledge base to help job seekers to improve their resume with the competences necessary to achieve an specific job position.

In this case study, the goal is to analyze and represent professional LinkedIn profiles using FCA. This will be done by building a conceptual model, transforming it into a formal context, scraping and pre-processing the data into formal concepts, then finally extracting the implication sets to be analyzed.

Figure 6 presents the flowchart of the process used to compute the proper implications set using our *ImplicPBDD* algorithm.



	Skills			Positions				
	developer	professor	test analyst	agile meth.	c. lang	databases	telecom	inf. syst.
user 23	x				x	x	x	
user 24		x			x	x		
user 25		x				x	x	x
user 26			x		x			x
user 27			x	x				x
user 28	x			x	x	x	x	x

Figure 6. Flowchart of the proposed approach.

Before building the formal context, it was necessary to define the conceptual model that represents the problem domain. For this purpose, we adopted a *model of competence* [38], where the authors define professionals by their competence in accomplishing a certain purpose. The competence was divided into three dimensions: knowledge, attitudes and skill. Then, following the methodology by [39], the on-line professional resume categories were mapped into the competence model dimensions, as follows: the academic education information was classified into knowledge dimension, the professional experience was linked to the skill dimension, and the user's interaction with the network were linked to the attitudes dimension.

After formally representing the problem, the next step was to collect the LinkedIn users' data. This process was divided into two phases: selecting the initial seeds and collecting the public profiles. It is important to mention that this process was executed from October 2016 to February 2017, and during this period some LinkedIn profiles were publicly available and could be accessed without logging into the network.

The initial seeds was selected randomly following basic definitions such as location and occupation area. An approach named open collection was applied to extract the data of public profiles, because the network did not provide an API (Application Programming Interface) to extract data directly from the server. This process started by accessing the initial seed. In the public profile there is a section denoted "People also viewed"—a list with the ten most similar profiles related to the visited profile [40]. The automated collector visited these ten profiles, and verified which fit into the collection scope. Valid profiles are stored and each one becomes a new seed to extract new links, restarting the scrapping process until it reaches the stop criteria, which was based on the percentage of new extracted profiles. Each iteration checks if 65% of the profiles were already in the database in the last data collection iteration. If the response is "yes", the process stops and restarts with a new seed.

From the variables extracted in the data collection phase and the mapping of the informational categories of LinkedIn proposed by [39], for this paper it was chosen to address only the variables *ability* and *experience* was sufficient to determine the skills needed to hold professional positions.

6.1. Building the Formal Context

To construct the formal context, the values of the variables in skill and experience dimensions were considered as the formal context's attributes. In this case, the formal context consisted of two types of attributes: positions and skills. Finally, each user (LinkedIn professional, whose identification was anonymised during the data collection process), was considered as an object in the formal context. This generated an initial context with 5265 attributes and 1652 objects.

Before proceeding to the stage of extracting proper implications, an exploratory analysis of the attributes was performed. The attribute values were textual, and the analysis identified that the attributes of type *position* were partially standardised, and that there was no standardisation for the attributes of the type *skills*. Therefore, we employed process of transformation and cleaning of the attributes. In the first part of the process, terms with semantic proximity that could be merged were identified, considering the prior knowledge of the domain. This transformation process resulted in an *extended base* [40], composed of specific terms and generic terms. Table 4 shows an example of how the first phase of attribute preprocessing was performed.

Table 4. Table of generic and specific terms.

Generic Term	Specific Termo Specific
Java	Java language JPA JSF
Software developer	Developer Programmer Program developer

In the second part, attributes of the type *skills* were described at a superior level of abstraction, found in the previous phase, resulting in a classification structure. This abstraction allows the reduction of the number of attributes in the formal context, which positively impacts the phases of extraction and selection of implications as the reduced context allows for a quicker computation of the implication sets.

In addition, we expected a gain in the task of selecting implications, related to the quality of the extracted implications. This, because when defining a concise set of attributes, one can obtain a set of smaller implications with a greater statistical significance. The *table of knowledge areas* (Available at: http://www.cnpq.br/documents/10157/186158/TabeladeAreasdoConfecimento.pdf) provided by *CNPq*, was used as a starting point for the elaboration of the classification structure. From this table, terms related to professional skills were categorized within their sub-divisions, and new sub-divisions were created in order to fit all terms. It is also important note that, for this case study, attributes of the type *position* were selected from the ten most frequent positions in the database. The preprocessing resulted in a formal context (Table 5) with 24 attributes of the skills type, ten attributes of the position type and 1269 objects.

Table 5. Part of the formal context.

