ORIGINAL RESEARCH

Parallelization of the next Closure algorithm for generating the minimum set of implication rules

Nilander R. M. de Moraes[1], Sérgio M. Dias[2,3], Henrique C. Freitas[1], Luis E. Zárate*[1]

[1] Department of Computer Science, Pontifical Catholic University of Minas Gerais (PUC Minas), Minas Gerais, Brazil
[2] Federal Service of Data Processing (SERPRO), Belo Horizonte, Minas Gerais, Brazil
[3] Department of Computer Science, Federal University of Minas Gerais (UFMG), Minas Gerais, Brazil

Received: December 29, 2015 **Accepted:** February 15, 2016 **Online Published:** March 2, 2016
DOI: 10.5430/air.v5n2p40 **URL:** http://dx.doi.org/10.5430/air.v5n2p40

ABSTRACT

This paper addresses the problem of handling dense contexts of high dimensionality in the number of objects, which is still an open problem in formal concept analysis. The generation of minimal implication basis in contexts with such characteristics is investigated, where the *NextClosure* algorithm is employed in obtaining the rules. Therefore, this work makes use of parallel computing as a means to reduce the prohibitive times observed in scenarios where the input context has high density and high dimensionality. The sequential and parallel versions of the *NextClosure* algorithm applied to generating implications are employed. The experiments show a reduction of approximately 75% in execution time in the contexts of greater size and density, which attests to the viability of the strategy presented in this work.

Key Words: Minimum set of implication rules, Formal concept analysis, Parallel computing

1. INTRODUCTION

Formal concept analysis (FCA) has been considered an important formalism for knowledge representation, extraction and analysis.[1–3] Its formalization was born in 1982 with the work of Wille,[4] who proposed considering each lattice element as a formal concept and the lattice itself as representing a conceptual hierarchy.[5] However, the applicability of FCA is limited by its capacity of dealing with huge data sets.[6,7] FCA induces a potentially high combinatorial complexity and the structures obtained, even from a small dataset, may become prohibitively large. Despite the fact that the worst case of the lattice size ($2^{min(|G|,|M|)}$) is rarely found in practice, the computational cost is still too prohibitive for many applications. There is an enormous interest in the community to present efficient solutions, since this restriction is closely related to an open problem in FCA: handling dense and highly dimensional contexts.[8,9] To address this situation techniques capable of reducing the complexity of constructing lattices,[10] selecting specific concepts[11,12] or even acting directly on the input context,[13,14] have recently been addressed. However, even with the aid of these techniques, obtaining the lattice can be infeasible. In this sense, the use of parallel architectures is shown to be a viable alternative in reducing the execution time.

Proposals and mathematical fundamentals for parallel and distributed solutions have been presented by several authors.[15–17] However, these contributions fail to assess the impact of density on the input context in the final perfor-

*Correspondence: Luis E. Zárate; Email: zarate@pucminas.br; Address: Department of Computer Science, Pontifical Catholic University of Minas Gerais (PUC Minas) - Av. Dom José Gaspar, 500 - Belo Horizonte, Brazil.

mance of the proposal. Furthermore, they lack information on the resources used to obtain the results such as the data structures used, the configuration of the test environment, the number of computers that comprise the cluster, the number of cores involved, among other information.

On the other hand, implication rules drawn from the formal concepts allow the identification of rare patterns in the data, or regularities that occur in a small number of records (objects). Although they are less used for data analysis,[18] the implications are of practical interest, since it can expose innovation that is little known regarding the data in the study.[19] However, as in the generation of lattices, generating the complete set of implications may also present an exponential behavior in the worst case scenario.[20]

Literature presents several algorithms for the acquisition of implication databases.[21–23] Among these algorithms, there is the *NextClosure*[24] algorithm that, as suggested by Guigues and Duquenne,[22] can be employed for the generation of minimal implication bases (also called stem base or *Duquenne-Guigues* basis). For providing the complete, minimal and non-redundant set of implications, the stem base is of special interest for summarizing the set of valid rules. In other words, the stem base corresponds to the smallest number of implications from which all valid implications in a *formal context* can be obtained. Recently, FCA has received attention for representing and analyzing social networks. For example in Jota Resende *et al.*,[25] the *NextClosure* algorithm was used to detect hidden substructures using a minimum set of rules. The paper points out the difficulty in manipulating database of high dimensionality. The authors suggest the development of parallel algorithms in order to improve the computational performance, in order to apply the FCA theory in social networks analysis.

In this work, the behavior of the *NextClosure* algorithm in obtaining the stem base is considered. For such, we implemented a parallel version of Ganter's algorithm for computing the Duquenne–Guigues basis. The sequential and parallel approaches are evaluated and compared to determine in which scenarios which strategy is more suitable than the other, *i.e.* situations in which the gain achieved with parallelism is significant.

Experiments reveal interesting results concerning the impact in the the size and density parameters in obtaining the stem base. It shows that this performance is also associated with the number of rules extracted from the input context. These findings allowed us to identify the situation in which parallelization would provide greater performance. Furthermore, parallelization of the stem base provided a reduction around $3/4$ in execution runtime, compared to the corresponding sequential solution. This demonstrates the efficiency in the use of parallel architectures as a means of reducing the prohibitive times observed in handling large scale contexts.

The remainder of this paper is organized as follows. Section 2 presents the theoretical foundations of FCA. Section 3 shows the work related to this paper. Section 4 exposes the development of our parallel version of the algorithm *NextClosure* and the methodology used in the experiments. In Section 5, the empirical results are presented. Finally, conclusions and future work are presented in Section 6.

2. THEORETICAL REVIEW REGARDING THE FCA

Two fundamental concepts of FCA are *formal contexts* and *formal concepts*. A third important body of knowledge in the context of this work are the *implication rules*. These concepts are discussed in the following subsections.

2.1 Formal context

A formal context $\mathcal{K}(G, M, I)$ consists in two sets G and M, and a binary relation I between these sets. The elements of G are known as objects, while the ones of M are denominated attributes. If an object g contains a relation I with an attribute m, this relation is expressed by gIm or $(g, m) \in I$.

The derivation operator $'$ is used to map a set of objects onto a set of attributes and vice-versa. For a set $A \subseteq G$ of objects, A' is defined as the set of attributes common to the objects in A and mathematically, it is expressed as:

$$A' : \{m \in M \mid gIm \, \forall g \in A\}. \tag{1}$$

In correspondence, for the set $B \subseteq M$ of attributes, B' is defined as the set of objects with all attributes in B. It is formally expressed by

$$B' : \{g \in G \mid gIm \, \forall m \in B\}. \tag{2}$$

2.2 Formal concept

A formal concept of a context $\mathcal{K}(G, M, I)$ consists in an pair (A, B), where the following property is applied:

$$A \subseteq G, \; B \subseteq M, \; A' = B \; and \; B' = A. \tag{3}$$

The object subset A is called *extent*, while the attribute subset B is denoted *intent*.

The derivation operator $'$ can be combined in a pair of composite operators, denoted by $''$, which map a set onto itself. Therefore, for a set $A \subseteq G$, $A \subseteq A''$.

2.3 Implication rules

A context $\mathcal{K}(G, M, I)$ satisfies an implication $Q \rightarrow R$, with $Q, R \subseteq M$, if $\forall g \in G$, gIq for all $q \in$

Q implies gIr for all $r \in R$.

That is, an implication rule between two sets Q and R is valid for a given context, if the set of objects described by attributes in Q is also described by attributes in R. The sets Q and R are called, respectively, premise and conclusion.

$$\frac{}{A \to A}, \frac{A \to B}{A \subset C \to B}, \frac{A \to B, B \subset C \to D}{A \subset C \to D}. \quad (4)$$

The Duquenne-Guigues basis or stem base is a minimum (in the number of implications) subset from which every implication of a formal context can be generated by means of Armstrong's axioms.[22] It is formed by the set of implications of the form $P \to P''$, where P is a pseudo-intent.

Definition 2.1 (Pseudo-intent). A set $P \subseteq M$ is called **pseudo-intent** if $P \neq P''$ and $Q'' \subseteq P$ for all pseudo-intent $Q \subseteq P, Q \neq P$.

Pseudo-intents can be generated by means of Quasi-intents.[20]

Definition 2.2 (Quasi-intent). A set is a **quasi-intent** if, and only if, the set itself is an intent or a pseudo-intent.

