Expert Systems With Applications 96 (2018) 271–283

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Closed frequent similar pattern mining: Reducing the number of frequent similar patterns without information loss

Ansel Y. Rodríguez-González[a,*], Fernando Lezama[a], Carlos A. Iglesias-Alvarez[b], José Fco. Martínez-Trinidad[a], Jesús A. Carrasco-Ochoa[a], Enrique Munoz de Cote[a]

[a] Department of Computer Sciences, Institute of Astrophysics, Optics and Electronics (INAOE), Luis Enrique Erro 1, Tonantzintla, Puebla, 72840, Mexico
[b] Faculty of Mathematics and Computer Science, University of Havana, San Lazaro and L, Plaza de la Revolución, Havana, Cuba

ARTICLE INFO

Article history:
Received 7 December 2016
Revised 7 December 2017
Accepted 8 December 2017
Available online 9 December 2017

Keywords:
Data mining
Frequent patterns
Mixed data
Similarity functions
Downward closure

ABSTRACT

Frequent pattern mining is considered a key task to discover useful information. Despite the quality of solutions given by frequent pattern mining algorithms, most of them face the challenge of how to reduce the number of frequent patterns without information loss. Frequent itemset mining addresses this problem by discovering a reduced set of frequent itemsets, named *closed frequent itemsets*, from which the entire frequent pattern set can be recovered. However, for *frequent similar pattern mining*, where the number of patterns is even larger than for Frequent itemset mining, this problem has not been addressed yet. In this paper, we introduce the concept of *closed frequent similar pattern mining* to discover a reduced set of frequent similar patterns without information loss. Additionally, a novel *closed frequent similar pattern mining* algorithm, named *CFSP-Miner*, is proposed. The algorithm discovers frequent patterns by traversing a tree that contains all the closed frequent similar patterns. To do this efficiently, several lemmas to prune the search space are introduced and proven. The results show that *CFSP-Miner* is more efficient than the state-of-the-art frequent similar pattern mining algorithms, except in cases where the number of frequent similar patterns and closed frequent similar patterns are almost equal. However, *CFSP-Miner* is able to find the closed similar patterns, yielding a reduced size of the discovered frequent similar pattern set without information loss. Also, *CFSP-Miner* shows good scalability while maintaining an acceptable runtime performance.

1. Introduction

Frequent pattern mining (Agrawal, Imieliński, & Swami, 1993; Agrawal, Srikant et al., 1994) is a technique that consists of finding patterns (i.e., feature sets with their corresponding values) that frequently occur (more than or equal to a minimum frequency threshold) in a dataset. It is considered a key task in data mining because of its application to discover useful information, such as risk factors (Li, Fu, & Fahey, 2009; Li et al., 2005; Nahar, Imam, Tickle, & Chen, 2013), user's profiles (Chiu, Yeh, & Lee, 2013), human behavior (Wen, Zhong, & Wang, 2015), malicious software (Fan, Ye, & Chen, 2016) among others. In addition, *Frequent pattern mining* can be used as a previous or internal step for other data mining tasks, like association rule mining (Alatas, Akin, & Karci, 2008; Kalpana & Nadarajan, 2008; Lopez, Blanco, Garcia, Cano, & Marin, 2008), classification(Hernández-León, Carrasco-Ochoa, Martínez-Trinidad, & Hernández-Palancar, 2012; Nguyen & Nguyen, 2015) and clustering (Beil, Ester, & Xu, 2002).

Since 1990, most of the *frequent pattern mining* algorithms were based on the exact matching of *boolean* features to compare and count patterns. This subclass of *frequent pattern mining* algorithms was called *frequent itemset mining* (considered as the traditional approach for frequent pattern mining). However, real life objects, such as objects in sociology (Ruiz-Shulcloper & Fuentes-Rodríguez, 1981), geology (Gómez-Herrera, Rodríguez-Morn, Valladares-Amaro et al., 1994), medicine (Ortiz-Posadas, Vega-Alvarado, & Toni, 2009) or information retrieval (Baeza-Yates, Ribeiro-Neto et al., 1999)), are rarely equal or they can be described by *non boolean* features. Thus, similarity functions different from the exact matching were proposed to compare object descriptions giving rise to a new approach named *frequent similar pattern mining* which can handle datasets containing *non boolean* features by using similarity functions (Danger, Ruiz-Shulcloper, & Llavori, 2004; Rodríguez-González, Martínez-Trinidad, Carrasco-Ochoa, & Ruiz-Shulcloper, 2008; 2011; 2013). This approach produces pat-

* Corresponding author.
E-mail addresses: ansel@ccc.inaoep.mx (A.Y. Rodríguez-González), f.lezama@inaoep.mx (F. Lezama), ciglsias@gmail.com (C.A. Iglesias-Alvarez), fmartine@inaoep.mx (J.Fco. Martínez-Trinidad), ariel@inaoep.mx (J.A. Carrasco-Ochoa), jemc@inaoep.mx (E.M. de Cote).

https://doi.org/10.1016/j.eswa.2017.12.018
0957-4174/

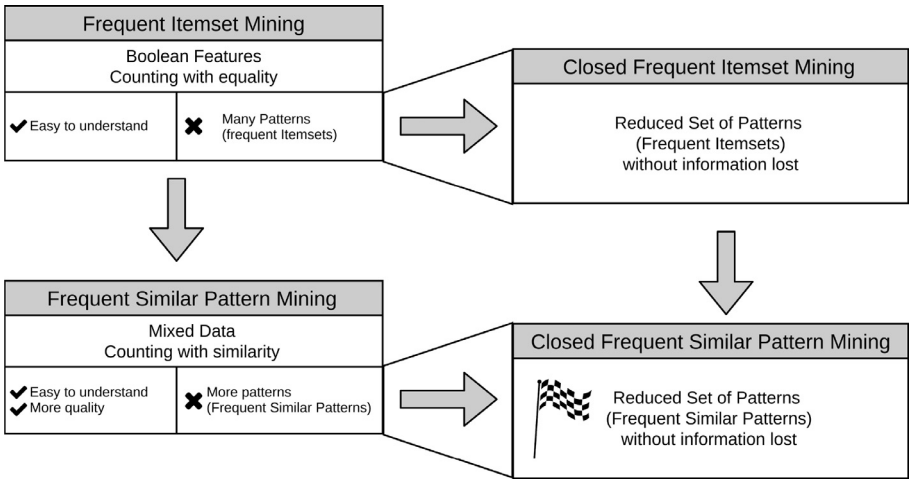

Fig. 1. From frequent itemset mining to closed frequent similar pattern mining.

terns which can not be found by those algorithms based on exact matching. The frequent patterns found using a similarity function are named *frequent similar patterns* (Rodríguez-González, Martínez-Trinidad, Carrasco-Ochoa, & Ruiz-Shulcloper, 2013).

Despite the quality of solutions given by a *frequent itemset mining* algorithm or a *frequent similar pattern* algorithm, a critical drawback to both these approaches is that, although a complete set of frequent itemsets or frequent similar patterns can be found, the number of frequent patterns is often too big (Burdick, Calimlim, & Gehrke, 2001; Hu, Sung, Xiong, & Fu, 2008; Pei, Han, Mao et al., 2000; Rodríguez-González et al., 2013; Zaki & Hsiao, 2002).

It is helpful, therefore, to obtain a reduced set of all the frequent patterns without information loss (i.e., from which the entire frequent pattern set can be recovered). One way to do that, is through the use of *closed frequent itemsets mining* (Prabha, Shanmugapriya, & Duraiswamy, 2013). *Closed frequent itemsets mining* algorithms define that a frequent itemset is closed if it has no super-patterns with the same frequency, and use this definition to find the closed frequent itemsets. From such closed itemsets, the complete set of frequent itemsets can be generated without information loss. The so-called *closed frequent itemsets mining* algorithms also have more efficient runtimes than *frequent itemset mining* algorithms (Pei et al., 2000; Uno, Asai, Uchida, & Arimura, 2003; Zaki & Hsiao, 2002).

However, the concept of a closed patterns has not been exploited for the *frequent similar patterns* to the best of our knowledge. In this paper, we introduce the concept of a *closed frequent similar pattern* and a novel *closed frequent similar pattern mining* algorithm, named *CFSP-Miner*, that finds a reduced closed set of frequent similar patterns without information loss (see Fig. 1 to see the scope of our work). The results show that this proposed algorithm, *CFSP-Miner*, has more efficient runtimes than the state-of-the-art *frequent similar pattern mining* algorithms, except when the number of frequent similar patterns and the closed frequent similar patterns are almost equal. From the scalability point of view, the *CFSP-Miner* algorithm also finds the closed frequent similar patterns in an acceptable runtime, regardless of size.

The outline of this paper is as follows. In Section 2 related work is reviewed. Section 3 provides basic concepts. In Section 4 several concepts are introduced and redefined as a result of combining frequent similar pattern and closed pattern concepts. In Section 5 a novel algorithm for mining *closed frequent similar patterns* is proposed. Section 6 presents the experimental results and discussion, and finally, in Section 7 some conclusions and future work are discussed.

2. Related work

The frequent pattern mining problem has attracted the attention of the data mining research community because of its potential application in many different domains. One research line focuses on discovering frequent patterns in mixed data (datasets whose objects are described by numerical and non-numerical features) using similarity functions to compare and count objects (frequent similar patterns) (Danger et al., 2004; Rodríguez-González et al., 2008; 2013). Another research line focuses on obtaining a reduced set of all the frequent patterns in boolean datasets without information loss (closed frequent itemsets) (Prabha et al., 2013; Uno et al., 2003; Zaki & Hsiao, 2002).

The results of these research lines related to the current work are presented in the following subsections.

2.1. Frequent similar pattern mining

In the literature there are two algorithms for mining frequent similar patterns: *ObjectMiner* (Danger et al., 2004) and *STreeDC-Miner* (Rodríguez-González et al., 2013). Both algorithms find the whole set of frequent similar patterns by using boolean similarity functions. The discovered set of patterns usually contains patterns hidden to the traditional approach.