	Skills				**Positions**			
	Developer	**Professor**	**Test Analyst**	**Agile Meth.**	**c. Lang**	**Databases**	**Telecom**	**Inf. Syst.**
user 23	x				x	x	x	
user 24		x			x	x		
user 25		x				x	x	x
user 26			x		x			x
user 27			x	x				x
user 28	x			x	x	x	x	x

6.2. Rules Obtained in the Formal Context of LinkedIn

This formal context, created based on a sample extracted from the LinkedIn social network, was used in order to test the *ImplicPBDD* algorithm on a real problem. In this case study, the set of proper implications expresses the competences, acquired by the people along their professional careers. Competences are defined as the set of professional skills required to perform an organizational function. The context attributes were divided into two types: *skills*, which compose the set of premises, and *positions*, which are the conclusions of the proper implications. Thus, the context contains as conclusion 10 professions, with a subset of the 24 skills as premises, and all the 1269 objects from the automated data collection (each a profile from a different LinkedIn user).

For the formal context of LinkedIn 430 proper implications were obtained. We did not apply metrics to select and evaluation the implications, which could be a possible future work. The evaluation and selection of implications was carried out by a domain expert.

When a person seeks some orientation to start their professional career or to return to the market, it is important to define the specific competences required for the positions that are sought by that person. This allows a job seeker to start a learning process focused on their professional goals. Table 6 presents three implications that were extracted from LinkedIn's formal context, with the conclusion of *Data Scientist*. As data science is a broad area and the professional's competence depends on the type of project they work on, based on these implications, three examples of different configurations of the minimum skills that a data scientist has were observed.

Note that the intersection between the premises of the implications 2 and 3 would result in *computer architecture* and *user interface*. However, these were not enough to guarantee the expertise of the data scientist.

Table 6. Sample of proper implications obtained from the formal context of LinkedIn for the *Data Scientist* position.

Rule	**Propoer Implications Obtained for *Data Scientist***
1	{algorithms, digital marketing] → [data scientist}
2	{computer architecture, office applications, user interface] → [data scientist}
3	{computer architecture, user experience, user interface] → [data scientist}

It is important to note that the use of the *ImplicPBDD* algorithm in the actual context based on the LinkedIn sample, which contains 24 attributes, 10 conclusions and 1269 objects, allowed to extract all its proper implications, while the *PropIm* algorithm did not return any result within the set time-limit.

7. Conclusions and Future Works

This article contains the proposal of a novel algorithm, *ImplicPBDD*, for extracting proper implications sets with a better performance than the *PropIm* algorithm, in high-dimensional contexts considering the number of objects (instances). After the experiments, we realized that the algorithm obtained substantial speed gains.

The proposed algorithm was faster in all contexts used in the experiments. In contexts with 100,000 objects, which is close to the challenge defined in [20], the *ImplicPBDD* algorithm was able to extract the set of proper implication up to 80% faster compared to the *PropIm* algorithm.

The *ImplicPBDD* algorithm only finds implications with support greater than 0, that is, only rules that are present in the formal context (unlike the algorithm *Impec*). This avoids the generation of rules with support value equal to 0, which are valid but irrelevant in formal contexts.

Considering that we aimed to find proper implications in only a certain number of attributes, that is, it is known *a priori* which attribute is desired as the conclusion of a rule (as presented in the case study for the LinkedIn dataset), the algorithm proposed in this article demonstrated to be an efficient alternative to determine the proper implications.

An important observation of using a Binary Decision Diagram (BDD) as the data structure was that in denser contexts, as more similar objects exists, the context becomes more optimized and performs better.

In order to compute proper implications from a formal context with a greater number of attributes, we suggest, as a future work, a modification in the algorithm to support distributed computing. Note that, as the number of attributes increases, the required computational time increases substantially. However, the workload can be distributed, i.e., each conclusion can be computed independently.

Additionally, we suggest that *ImplicPBDD* is evaluated in contexts of real-world problems with greater number of attributes. Since the high-density contexts presented the smallest gains, we suggest modifications in our heuristics to work with dual formal contexts, that is, converting a high density formal context into low density ones.

real-world problems normally tend to generate low density contexts (sparse). Our experiments showed that our algorithm can be performed in big data environment, with feasible execution time. To illustrate this, for 16 attributes and 10,000 objects, with 30% density, the algorithm needed 3.13 min to find the proper implications set.

Author Contributions: Conceptualization, P.S., L.E.Z. and M.A.S.; Formal analysis, P.H.B.R.; Investigation, P.S.; Methodology, P.S., P.H.B.R. and J.C.V.N.; Resources, P.S.; Software, P.S.; Supervision, L.E.Z. and M.A.S.; Validation, S.D.; Writing - original draft, P.H.B.R., J.C.V.N., L.E.Z. and M.A.S.; Writing - review and editing, P.H.B.R., J.C.V.N., P.S., S.D., L.E.Z. and M.A.S.

