A Hybrid Algorithm for Generating Formal Concepts and Building Concept Lattice Using NextClosure and Nourine Algorithms

Simin Jabbari and Kilian Stoffel

Information Management Institute, University of Neuchâtel,
Neuchâtel 2000, Switzerland
{simin.jabbari,kilian.stoffel}@unine.ch
https://www2.unine.ch/imi

Abstract. Concept lattice produced from a set of formal concepts is used for representing concept hierarchy and has many applications in knowledge representation and data mining. Different algorithms have been proposed in the past for efficiently generating formal concepts and building concept lattices. In this paper we introduce the idea of combining existing algorithms in FCA with the aim of benefiting from their specific advantages. As an example, we propose a hybrid model that utilizes the NextClosure (NC) algorithm for generating formal concepts and parts of the Nourine algorithm for building concept lattices. We compare the proposed hybrid model with two of its counterparts: pure NC and pure Nourine. Our experiments show that the hybrid model always outperforms pure NC and for very large datasets can surpass the pure Nourine, as well.

Keywords: Formal concept analysis, formal concept, concept lattice, NextClosure, Nourine algorithm.

1 Introduction

Formal Concept Analysis (FCA) [1, 2] provides a unique framework for analyzing data across various domains. There are two important components to this framework: (1) *formal concepts*, by which the usual notion of real concepts in the world are mathematically represented, and (2) *concept lattices*, that are used for describing hierarchies of the concepts.

Different algorithms have been proposed for generating formal concepts [1, 3–9]. The NextClosure (NC) algorithm [1], proposed by Ganter, is one of the best general purpose algorithms. It has been shown that NC requires much less execution time compared to its counterparts [10].

Apart from formal concepts, concept lattices have been shown to be useful for many applications specifically for transforming data into human understandable structures. They are frequently used for knowledge representation [11] and data mining [12]. According to Wille [13], each element in a lattice can represent a concept and the corresponding graph indicates the relation between the concepts.

As such, a concept lattice is an essential tool in FCA for describing generalization and specialization relationships between formal concepts.

Many different algorithms have been also proposed for building concept lattices from a binary relation [14–16, 2, 17–19]. One of the most promising one is the Nourine algorithm [3] that has been shown to improve previous algorithms such as Bordat's algorithm [14] and the Ganter and Kuznetsov [20] algorithm. This model is proposed for building lattices in a general framework and can be used to compute Galois lattices, the maximal antichains lattices, as well as the Dedekind-MacNeille completion of a partial order.

The Nourine algorithm is suitable for both, generating formal concepts and building concept lattices. Although Nourine's algorihtm is fast in building concept lattices compared to many other algorithms, it is not as fast as NC for generating formal concepts. Here, we would like to combine the advantageous of NC (for generating formal concepts) and Nourine's algorithm (for building concept lattice) to develop a hybrid model that quickly generates formal concepts and builds their corresponding concept lattice. We show through implementation that such a hybrid model always outperforms *pure NC* (where formal concepts are generated by NC algorithm 1 and concept lattice is built by a separate algorithm 3). Moreover, we show that if datasets are large enough (with high density), the hybrid model can also outperform *pure Nourine* (where formal concepts generation and concept lattice building are both implemented using Nourine's algorithms 2, 4, respectively).

2 The Theory

A *formal context* is denoted by a triplet $\langle X, Y, I\rangle$ where I is a $|X| \times |Y|$ binary table that represents relation between a set of *objects* $X = \{1, 2, ..., m\}$ (rows) and a set of *attributes* $Y = \{1, 2, ..., n\}$ (columns). $I(x, y) = 1$ indicates that object $x \in X$ has attribute $y \in Y$.

Assume $A \subset X$ and $B \subset Y$ denote a set of objects and attributes, respectively. Then a pair (A, B) is a *formal concept* if and only if $A^u = B$ and $B^d = A$. Here, A^u denotes a set of those attributes that all objects in A have in common. Similarly, B^d denotes a set of those objects that all attributes in B have in common. We denote by $\{(A_1, B_1), (A_2, B_2), ..., (A_p, B_p)\}$ the set of all p formal concepts embedded in a given formal context $\langle X, Y, I\rangle$.