Definition 2.3 (Quasi-intent). A set $Q \subseteq M$ is a quasi-intent if, and only if, Q satisfies the following condition: if $P \subset Q$, $P \neq Q$, is a pseudo-intent, then $P'' \subseteq Q$.

3. RELATED WORK

A notable contribution for the formal concept analysis field is presented by Qi *et al.*,[15] where the authors propose a new distributed algorithm for extracting formal concepts through the concept search space partitioning. In that work, the approach for verifying the validity of the subspaces, *i.e.* the subsets which have a higher probability to generate formal concepts, is unprecedented. However, in the work of Qi *et al.*[15] only the mathematical formalism is presented, which makes it under the practical point of view, of low impact, because the empirical results of the proposed approach are not presented.

Fu and Nguifo[17] also present an algorithm for obtaining the concepts from search spaces in a parallel manner. In that proposal, redundant attributes are removed from the input formal context before the step where the partitions are generated, in order to decrease the algorithm effort in extracting the formal concepts. Another important feature to be highlighted on the proposed algorithm by the authors is that it is based on the *NextClosure* algorithm for obtaining in an effective manner the concept set. Moreover, the approach adopted by the authors does not take into account the workload balance, which reveals itself as one of the limiting factors in the employed strategy. In that work, the authors conclude by means of experimental results that due to the use of the *NextClosure* algorithm for obtaining the concepts, the parallel proposal is able to handle input contexts of high dimensionality, but due to the workload mismatch, the algorithm can present quick variations in performance depending on the load factor chosen.

In contrast to the results of the work in Fu and Nguifo,[17] Moraes *et al.*[7] exhibit a strategy for creating formal concepts which takes into consideration workload balancing. In fact, that feature is an advantage over Fu and Nguifo[17] approach because as emphasized by the authors, the execution time can be estimated by the means of empirical experiments.

Similarly, Krajca *et al.*[26] also explore the parallelization algorithms in the process of generating concepts. In fact, the parallel version discussed in that work is based on Vychodil[27] algorithm. In that work the strategy of generating concepts is very similar to Kuznetsov's *CbO* algorithm,[28] differing only in the order of concepts generated.

In Krajca *et al.*,[26] two versions (sequential and parallel) of the same algorithm are presented. The authors make a brief explanation on the technique used for obtaining concepts and attest the feasibility in employing the depth-first search as the strategy for not generating repeated concepts. Moreover, they perform comparative experimental tests between the sequential and the parallel versions as well as with other algorithms in the literature.[24,29,30] However, the authors do not mention what data structures are used in implementing these algorithms.

In turn, Hu *et al.*[16] use a partitioning approach in the input formal context, which differs from the above-mentioned work, since in that strategy the context is either split horizontally or vertically, and as a result of that operation subcontexts are obtained, *i.e.* partial formal contexts. With these subcontexts, the subconcepts are then extracted and united by the means of a specific union operator, in order to find the final set of formal concepts. This strategy has the advantage of considerably reducing the degree of complexity required to extract the set of relationships between objects and attributes. However, that gain is practically discarded in the process of uniting the subconcepts. In fact, at first, splitting the context seems to be the obvious choice to be made in order to reduce the complexity in generating concepts. However, as in generating concepts, uniting them also presents exponential order.

In Valtchev and Duquenne,[31] a divide and conquer strategy is presented for the construction of the concept lattice. In that strategy, the lattice is constructed from partial lattices which,

taken two by two by means of the cartesian product, they form a complete lattice. In that study, the authors present a new technique for identifying invalid nodes, which consists in calculating, along with the Lattice, the set of implication rules from the partial lattices. Valtchev and Duquenne[31] claim that that technique is effective, since it enables the rapid identification of invalid nodes, that is, the cartesian product of partial Reticulates that do not constitute a concept. The authors also claim that the proposed method performs better than the *NextClosure* algorithm, however they do not show that behavior in an experimental manner.

Finally, Li *et al.*[32] in turn present all the mathematical formalism necessary for the effective generation of formal concepts in distributed systems, thereby ensuring the viability of its usage. Although the practical aspects of applying parallel architectures are not discussed.

4. SCALABLE ALGORITHM AND EXPERIMENTAL METHODOLOGY

The formal contexts used to perform the simulations in this work were randomly generated, but with the specification parameters, in order to obtain a test scenario that is the most comprehensive and reliable. Moreover, the minimum, intermediate and maximum densities for formal contexts employed in the experiments were specified, since this parameter is directly related to the performance of the FCA algorithms, as much in the concept generation as in rule acquisition.

The *SCGaz* tool was used to generate the workloads[33] (Available at: `http://www.icei.pucminas.br/projetos/dsrgroup/?wpdmpro=scgaz`). This tool enables the specification of the number of objects and attributes for the context to be generated, as well as the specification of an arbitrary density that is calculated based on the dimensions adopted for the context. At this point, it is important to note that the contexts generated by *SCGaz* are irreducible.

Since the high-performance computing to be achieved when obtaining the implication rules is also an objective of this work, the implementation of both approaches (sequential and parallel versions) for obtaining the implication rules was written in the *C* programming language. Since that, for the parallel approach, we used the *OpenMP* (Available at: `www.openmp.org`) library for accessing the resources for intra-node parallelism (*i.e.* multithreading).

One may also note that the data structures for both approaches adopted were the same. This was done in order to minimize the appearance of eventual distortions that could affect the experimental results, due to the adoption of divergent data structures.

4.1 Generation of synthetic formal contexts

The selection of synthetic contexts justifies itself in the fact that the use of real databases can specialize rather than generalize the result analysis, and require a discretization process, which would imply a further aspect to be considered in the experiments. Thus, for the generation of synthetic contexts, it was decided to select a set of densities that could adequately represent the behavior of the algorithm in several situations. Thus, in the simulations the following densities were employed: minimum, 30%, 50%, 70% and maximum.

Since the contexts generated by *SCGaz* have the characteristic of being irreducible, the minimum and maximum densities can vary according to the dimensionality specified (*i.e.* number of objects and attributes). Therefore, comparing the behavior of two or more contexts of different dimensions in these extremes can become inconvenient, because the density parameter will be different for each input. Thus, to enable comparison between different contexts, we opted the usage of intermediate densities (*i.e.* 30%, 50% and 70%), with a view to the evaluation of the algorithm in relation to the variation of other parameters.

Considering that two or more formal contexts may differ in their incidence matrices even when presenting characteristics in common such as dimensions and densities, for the simulations it was employed samples of size 3 for each synthetic context considered. This was done in order to consider the impact on the performance of the density from both approaches. Moreover, such a choice was also guided by the high execution time that is expected to observe in the simulations with the contexts considered. Table 1 presents the synthetic contexts employed in the experiments, where each line represents a group sample.

Table 1. Synthetic contexts used in the experiments

|M|×|G|	Density (%)				
	Min.	Intermediate			Max.
15×1,000	21.93	30	50	70	78.07
15×5,000	29.88	30	50	70	70.12
15×10,000	34.97	-	50	-	65.03
20×1,000	13.85	30	50	70	86.15
20×5,000	18.42	30	50	70	81.58
20×10,000	21.11	30	50	70	78.88
25×1,000	10.6	30	50	70	89.4
25×5,000	13.62	30	50	70	86.38
25×10,000	14.81	30	50	70	85.19

Note in Table 1 that the set of synthetic contexts considered presents few attributes in relation to the amount of objects. This is justified since the algorithm to obtain the minimal implication basis contains the worst case scenario in the number

of attributes from the input context. Moreover, there is also the fact that this is the situation that occurs most frequent in real application cases for the FCA.

4.2 Metrics for the evaluation of experiments

For the sequential approach, the execution time will be evaluated together with the dimensionality and the density of the input context. With this condition, it is intended to justify the execution time obtained due to these two parameters. As for the parallel approach, the same metrics will be evaluated as in its sequential counterpart. Moreover, the *efficiency* and *scalability* of the algorithm as for its parallelization will also be evaluated.

Table 2 presents a summary of the metrics for the performance evaluation used in this work.

Table 2. Metrics for the performance evaluation

Metric	Expression	Description
Sequential time	t_s	Sequential execution time
Parallel time	t_p	Parallel execution time
# Threads	n	No. of threads of execution
# Cores	p	No. of cores
Speedup	$s(n) = t_s / t_p$	Where n is the no. of threads
Effiency	$E(p) = S(n) / p$	Where $S(n)$ is the speedup

At this point, it is important to note that due to the higher execution time expected, for certain sample groups, a threshold is imposed to this metric. The maximum execution time allowed for the contexts that will be simulated is set to 50 hours. Exceeding this limit, the simulation for the corresponding sample group will be interrupted.