Funding: This research received no external funding.

Acknowledgments: The authors acknowledges the financial support of CAPES, FAPEMIG and CNPq. The authors would like to thank PUC-MG researchers for raising important questions and making relevant suggestions that helped improve the quality of the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ganter, B.; Wille, R. *Formal Concept Analysis: Mathematical Foundations*; Springer: Secaucus, NJ, USA, 1997.
2. Ganter, B.; Stumme, G.; Wille, R. *Formal Concept Analysis: Foundations and Applications*; Springer: Berlin, Germany, 2005; Volume 3626.
3. She, S.; Ryssel, U.; Andersen, N.; Wąsowski, A.; Czarnecki, K. Efficient synthesis of feature models. *Inf. Softw. Technol.* **2014**, *56*, 1122–1143. [CrossRef]
4. De Maio, C.; Fenza, G.; Gallo, M.; Loia, V.; Senatore, S. Formal and relational concept analysis for fuzzy-based automatic semantic annotation. *Appl. Intell.* **2014**, *40*, 154–177. [CrossRef]
5. Kumar, C.A.; Radvansky, M.; Annapurna, J. Analysis of a vector space model, latent semantic indexing and formal concept analysis for information retrieval. *Cybern. Inf. Technol.* **2012**, *12*, 34–48.
6. Croitoru, M.; Nir, O.; Simon, M.; Michael, L. Graphical norms via conceptual graphs. *Knowl.-Based Syst.* **2012**, *29*, 31–43. [CrossRef]
7. Poelmans, J.; Elzinga, P.; Dedene, G. Retrieval of criminal trajectories with an FCA-based approach. In Proceedings of the FCAIR 2013 Formal Concept Analysis Meets Information Retrieval Workshop, Co-Located with the 35th European Conference on Information Retrieval (ECIR 2013), Moscow, Russia, 25–27 March 2013; Volume 977, pp. 83–94.