2.1 NextClosure versus Nourine's Algorithm

An important property of the NC algorithm is that it calculates the list of all *intents* $\{B_1, B_2, ..., B_p\}$ in *lexicographic ordering* fashion. The list of all *extents* $\{A_1, A_2, ..., A_p\}$ then can be easily derived by applying the down operator d on each element of the intent set, i.e., $A_i = (B_i)^d, \forall i$ (see algo 1, and [21] for more information on NC).

Given a formal context $\langle X, Y, I\rangle$, the Nourine algorithm receives a *basis* $\mathcal{B} = \{X \backslash y^d \mid y \in Y\}$ as a set of subsets of objects, and returns a set of pairs

Algorithm 1 Generating formal concepts using NextClosure algorithm [1, 21]

Input: $\langle X, Y, I\rangle$ # a formal context
Output: $\{(A_i, B_i)\}$ # formal concepts with lexic. ordered intents $B_1 < ... < B_p$

$B_1 = (\emptyset^d)^u$ # the least intent
$i = 1$
while $B_i \neq Y$ **do**
  $B_i = (B_i)^+$ # replace the current intent B_i with the next intent B_i^+ (see below)
  $A_i = (B_i)^d$
  $i++$
end

$B^+ = B \oplus k$ where $k \in \{1, ..., n\}$ is the greatest one for which $B <_k B \oplus k$
$B \oplus k = \left((B \cap \{1, 2, ..., k-1\} \cup \{k\})^d\right)^u$
$B <_k B' \iff k \in B' \setminus B$ and $B \cap \{1, 2, ..., k-1\} = B' \cap \{1, 2, ..., k-1\}$

$\{(F_i, \gamma(F_i))\}$ that is isomorphic to the set of formal concepts $\{(A_i, B_i)\}$, calculated by NC. Here, $F_i = X\backslash A_i$ and $\gamma(F_i) = B_i$. In other words, the Nourine algorithm returns the list of all intents B_i and the list of the *extent complements* $X\backslash A_i$ (see algorithm 2).

Algorithm 2 Generating formal concepts using Nourine algorithm [3, 22]

Input: $\langle X, Y, I\rangle$ # a formal context
Output: $\{(F_i, \gamma(F_i))\}$ # a family of pairs describing formal concepts

$\mathcal{B} = \{X\backslash y^d \mid y \in Y\}$ # the basis: a set of subsets of X
$\mathcal{F} = \{\emptyset\}$ # the root of lattice
for each B in $\mathcal{B}$ **do**
  for each F in $\mathcal{F}$ **do**
    $F' = F \cup B$
    if $F' \notin \mathcal{F}$
      $\mathcal{F} = \mathcal{F} \cup \{F'\}$ # append F' to $\mathcal{F}$
      $\gamma(F') = \gamma(F) \cup \{y_B\}$ # note that $B = X\backslash(y_B)^d$ and $y_B \in Y$
    end
  end
end

2.2 The Concept Lattice as a Directed Graph

Each formal concept is indicative of the usual notion of a real concept in the world. Concept lattices are a way to demonstrate hierarchical relations the concepts. For instance, *dog* is a sub-concept of *mammal* (or equivalently mammal is a super-concept of dog). That is all dogs are mammals but not all the mammals

are dog. There might also be no relation between two concepts. These hierarchical relations between the concepts have a mathematical interpretation as shown in the following.

A formal concept $C_j = (A_j, B_j)$ (e.g., dog) is a sub-concept of $C_i = (A_i, B_i)$ (e.g., mammal), if and only if B_i (attributes that are required for a creature to be considered as a mammal) is a subset of B_j (attributes that are required for a creature to be considered as a dog). In other words, C_j is more specific than C_i. This also implies that A_j (objects in the family of dogs) is a subset of A_i (objects in the family of mammals, which also include all the dogs). Mathematically, we say

$$C_j \prec C_i \Longleftrightarrow (A_j \subset A_i \Longleftrightarrow B_j \supset B_i) \tag{1}$$

A concept lattice is a *directed graph* with p nodes each of which corresponds to a formal concept. The concept C_i is connected to C_j with an arrow pointing from C_i to C_j (denoted by $C_i \rightarrow C_j$), only if two conditions are simultaneously fulfilled: (1) C_i is a super-concept of C_j, i.e., $C_j \prec C_i$ and (2) no other concept C_k exists such that $C_j \prec C_k \prec C_i$.