4.3 Parallel algorithm for acquiring the minimal implication basis

Both approaches studied in this work follow the same syntactic structure, differing only on call points of the concurrency access primitives. Therefore, generally, the structure of the algorithm can be summarized in steps below:

(1) Generate the initial set of attributes;
(2) Initialize the list of implication rules;
(3) Acquire the implication rule;
(4) Maintain the rule list;
(5) Generate the next quasi-closed set.

Basically, obtaining the minimal implication basis consists in generating the entire quasi-closed set of a formal context. However, as the minimal basis is composed of pseudo-closed sets, subsets of interest are precisely those that are not closed with respect to the derivation operator. As the closure operator is defined in terms of this input context, one can refer to the pseudo-closed or quasi-closed sets as pseudo-intentions or quasi-intentions, respectively.

Steps 1 and 2 are responsible for initializing the data structures employed by the algorithm, namely the empty attribute subset (empty intent) and the empty list of implication rules. The remaining outline steps are included within a loop, where the stop condition does not occur until the current subset (of quasi-intents) is equal to the full set of attributes.

At each iteration in the algorithm, a new subset is generated and evaluated. Thus, step 3 deals with obtaining probable rules. They are probable because, as mentioned above, the minimal implication basis is defined in terms of pseudo-intentions, that is, non-closed quasi-intentions in relation to the derivation operator. Therefore, at the time when the quasi-intent is closed, it will not result in a new implication rule.

Thus, if a pseudo-intent is generated, then in step 4, the new subset is added to the list of rules. Again, this structure is maintained, due to the recursive nature of the algorithm, since the minimal basis is defined in terms of itself, this structure is used as feedback in the next step.

Finally, in step 5, the next quasi-intent is generated. In this step, the list of rules held in the previous step as well as the current quasi-intent are employed for generating the next subset. Naturally, the generation of subsets occurs in lexicographical order, since for generating these, the algorithm *NextClosure* is used.

4.3.1 *Definition of the load balancing criteria*

Since the present work deals with the parallelization of an algorithm, the problem of specifying the ideal size for workloads arises naturally. In fact, parallelization of FCA algorithms is a recent topic and still under discussion,[34–38] however specifying the ideal size for partitions is a point that is still open.

It is known that the size employed for the workloads has a relevant impact on the performance of parallel/distributed algorithms. Thus, in order to minimize possible distortions that this parameter may inflict on the results of this work, the size specification for the workloads takes into account their balance. It is important to mention that, in this work, balancing is defined as the equitable grain size (*i.e.* amount of objects, attributes and rules) for each *thread*.

Since performance is closely related to dimensionality, the density and the number of rules generated, the efforts were put in the parallelization of the points where these parameters occur. Therefore, the grains are defined in terms of the quantity of objects, attributes and rules. Consequently, it is expected that the performance observed with the parallelization is influenced by the granularity of these parameters.

Despite the selected balancing criteria not being guaranteed as the best, it is expected that the metrics and analysis exposure of the experimental results can assist related work in the calculating the ideal size of this parameter.

4.3.2 *Implementation of the scalable algorithm*

In consonance with what has been exposed previously, the parallel implementation of the algorithm for obtaining the minimal implication basis takes into consideration, mainly, the equitable division of the workloads. This characteristic was reached with the aid of the primitives available in the *OpenMP* library.

It can also be noted that the structural changes were only performed when strictly necessary in order to reach a parallel behavior. In this manner, the greatest part of the parallelization effort was concentrated in the identification and inclusion of *OpenMP* directives in the parts where a parallelization was possible.

Also, since the largest part of the algorithm logic for obtaining the minimal implication basis concentrates in operations over sets and in the usage of loop structures, the efforts for implementing the parallel approach were focused on these structures.

Therefore, in order to guarantee the workload balance, in the parallelization of the loops the `omp for` directive was employed. This directive enables the parallel execution of loop structures, dividing the workload amongst the *thread* team, following a previously defined scheduling scheme. And, since it is intended to employ workloads of equitable size, along with this directive, the clause `static` was used as the scheduling strategy.

Regarding the implementation of data structures, since much of the operations is concentrated in operations over sets, a representation at the *bit* level for these sets was employed. This was done in order to reduce the space required for storage of such structures and for a higher performance in operations over sets.

Algorithm 1 DUQUENNE-GUIGUES minimal implcation basis

Input: Formal Context $\mathcal{K}(G, M, I)$
Output: The stem base $\mathcal{L}$
$\mathcal{L} \leftarrow \emptyset \quad A \leftarrow \emptyset$
while $A \neq M$ **do**
  $B \leftarrow$ `double_prime(`$\mathcal{K}$`(G,M,I), A)` **if** $A \neq B$ **then**
    $obj \leftarrow$ `extent(`$\mathcal{K}$`(G,M,I),` $A \cup B$`)` `supp(`i`)` $\leftarrow |obj| \,/\, |G| \quad \mathcal{L} \leftarrow \mathcal{L} \cup \{A \rightarrow B \setminus A\}$
  end
  $A \leftarrow$ `next_closure(`A`, lin_closure)`
end
return $\mathcal{L}$

Algorithm 2 Double prime derivation operator

Input: Formal Context $\mathcal{K}(G, M, I)$, subset $A \subseteq M$
Output: Closure of A
$B \leftarrow$ `extent(`$\mathcal{K}$`(G,M,I), A)` $\quad C \leftarrow$ `intent(`$\mathcal{K}$`(G,M,I), B)` **return** C

Algorithm 3 Next closure

Input: A closure operator $X \rightarrow X''$ over a set M
Output: A is replaced with the next closed-set in lexicographic order
$i \leftarrow$ the greatest element of $M \quad i \leftarrow$ `succ(`i`)` $\quad success = F$
repeat
  $i \leftarrow$ `pred(`i`)` **if** $i \notin A$ **then**
    $A \leftarrow A \cup \{i\} \quad B \leftarrow A''$ **if** $B \setminus A$ *does not contain an element* $< i$ **then**
      $A \leftarrow B \quad success \leftarrow V$
    end
  else
    $A \leftarrow A \setminus \{i\}$
  end
until *success* ***ou*** i = *smallest element of* M
return A

Finally, the set of parallel algorithms that make up the parallel approach for obtaining the stem base is expressed through the Algorithms 1-6. The parallelized parts are defined between `Begin Parallel Region` and `End` blocks. It is also noteworthy that since the workload division is performed by the *OpenMP* library, the calculation of specific work threasholds of each *thread* was omitted in the algorithms. Therefore, in the sections where variables `workload_st`

and `workload_end` occur, it should be understood that these are automatically calculated, in an equal manner.

The Algorithms 4 and 5 are the derivative operations over the input context, namely: the concept extent and intent. In the first algorithm, grains are defined in terms of the amount of context objects and the size of the attribute subset $A \subseteq M$. As for the latter, the grains correspond to the amount of context attributes and the size of the object subset $B \subseteq G$.

Algorithm 4 Extent concept excent

Input: Formal Context $\mathcal{K}(G, M, I)$, subset $A \subseteq M$
Output: The set of objects with all elements from A
$NUM\_THREADS \leftarrow$ `num_threads()` $A' \leftarrow \{0, 0, \ldots, 0_{G-2}, 0_{G-1}\}$
for $i \leftarrow 0$ ***to*** $NUM\_THREADS$ **do**
  $objects[i] \leftarrow \{0, 0, \ldots, 0_{G-2}, 0_{G-1}\}$
end
Begin Paralell Region
  $tid \leftarrow$ `thread_id()` `/* Values in the interval` $0 \leq i < |G|$.
  for $i \leftarrow workload\_st$ ***to*** $workload\_end$ **do**
    $all \leftarrow T$ **foreach** $j \in A$ **do**
      if $(i, j) \notin I$ **then**
        $all \leftarrow F$
      end
    end
    if $all = T$ **then**
      $objects[tid] \leftarrow objects[tid] \cup \{i\}$
    end
  end
  `/* Synchronization point: only the thread with id = 0 (zero) executes the following block.`
  if $tid = 0$ **then**
    for $i \leftarrow 0$ ***to*** $NUM\_THREADS$ **do**
      $A' \leftarrow A' \cup objects[tid]$
    end
  end
End
return A'