ObjectMiner (Danger et al., 2004) was the first algorithm for mining frequent similar pattern that used similarity functions different from equality, and it was inspired by the *Apriori* algorithm (Agrawal et al., 1994). *ObjectMiner* works by following a breadth first search strategy. Given a dataset $\mathcal{D}$, a similarity function, and a minimum frequency threshold, *ObjectMiner* finds the frequent similar patterns in $\mathcal{D}$ with only one feature. The frequency of patterns is computed by adding the occurrences of itself and the occurrences of its similar patterns. Each pattern with a frequency greater than the minimum frequency threshold is consider a frequent similar pattern. In the iteration k (starting with $k = 2$) *ObjectMiner* finds the frequent similar patterns in $\mathcal{D}$ with k features. This is done by merging the frequent similar patterns with $k - 1$ features. This process finishes after no frequent similar patterns are found.

The main weakness of *ObjectMiner*, is that the similarity between a pattern and its repetitions is computed in each iteration of the algorithm, causing an additional and unnecessary computation. *ObjectMiner* also stores the set of all similar subdescriptions (including its repetitions) of each frequent subdescription, which slows the performance as was shown in Rodríguez-González et al. (2013).

STreeDC-Miner (Rodríguez-González et al., 2013) is another algorithm for mining frequent similar patterns. *STreeDC-Miner* works by following a depth first search strategy, using a total order defined over the feature set in $\mathcal{D}$. *STreeDC-Miner* starts analyzing each set with only one feature *A*, and recursively adds a new feature to this set which is after the last feature in the set according to the defined order. The base case in the recursion is fulfilled when no more frequent similar patterns are discovered for the current set of features under analysis. In order to efficiently compute the frequency of patterns, *STreeDC-Miner* uses a tree structure called *STree*. Each leaf in a *STree* stores the repetitions of the pattern under this branch and also stores the similarity among this pattern and its similar patterns. The similarity between two patterns in *STree* is only computed if their subpatterns in *STree* are similar, and if one of them is a frequent similar pattern. In this way *STreeDC-Miner* reduces the computational effort to compute the frequency of each pattern by reducing the similarity function evaluations. However, *STreeDC-Miner* like *ObjectMiner* has the same drawback of finding too many frequent similar patterns.

2.2. Closed frequent itemset mining

The closed frequent itemset mining consists in finding only those frequent itemsets where there is not another itemset containing it with the same frequency (Prabha et al., 2013). The advantage of computing the closed frequent itemsets is that they retains all the information needed to obtain all the frequent patterns and their exact frequencies without the need of the original data set. Two of the first and most referenced closed frequent itemset mining algorithms are *Closet* and *CHARM*.

Closet (Pei et al., 2000) is based on: *i*) compressing frequent patterns in tree structure containing the frequent patterns for mining closed itemsets without candidate generation, *ii*) compressing a single path in the tree to do a fast identification of the frequent closed itemsets, *iii*) performing a partition-based projection mechanism for scalable mining in large databases.

Closet uses a divide and conquer method for mining frequent closed patterns. First, frequent items are found and sorted in descending frequency order. Then, the search space is divided into non-overlapping subsets and each subset of frequent closed itemsets is mined recursively by constructing related conditional databases.

CHARM (Zaki & Hsiao, 2002), on the other hand, uses a bottom up approach for mining the closed frequent itemsets. It explores both itemset and transaction spaces, through a dual itemset-tidset search tree, using an efficient hybrid search that skips many levels in the tree during the search. *CHARM* also uses a technique called diffsets to reduce the memory footprint of intermediate computations. Finally, it uses a fast hash-based approach to remove any non-closed sets found during the search.

LCM (Uno et al., 2003) is another closed frequent itemsets mining algorithm. That defines a parent-child relationship between closed patterns. It was proven that each parent-child relationship forms a tree from which all closed patterns can be found by traversing it. *LCM*, also introduced an efficient way to traverse each tree in polynomial time with respect to the amount of closed frequent itemsets in the datasets.

From *CHARM, Closet* and *LCM*, other algorithms have also been proposed for closed frequent itemsets mining such as *COBBLER* (Pan, Tung, Cong, & Xu, 2004), *TD-Close* (Han & Shao, 2006), *PGMiner* (Moonesinghe, Fodeh, & Tan, 2006), *Patricia Tree Close* (Nezhad & Sadreddini, 2007), *ICMiner* (Lee, Wang, Weng, Chen, & Wu, 2008), *TTD-Close* (Liu et al., 2009) *CFIM-P* (Nair & Tripathy, 2011), *DBV-Miner* (Vo, Hong, & Le, 2012), *NAFCP* (Le & Vo, 2015), and more recently *BVCL* (Hashem, Karim, Samiullah, & Ahmed, 2017).

Unlike all these previous algorithms, our work introduces an algorithm for mining closed frequent patterns in datasets where the objects are described by numerical and non-numerical features.

3. Basic concepts and notations

In this Section, some concepts related to frequent similar pattern mining and closed frequent pattern mining are introduced. First, common concepts are described. Secondly, frequent similar pattern mining concepts are enumerated. Thirdly, closed frequent pattern mining concepts are also enumerated.

Consider a dataset as a tuple $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ where $\mathcal{O}$ is a non-empty and finite set of objects, $\mathcal{A}$ is a non-empty and finite set of features, $\mathcal{V}$ is a non-empty and finite set of values and $\mathcal{P}$ is an application such that $\mathcal{P} : (\mathcal{O} \times \mathcal{A}) \longrightarrow \mathcal{V}$. For simplicity, $O[A]$ is denoted as $\mathcal{P}(O, A)$, $\forall O \in \mathcal{O}$, $\forall A \in \mathcal{A}$.

Definition 1 (Domain of a feature)**.** The Domain of a feature $A \in \mathcal{A}$ is the application $Domain : \mathcal{A} \longrightarrow 2^{\mathcal{V}}$ defined as $Domain(A) = \{V \in \mathcal{V} \mid \exists O \in \mathcal{O} \ : \ V = O[A]\}$.

Definition 2 (Pattern)**.** A pattern in $\mathcal{D}$ is a pair $T = (O, \mathcal{A}_T) \in (\mathcal{O} \times \{2^{\mathcal{A}} \backslash \{\}\})$. $T.O$ denotes O and $T.\mathcal{A}$ denotes $\mathcal{A}_T$. Also, $\mathcal{T}$ denotes the set of all patterns in $\mathcal{D}$. $\mathcal{T}$ is a non-empty and finite set. Given two patterns $T_1 \in \mathcal{T}$ and $T_2 \in \mathcal{T}$, $T_1 = T_2$ iff $T_1.\mathcal{A} = T_2.\mathcal{A}$ and $\forall A \in T_1.\mathcal{A}$; $T_1.O[A] = T_2.O[A]$.

Definition 3 (Super-pattern)**.** A *SupPatterns* of a pattern is the application $SupPatterns : \mathcal{T} \longrightarrow 2^{\mathcal{T}}$ defined as: $SupPatterns(T) = \{T_{sup} \in \mathcal{T} \mid T.\mathcal{A} \subseteq T_{sup}.\mathcal{A}$ and $T_{sup}.O[A] = T.O[A] \ \forall A \in T.\mathcal{A}\}$. If $T_{sup} \in SupPatterns(T)$; then we say that T_{sup} is a super-pattern of T and T is sub-pattern of T_{sup}.

Definition 4 (Well-order of the features)**.** A well order can be set in $\mathcal{A}$ due it is a is a non-empty and finite set. $\leq_{\mathcal{A}}$ denotes a total order that defines the well order. If $A_1 \leq_{\mathcal{A}} A_2$ and $A_1 \neq A_2$ then $A_1 <_{\mathcal{A}} A_2$.

Definition 5 (Prefix)**.** The prefix of a pattern until a feature is the application $Prefix : (\mathcal{T} \times \mathcal{A}) \longrightarrow \mathcal{T}$ defined as: $Prefix(T, A) = T_{pr} \in \mathcal{T} \mid T_{pr}.O = T.O$ and $\forall A_{pr} \in T_{pr}.\mathcal{A}$ $A_{pr} \in T.\mathcal{A}$ and $A_{pr} \leq_{\mathcal{A}} A$.

Definition 6 (Last feature)**.** The last feature of a patterns is the application $Tail : \mathcal{T} \longrightarrow \mathcal{A}$ defined as: $Tail(T) = A_u \in T.\mathcal{A} \mid \forall A \in T.\mathcal{A}$ $A \leq_{\mathcal{A}} A_u$.

Definition 7 (Previous feature)**.** The previous feature of a feature is the application $Previous : \mathcal{A} \longrightarrow (\mathcal{A} \cup \{A_0\})$ defined as:

$$Previous(A) = \begin{cases} A_0 & \text{if } \forall A' \in \mathcal{A} \ A \leq_{\mathcal{A}} A' \\ Previous_0(A) & \text{otherwise} \end{cases} \quad (1)$$

where A_0 is a special feature, such that $A_0 \notin \mathcal{A}$. $Previous_0(A) = A_{prev} \in \mathcal{A} \mid \forall A' \in \mathcal{A}$ $A_m <_{\mathcal{A}} A$ and $A' \leq_{\mathcal{A}} A_{prev}$

The frequent similar pattern mining concepts are described below.

Definition 8 (Boolean similarity function)**.** A Boolean similarity function in $\mathcal{D}$ is an application $F : (\mathcal{T} \times \mathcal{O}) \longrightarrow$ { True,False}, such that $\forall \ T \in \mathcal{T}$, $O \in \mathcal{O}$; $T.O = O \Rightarrow F(T, O) = True$. $\mathcal{B}$ denotes the set of all Boolean similarity functions that can be defined in $\mathcal{D}$.

Definition 9 (Occurrences)**.** The occurrences of a pattern in $\mathcal{D}$ is the application $Occurrences_F : \mathcal{T} \longrightarrow 2^{\mathcal{O}}$, such that $Occurrences_F(T) = \{O \in \mathcal{O} \mid F(T, O) = True\}$.