In other words, each concept is only allowed to be connected to its *parents* (infimum of its super-concepts) and its *children* (supremum of its sub-concepts). For instance if *animal* $\rightarrow$ *mammal* and *mammal* $\rightarrow$ *dog*, then there must be no direct edge in the graph from *animal* to *dog*, although *animal* is a super-concept of *dog*, i.e., *dog* $\prec$ *animal* but *animal* $\nrightarrow$ *dog*.

2.3 Generating Concept Lattices using Lexicographically Ordered Intents

For building a concept lattice (or equivalently a graph), we need to create a $p \times p$ adjacency matrix M to indicate which concepts are adjacent. We set $M(i, j) = 1$ if $C_i \rightarrow C_j$ and $M(i, j) = 0$ otherwise.

In the following we assume that all formal concepts of a given context are available and so we have access to the list of all intents $\{B_1, ..., B_p\}$. If the formal concepts are generated by the NC algorithm, the list of intents is lexicographically ordered. Without loss of generality, we can assume that the intents are sorted as follows

$$B_1 < B_2 < ... < B_p, \tag{2}$$

where B_1 and B_p correspond to the attributes of *the most general* concept C_1 and *the most specific* concept C_p, respectively.

Thanks to the lexicographically ordered intents (2), a concept C_i is *never* a super-concept of C_j (i.e., $C_j \not\prec C_i$), if $j < i$. This is because B_i is never a subset of B_j when B_j is lexicographically less than B_i (i.e., $B_j < B_i \Rightarrow B_j \not\supset B_i \Rightarrow C_j \not\prec C_i$, $\forall j < i$). For further details about lexicographically ordered intents and their properties see [21].

A direct consequence of the statements above is that the entries of the adjacency matrix $M(i, j)$ for all $j < i$ is 0. In other words, the adjacency matrix M is an *upper-triangular* matrix. Note that if we did not have access to the list of

lexicographically ordered intents, we could not easily argue that the lower triangular entries of the adjacency matrix M are all zero. This feature significantly speeds up generating concept lattices using naive methods, because we do not need to spend time for calculating half of the entries of the adjacency matrix M. However, even by using this trick we will show later that it does *not* outperform Nourine's algorithm for building concept lattice.

Algorithm 3 illustrates how concept lattice can be built using lexicographically ordered intents calculated by NC. In summary, if $j \leq i$, we set $M(i,j) = 0$ and for the rest of the entries in the adjacency matrix (i.e., for $j > i$), we set $M(i,j) = 1$ (i.e., $C_i \rightarrow C_j$) only if three conditions are simultaneously fulfilled: (1) $j > i$, (2) $B_i \subset B_j$ and (3) no other B_m exists such that $B_j \subset B_m \subset B_j$.

Algorithm 3 Building concept lattice using lexicographically ordered intents

Input: a list of lexicographically ordered intents $B_1 < B_2 < ... < B_p$
Output: a $p \times p$ adjacency matrix M, where $M(i,j) = 1$ indicates $C_i \rightarrow C_j$

$M(i,j) = 0,\ \forall i,j$
for $i = 1 : p$ **do**
  $temp = [True, ..., True]$ # a list of p Boolean $True$
  for $j = i + 1 : p$ **do** # we only need to check for $j > i$
    if $temp[j] == False$ # there was an $m < j$ such that $B_i \subset B_m \subset B_j$
      jump to the next j
    if $B_i \not\subset B_j$
      jump to the next j
    for $k = j + 1 : p$ **do**
      If $B_j \subset B_k$ # if the condition is true then $B_i \subset B_j \subset B_k \Rightarrow M(i,k) = 0$
        $temp[k] = False$ # to prevent unnecessary check for $M(i,k)$ with $k > j$
    end
    $M(i,j) = 1$
  end
end

For building a concept lattice using the Nourine algorithm, we do not use the algorithm 3. In the original description of Nourine in [3], a covering graph algorithm is proposed for this purpose that we rephrased in algorithm 4. The Nourine algorithm for building concept lattices is fast because it benefits from an auxiliary tree structure that is used for the representation of concepts.