Algorithm 5 Intent concept intent

Input: Formal Context $\mathcal{K}(G, M, I)$, subset $B \subseteq G$
Output: The set of attributes with all elements from B
$NUM\_THREADS \leftarrow$ `num_threads()` $B' \leftarrow \{0, 0, \ldots, 0_{M-2}, 0_{M-1}\}$
for $i \leftarrow 0$ ***to*** $NUM\_THREADS$ **do**
  $attributes[i] \leftarrow \{0, 0, \ldots, 0_{M-2}, 0_{M-1}\}$
end
Begin Paralell Region
  $tid \leftarrow$ `thread_id()` `/* Values in the interval` $0 \leq i < |M|$.
  for $i \leftarrow workload\_st$ ***to*** $workload\_end$ **do**
    $all \leftarrow T$ **foreach** $j \in B$ **do**
      if $(j, i) \notin I$ **then**
        $all \leftarrow F$
      end
    end
    if $all = T$ **then**
      $attributes[tid] \leftarrow attributes[tid] \cup \{i\}$
    end
  end
  `/* Synchronization point: only the thread with id = 0 (zero) executes the following block.`
  if $tid = 0$ **then**
    for $i \leftarrow 0$ ***to*** $NUM\_THREADS$ **do**
      $B' \leftarrow B' \cup attributes[tid]$
    end
  end
End
return B'

At line 7 of these two algorithms, the `tid` variable corresponds to the identifier of the *thread* in execution. More especifically, the pseudo-code between `Begin Parallel Region` and `End` blocks will be executed in parallel by a team of execution *threads*. Each *thread* belonging to a team receives an unique identifier, so as to facilitate the merge of

the partial results in line 21.

Again, the calculation of the specific workload limits of each execution *thread* is defined in an automatic manner and calculated according to the domain limits of the loop. In the case of the Algorithm 4, limits in the interval $0 \leq i < |G|$ are calculated. For the Algorithm 5, limits are between $0 \leq i < |M|$.

The Algorithm 6, in turn, corresponds to the parallel version of the closure operator for quasi-closed sets. In this algorithm, the grains of the first parallel block, between lines 1 and 11, are defined in terms of the number of context attributes and the amount of rules (*i.e.* obtained up to the moment). In this block, the work ranges of the *thread* team are defined within the range $1 \leq x \leq |M|$. As for the second parallel block (lines $18 - 31$), the granularity is defined by the size of the subsets `T` and `new_closure`.

Algorithm 6 Linclosure

Input: A list $\mathcal{L} \leftarrow [\mathcal{L}[1], \ldots, \mathcal{L}[n]]$ of implications in M and a set $X \subseteq M$
Output: The closure $\mathcal{L}(X)$ of X
Begin Paralell Region
  /* Values in the interval $0 \leq x < |M|$.
  for $x \leftarrow workload\_st$ *to* $workload\_end$ **do**
    $avoid[x] \leftarrow \{1, \ldots, n\}$ **for** $i \leftarrow 1$ *to* n **do**
      $A \rightarrow B \leftarrow \mathcal{L}[i]$ **if** $x \in A$ **then**
        $avoid[x] \leftarrow avoid[x] \setminus \{i\}$
      end
    end
  end
End
$used\_imps \leftarrow \emptyset$ $old\_closure \leftarrow \{-1\}$ $new\_closure \leftarrow X$
while $new\_closure \neq old\_closure$ **do**
  $old\_closure \leftarrow new\_closure$ $T \leftarrow M \setminus new\_closure$
  Begin Paralell Region
    Parallel Section
      $tmp1 \leftarrow \{1, 1, \ldots, 1_{M-1}, 1_M\}$ **foreach** $x \in T$ **do**
        $tmp1 \leftarrow tmp1 \cap avoid[x]$
      end
    End
    Parallel Section
      $tmp2 \leftarrow \{0, 0, \ldots, 0_{M-1}, 0_M\}$ **foreach** $x \in new\_closure$ **do**
        $tmp2 \leftarrow tmp2 \cup avoid[x]$
      end
    End
  End
  $usable\_imps \leftarrow tmp1 \cap tmp2$ $use\_now\_imps \leftarrow usable\_imps \setminus used\_imps$ $used\_imps \leftarrow usable\_imps$
  foreach $i \in use\_now\_imps$ *com* $A \rightarrow B \leftarrow \mathcal{L}[i]$ **do**
    $new\_closure \leftarrow new\_closure \cup B$
  end
end
$\mathcal{L}(x) \leftarrow new\_closure$ **return** $\mathcal{L}(x)$

Unlike the other parallel regions, the second parallel region presents section parallelism. In the *OpenMP* implementation, this behavior is described by the usage `omp section` directives, which correspond to a non-iterative type of parallelism, where each parallel section is assigned to an execution *thread*. According to the *OpenMP* documentation, it may occur in the same *thread* the execution of more than one parallel section, if its implementation permits. This type of parallelism has been adopted for this block due to the low granularity presented in this region.

Finally, although it is possible to parallelize the loop, it has not been parallelized, due to its low granularity (All implementation of the algorithm for obtaining the minimal implication basis are available at: `http://www.icei.pucminas.br/projetos/dsrgroup/?wpdmpro=dgbasis`.

Figure 1 shows the mechanics of the algorithms presented.

5. Result analysis

This section presents the results of the experiments conducted for the set of formal contexts considered.

5.1 Experimental results

The experiments were done in an Intel Xeon E5430 2.6Ghz / 8 cores, with 8Gb of RAM and operating system RHEL v4.1.2-44. That said, below, in Table 3a, the performance of the sequential approach is presented. For each synthetic context, the execution times obtained with density variation are presented; hyphens denote that the simulation was aborted due to the 50h threshold defined in the methodology. Some contexts that exceeded this threshold are shown and indicated with asterisks.

In general, the acquisition of the minimal basis presents the worst case when the input context density is 50%, which is notedly different from the observed behavior when obtaining

formal concepts, where the worst case scenario is achieved when the entry presents maximum density. In particular, for the context containing 15 attributes and 5,000 objects, this behavior is not checked. In part, because it is a relatively small context, but mainly due to the need of a number of iterations (*i.e.* quasi-intents) very close to the contexts with densities close to 70% and 70.12% (see Table 1).

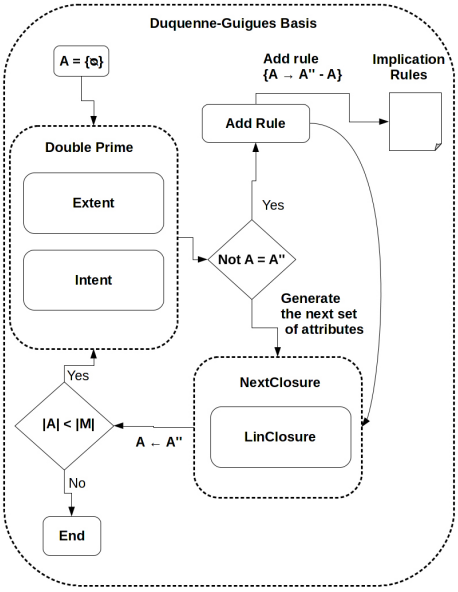


Figure 1. Flow chart of the algorithms for generating the Duquenne-Guigues basis of implications

Table 3. Average for each sample group

(a) Sequential time (in seconds)

|M| × |G|	Density (%) min.	30	50	70	max.
	time (in seconds)				
15×1,000	1.35	3.81	12.50	9.33	9.31
15×5,000	27.35	24.25	41.58	43.40	43.39
15×10,000	69.01	-	82.56	-	85.57
20×1,000	4.07	122.22	2,350.07	1,344.65	408.35
20×5,000	49.41	836.65	7,360.39	1,851.14	1,897.59
20×10,000	226.77	1130.12	9,382.43	3,677.29	3,711.39
25×1,000	8.40	2149.90	671,753.12	-	-
25×5,000	211.48	54,957.18	-	-	-
25×10,000	254.07	165,507.72	-	-	-

(b) No. of rules obtained

|M| × |G|	Density (%) min.	30	50	70	max.
	# rules (n)				
15×1,000	952	1,515	1,389	7	0
15×5,000	4,948	4,324	594	0	0
15×10,000	6,353	-	208	-	0
20×1,000	1,466	7,830	22,820	3,972	0
20×5,000	4,953	20,614	33,172	53	0
20×10,000	11,702	19,693	31,385	5	0
25×1,000	1,724	28,074	-	-	-
25×5,000	10,287	94,810	-	-	-
25×10,000	8,848	137,028	-	-	-

Another point to be noted, with respect to contexts $15\times5,000$ with minimum and 30% densities. In both cases, we find that the context with minimum density (*i.e.* 29.88%) showed a superior execution time in relation to its correspondent by 30%. This is justified by the density proximity. Besides, of course, that the entry with minimum density has provided a higher quantity of as many quasi-intentions as of rules, which suggests that, as in dealing with formal concepts, the context incidence relationship has a significant weight in obtaining minimal implication basis.