Definition 10 (Frequency)**.** The frequency of a pattern in $\mathcal{D}$ is the application $Frequency_F : \mathcal{T} \longrightarrow \{1, 2, \ldots, \|\mathcal{O}\|\}$, such that $Frequency_F(T) = \|Occurrences_F(T)\|$.

Definition 11 (Frequent Similar Pattern)**.** A Pattern $T \in \mathcal{T}$ is a frequent similar pattern in $\mathcal{D}$ if $Frequency_F(T) \geq M$ where M is a minimum threshold. $\mathcal{M}$ denotes the domain of M, $\mathcal{M} = \{1, 2, \ldots, \|\mathcal{O}\|\}$. Also, $\mathcal{S}$ denotes the set of all frequent similar patterns in $\mathcal{D}$.

With the above definitions, a *Frequent Similar Pattern Mining* problem can be stated as follows: Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$, $F \in \mathcal{B}$ and $M \in \mathcal{M}$, the frequent similar pattern mining problem consists in finding the set of all frequent similar patterns $\mathcal{S}$.

However, pruning the search space of frequent similar patterns is needed for frequent similar pattern mining. Some definitions useful to this end are:

Definition 12 (Non-increasing monotonic boolean similarity function)**.** F is a non-increasing monotonic boolean similarity function iff $\forall O, T, T_{sup}$; $O \in \mathcal{O}$; $T \in \mathcal{T}$; $T_{sup} \in SupPatterns(T)$ $[F(T, O) = False] \Rightarrow [F(T_{sup}, O) = False]$. $\mathcal{N}$ denotes the set of all non-increasing monotonic boolean similarity functions that can be defined in $\mathcal{D}$.

Sample 1. Sample of non-increasing monotonic boolean similarity function: Equality

$$F_{eq}(T, O) = \begin{cases} True & \text{if } \forall A \in T.\mathcal{A}\ T.O[A] = O[A] \\ False & \text{otherwise} \end{cases} \quad (2)$$

Sample 2. Sample of non-increasing monotonic boolean similarity function: Product

$$F^{\alpha}_{prod}(T, O) = \begin{cases} True & \text{if } \prod_{A \in T.\mathcal{A}} C_A(T.O[A], O[A]) \geq \alpha \\ False & \text{otherwise} \end{cases} \quad (3)$$

where C_A is defined $\forall A \in \mathcal{A}$ as $C_A : (Domain(A) \times Domain(A)) \longrightarrow [0, 1]$

Lemma 1 (Monotony of the frequency)**.** $\forall T, T_{sup}; T \in \mathcal{T}$; $T_{sup} \in SupPatterns(T)$ $Frequency_F(T) \geq Frequency_F(T_{sup})$

The proof of this Lemma is a direct consequence of $F \in \mathcal{N}$ (Rodríguez-González et al., 2013).

Lemma 2 (Downward closure property)**.** $\forall T \in \mathcal{T}$ *if* $T \notin \mathcal{S}$ *then* $\forall T_{sup} \in SupPatterns(T)$ $T_{sup} \notin \mathcal{S}$.

The proof of this Lemma is a direct consequence of the monotony of the frequency (Rodríguez-González et al., 2013).

Finally, two important closed frequent pattern mining concepts (i.e., *Closed Pattern* and *Closure*) are formalized using the same notation.

Definition 13 (Closed Pattern)**.** Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ and $F = F_{eq}$, a pattern $T_e \in \mathcal{T}$ is a closed pattern in $\mathcal{D}$ if $\forall T_{eSup} \in SupPatterns(T_e)$ $Frequency_F(T_{eSup}) < Frequency_F(T_e)$. $\mathcal{E}_{eq}$ denotes the set of all closed patterns in $\mathcal{D}$ using $F = F_{eq}$.

Definition 14 (Closure)**.** Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ and $F = F_{eq}$, the closure is the application $Closure : \mathcal{T} \longrightarrow \mathcal{E}_{eq}$, such that $Closure(T) = T_{cl} \in \mathcal{E}_{eq} \mid T_{cl} \in SupPatterns(T)$ y $Frequency_F(T) = Frequency_F(T_{cl})$.

4. Combining frequent similar pattern and closed frequent similar pattern concepts

In this section, closed pattern concepts for *frequent itemset mining* will be extended for *frequent similar pattern mining*.

Definition 15 (Closed Similar Pattern)**.** Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ and $F \in \mathcal{N}$, a closed similar pattern in $\mathcal{D}$ is a pattern $T_e \in \mathcal{T} \mid \forall T_{eSup} \in SupPatterns(T_e)$ $Frequency_F(T_{eSup}) < Frequency_F(T_e)$, where $\mathcal{E}$ denotes the set of all closed similar pattern in $\mathcal{D}$ using F.

Definition 16 (Closure')**.** The closure' is the application $Closure' : \mathcal{T} \longrightarrow \mathcal{E}$, such that $Closure'(T) = T_{cl} \in \mathcal{E} \mid T_{cl} \in SupPatterns(T)$ and $Frequency_F(T) = Frequency_F(T_{cl})$.

Definition 17 (Closed Frequent Similar Pattern)**.** The closed frequent similar pattern mining problem consists in finding the set of all frequent similar patterns in $\mathcal{S} \cap \mathcal{E}$.

Definitions 15 and 16 are extensions of Definitions. 13 and 14 from Section 3, and by changing $F = F_{eq}$ to $F \in \mathcal{N}$ allows the use of any non-increasing monotonic boolean similarity function given as a result Definition 17.

Notice that, if Definition 16 is used in certain datasets with certain similarity functions, then the images of some patterns are not univocally defined. The following two samples show the contradictions that arise using Definition 16 given a dataset $\mathcal{D}$ and a non-increasing monotonic boolean similarity function $F \in \mathcal{N}$.

Sample 3. Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$, such that $\mathcal{O} = \{O_1, O_2\}$, $\mathcal{A} = \{X, Y\}$, $\mathcal{V} = \{x, y_1, y_2\}$ and $\mathcal{P}$ is defined as follow :

$\mathcal{P}$	X	Y
O_1	x	y_1
O_2	x	y_2

It can be noticed that $Domain(X) = \{x\}$ and $Domain(Y) = \{y_1, y_2\}$. Also, given $F \in \mathcal{N}$, $F = F^1_{prod}$ (F^{α}_{prod} defined on Sample 2) and the following comparison criteria:

C_X	x
x	1

C_Y	y_1	y_2
y_1	1	1
y_2	1	1

Consider the patterns $T_1 = (O_1, \{X\})$, $T_2 = (O_1, \{X, Y\})$ and $T_3 = (O_2, \{X, Y\})$. Notice that T_2 and T_3 are super-patterns of T_1 and both are closed similar patterns. Therefore, T_2 and T_3 can be the image of T_1 for the application $Closure'$ (from Definition 16) which is a contradiction of its own definition.

Sample 4. Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$, such that $\mathcal{O} = \{O_1, O_2\}$, $\mathcal{A} = \{X, Y, Z\}$, $\mathcal{V} = \{x, y_1, y_2, z_1, z_2\}$ and $\mathcal{P}$ is defined as follow:

$\mathcal{P}$	X	Y	Z
O_1	x	y_1	z_1
O_2	x	y_2	z_2

It can be noticed that $Domain(X) = \{x\}$, $Domain(Y) = \{y_1, y_2\}$ and $Domain(Z) = \{z_1, z_2\}$. Also, given $F \in \mathcal{N}$, $F = F^{0.5}_{prod}$ (F^{α}_{prod} defined on Sample 2) and the following comparison criteria:

C_X	x
x	1

C_Y	y_1	y_2
y_1	1	0.5
y_2	0.5	1

C_Z	z_1	z_2
z_1	1	0.5
z_2	0.5	1

Consider the patterns $T_1 = (O_1, \{X\})$, $T_2 = (O_1, \{X, Y\})$, $T_3 = (O_2, \{X, Y\})$, $T_4 = (O_1, \{X, Z\})$ and $T_5 = (O_2, \{X, Z\})$. Notice that $Frequency_F(T_1) = 2$, $Frequency_F(T_2) = 2$, $Frequency_F(T_3) = 2$, $Frequency_F(T_4) = 2$ and $Frequency_F(T_5) = 2$. Also, T_2, T_3, T_4 and T_5 are super-patterns of T_1 and are closed similar patterns. Therefore, T_2, T_3, T_4 and T_5 can be the image of T_1 for the application $Closure'$ (Definition 16) resulting again in a contradiction of its own definition.

The above samples suggest a new *Closure* definition (Definition 21) exclusive for *frequent similar patterns*. To that

end, the application *FClosure* is introduced jointly with some related lemmas as a consequence.

Definition 18 (FClosure)**.** The FClosure is the application $FClosure : \mathcal{T} \longrightarrow 2^{\mathcal{E}}$, such that $FClosure(T) = \{T_e \in \mathcal{E} \mid T_e \in SupPatterns(T),\ Frequency_F(T_e) = Frequency_F(T)\}$.

As example, Definition 18 applied on Sample 4 gives as a result $FClosure(T_1) = \{T_2, T_3, T_4, T_5\}$.

Definition 19 (Well-order in the domain of features)**.** Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$, $\forall A \in \mathcal{A}$ a well-order over $Domain(A)$ can be established due to it is a non-empty and finite set.

$\preceq_A$ denotes a total order that defines the well-order over $Domain(A)$. If $A_{v1} \preceq_A A_{v2}$ and $A_{v1} \neq A_{v2}$ then $A_{v1} \prec_A A_{v2}$

Definition 20 ($\preceq_L$ Relation)**.** $\preceq_L$ Relation is a binary relation in $\mathcal{T}$, given that $T_1 \preceq_L T_2$ iff $T_2 \in SupPatterns(T_1)$ or $\exists A_{dif} \in T_1.\mathcal{A}$, such that:

- $T_1.O[A_{dif}] \prec_{A_{dif}} T_2.O[A_{dif}]$.
- $\forall A_m \in T_2.\mathcal{A} \mid A_m \prec_{\mathcal{A}} A_{dif}\ \ A_m \in T_1.\mathcal{A}$ and $T_1.O[A_m] = T_2.O[A_m]$.