2.4 The Hybrid Model: Combination of NextClosure and Nourine

Although generating formal concepts is significantly sped up by lexicographically ordered intents (as it is the case in NC), naive methods for building concept lattices (from those calculated formal concepts), e.g. algorithm 3, is not the most efficient method. This is because the NC algorithm is originally designed for generating formal concepts and not for building concept lattices. In contrast

Algorithm 4 Building concept lattice using Nourine algorithm [3, 22]

Input: $\mathcal{F} = \{F_i\}$, $\{\gamma(F_i)\}$, $\mathcal{B}$ # basis $\mathcal{B}$ and the family of pairs $\{(F_i, \gamma(F_i))\}$
Output: $\{ImSucc(F_i)\}$ # immediate successors of $\{F_i\}$, i.e., $ImSucc(F_i) \rightarrow F_i$

for each F in $\mathcal{F}$ **do**
  $count(F) = 0$
  for each $B \in \mathcal{B} \backslash \mathcal{B}_F$ **do** # we define $\mathcal{B}_F = \{X \backslash y^d \mid y \in \gamma(F)\}$
    $F' = F \cup B$
    $count(F') + +$
    if $|\gamma(F')| = count(F') + |\gamma(F)|$
      $ImSucc(F) = ImSucc(F) \cup \{F'\}$
    end
  end
end

to NC, Nourine's algorithm was originally designed for building lattices including concept lattice, although it can also be used for generating formal concepts.

The NC algorithm relays on a list of intents which are lexicographically ordered, the Nourine algorithm does not provide an ordered list of intents at all. Therefore, Nourine's algorithm is not as fast as NC for generating formal concepts. However, due to the maintenance of an auxiliary tree structure for representing the formal concepts, Nourine is one of the most efficient algorithms that currently exist for building concept lattices (see [3, 22] for more information on Nourine's algorithm).

Our proposed hybrid model (algorithm 5) benefits from the advantages of both NC and Nourine's algorithm. It generates the set of formal concepts $\{(A_i, B_i)\}$ using the NC algorithm 1 and then transform it to a set of $\{(F_i, \gamma(F_i))\}$ using $F_i = X \backslash A_i$ and $\gamma(F_i) = B_i$. Then it applies Nourine's algorithm 4 on the set $\{(F_i, \gamma(F_i))\}$ to build the concept lattice.

Algorithm 5 Generating formal concepts and concept lattice using hybrid

Input: $\langle X, Y, I \rangle$ # a formal context
Output: a $p \times p$ adjacency matrix M, where $M(i, j) = 1$ indicates $C_i \rightarrow C_j$

$M(i, j) = 0 \ \forall i, j$
Generate formal concepts $\{(A_i, B_i)\}$ using NC algorithm 1
Create a set $\{(F_i, \gamma(F_i))\} = \{(X \backslash A_i, B_i)\}$
Find $\{ImSucc(F_i)\}$ using Nourine algorithm 4
$M(k, i) = 1$ for all $F_k \in ImSucc(F_i)$ # F_k is connected to F_i, i.e., $F_k \rightarrow F_i$

3 Results

We first illustrate that the concept lattices generated by algorithm 3 (Fig. 1) and algorithm 4 (Figs 2) for a given formal context $\langle X, Y, I \rangle$ (Table 1) are isomorphic. Each formal concept in these two figures are represented by an ellipse. Note that each red ellipse in the lattice of Fig. 2 corresponds to a blue ellipse in the lattice of Fig. 1 with $(F_i, \gamma(F_i)) = (X \backslash A_i, B_i)$.

Table 1. A formal context with the object set $X = \{0, 1, 2, 3, 4\}$ and the attribute set $Y = \{a, b, c, d, e\}$. For this formal context, the corresponding *basis* that is given as input to the Nourine algorithm is $\mathcal{B} = \{X \backslash y^d \mid y \in Y\} = \{\{0, 2, 3\}, \{1, 3, 4\}, \{1, 2, 3\}, \{0, 3, 4\}, \{1, 2, 4\}\}$.

I(x,y)	**a**	**b**	**c**	**d**	**e**
0	0	1	1	0	1
1	1	0	0	1	0
2	0	1	0	1	0
3	0	0	0	0	1
4	1	0	1	0	0