Still in Table 3a, we note that for the contexts with more than 20 attributes, the explosion in execution time is clear due to the increased input density. It is also possible to observe a certain trend towards execution time obtained by the variation of context density. Of course, it can be stated that after a density of 50%, there is a decline in execution time. Table 3b helps to interpret such behavior.

Comparing the data in Table 3b with the execution times shown in Table 3a, it is observed that the algorithm performance is closely related to the number of rules obtained. In general, as the density increases, that is, from the minimum possible up to 50%, the number of rules also increases. In contrast, after 50%, there is a decline in the number of rules. In particular, when the maximum density was employed, for all contexts used, the number of rules was always null. This occurs, since in this case all subsets generated are closed in relation to the derivation operator. However, the subsets of interest for the construction of the stem base are precisely those that are not closed (see Algorithm 1).

Despite the fact that the number of rules affects the algorithm performance, this factor does not provide a longer execution time. An example of this situation can be observed in the context $15 \times 5,000$ with minimum density. In this case, the context with such configuration presented the largest number of rules, however, this characteristic did not result in the longest time taken. This shows that, although the performance is also dependant on the number of rules, this measure does not contribute alone to the increase in execution time. In fact, as it can be seen in Figure 2, the acquisition of the set of rules is also conditional to the number of quasi-intent subsets obtained from the input context.

In the graph of Figure 2 it is easy to see that the number of subsets (*i.e.* quasi-intents) grows with the increase of the context density. In fact, this number is exponential in the number of input attributes, where in the case where the context presents maximum density, the number of corresponding quasi-intents reaches its largest value (*i.e.* $2^{|M|}$). Another important consideration refers to the relationship between the number of subsets generated and of objects. As the re-

sults show, the number of objects has a linear impact on the number of quasi-intents generated.

A closer look at the results from Table 3b and the graph from Figure 2 reveals that the increase in the number of objects in the context causes a decline in the number of rules to be steeper, as the density approaches the maximum value. This is expected, since the increase in the number of objects makes the rule acquisition process more difficult, either by increasing the number of subsets evaluated or even by the effort required to perform derivative operations.

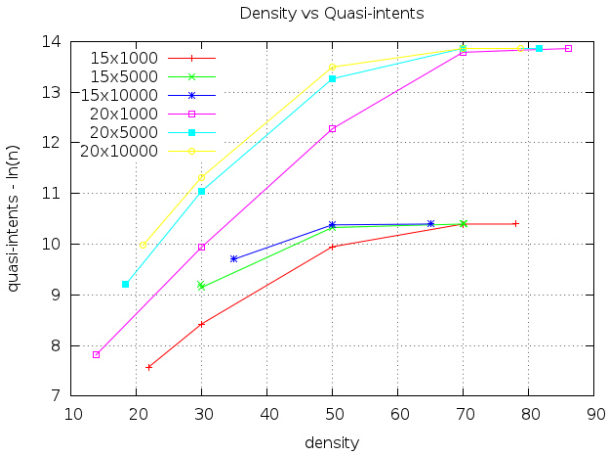


Figure 2. Variation of density of the input context and the corresponding no. of quasi-intents

Table 4a shows the standard deviation of the execution time of the sample groups used in the simulations. In this Table, the results explain the impact of the context incedence relation over the algorithm performance. Furthermore, it shows that although contexts belonging to the same sample group are similar in size and density, they still refer to different contexts. For example, taking the contexts $20 \times 5,000$ to $25 \times 10,000$ with 50% and 30% density, respectively. The average $\mu(x)$, for a confidence level of 90%, using the t-student distribution, the confidence interval for the first context is $7,119.2 \leq \mu(x) \leq 7,600.8$; for the second, the range is $149,617.88 \leq \mu(x) \leq 181,397.56$.

In Table 4a, it is observed that the increase in the standard deviation is closely related to the increase in the number of input attributes. This is clear for contexts over 20 attributes. Furthermore, with the aid of Table 4b, which presents a dispersion degree in the amount of rules for each sample group, it is possible to notice that the execution time is also related to the number of rules generated, thus corroborating with the observations raised about the input density.

Table 4. Standard deviation for each sample group

(a) Execution time (in seconds)

|M|×|G|	Density (%)				
	min.	30	50	70	max.
	time (in seconds)				
15×1,000	0.02	0.13	0.29	0.19	0.02
15×5,000	0.01	0.12	0.15	0.13	0.13
15×10,000	0.29	-	0.06	-	0.05
20×1,000	0.08	2.50	40.44	184.27	1.20
20×5,000	0.14	7.90	227.53	7.06	1.51
20×10,000	0.32	19.24	589.28	4.72	9.58
25×1,000	0.10	51.52	-	-	-
25×5,000	0.78	515.57	-	-	-
25×10,000	0.92	15,013.92	-	-	-

(b) No. of rules obtained

|M|×|G|	Density (%)				
	min.	30	50	70	max.
	# rules (n)				
15×1,000	2	26	69	2	0
15×5,000	0	24	65	0	0
15×10,000	4	-	33	-	0
20×1,000	7	110	479	903	0
20×5,000	30	317	1,784	33	0
20×10,000	11	60	3,123	3	0
25×1,000	18	462	-	-	-
25×5,000	6	136	-	-	-
25×10,000	47	1,985	-	-	-

In Tables 5 and 6 are shown the speedup and efficiency of algorithms 2 and 6 when applied 8 logical cores. As the paralelization effort was focused in these two rotines, it was possible to measure the contribution of these portions in the perfomance of the proposal.

It can be seen in Table 5 that the speedup of DOUBLEPRIME is related with the number of quasi-intents of the input context. It also could be noted that the increase of density provides an improvement of the performance of this rotine. On the other hand, Table 6 shows that the greater the number of pseudo-intentions generated, the greater the speedup of LINCLOSURE.

Table 7 lists the times obtained with parallelization of the minimal implication basis. It is shown for each synthetic context, the execution times obtained with the variation of the density as well as the number of *threads*. Also in this Table, hyphens denote the input context with that density and number of *threads* was not simulated, due to the threshold defined in methodology (see Section 4.2).

Table 7, it is noted that, as in its corresponding sequential one, the parallel time also depends on the density of the input context. A comparison of execution times for the parallel and the sequential versions, shown in Table 3a, also reveals that the performance is only meaningful for contexts with 15

attributes when 4 or more execution *threads* are employed.

Table 5. Speedup and effiency achieved in DoublePrime with 8 threads of execution

|M| x |G|	Density (%)				
	min.	30	50	70	max.
	Speedup / Efficiency				
15×1,000	0.901 / 0.113	1.079 / 0.135	1.389 / 0.174	1.522 / 0.19	1.557 / 0.195
15×5,000	1.128 / 0.141	1.164 / 0.146	1.599 / 0.2	1.842 / 0.23	1.836 / 0.229
15×10,000	1.229 / 0.154	-	1.491 / 0.186	-	1.835 / 0.229
20×1,000	0.932 / 0.116	1.333 / 0.167	1.589 / 0.199	1.776 / 0.222	1.644 / 0.205
20×5,000	1.209 / 0.151	1.554 / 0.194	1.858 / 0.232	1.813 / 0.227	2.2 / 0.275
20×10,000	1.297 / 0.162	1.626 / 0.203	1.889 / 0.236	1.802 / 0.227	1.66 / 0.208
25×1,000	1.014 / 0.127	1.554 / 0.194	1.836 / 0.23	-	-
25×5,000	1.293 / 0.162	1.825 / 0.228	-	-	-
25×10,000	1.432 / 0.179	1.894 / 0.237	-	-	-