Lemma 3 (Lexicographic order)**.** *$\preceq_L$ defines a well-order in $\mathcal{T}$.*

Proof. $\forall A \in \mathcal{A}$, $(\mathcal{A}, \preceq_A)$ and $(Domain(A), \preceq_A)$ are well-ordered sets. Therefore $\preceq_L$ defines a lexicographic order in $\mathcal{T}$ (Weisstein, 2002). □

Sample 5 (Sample of lexicographic order)**.** Given the dataset $\mathcal{D}$ and the similarity function F of Sample 4, $\preceq_{\mathcal{A}}$, such that $X \prec_{\mathcal{A}} Y \prec_{\mathcal{A}} Z$, a total order $\preceq_Y$, such that $y_1 \preceq_Y y_2$, and a total order $\preceq_Z$, such that, $yz_1 \preceq_Y z_2$, the lexicographic order $\preceq_L$ in $\mathcal{T}$ follows the well-order:

1. $(O_1, \{X\}) = x$
2. $(O_1, \{X, Y\}) = xy_1$
3. $(O_1, \{X, Y, Z\}) = xy_1z_1$
4. $(O_2, \{X, Y\}) = xy_2$
5. $(O_2, \{X, Y, Z\}) = xy_2z_2$
6. $(O_1, \{X, Z\}) = xz_1$
7. $(O_2, \{X, Z\}) = xz_2$
8. $(O_1, \{Y\}) = y_1$
9. $(O_1, \{Y, Z\}) = y_1z_1$
10. $(O_2, \{Y\}) = y_2$
11. $(O_2, \{Y, Z\}) = y_2z_2$
12. $(O_1, \{Z\}) = z_1$
13. $(O_2, \{Z\}) = z_2$

Now, the new *Closure* definition, named Lexicographic closure, is introduced:

Definition 21 (Lexicographic closure for non-increasing monotonic boolean similarity function)**.** The lexicographic closure of a pattern is the application $FCL : \mathcal{T} \longrightarrow \mathcal{E}$, such that, $FCL(T) = T_{fcl} \in \mathcal{E} \mid T_{fcl} \in FClosure(T)$, T_{fcl} is the minimum pattern in $FClosure(T)$.

Sample 6. Sample of the lexicographic closure. Using Sample 5 and applying Definition 21:

- $FCL(O_1, \{X\})) = (O_1, \{X, Y\})$
- $FCL(O_1, \{X, Y\}) = (O_2, \{X, Y, Z\})$

The lexicographic closure *FCL* is an application defined for all its domain due to Lemma 5, whose proof is based on the following lemma:

Lemma 4 (FClosure is never empty)**.** *Given a dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ and $F \in \mathcal{N}$, $\forall T \in \mathcal{T}$ $\|FClosure(T)\| \geq 1$.*

Proof. $\forall T \in \mathcal{T}$ $T \in \mathcal{E}$ or $T \notin \mathcal{E}$.

If $T \in \mathcal{E}$:

1. $T \in FClosure(T)$ because $T \in \mathcal{E}$ and $T \in SupPatterns(T)$.
2. $FClosure(T) \geq 1$ by 1.

If $T \notin \mathcal{E}$:

1. Build $H(T) \subseteq \mathcal{T}$ as follows:

$$H(T) = \{H \in SupPatterns(T) \mid Frequency_F(H) = Frequency_F(T)\}.$$

2. $H(T)$ is non-empty because $T \notin \mathcal{E}$. Also, $H(T)$ is a finite set because $H(T) \subseteq \mathcal{T}$.
3. The relation $H_{tam} \subseteq (H(T) \times H(T))$ is defined as follows:

$$H_{tam} = \{(H_1, H_2) \in (H(T) \times H(T)) \mid \|H_1.\mathcal{A}\| \geq \|H_2.\mathcal{A}\|\}.$$

H_{tam} is a total order in $H(T)$.
4. The total ordered set $(H(T), H_{tam})$ is a well-ordered set because $H(T)$ is a non-empty and finite set and H_{tam} is a total order. Consequently, there is a minimum element in $H(T)$, that is $\exists H_m \in H(T) \mid \forall H \in H(T)\ \|H_m.\mathcal{A}\| \geq \|H.\mathcal{A}\|$.
5. $H_m \in \mathcal{E}$ by 4.
6. $H_m \in FClosure(T)$ by 5.
7. $FClosure(T) \geq 1$ by 6.

□

Lemma 5 (FCL is well-defined)**.** *FCL is well-defined $\forall T \in \mathcal{T}$.*

Proof.

1. $\forall T \in \mathcal{T}$ $FClosure(T)$ is non-empty by Lemma 4.
2. $\forall T \in \mathcal{T}$ $FClosure(T) \subseteq \mathcal{E} \subseteq \mathcal{T}$ and $\mathcal{T}$ has a well-order using $\preceq_L$.
3. $\forall T \in \mathcal{T}$ $FCL(T)$ is well-defined by 1, 2 and because for all well-ordered and non-empty sets there is a minimum element.

□

5. Closed frequent similar pattern algorithm (CFSP-Miner)

In this section we introduce the *CFSP-Miner* algorithm for mining closed frequent similar patterns when similarity functions hold the downward closure property from Lemma 2. The algorithm works by traversing a tree, defined by a father-child relation that contains all the closed frequent similar patterns. Before we introduce the father-child relation definition, we need to introduce a new definition:

Definition 22 (Minor prefix with equal FCL)**.** The minor prefix with equal FCL of a closed similar pattern is the application $PrFCL : \mathcal{E} \longrightarrow \mathcal{A}$, such that $PrFCL(T) = A_{pr} \in T.\mathcal{A} \mid FCL(Prefix(T, A_{pr})) = T$ and $\forall A_m \in T.\mathcal{A} \mid A_m \prec_{\mathcal{A}} A_{pr})\ \ FCL(Prefix(T, A_m)) \neq T$.

Definition 23 (Father-child relation among closed similar patterns)**.** Father of closed similar pattern is the application $Father : \mathcal{E} \longrightarrow \mathcal{E} \cup T_{ROOT}$, such that:

$$Father(T_e) = \begin{cases} FCL(Prefix(T_e, Previous(PrFCL(T_e)))) & if\ Previous(PrFCL(T_e)) = A_0 \\ T_{ROOT} & otherwise \end{cases} \quad (4)$$

where T_{ROOT} is a special pattern, such that $T_{ROOT} \notin \mathcal{T}$. Notice that *Father* is defined for all its domain.

Sample 7. Sample of the Father-child relation. Using Definition 23 in Sample 5, it can be seen that:

- $Father((O_1, \{X, Y, Z\})) = (O_1, \{X, Y\})$
- $Father((O_2, \{X, Y\})) = (O_1, \{X, Y\})$
- $Father((O_1, \{X, Y\})) = T_{ROOT}$
- $Father((O_2, \{X, Y, Z\})) = (O_2, \{X, Y\})$

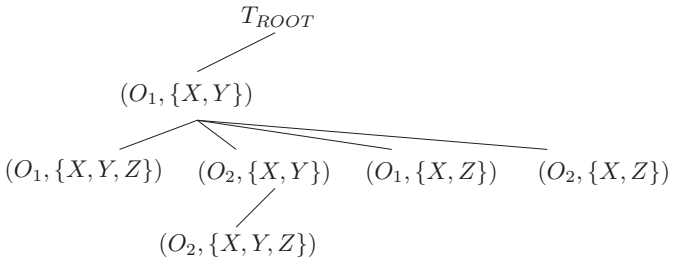

Fig. 2. Sample of the father-child relation tree.

Definition 23 (father-child relations) can be use to built a graph which is actually a tree.

Definition 24 (Father-child relation graph)**.** G is a the undirected graph, such that $G = (V, A)$ where $V = \mathcal{E} \cup \{T_{ROOT}\}$ and $A = \{(T_1, T_2) \in (V \times V) \mid Father(T_2) = T_1\}$.

Lemma 6 (G is a tree)**.** *The father-child relation graph G is a tree.*

The proof of this Lemma is based on Lemma 7.

Proof.

1. G is acyclic by Lemma 7.
2. $\|A\| = \|V\| - 1$ because *Father* is an application defined $\forall T \in \mathcal{E}$ but not for T_{ROOT}.
3. G is a tree by 1 and 2.

□

As example, Fig. 2 shows the father-child relation tree from Sample 5.

Lemma 7 (PrFCL grows from father to child)**.** $PrFCL(T_{father}) <_{\mathcal{A}} PrFCL(T)\ \forall T \in \mathcal{E} \mid T_{father} = Father(T) \neq T_{ROOT}$.

Proof.

1. $T_{father} = FCL(Prefix(T, Previous(PrFCL(T))))$ because $T_{father} = Pather(T)$.
2. $PrFCL(T_{father}) \leq_{\mathcal{A}} Previous(PrFCL(T))$ by 1 and Lemma 8.
3. $PrFCL(T_{father}) <_{\mathcal{A}} PrFCL(T)$ by 2.

□

Lemma 8 (PrFCL lower bound)**.** $Tail(T) \leq_{\mathcal{A}} PrFCL(T_{fcl})\ \forall T \in \mathcal{T}$ *and* $T_{fcl} = FCL(T)$.

Proof.

1. Build $Intermediate(T) \subseteq \mathcal{T}$ as follows: $Intermediate(T) = \{Z \in SupPatterns(T)\ y\ T_{fcl} \in SupPatterns(Z)\}$.
2. $\forall Z \in SupPatterns(T)$, $FCL(Z) = T_{fcl}$.
3. $\forall A_{int} \in \mathcal{A} \mid Tail(T) \leq_{\mathcal{A}} A_{int} \leq_{\mathcal{A}} Tail(T_{fcl})$, $Prefix(T_{fcl}, A_{int}) \in Intermediate(T)$.
4. By 2 and 3: $\forall A_{int} \in \mathcal{A} \mid Tail(T) \leq_{\mathcal{A}} A_{int} \leq_{\mathcal{A}} Tail(T_{fcl})$, $FCL(Prefix(T_{fcl}, A_{int})) = T_{fcl}$.
5. $Tail(T) \leq_{\mathcal{A}} PrFCL(T_{fcl})$ by 4.