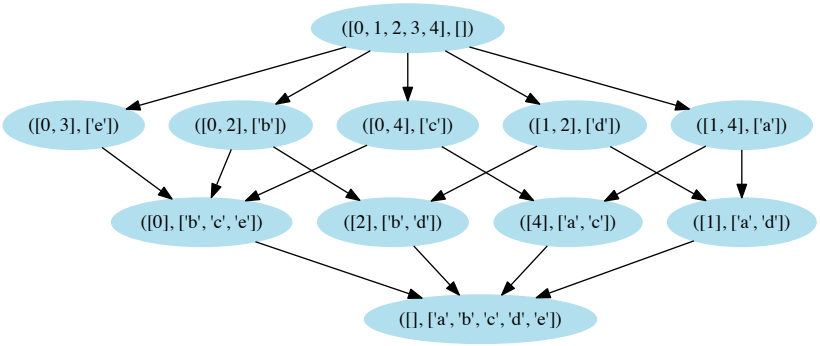

Fig. 1. The concept lattice in Table 1 generated by algorithm 3 using lexicographically ordered intents derived using the NextClosure algorithm 1. Each concept C_i is represented by a blue ellipse and contains its corresponding (extent, intent), i.e., (A_i, B_i). The very top ellipse corresponds to the most general concept and the very bottom one corresponds to the most specific concept.

We then compare the time spent for generating formal concepts and building concept lattices using the different algorithms mentioned before in this paper.

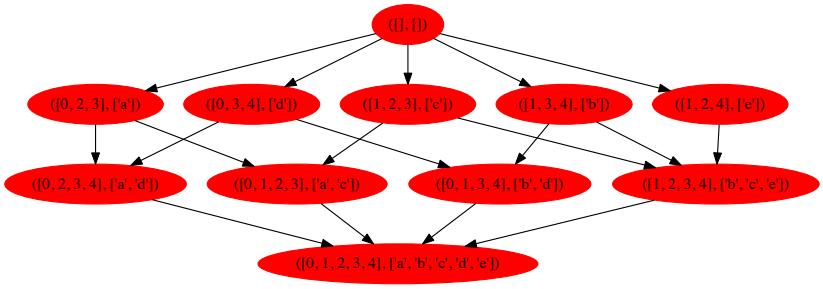

Fig. 2. The concept lattice in Table 1 generated by Nourine's algorithm [3]. Each concept C_i is represented by a red ellipse and contains its corresponding $(F_i, \gamma(F_i))$. Note that this lattice is isomorphic to the lattice of Fig. 1. Note also that each red ellipse in this lattice corresponds to a blue ellipse in the lattice of Fig. 1 with $F_i = X \backslash A_i$ and $\gamma(F_i) = B_i$.

For this purpose we created a variety of artificial formal contexts with different size and sparsity (see Table 2).

Table 2. List of formal contexts, created artificially, that are used for the comparisons. Here, Density determines the fraction of 1 in the corresponding binary table.

Formal Contexts			
Name	Objects	Attributes	Density
D1	25	26	50%
D2	26	17	76%
D3	500	15	30%
D4	1000	20	15%
D5	1000	15	60%
D6	2688	30	11%
D7	500	15	60%
D8	600	15	50%

Fig. 3 clearly shows that NC requires much less time than Nourine for generating formal concepts. This difference gets bigger when datasets are getting bigger.

Fig. 4 depicts the time spent for building the concept lattices using Nourine's algorithm 2 and the naive algorithm 3 (that uses formal concepts calculated by NC). Results indicate that Nourine performs much better than its counterpart. Note that the time reported for building concept lattices is only for the calculation of the adjacency matrix and not drawing the graph.

Although NC performs faster than Nourine for generating formal concepts, Nourine builds concept lattice much faster than the naive algorithm 3. Therefore, the *pure Nourine* (algorithms 2, 4) outperforms the *pure NC* (algorithms 1, 3). The *hybrid* model (algorithm 5, or equivalently algorithms 1, 4), however, can compete with the *pure Nourine algorithm* (see Fig. 5). That means, although as our results show the hybrid model not always outperforms *pure Nourine*, we can see that for large datasets with high density (i.e., a large percentage of 1 in the binary table of the corresponding formal context), the hybrid model surpasses the pure Nourine algorithm. Therefore, in the context of Big Data and depending on the characteristics of the formal context (such as the density), it might be worth to first generate formal concepts using the NC algorithm, and then to build the concept lattice using Nourine's algorithm. Our prediction is that for very big datasets with large density, the hybrid model surpass the pure Nourine algorithm while for very large datasets with low density, the pure Nourine algorithm outperforms the hybrid model.