Table 6. Speedup and efficiency achieved LinClosure with 8 threads of execution

|M|×|G|	Density (%)				
	min.	30	50	70	max.
	Speedup / Efficiency				
15×1,000	1.805 / 0.226	1.907 / 0.238	2.519 / 0.315	0.096 / 0.012	0.075 / 0.009
15×5,000	3.426 / 0.428	3.152 / 0.394	1.315 / 0.164	0.073 / 0.009	0.079 / 0.01
15×10,000	3.667 / 0.458	-	0.414 / 0.052	-	0.08 / 0.01
20×1,000	2.441 / 0.305	3.951 / 0.494	4.66 / 0.582	4.186 / 0.523	0.088 / 0.011
20×5,000	3.427 / 0.428	4.423 / 0.553	5.007 / 0.626	0.236 / 0.029	0.087 / 0.011
20×10,000	3.929 / 0.491	4.373 / 0.547	5.115 / 0.639	0.093 / 0.012	0.1 / 0.013
25×1,000	2.744 / 0.343	4.398 / 0.55	3.662 / 0.458	-	-
25×5,000	3.774 / 0.472	4.03 / 0.504	-	-	-
25×10,000	3.855 / 0.482	3.845 / 0.481	-	-	-

Table 7. Average parallel time (in seconds) for the construction of the stem base of contexts from Table 1

Context	# threads	Density (%)				
		min.	30	50	70	max.
		time (in seconds)				
15×1,000		1.33	3.30	10.86	9.79	9.84
15×5,000		24.32	22.28	40.45	44.92	46.36
15×10,000		66.28	-	86.01	-	77.81
20×1,000		3.25	73.90	1,313.53	888.01	403.27
20×5,000	2	38.25	508.71	4,306.58	1,660.34	1,599.46
20×10,000		171.06	745.97	5,954.80	3,228.70	3,409.13
25×1,000		6.24	1,241.77	-	-	-
25×5,000		144.78	35,016.49	-	-	-
25×10,000		180.25	101,669.18	-	-	-
15×1,000		1.43	2.43	7.75	7.03	7.11
15×5,000		16.68	15.47	28.49	29.56	30.53
15×10,000		43.80	-	56.76	-	56.95
20×1,000		2.32	45.82	756.00	554.59	261.94
20×5,000	4	24.71	307.21	2,473.64	1,048.54	1,025.64
20×10,000		107.71	454.05	3,450.38	2,159.62	2,322.41
25×1,000		4.32	749.47	-	-	-
25×5,000		90.68	24,259.85	-	-	-
25×10,000		115.76	80,372.78	-	-	-
15×1,000		0.94	2.13	6.82	7.33	7.76
15×5,000		15.19	13.81	26.60	24.94	25.16
15×10,000		41.36	-	54.95	-	50.05
20×1,000		2.60	35.01	537.62	433.82	297.05
20×5,000	8	21.39	235.88	1,805.16	1,066.64	913.23
20×10,000		90.57	364.94	2,656.57	2,087.76	2,277.62
25×1,000		3.51	505.03	183,897.60*	-	-
25×5,000		68.14	13,872.32	-	71,873.45	18,403.50
25×10,000		90.78	43,701.65	-	-	-
15×1,000		4.17	8.30	25.92	33.18	33.48
15×5,000		30.15	27.70	58.52	58.39	57.06
15×10,000		68.54	-	101.92	-	91.57
20×1,000		8.08	69.32	771.35	1,180.88	1,087.44
20×5,000	16	42.44	340.78	2,456.46	1,981.20	2,048.81
20×10,000		141.62	523.04	3,611.47	3,393.92	3,385.14
25×1,000		12.16	647.78	-	-	-
25×5,000		110.00	14,939.25	-	-	-
25×10,000		146.39	45,030.98	-	-	-

In fact, not only the performance of the sequential implementation, but also from the parallel version depend on the context density. Thus, in the carried simulations, it was observed that the maximum *speedup* (Maximum *Speedup* compared to the other contexts of the same dimension, but with different densities) is obtained when the incidence context matrix presents sparse density and a relatively high number of rules, as illustrated by the data in Table 8, which shows the gain obtained with 8 execution *threads*.

In Table 8, considering the finding about acquiring the maximum performance of the algorithm, it is possible to note that the *speedup* reaches its peak when the input context presents a high number of rules and sparse density. This is corroborated by the contexts $15 \times 5,000$ with minimum density, $25 \times 1,000$ and $25 \times 5,000$ with 30% density and $15 \times 1,000$, $20 \times 1,000$, $20 \times 5,000$ and $20 \times 10,000$ with 50% density. In such cases, as shown in Table 3b, the contexts considered showed high amounts of rules, compared with their counterparts in other densities.

Table 8. Speedup achieved with 8 threads and 8 cores

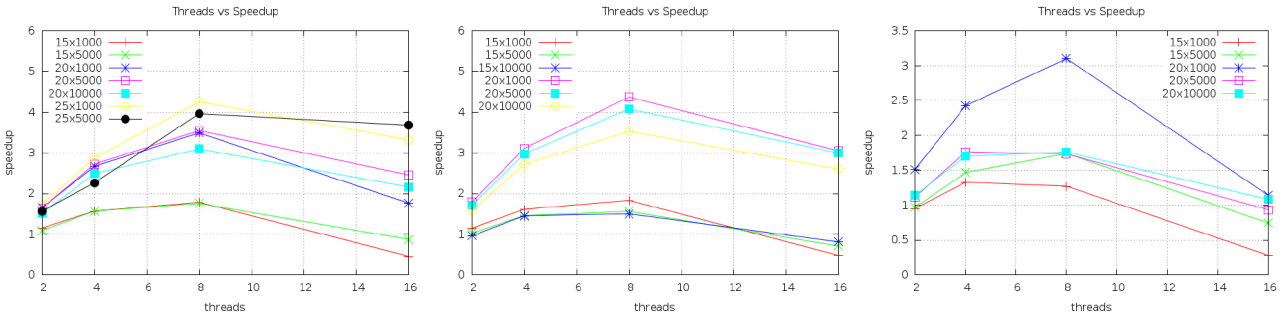
|M|×|G|	Density (%)				
	min.	30	50	70	max.
	Speedup				
15×1,000	1.439	1.785	1.832	1.273	1.20
15×5,000	1.801	1.756	1.563	1.740	1.724
15×10,000	1.669	-	1.502	-	1.710
20×1,000	1.565	3.491	4.371	3.10	1.375
20×5,000	2.311	3.547	4.077	1.735	2.078
20×10,000	2.504	3.097	3.532	1.761	1.630
25×1,000	2.397	4.257	3.653	-	-
25×5,000	3.103	3.962	-	-	-
25×10,000	2.799	3.787	-	-	-

Reinforcing these findings, it can also be seen that the contexts that presented results (synthetic contexts that showed the execution time below the threshold defined.) with 50% density (*i.e.* $15 \times 5,000$ and $15 \times 10,000$), but resulted in a smaller number of rules that obtained lower *speedup*. This is in agreement with the previous claim about the performance of the parallel approach. In fact, as the grain size in the parallel approach is defined in terms of the quantity of objects, attributes and rules, the low index of rules in this case incurred in finer grains and, hence, a lower performance.

Another point that stands out is the result obtained for the context $15 \times 10,000$ with minimum density. Unlike other sparse contexts which produced many rules, this context, with 8 *threads*, presented lower *speedup* than the context with maximum density, which generated less rules. However, this behavior is due to the number of *threads* used in this simulation, since in the scenario where 4 execution *threads* were employed, such input context showed higher performance than the others. More specifically, in this situation,

the simulations with contexts $15 \times 10,000$ with minimal, 50% and maximum densities presented respectively, *speedup* of 1.575, 1.455 and 1.502.

The graphs from Figure 3 present the performance of the parallel approach with the increase in the number of execution *threads* for synthetic contexts with density of 30%, 50% and 70%, respectively. The graphs clearly demonstrate the scalability achieved with the parallelization of the minimal implication basis in obtaining the set of rules. It is shown only the contexts that presented an execution time lower or equal to the threshold defined in the methodology.



(a) Density of 30% **(b)** Density of 50% **(c)** Density of 70%

Figure 3. Speedup obtained with the increase of the no. of threads of execution

In Figure 3a, it is observed that the maximum *speedup* is obtained when the number of *threads* is equal to 8. It should also be noted that the performance achieved is closely related to the size of grains. More specifically, contexts with a higher amount of attributes which also generated more rules obtained higher *speedup*.

However, it is noted that there is a loss of performance due to the increase in the number of objects, for the contexts which presented a high number of rules. This behavior can be clearly observed in contexts with 15, 20 and 25 attributes, when applied to 8 execution *threads*. This is due to the extension calculation in line 7 in the Algorithm 1, where the support (*i.e.* the number of objects involved in the implication) of the obtained rule is calculated. Since the extension calculation deals with objects, the higher this parameter, the greater the effort involved in such operation.