8. Watmough, M. Discovering the Hidden Semantics in Enterprise Resource Planning Data Through Formal Concept Analysis. In *Inter-Cooperative Collective Intelligence: Techniques and Applications*; Xhafa, F., Bessis, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 291–314.
9. Aufaure, M.A.; Le Grand, B. Advances in FCA-based applications for social networks analysis. *IJCSSA* **2013**, *1*, 73–89. [CrossRef]
10. Missaoui, R.; Kuznetsov, S.O.; Obiedkov, S. *Formal Concept Analysis of Social Networks*; Lecture Notes in Social Networks; Springer: Cham, Switzerland, 2017.
11. Missaoui, R.; Abdessalem, T.; Latapy, M. *Trends in Social Network Analysis*; Lecture Notes in Social Networks; Springer: Cham, Switzerland, 2017; pp. 169–187.
12. Rouane-Hacene, M.; Huchard, M.; Napoli, A.; Valtchev, P. Relational concept analysis: Mining concept lattices from multi-relational data. *Ann. Math. Artif. Intell.* **2013**, *67*, 81–108. [CrossRef]
13. Cigarrán, J.; Castellanos, Á.; García-Serrano, A. A step forward for Topic Detection in Twitter: An FCA-based approach. *Expert Syst. Appl.* **2016**, *57*, 21–36. [CrossRef]
14. Negm, E.; AbdelRahman, S.; Bahgat, R. PREFCA: A portal retrieval engine based on formal concept analysis. *Inf. Process. Manag.* **2017**, *53*, 203–222. [CrossRef]
15. Singh, P.K.; Kumar, C.A.; Gani, A. A comprehensive survey on formal concept analysis, its research trends and applications. *Int. J. Appl. Math. Comput. Sci.***2011**. *26*, 495–516. [CrossRef]
16. Taouil, R.; Bastide, Y. Computing Proper Implications. In Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, 30 July–3 August, 2001; pp. 46–61.
17. Ryssel, U.; Distel, F.; Borchmann, D. Fast Algorithms for Implication Using Proper Premises. *Ann. Math. Artif. Intell.* **2014**, *70*, 25–53. [CrossRef]
18. Silva, P.R.C.; Dias, S.M.; Brandão, W.C.; Song, M.A.; Zárate, L.E. *Formal Concept Analysis Applied to Professional Social Networks Analysis*; ICEIS: Setubal, Portugal, 2017; Volume 1, pp. 123–134.
19. Mannila, H.; Räihä, K.J. Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.* **1994**, *12*, 83–99. [CrossRef]
20. Priss, U. Some open problems in formal concept analysis. In Proceedings of the 4th International Conference—ICFCA 2006, Dresden, Germany, 13–17 February 2006.
21. Santos, P.; Neves, J.; Silva, P.; Dias, S.M.; Zárate, L.; Song, M. An Approach to Extract Proper Implications Set from High-dimension Formal Contexts using Binary Decision Diagram. In Proceedings of the 20th International Conference on Enterprise Information Systems, Funchal, Portugal, 21–24 March 2018; Volume 1, pp. 50–57.
22. Bertet, K. Some algorithmical aspects using the canonical direct implicationnal basis. In Proceedings of the Fourth International Conference—CLA 2006, Tunis, Tunisia, 30 October–1November 2006; pp. 101–114.
23. Bertet, K.; Monjardet, B. The multiple facets of the canonical direct unit implicational basis. *Theor. Comput. Sci.* **2001**, *411*, 2155–2166. [CrossRef]
24. Bryant, R.E. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.* **1986**, *35*, 677–691. [CrossRef]
25. Mo, Y.; Xing, L.; Zhong, F.; Pan, Z.; Chen, Z. Choosing a heuristic and root node for edge ordering in BDD-based network reliability analysis. *Reliab. Eng. Syst. Saf.* **2014**, *131*, 83–93. [CrossRef]
26. Rimsa, A.; Zárate, L.E.; Song, M.A. Evaluation of Different BDD Libraries to Extract Concepts in FCA — Perspectives and Limitations. In Proceedings of the 9th International Conference on Computational Science: Part I, Baton Rouge, LA, USA, 25–27 May 2009; pp. 367–376.
27. Burmeister, P. *Formal Concept Analysis with ConImp: Introduction to the Basic Features*; Fachbereich Mathematik, Technische Universität Darmstadt: Darmstadt, Germany, 2003.
28. Qiu, S.; Ming, H.X. Binary decision diagram–based methods for risk assessment of systems subject to propagated failures and parametric uncertainties. *Qual. Reliab. Eng. Int.* **2018**, *34*, 1339–1351. [CrossRef]
29. Vion, J.; Piechowiak, S. From MDD to BDD and Arc consistency. *Constraints* **2018**, *23*, 451–480. [CrossRef]
30. Hirano, D.; Tanaka-Ishii, K.; Finch, A. Extraction of templates from phrases using Sequence Binary Decision Diagrams. *Nat. Lang. Eng.* **2018**, *24*, 763–795. [CrossRef]
31. Castellanos, A.; Cigarrán, J.; García-Serrano, A. Formal concept analysis for topic detection: A clustering quality experimental analysis. *Inf. Syst.* **2017**, *66*, 24–42. [CrossRef]
32. De Maio, C.; Fenza, G.; Loia, V.; Orciuoli, F. Unfolding social content evolution along time and semantics. *Future Gener. Comput. Syst.* **2017**, *66*, 146–159. [CrossRef]

33. Yevtushenko, S. Bdd-based algorithms for the construction of the set of all concepts. In *Foundations and Applications of Conceptual Structures*; Contributions to ICCS; Bulgarian Academy of Sciences: Sofia, Bulgaria, 2002; Volume 2002, pp. 61–73.
34. Neto, S.M.; Zárate, L.E.; Song, M.A.J. Handling high-dimensionality contexts in formal concept analysis via binary decision diagrams. *Inf. Sci.* **2018**, *429*, 361–376. [CrossRef]
35. Salleb, A.; Maazouzi, Z.; Vrain, C. Mining Maximal Frequent Itemsets by a Boolean Based Approach. In Proceedings of the 15th European Conference on Artificial Intelligence, Lyon, France, 21–26 July 2002; pp. 385–389.
36. Lind-Nielsen, J.; Somenzi, F.; Vahidi, A. JavaBDD. 2007. Available online: http://javabdd.sourceforge.net (accessed on 3 September 2018)
37. LinkedIn. About LinkedIn. 2018. Available online: https://about.linkedin.com (accessed on 7 September 2018)
38. Durand, T. Forms of incompetence. In Proceedings of the Fourth International Conference on Competence-Based Management, Oslo, Norway, 18–20 June 1998.
39. Silva, P.R.; Dias, S.M.; Brandão, W.C.; Song, M.A.; Zárate, L.E. Professional Competence Identification Through Formal Concept Analysis. In *Enterprise Information Systems*; Hammoudi, S., Śmiałek, M., Camp, O., Filipe, J., Eds.; Springer: Cham, Switzerland, 2018; pp. 34–56.
40. Chen, X.; Zhou, X.; Scherl, R.; Geller, J. Using an interest ontology for improved support in rule mining. *Data Warehous. Knowl. Discov.* **2003**, *2737*, 320–329.