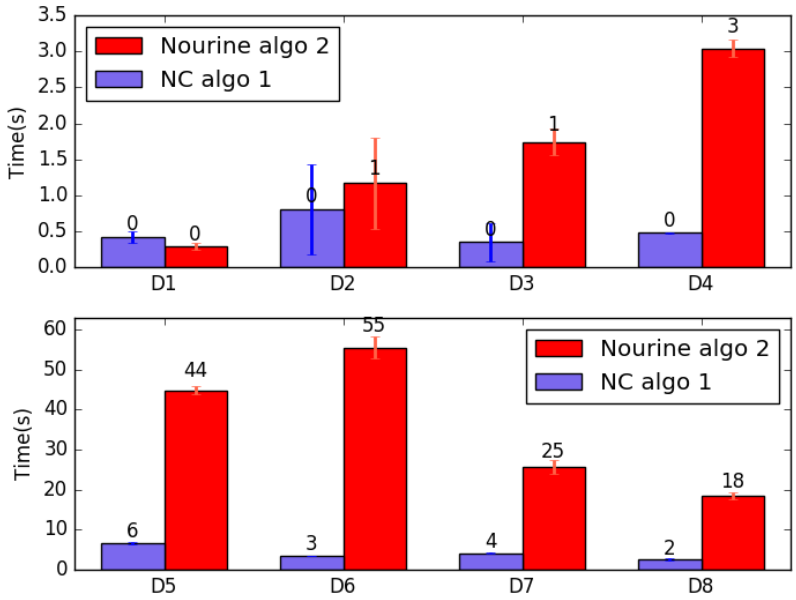

Fig. 3. The average time spent for calculating formal concepts using the NC algorithm 1 (blue) and the Nourine algorithm 2 (red) for different datasets in Table 2. Error bars indicate the standard deviation (each implementation is executed 5 times with a different random dataset).

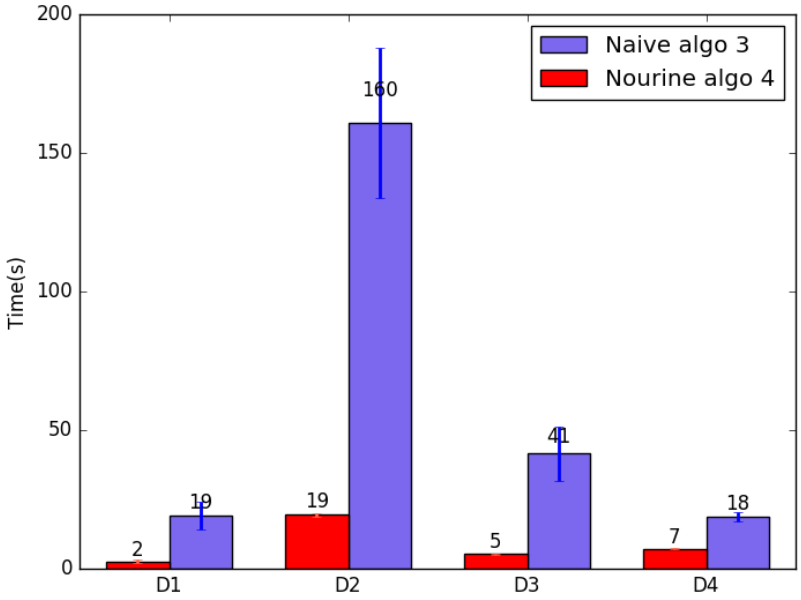

Fig. 4. The average time spent for building the concept lattice using the Nourine algorithm 4 (red) and the naive algorithm 3 (blue) for some of the datasets in Table 2. Error bars indicate the standard deviation (each implementation is run 5 times with a different random dataset).

4 Conclusion

We proposed a hybrid model that combines the benefits of the NextClosure algorithm for generating formal concepts and Nourine's algorithm for building concept lattices. We showed that our proposed hybrid model outperforms pure NC and only for very large datasets can surpass the pure Nourine algorithm. Our results suggest that we can derive a new framework for generating formal concepts and building concept lattices by combining existing algorithms to benefit from their advantages and overcome their shortcomings.