The decline in performance due to the increase in the number of objects can also be seen in Figure 3b, which shows the scalability of parallel approach for contexts with 50% density. This behavior is presented only for contexts with 20 attributes because, as mentioned above, contexts with 15 attributes showed a reduction in the number of rules with an increase in the amount of objects (see Table 3b).

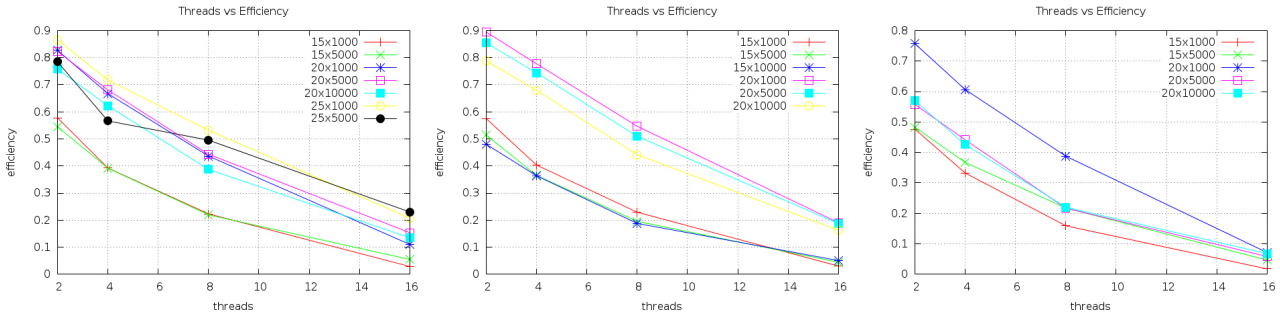
Figure 3c, in turn, shows the scalability for contexts with 70% density. In this graph the relationship between the *speedup* and the number of rules is clear, where the $20 \times 1,000$ context, presenting a higher number of rules in comparison to the other contexts, obtained the highest performance.

In the graph from Figure 3c, it is also noted that the maximum *speedup* is achieved with 4 and 8 execution *threads*. In the situation which 4 *threads* were employed, the contexts $15 \times 1,000$ and $20 \times 5,000$ obtained better results. On the other hand, when 8 *threads* were employed, these contexts showed a slight decline in the *speedup*, while the other contexts showed an increase in performance. Also this behavior is justified by the grain size. Since few rules were generated in this scenario, the performance of the parallel approach was due to the parallelism achieved in the Algorithm 2.

It is also important to note the behavior of the parallel approach when 16 execution *threads* were used, as shown in graphs of Figures 3a, 3b and 3c. In such situation, for all densities considered, there was a reduction in the gain obtained. This behavior is justified, since this scenario has incurred greater competition between *threads*, once that the number of available cores was 8. It can also be seen that some contexts, in this scenario, presented worse results, compared to the sequential approach, with parallelization (*i.e.* $15 \times 1,000$ and $15 \times 5,000$ with 30%; $15 \times 1,000$, $15 \times 5,000$ and $15 \times 10,000$ with 50%; $15 \times 1,000$, $15 \times 5,000$ and $20 \times 5,000$ with 70%), which is related to the granularity.

It is presented below, in Figure 4 the efficiency acquired with the increase in the number of *threads* is presented, for the contexts with 30%, 50% and 70% density, respectively.

As the results show in Figures 4a, 4b and 4c, the decline in efficiency due to the increase in the number of *threads* when the input size is maintained is clear. Furthermore, it is noted that the increase in the number of context attributes provides an improvement for this metric.



(a) Density of 30% (b) Density of 50% (c) Density of 70%

Figure 4. Efficiency obtained with the increase of the no. of threads of execution

5.2 Dresden challenge

As mentioned earlier, there is a special interest from researchers in the development of techniques and algorithms capable of handling dense and highly dimensional contexts. This fact gave rise to recurring challenge exposed in the 4th edition of the *ICFCA*.

However, in this work, as demonstrated in the experiments, the algorithm for obtaining the minimal implication basis has the worst case scenario in the number of context attributes. So in this section, experiments with a focus on high dimensionality in the number of objects are performed. The experiments were conducted in contexts of 25 attributes with minimal and 30% densities. The purpose of these simulations is to verify that the parallel approach can deal with the specified contexts, according to the threshold defined in the methodology.

In Table 9, the execution times and the number of rules for formal contexts considered are presented.

Table 9. Simulations with 8 threads of execution

|M| × |G|	Density (%)	Time (in seconds)	# rules
25×30,000	17.57	1,131.40	38,496
25×30,000	30	174,873.88	227,064
25×50,000	18.54	1,466.93	32,285
25×50,000	30	372,129.97*	354,608
25×70,000	19.05	28,669.74	175,505
25×70,000	30	-	-
25×90,000	20.15	26,565.92	155,505
25×90,000	30	-	-
25×120,000	21.11	22,704.28	125,603
25×120,000	30	-	-

As discussed in Section 5.1, the results from Table 9 explain the impact of the input context density on the algorithm performance when obtaining the minimal implication basis. However, not only this parameter impacts the execution time, but also the number of rules parameter, which also has significant influence on this metric. This is because, in order to obtain the set of implications, the already obtained rules are employed to generate the other ones (see Algorithm 6).

Analyzing the results presented, it is easy to verify the behavior of the algorithm in relation to the parameter density and number of rules. For example, in context results with minimum density, a significant increase in the execution time starting from $70,000$ objects is perceivable. This increase was mainly due to the increase in the number of rules. In fact, from the contexts with minimum density considered in this simulation, the one that showed the highest execution time was precisely the one that generated the largest number of rules (*i.e.* $25 \times 70,000$).

Such observations are also valid for the other contexts with 30% density. In this case, it is observed that the context that showed the longest execution time was also the one that resulted in the largest number of rules (*i.e.* $25 \times 50,000$). However, unfortunately, due to the defined threshold, it was not possible to perform tests with a higher number of objects for this density.

6. Conclusions and Future Work

In this work, the usage of parallel processing strategies was discussed as a means of reducing the high execution time observed in situations where the formal context presents an elevated degree of density and high dimensionality. It was shown, through experiments, that the parallelization of the stem base provides a significant reduction in the execution time when the input context presents a sparse density and a high number of rules. However, even with this reduction, the observed times are still prohibitive when employed densities exceeding 30% in large-scale contexts, as shown in Section 5.2.

The experiments also reveal that the number of execution *threads* has a significant influence on the performance of the parallel approach. Particularly, in the simulations that used 16 execution *threads*, there was a decline in the obtained *speedup*. This behavior could be observed for all contexts considered in the experiments, and it is justified, since this sit-

uation provides greater competition among the *thread* teams, once that the environment in which these simulations were performed contains only 8 processing cores.

In contrast, when an amount equal to or less than the number of available cores is employed, a higher *speedup* had occurred. At the first moment, such behavior may suggest that the parallel implementation adopted in this study shows better performance when the number of *threads* per core is equal to 1. However, in the simulations with 16 *threads*, it may be noted that the increase in the number of attributes at the input context provides an improvement in the gain obtained. Despite such performance being lower compared to simulations with a smaller number of *threads*. This shows that the behavior observed for 16 execution *threads* is related to the grain size.

This study, however, evaluated the formal contexts in which the number of objects is greater than the number of attributes. The behavior of the algorithm when obtaining the stem base in situations where the input presents large numbers of attributes in comparison with the number of objects was not evaluated. Nor were the contexts where the object and attribute dimensionality are the same.

It is important to mention that in real problems as in social networks analysis, the number of objects is significantly superior to the number of attributes. In this kind of sceneries that our scalable parallel version of the *NextClosure* algorithm can be used.

ACKNOWLEDGEMENTS

The authors would like to thank CNPq, CAPES, FAPEMIG, and SERPRO.

REFERENCES

[1] Kuznetsov SO, Poelmans J. Knowledge representation and processing with formal concept analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2013; 3(3): 200-15. http://dx.doi.org/10.1002/widm.1088

[2] Poelmans J, Kuznetsov SO, Ignatov DI, *et al*. Formal concept analysis in knowledge processing: a survey on models and techniques. Expert Syst. Appl. 2013; 40(16): 6601-23.

[3] Poelmans J, Elzinga P, Viaene S, *et al*. Formal concept analysis in knowledge discovery: a survey. In: Croitoru, M., Ferré, S., Lukose, D., eds.: ICCS. Volume 6208 of Lecture Notes in Computer Science., Springer; 2010. p. 139-53.