□

Having the tree that contains all closed similar patterns, a traversal tree from the root can be defined to find all of them. This can be done using a children application defined below:

Definition 25 (Children)**.** Children of a closed similar pattern is the application $Children : \mathcal{E} \cup T_{ROOT} \longrightarrow 2^{\mathcal{E}}$, such that $Children(T) = \{T_{child} \in \mathcal{E} \mid Father(T_{child}) = T\}$.

To find the children of a closed similar pattern, we will use Lemma 9. Before introducing Lemma 9, the following two definitions are needed:

Definition 26 (Extensions of a pattern)**.** Extensions of a pattern is the application $Extensions : \mathcal{T} \longrightarrow 2^{\mathcal{E}}$, given that

$$Extensions(T) = \left\{ T_{ext} \in \mathcal{E} \mid T_{ext} \neq FCL(T) \text{ and } \begin{array}{l} \exists T_{+1} \in \mathcal{T},\ such that: \\ T = Prefix(T_{+1}, Prevoius(Tail(T_{+1}))) \\ T_{ext} = FCL(T_{+1}) \\ T_{+1} = Prefix(T_{ext}, Tail(T_{+1}) \end{array} \right\}$$

As example, using Definition 26 in Sample 5, it can be seen that:

- $(O_2, \{X, Y\}) \in Extensions((O_1, \{X\}))$, in this case $T_{+1} = (O_2, \{X, Y\})$
- $(O_2, \{X, Y, Z\}) \in Extensions((O_2, \{X, Y\}))$, in this case $T_{+1} = (O_2, \{X, Y, Z\})$

Definition 27 (inverse of FCL)**.** The inverse of FCL of a closed similar pattern is the application $IFCL : \mathcal{E} \longrightarrow 2^{\mathcal{E}}$, such that $IFCL(T) = \{T_{ifcl} \in \mathcal{E} \mid FCL(T_{ifcl}) = T\}$.

Lemma 9 (Finding children)**.** $\forall T \in \mathcal{E}$ *and* $\forall T_{child} \in \mathcal{E}$, $T_{child} \in Children(T)$ *iff* $T_{child} \in Extensions(T_{ifcl})$ *where* $T_{ifcl} \in IFCL(T)$.

The proof of Lemma 9 requires Lemmas 10, 11 and 12 that will be provided later.

Proof. It starts proving that if $T_{child} \in Extensions(T_{ifcl})$ for some $T_{ifcl} \in IFCL(T)$ then $T_{child} \in Children(T)$.

1. $T_{ifcl} = Prefix(T_{child}, Previous(PrFCL(T_{child})))$ by $T_{child} \in Extensions(T_{ifcl})$ and by Lemma 10.
2. $FCL(T_{ifcl}) = T$ because $T_{ifcl} \in IFCL(T)$.
3. $Father(T_{child}) = T$ by 1 and 2.
4. $T_{child} \in Children(T)$ by 3.

Now, it is proved that if $T_{child} \in Children(T)$ then $T_{child} \in Extensions(T_{ifcl})$ for some $T_{ifcl} \in IFCL(T)$.

1. Give $T_{ifcl} = Prefix(T_{child}, Previous(PrFCL(T_{child})))$:
2. $T_{child} \in Extensions(T_{ifcl})$ by Lemma 10.
3. $T = FCL(T_{ifcl})$ because $T_{child} \in Children(T)$.
4. $T_{ifcl} \in IFCL(T)$ by 3.

□

Lemma 10 (Equivalence with Extensions)**.** $\forall T_{ext} \in \mathcal{E}$ *and* $\forall T \in \mathcal{T}$, $T_{ext} \in Extensions(T)$ *iff* $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$.

Proof. It starts proving that if $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$ then $T_{ext} \in Extensions(T)$.

1. $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$.
2. Let $T_{+1} = (T.O, T.\mathcal{A} \cup PrFCL(T_{ext}))$ be a pattern, notice that $T = Prefix(T_{+1}, Previous(Tail(T_{+1})))$ and $T_{+1} = Prefix(T_{ext}, Tail(T_{+1}))$.
3. $T_{ext} = FCL(T_{+1})$ by definition of PrFCL.
4. $T_{ext} \neq FCL(T)$ by 1 and definition of PrFCL.
5. $T_{ext} \in Extensions(T)$ by 2, 3 and 4.

Now, it is proved that if $T_{ext} \in Extensions(T)$ then $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$.

1. $T_{ext} \in Extensions(T)$ implies that $\exists T_{+1} \in \mathcal{T}$.
2. $T_{ext} = FCL(T_{+1})$
3. $T_{ext} \neq FCL(T)$ because $T_{ext} \in Extensions(T)$.
4. $PrFCL(T_{ext}) = Tail(T_{+1})$ by 2 and 3.

5. $Prefix(T_{ext}, Previous(PrFCL(T_{ext}))) = T$ by 4 and because $Previous(Tail(T_{+1})) = Tail(T)$.

□

Lemma 11 (The image of a closed similar pattern is itself). $FCL(T) = T \ \forall T \in \mathcal{E}$

Proof.

1. $\forall T \in \mathcal{E} \quad T \in FClosure(T)$.
2. $\forall T \in \mathcal{E}$ s $\nexists T_x \in FClosure(T) \mid T_x \neq T$ because if there is T_x, then it contradicts that fact that $T \in \mathcal{E}$.
3. $\forall T \in \mathcal{E} \ FCL(T) = T$ by 1 and 2.

□

Lemma 12 (IFCL is never empty). $\|IFCL(T)\| \geq 1 \ \forall T \in \mathcal{E}$.

Proof. It is sufficient to keep in mind that $FCL(T) = T \ \forall T \in \mathcal{E}$. □

The baseline of *CFSP-Miner* (Algorithm 1) is traversing the

Algorithm 1: CFSP-Miner $_{Baseline}(\mathcal{D}, F, M, T)$.

Input: Dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$,
Similarity Function $F \in \mathcal{N}$,
Minimum Frequency Threshold $M \in \mathcal{M}$
Frequent Closed Similar Pattern $T \in \mathcal{S} \cap \mathcal{E}_{eq}$
Output: Frequent Closed Similar Patterns Set $\mathcal{SE}_{eq}$
foreach $T_{ifcl} \in IFCL(T)$ **do**
  foreach $T_{child} \in Extensions(T_{ifcl})$ **do**
    if $Frequency_F(T_{child}) \geq M$ **then**
      $\mathcal{SE}_{eq} \leftarrow$
      $\mathcal{SE}_{eq} \cup T_{child} \cup$ CFSP-Miner$_{Baseline}(\mathcal{D}, F, M, T_{child})$

frequent closed similar patterns set starting from the special pattern T_{ROOT} by means of Lemma 9, which is used to find the set children of a closed similar pattern.

Let's see an example of a run of the *CFSP-Miner*$_{Baseline}$ algorithm, for dataset and similarity function of Sample 4 and minimum frequency threshold $M = 2$. The sample of the father-child relation tree (Fig. 2) will help us. Since the algorithm is recursive, by simplicity we will focus on explaining only one iteration. The selected iteration is that which has as input $T = (O_1, \{X, Y\})$, $Frequency_F(T) = 2$.

First, all elements that belong to $IFCL(T)$ are found. Remember that $IFCL(T)$ is the set of all elements that have T as lexicographic closure. Then $IFCL(T) = \{(O_1, \{X, Y\}), (O_1, \{X\}), (O_1, \{Y\})\}$ because:

- Always T belongs to $IFCL(T)$,
- $FCL((O_1, \{X\})) = T$ and $Frequency_F((O_1, \{X\})) = 2$.
- $FCL((O_1, \{Y\})) = T$ and $Frequency_F((O_1, \{Y\})) = 2$.

For each element in $IFCL(T)$, its extensions are searched. *Extensions* of a pattern T' is the set that contains the lexicographic closure of each element obtained by adding one feature to T' after its last feature.

- $Extensions((O_1, \{X, Y\})) = \{FCL((O_1, \{X, Y, Z\}))\}$ because in Sample 4 there is only another feature, the Z feature, and $\{FCL((O_1, \{X, Y, Z\}))\} = \{(O_1, \{X, Y, Z\})\}$ because there is no pattern that contains it. Then $Extensions((O_1, \{X, Y\})) = \{(O_1, \{X, Y, Z\})\}$.
- $Extensions((O_1, \{X\})) = \{FCL((O_1, \{X, Z\})), FCL((O_2, \{X, Y\})), FCL((O_2, \{X, Z\}))\}$ because:
  - $(O_1, \{X, Z\})$ is the first way to add a feature to $(O_1, \{X\})$
  - $(O_2, \{X, Y\})$ is another way to add a feature to $(O_1, \{X\})$. Notice that $O_1[X] = O_2[X]$
  - $(O_2, \{X, Z\})$ is analogous to the previous case.

  Also, $FCL((O_1, \{X, Z\})) = (O_1, \{X, Z\})$ because its frequency is 2 and there is no pattern that contains it. By the same reason $FCL((O_2, \{X, Y\})) = (O_2, \{X, Y\})$ and $FCL((O_2, \{X, Z\})) = (O_2, \{X, Z\})$. Then $Extensions((O_1, \{X\})) = \{(O_1, \{X, Z\}), (O_2, \{X, Y\}), (O_2, \{X, Z\})\}$.
- $Extensions((O_1, \{Y\}))$ is empty because there is no $T_ext \neq FCL(T)$, such that $T_ext = FCL(T')$ and $T' = Prefix(T_{ext}, Tail(T'))$

The children of $(O_1, \{X, Y\})$ are $(O_1, \{X, Y, Z\})$, $(O_1, \{X, Z\})$, $(O_2, \{X, Y\})$, $(O_2, \{X, Z\})$. The children with frequency greater than or equal to 2 are added to the frequent closed similar patterns set, and each one is the input of another recursive iteration.