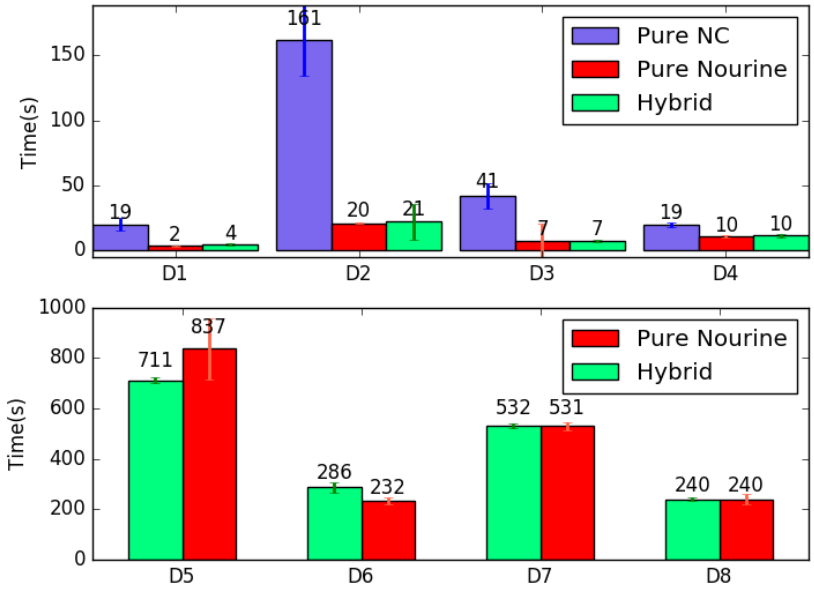

Fig. 5. The average time spent for both generating formal concepts as well as building concept lattices using pure NC (algorithms 1, 3) (blue), pure Nourine (algorithms 2, 4) (red) and our proposed hybrid algorithm 5 (green), for different datasets in Table 2. Time for pure NC is not reported for datasets **D5**-**D8** because it was shown to be more time consuming than pure Nourine and the hybrid model for large datasets.

References

1. Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.
2. Bernhard Ganter. *Two basic algorithms in concept analysis*. Springer, 2010.
3. Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Information processing letters*, 71(5):199–204, 1999.
4. Petr Krajca and Vilem Vychodil. Distributed algorithm for computing formal concepts using map-reduce framework. In *Advances in Intelligent Data Analysis VIII*, pages 333–344. Springer, 2009.
5. Biao Xu, Ruairí de Fréin, Eric Robson, and Mícheál Ó Foghlú. Distributed formal concept analysis algorithms based on an iterative mapreduce framework. In *Formal Concept Analysis*, pages 292–308. Springer, 2012.
6. Sergei O Kuznetsov. A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automatic documentation and Mathematical linguistics*, 27(5):11–21, 1993.
7. Sergei O Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In *PKDD*, volume 99, pages 384–391. Springer, 1999.

8. Petr Krajca, Jan Outrata, and Vilém Vychodil. Advances in algorithms based on cbo. In *CLA*, pages 325–337, 2010.
9. Simon Andrews. In-close, a fast algorithm for computing formal concepts. 2009.
10. Sergei O Kuznetsov and Sergei A Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.
11. Rudolf Wille. *Lattices in data analysis: How to draw them with a computer*. Springer, 1989.
12. Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, et al. New algorithms for fast discovery of association rules. In *KDD*, volume 97, pages 283–286, 1997.
13. Rudolf Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts*. Springer, 1982.
14. Jean Paul Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, 96:31–47, 1986.
15. Claudio Carpineto and Giovanni Romano. Galois: An order-theoretic approach to conceptual clustering. In *Proceedings of ICML*, volume 293, pages 33–40, 1993.
16. Michel Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, pages 21–25, 1969.
17. Robert Godin, Rokia Missaoui, and Hassan Alaoui. Learning algorithms using a galois lattice structure. In *Tools for Artificial Intelligence, 1991. TAI'91., Third International Conference on*, pages 22–29. IEEE, 1991.
18. Y Malgrange. Recherche des sous-matrices premières d'une matrice à coefficients binaires. applications à certains problèmes de graphe. In *Proceedings of the Deuxième Congrès de l'AFCALTI*, pages 231–242, 1962.
19. Eugene M Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.
20. Bernhard Ganter and Sergei O Kuznetsov. Stepwise construction of the dedekind-macneille completion. In *Conceptual framearc=Structures: Theory, Tools and Applications*, pages 295–302. Springer, 1998.
21. Radim Belohlavek. Introduction to formal concept analysis. *Palacky University, Department of Computer Science, Olomouc*, 2008.
22. Lhouari Nourine and Olivier Raynaud. A fast incremental algorithm for building lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):217–227, 2002.