[4] Wille R. Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered sets, Dordrecht–Boston, Reidel; 1982. p. 445-70.

[5] Ganter B, Wille R. Formal concept analysis: mathematical foundations. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 1997.

[6] Baixeries J, Szathmary L, Valtchev P, *et al*. Yet a faster algorithm for building the hasse diagram of a concept lattice. In: Ferré, S., Rudolph, S., eds.: ICFCA. Volume 5548 of Lecture Notes in Computer Science. Springer; 2009. p. 162-77.

[7] Moraes NRM, Zárate LE, Freitas HC. A distributed algorithm for formal concepts processing based on Search Subspaces. In: Filipe J, Cordeiro J, eds.: ICEIS (1), SciTePress; 2010. p. 105-11.

[8] Gély A, Medina R, Nourine L, *et al*. Uncovering and reducing hidden combinatorics in guigues-duquenne Bases. In: Ganter, B., Godin, R., eds.: ICFCA. Volume 3403 of Lecture Notes in Computer Science., Springer; 2005. p. 235-48.

[9] Gély A, Medina R, Nourine L. About the enumeration algorithms of closed sets. In: Kwuida, L., Sertkaya, B., eds.: ICFCA. Volume 5986 of Lecture Notes in Computer Science., Springer; 2010 p. 1-16.

[10] Dias SM, Vieira NJ. Concept lattices reduction: definition, analysis and classification. Expert Systems with Applications. 2015; 42(20): 7084-97. http://dx.doi.org/10.1016/j.eswa.2015.04.044

[11] Kuznetsov SO, Obiedkov SA, Roth C. Reducing the representation complexity of lattice-based taxonomies. In: Priss U, Polovina S, Hill R, eds.: ICCS. Volume 4604 of Lecture Notes in Computer Science. Springer; 2007. p. 241-54.

[12] Jay N, Kohler F, Napoli A. Analysis of social communities with iceberg and stability-based concept lattices. In: Medina R, Obiedkov SA, eds.: ICFCA. Volume 4933 of Lecture Notes in Computer Science. Springer; 2008. p. 258-72.

[13] Dias SM, Vieira NJ. Reducing the size of concept lattices: the jbos approach. In: Kryszkiewicz M, Obiedkov SA, eds.: CLA. Volume 672 of CEURWorkshop Proceedings., CEUR-WS.org; 2010. p. 80-91.

[14] Rimsa A, Zárate LE, Song MAJ. Handling large formal context using bdd – perspectives and limitations. In: Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA 2009). Volume 5548 of LNCS/LNAI., Darmstadt, Germany, Springer-Verlag; May 2009. p. 194-206.

[15] Priss U. Some open problems in formal concept analysis. Problems presented at International Conference on Formal Concept Analysis (ICFCA) 2006 in Dresden (2006). Available from: http://www.upriss.org.uk/fca/problems06.pdf(last access - April 2014).

[16] Qi H, Liu D, Hu C, *et al*. A parallel algorithm based on search space partition for generating concepts. In: 10th International Conference on Parallel and Distributed Systems, ICPADS 2004, Newport Beach, CA, USA, July 7-9, 2004, IEEE Computer Society. 2004: 241-8.

[17] Hu X, Wei X, Wang D, *et al*. A parallel algorithm to construct concept lattice. Fuzzy Systems and Knowledge Discovery. FSKD; 2007. p. 119-23.

[18] Fu H, Nguifo E. Partitioning large data to scale up lattice-based algorithm. Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference. 2003: 537-41.

[19] Vimieiro R. An study of algorithms for extraction of rules based on formal concept analysis (in portuguese: um estudo de algoritmos para a extração de regras baseados em análise formal de conceitos). Master's thesis, Universidade Federal de Minas Gerais (UFMG), Instituto de Ciências Extatas, Departamento de Ciência da Computação, Belo Horizonte, Minas Gerais, Brasil. 2007.

[20] Cohen E, Datar M, Fujiwara S, *et al*. Finding interesting associations without support pruning. IEEE Trans. Knowl. Data Eng. 2001; 13(1): 64-78.

[21] Ganter B. Formal concepts analisys: algorithmic aspects. TU Dresden, Germany, Tech. Report. 2002.

[22] Taouil R, Bastide Y. Computing proper implications. In: 9th International Conference on Conceptual Structures: Broadening the Base - ICCS'2001, Stanford, CA US; July 2001. 13 p.

[23] Guigues J, Duquenne V. Minimum families of informative implications resulting from an array of binary data (in French: Familles minimales d'implications informatives résultant, d'un tableau de données binaires). Mathématiques et Sciences Humaines. 1986; 95: 5-18.

[24] Carpineto C, Romano G, D'Adamo P. Inferring dependencies from relations: a conceptual clustering approach. Computational Intelligence. 1999; 15.

[25] Ganter B. Two basic algorithms in concept analysis. In: Kwuida, L., Sertkaya, B., eds.: ICFCA. Volume 5986 of Lecture Notes in Computer Science. Springer; 2010. p. 312-40.

[26] Jota Resende G, De Moraes N, Dias S, *et al*. Canonical computational models based on formal concept analysis for social network analysis and representation. In: International Conference on Web Services (ICWS) - IEEE. June 2015: 717-20.

[27] Krajca P, Outrata J, Vychodil V. Parallel recursive algorithm for fca. In: 6th Internation Conference on Concept Lattices and Their Applications (CLA 2008). Volume 433., Oulomouc, Czech Republic. October 2008: 71-82.

[28] Vychodil V. A new algorithm for computing formal concepts. In: Proceedings of the 19th European Meeting on Cybernetics and Systems Research (EMCSR 2008). 2008: 15-21.

[29] Kuznetsov SO. Learning of simple conceptual graphs from positive and negative Examples. In: Zytkow, J.M., Rauch, J., eds.: PKDD. Volume 1704 of Lecture Notes in Computer Science. Springer; 1999. p. 384-91.

[30] Lindig C, Gbr GD. Fast concept analysis. In: Working with Conceptual Structures – Contributions to ICCS 2000, Shaker Verlag; 2000. p. 152-61.

[31] Berry A, Bordat JP, Sigayret A. A local approach to concept generation. Ann. Math. Artif. Intell. 2007; 49(1-4): 117-36. http://dx.doi.org/10.1007/s10472-007-9063-4

[32] Valtchev P, Duquenne V. Towards divide-and-conquer methods for computing concepts and implications. In: Fourth International Conference on Knowledge Discovery and Discrete Mathematics–Journées de l'informatique Messine – JIM'03, Metz, France, INRIA. Sep 2003: 3-15.

[33] Li Y, Liu ZT, Shen XJ, *et al*. Theoretical research on the distributed construction of concept lattices. In: Machine Learning and Cybernetics, 2003 International Conference on. Volume 1. Nov. 2003: 474-9.

[34] Rimsa A, Song MAJ, Zárate LE. SCGaz - a synthetic formal context generator with density control for test and evaluation of fca algorithms. In: IEEE International Conference on Systems, Man, and Cybernetics, Manchester, SMC 2013, United Kingdom, October 13-16, 2013, IEEE. 2013: 3464-70.

[35] Fu H, Nguifo EM. A parallel algorithm to generate formal concepts for large data. In: Eklund PW, ed.: ICFCA. Volume 2961 of Lecture Notes in Computer Science., Springer; 2004. p. 394-401.

[36] Kengue JFD, Valtchev P, Djamégni CT. A parallel algorithm for lattice construction. In: Ganter B, Godin R, eds.: ICFCA. Volume 3403 of Lecture Notes in Computer Science. Springer; 2005. p. 249-64.

[37] Kengue JFD, Valtchev P, Djamégni CT. Parallel computation of closed itemsets and implication rule bases. In: Stojmenovic I, Thulasiram RK, Yang LT, Jia W, Guo M, de Mello RF, eds.: ISPA. Volume 4742 of Lecture Notes in Computer Science. Springer; 2007. p. 359-70.

[38] Krajca P, Vychodil V. Distributed algorithm for computing formal concepts using map-reduce framework. In: Adams NM, Robardet C, Siebes A, Boulicaut JF, eds.: IDA. Volume 5772 of Lecture Notes in Computer Science. Springer; 2009. p. 333-44.

[39] Krajca P, Outrata J, Vychodil V. Parallel algorithm for computing fixpoints of Galois connections. Ann. Math. Artif. Intell. 2010; 59(2): 257-72. http://dx.doi.org/10.1007/s10472-010-9199-5