The most important step of CFSPs-Miner consists in finding the *IFCL* set of a closed similar pattern. To perform this step efficiently, a prune property (Lemma 13), and a necessary and sufficient condition (Lemmas 14 and 16) for a pattern belonging to *IFCL*, are introduced.

Lemma 13 (Pruning the search space of the *IFCL set*). *If* $FCL(T_{sup}) \neq T_e$, *then* $FCL(T) \neq T_e \ \forall T \in \mathcal{T}, \ \forall T_{sup} \in SupPatterns(T), \forall T_e \in \mathcal{E} \mid T_e \in SupPatterns(T_{sup})$.

Proof. There are two cases: $Frequency_F(T_{sup}) \neq Frequency_F(T_e)$ or $Frequency_F(T_{sup}) = Frequency_F(T_e)$.

If $Frequency_F(T_{sup}) \neq Frequency_F(T_e)$:

1. $Frequency_F(T) \geq Frequency_F(T_{sup})$ by Lemma 1.
2. $Frequency_F(T_{sup}) > Frequency_F(T_e)$ by Lemma 1.
3. $Frequency_F(T) > Frequency_F(T_e)$ by 1 and 2.
4. $FCL(T) \neq T_e$ by 3.

If $Frequency_F(T_{sup}) = Frequency_F(T_e)$:

There are two subcases: $Frequency_F(T_{sup}) < Frequency_F(T)$ or $Frequency_F(T_{sup}) = Frequency_F(T)$. If $Frequency_F(T_{sup}) < Frequency_F(T)$ it should be noted that $Frequency_F(T) \neq Frequency_F(T_e)$.

If $Frequency_F(T_{sup}) = Frequency_F(T)$:

1. $T_e \in FClosure(T_{sup})$ by Definition 18.
2. $FCL(T_{sup}) <_L T_e$ by Definition 18 and due to $T_e \neq FCL(T_{sup})$.
3. $T_e \in FClosure(T)$ and $FCL(T_{sup}) \in FClosure(T)$ by Definition 18.
4. $FCL(T) \leq_L FCL(T_{sup}) <_L T_e$ by 2 and 3.
5. $FCL(T) \neq T_e$ by 4.

□

Lemma 14 (Necessary condition for belonging to *IFCL*). *If* $Frequency_F(T) \neq Frequency_F(T_e)$ *or* $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$, *then* $FCL(T) \neq T_e \ \forall T \in \mathcal{T}, \ \forall T_e \in \mathcal{E} \mid T_e \in SupPatterns(T)$.

Proof. For the case $Frequency_F(T) \neq Frequency_F(T_e)$, it is only necessary to note that the definition of *FCL* requires equal $Frequency_F$.

For the other case, $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$:

1. $Prefix(T_e, Previous(PrFCL(T_e))) \in SupPatterns(T)$ because $T_e \in SupPatterns(T)$ and $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$.
2. $FCL(Prefix(T_e, Previous(PrFCL(T_e)))) \neq T_e$ by Definition 22.
3. $FCL(T) \neq T_e$ by 1, 2 and Lemma 14.

□

To introduce the sufficient condition for a pattern belonging to the *IFCL*, Definitions 28 and 29, and Lemma 15 are presented below.

Definition 28 (Super patterns with one more feature and equal frequency). Super patterns with one more feature and equal frequency of a pattern is the application $SupPatterns^{=}_{+1} : \mathcal{T} \longrightarrow 2^{\mathcal{T}}$, such that:

$SupPatterns^{=}_{+1}(T) = \{T^{=}_{+1} \in SupPatterns(T) \mid \|T^{=}_{+1}.\mathcal{A}\| = \|T.\mathcal{A}\| + 1$ and $Frequency_F(T^{=}_{+1}) = Frequency_F(T)\}$

Definition 29 (Derivation of a pattern)**.** Derivation of a pattern is the application $Derivation : \mathcal{T} \longrightarrow \mathcal{T}$, such that:

$$Derivation(T) = \begin{cases} \min\limits_{T' \in SupPatterns^{=}_{+1}(T)} T' & if\ \|SupPatterns^{=}_{+1}(T)\| > 0 \\ T & otherwise \end{cases} \quad (5)$$

Notice that *Derivation* is defined for all its domain.

Lemma 15 (Property of derivation)**.** $FCL(T) = FCL(Derivation(T))\ \forall T \in \mathcal{T}$.

Proof. First, $FCL(Derivation(T)) \leq_L FCL(T)$ is proved:

1. $FCL(Derivation(T)) \in SupPatterns(T)$ because $FCL(Derivation(T)) \in SupPatterns(Derivation(T))$ and $Derivation(T) \in SupPatterns(T)$.
2. $Frequency_F(FCL(Derivation(T))) = Frequency_F(Derivation(T)) = Frequency_F(T)$ by Definitions 21 and 29.
3. $FCL(Derivation(T)) \in FClosure(T)$ by 1 and 2.
4. $FCL(Derivation(T)) \leq_L FCL(T)$ by 3 and by Definition 21.

Now, $FCL(T) \leq_L FCL(Derivation(T))$ is proved:

1. $FCL(T) \in SupPatterns(Derivation(T))$ by Definitions 21 and 29.
2. $Frequency_F(T) = Frequency_F(Derivation(T)) = Frequency_F(FCL(Derivation(T)))$ by Definitions 21 and 29.
3. $FCL(T) \in FClosure(Derivation(T))$ by 1 and 2.
4. $FCL(T) \leq_L FCL(Derivation(T))$ by 3 and by Definition 21.

After that, $FCL(T) = FCL(Derivation(T))$ because $\leq_L$ is a total order. □

Lemma 16 (Sufficient condition to belong to IFCL)**.** *If* $FCL(Derivation(T)) = T_e$, *then* $FCL(T) = T_e\ \ \forall T \in \mathcal{T}$ *and* $\forall T_e \in \mathcal{E}$.

Proof. This Lemma is a direct consequence of Lemma 15. □

At this point, it can be obtained the $IFCL(T_e)\ \forall T_e \in \mathcal{E}$ in an efficient way, traversing all subpatterns of T_e using Lemmas 13 and 14 to prune the search space, and Lemma 16 to check if a subpattern belongs to $IFCL(T_e)$.

To achieve an efficient implementation of *CFSP-Miner* (Algorithm 2) , the sets *IFCL* and *Extensions* are simultaneously built. This is accomplished by expanding, in lexicographical order, the patterns by adding new features and their values.

When the new expanded pattern is the first to have the same frequency as its non-expanded version, its subpatterns, including the feature *Tail* and with equal *FCL*, are also expanded. Each subpattern can be used to obtain new children.

Before expanding a pattern and to avoid repeating the same analysis, it should be verified that there is no superpattern less (using $\leq_L$) than it with the same frequency.

To define $\leq_L$ it is necessary to establish a well-order in $\mathcal{A}$ from the cardinality of the domains of the features. However, to define $\leq_{Domain(A)}\ \forall A \in \mathcal{A}$ for numerical features, the conventional order defined by $\leq$ is used, whereas for non-numerical features, the order in which values appear in the data set is used.

Let's see an example of a run of the *CFSP-Miner* algorithm for dataset and similarity function of sample 4 and minimum frequency threshold $M = 2$. *CFSP-Miner* starts from the special pattern T_{ROOT}.

Each T_{exp} is found by extending T_{ROOT} with one feature value. Then T_{exp} iterates over $\{(O_1, \{X\}), (O_1, \{Y\}), (O_1, \{Z\}), (O_2, \{Y\}), (O_2, \{Z\})\}$. For each T_{exp}, $Frequency_F(T_{\text{exp}}) = 2$.

The flag $DerivationFound = True$ means that it has a *derivation*: a pattern with equal frequency and one more field and it is the smallest one that was found expanding in Lexicographic Order. In other words, $DerivationFound = True$ implies that it is not closed, but the path where will appear the *FCL* of this pattern, has been found.

Algorithm 2: CFSP-Miner$(\mathcal{D}, F, M, T)$.

Input: Dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$,
Similarity Function $F \in \mathcal{N}$,
Minimum Frequency Threshold $M \in \mathcal{M}$
Frequent Closed Similar Pattern $T \in \mathcal{S} \cap \mathcal{E}_{eq}$
Output: Frequent Closed Similar Patterns Set $\mathcal{SE}_{eq}$
if $\exists T_{men} \in SupPatterns^{=}_{+1}(T) \mid T_{men} <_L T$ **then**
  return
$DerivationFound \leftarrow False$
foreach $T_{\text{exp}} \in SupPatterns(T)$ *such that* $\|T_{\text{exp}}.\mathcal{A}\| = \|T.\mathcal{A}\| + 1$, $Tail(T) <_{\mathcal{A}} Tail(T_{\text{exp}})$ **do**
  if $\neg DerivationFound$ *and* $Frequency_F(T_{\text{exp}}) = Frequency_F(T)$ **then**
    $DerivationFound \leftarrow True$
    foreach $T_{equal-fcl} \in \mathcal{T}$ *such that* $T_{\text{exp}} \in SupPatterns(T_{equal-fcl}), Tail(T_{equal-fcl}) = Tail(T_{\text{exp}}), FCL(T_{equal-fcl}) = FCL(T_{\text{exp}})$ **do**
      $\mathcal{SE}_{eq} \leftarrow \mathcal{SE}_{eq} \cup$ CFSP-Miner$(\mathcal{D}, F, M, T_{equal-fcl})$
  else
    $\mathcal{SE}_{eq} \leftarrow \mathcal{SE}_{eq} \cup$ CFSP-Miner$(\mathcal{D}, F, M, T_{\text{exp}})$
if $\neg DerivationFound$ **then**
  $\mathcal{SE}_{eq} \leftarrow \mathcal{SE}_{eq} \cup T$

For each T_{exp}, the frequency of T_{exp} is different from the frequency of T_{ROOT}. Then a recursive call to the algorithm will be done for each T_{exp} . So far no closed pattern has been found.

We do not explain all recursive calls, instead, we focus on *CFSP-Miner*$(\mathcal{D}, F, M, (O_1, \{X\}))$. For this call each T_{exp} is found by extending $(O_1, \{X\})$ with one feature value. Then T_{exp} iterates over $\{(O_1, \{X, Y\}), (O_1, \{Y, Z\})\}$. For each T_{exp}, $Frequency_F(T_{\text{exp}}) = 2$.

Note that $T_{\text{exp}} = (O_1, \{X, Y\})$ is a *derivation* of $(O_1, \{X\})$ because it has the same frequency and it is the first pattern that was found expanding in Lexicographic Order. Therefore the next recursive calls will be for each $T_{equal-fcl}$: subsets of $(O_1, \{X, Y\})$ that have the same *FCL* of $(O_1, \{X, Y\})$ and retain the last feature of $(O_1, \{X, Y\})$. In this case the only one $T_{equal-fcl}$ is $(O_1, \{X, Y\})$ itself. Finally, in its recursive call, it is added to the Frequent Closed Similar Patterns Set because it has no derivation. Subsequent recursive calls will not be explained.

An analogous situation is observed for $T_{\text{exp}} = (O_1, \{Y, Z\})$, then $(O_1, \{Y, Z\})$ is also added to the Frequent Closed Similar Patterns Set.

Following all the recursive calls, all Closed Frequent Similar Patterns will appear.

5.1. Complexity analysis

To analyze the computational complexity of *CFSP-Miner*, we examine the worst case. The input of *CFSP-Miner* consists of a dataset of m objects $\mathcal{O}$, each one described by a set of n features $\mathcal{A}$. Therefore, the size of input is mn. The size of output, on the other hand, is the number c of frequent similar closed patterns. We assume that the cost of computing the similarity between a pattern and an object is $O(n)$, since n attributes have to be compared.

Let's focus on the number of operations that must be performed to find each frequent similar closed pattern. For each frequent similar closed pattern candidate, it must be compared against all objects in the dataset in order to compute its frequency. Therefore, computing the frequency of a pattern candidate is $O(mn)$. For each similar closed pattern candidate, its *IFCL* must also be computed, which is $O(n^2)$. Therefore, the complexity of *CFSP-Miner* is $O(cmn^3)$.

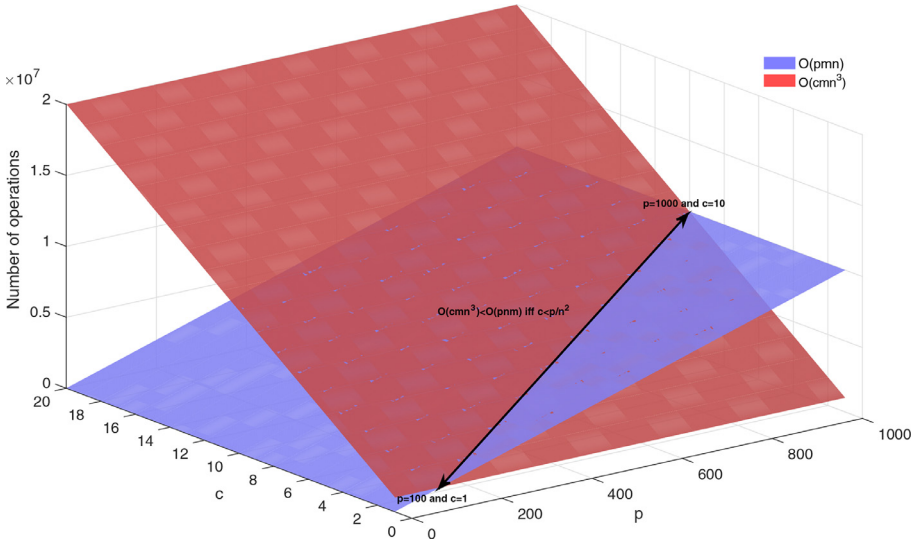

Fig. 3. A particular case of the number of operations required by *CFSP-Miner, ObjectMiner* and *STreeDC-Miner* varying the number of frequent similar patterns p from [1, 1000], the number of frequent similar closed patterns c from [1,20], and considering $m = 1000$ objects and $n = 10$ features.

A similar analysis can be done for the *ObjectMiner* and *STreeDC-Miner* algorithms. Given that *IFCL* does not have to be computed, the computational complexity in both cases is $O(pmn)$, where p is the number of frequent similar patterns obtained as output.

If the number of frequent similar closed patterns is similar to the number of frequent similar patterns, note that the computational complexity of *ObjectMiner* and *STreeDC-Miner* is smaller than the computational complexity of *CFSP-Miner*. If the number of similar closed patterns is much smaller than the number of frequent similar patterns, on the other hand the computational complexity of *CFSP-Miner* will be smaller than the computational complexity of *ObjectMiner* and *STreeDC-Miner*. These situations will be experimentally verified in the next section.

Analytically, $\forall c > 0,\ \exists p > 0,\ c < p/n^2$, such that, $cmn^3 < pmn$. Fig. 3 shows a particular case about the number of operations required by *CFSP-Miner* and *ObjectMiner* and *STreeDC-Miner* at varying the number of frequent similar patterns p from [1, 1000] and the number of frequent similar closed patterns c from [1,20]; and considering $m = 1000$ objects and $n = 10$ features. This figure illustrates that there is a region where the number of operations required by *CFSP-Miner* is less than the number of operations required by *ObjectMiner* and *STreeDC-Miner*

It is also worth noting that the number of patterns in the output for the three algorithms is exponentially bounded regarding the input size.

6. Experimental results

In this section, the performance of the proposed algorithm *CFSP-Miner* is evaluated. We divide the experimental results into two subsections. In the first subsection (Section 6.1), a comparison in terms of time needed for mining the frequent similar patterns by each algorithm (*CFSP-Miner, ObjectMiner* and *STreeDC-Miner*) is presented. It is important to highlight that *ObjectMiner* and *STreeDC-Miner* algorithms obtain the set of all frequent similar patterns $\mathcal{S}$, whereas the proposed algorithm, *CFSP-Miner*, obtains only the set of all closed frequent similar patterns $\mathcal{S} \cap \mathcal{E}$. In the second subsection (Section 6.2), the scalability of *CFSP-Miner* algorithm is shown.

The experiments were done on a PC with a Intel(R) Core(TM)2 Duo at 1.83 Ghz and 2Gb of RAM. The *CFSP-Miner* algorithm was implemented in CSharp. For *ObjectMiner* and *STreeDC-Miner* algorithms, the Java implementations of their authors were used.

Table 1
Description of datasets.

Datasets	Objects	Non-numerical features	Numerical features
Dermatology	366	34	1
Flags	194	20	10
Mushroom	8124	22	0
Waveform	5000	40	1
Vehicle	946	1	18
Wine	178	1	13

6.1. *Efficiency of* CFSP-Miner *algorithm*

Table 1 gives a description of the datasets[1] used. A different similarity function can be defined for each particular problem. However, the goal of these experiments is not to solve a particular problem but simply to assess the efficiency of the proposed algorithm. That is why the similarity functions are not tied to any particular problem in this work.

For this experiment, the similarity function F^{α}_{prod}, defined in Sample 2, with $\alpha = 0.1$ was used. As comparison criteria we used the following:

If A is numerical feature:

$$C_A(A_{v1}, A_{v2}) = \begin{cases} ss1 & \text{if } \frac{|A_{v1} - A_{v2}|}{Max_A - Min_A} \leq \theta \\ 0 & otherwise \end{cases}$$

If A is a non-numerical feature:

$$C_A(A_{v1}, A_{v2}) = \begin{cases} 1 & \text{if } A_{v1} = A_{v2} \\ 0 & otherwise \end{cases}$$

Fig. 4 shows the runtime of the algorithms for each data set when the frequency percent $M\%$ is varied from 10 to 90. Notice that, $M\% = \frac{M}{\|O\|}$ where M is the minimum frequency threshold. *CFSP-Miner* outperforms the runtimes of *ObjectMiner* and *STreeDC-Miner* in 4 (*Flags, Dermatology, Mushroom*, and *Waveform*) of the 6 datasets. While the opposite happens in *Vehicle* and *Wine* datasets.

This behavior is explained by observing the percent of frequent similar patterns that are closed (*Percent* in Fig. 5) and keeping in

[1] http://archive.ics.uci.edu/ml/datasets.html.

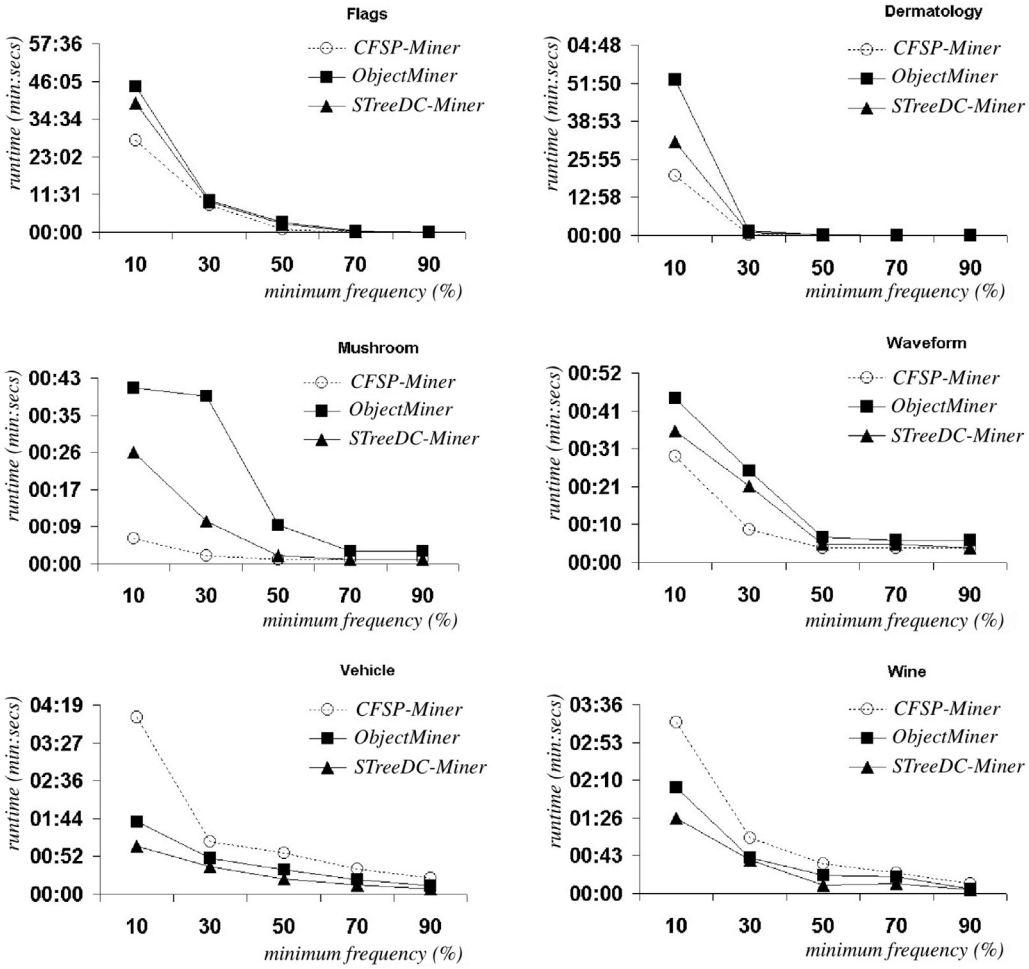

Fig. 4. Runtime of *CFSP-Miner, ObjectMiner* and *STreeDC-Miner* algorithms for the datasets of Table 1.

mind the complexity analysis of the algorithms. Remember that, for each similar closed pattern candidate, its *IFCL* must also be computed by *CFSP-Miner*, while *STreeDC-Miner* and *ObjectMiner* do not include this step for each similar pattern candidate. Then when the amount closed frequent similar patterns is almost equal to the amount of frequent similar patterns (*Vehicle* and *Wine* datasets), *STreeDC-Miner* and *ObjectMiner* outperform the runtime of *CFSP-Miner*. When the percent of frequent similar patterns that are closed decreases, on the other hand, the amount of similar closed pattern candidates verifications and the amount of the *IFCL* computed also decrease. Then the runtime of *CFSP-Miner* decreases compared to the runtimes of *STreeDC-Miner* and *ObjectMiner*. In this experiment *CFSP-Miner* outperforms the runtimes of *STreeDC-Miner* and *ObjectMiner* when less than 80% percent of frequent similar patterns are closed (*Flags, Dermatology, Mushroom*, and *Waveform* datasets).

It is important to highlight that the main characteristic of *CFSP-Miner* is its ability to find the "closed" similar patterns, yielding a reduction in the number of frequent similar patterns without information loss. *CFSP-Miner* can also outperform the runtimes of *ObjectMiner* and *STreeDC-Miner*, depending on the percent of frequent similar patterns that are closed.

To analyze in more detail the behavior of *CFSP-Miner*, different datasets[2] with a defined percent of frequent similar patterns that are closed were automatically generated.

First, three natural numbers X (for the amount of objects), Y (for the amount of features), and Z (for the cardinality of the feature domains) are fixed. Then, $\mathcal{D}$ is built such that $\|O\| = X$, $\|A\| = Y$ and $\mathcal{P}(O, A) = V_{random}\ \forall O \in \mathcal{O}$ *and* $\forall A \in \mathcal{A}$, where V_{random} is a value for the random feature V with uniform distribution over $\{1, 2, \ldots, Z\}$. $O(X)A(Y)D(Z)$ denotes the set of all datasets $\mathcal{D}$ defined by X, Y and Z.

For the first experiment, designed to proof that *CFSP-Miner* has a poor performance when *Percent* is close to 100 %, let $\mathcal{D}_0 \in O(10000)A(10)D(100)$ be a dataset, such that $Percent(M) = 100\%$ for the minimum frequency threshold values 10%, 30%, 50%, 70%, and 90%.

To obtain $\mathcal{D}_0$, it was only necessary to generate less than 10 times the random dataset. This can be claimed because the characteristics of a discrete random variable with uniform distribution over $\{1, 2, \ldots, 100\}$ guarantee a very low probability of $\exists T \in \mathcal{T} \mid FCL(T) \neq T$ and a high probability of $Percent(M) = 100\%$. Proving this is not relevant, but it is sufficient to say that $\mathcal{D}_0$ was obtained with those characteristics.

Fig. 6a) shows that *CFSP-Miner* has the worst performance from the tested algorithms for the dataset D_0, which is the expected behavior when *Percent* is near to 100%. However, as it is shown below, the performance of *CFSP-Miner* improves when *Percent* decreases.

To prove that, a continuous random variable with normal distribution $N(Z/2, 1/Z)$ is used to generate values for a feature. After that, A^* denotes such feature and $O(X)A(Y)D(Z)^*$ denotes the set of all datasets $\mathcal{D}$ that can be generated in this way.

[2] The similarity function used for these datasets was $F^{0.2}_{prod}$.

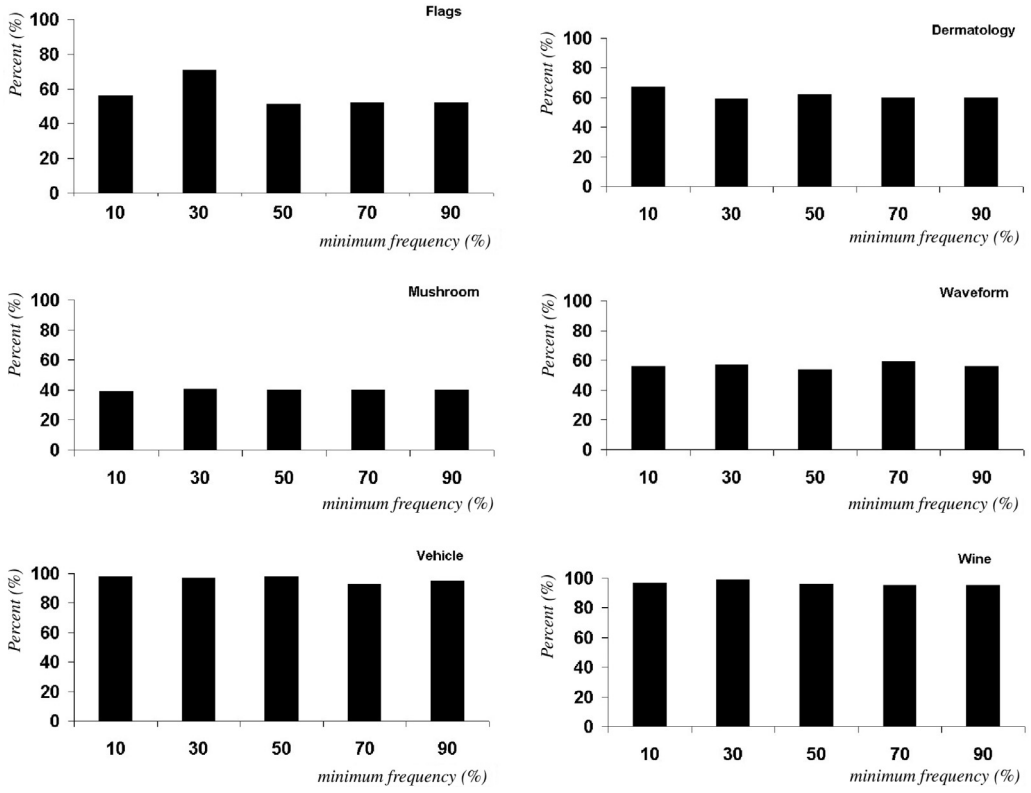

Fig. 5. *Percent* of closed frequent similar patterns respect to frequent similar patterns.

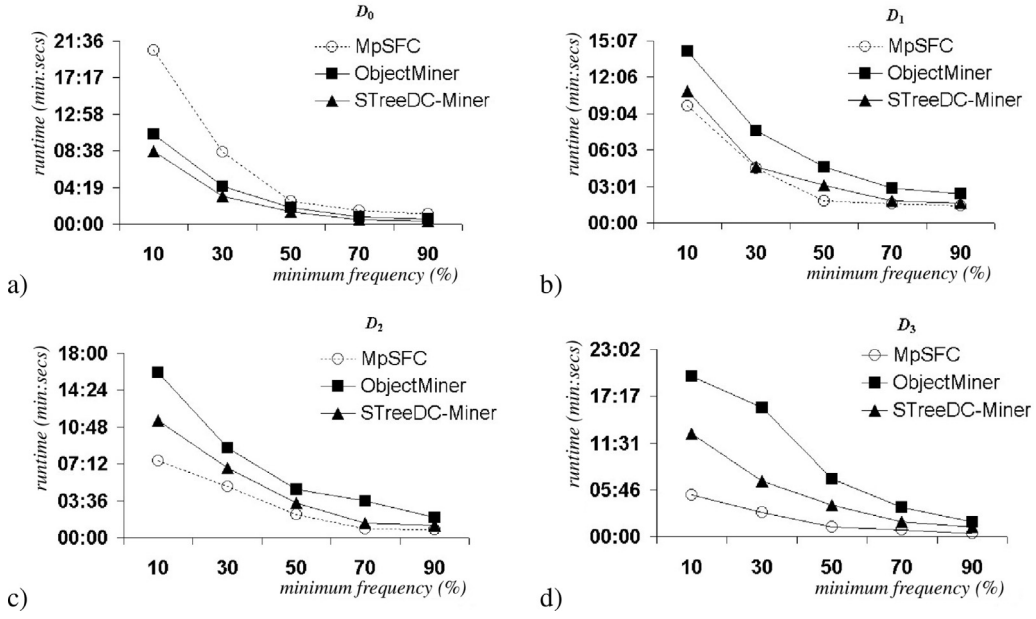

Fig. 6. Runtime of the algorithms *CFSP-Miner, ObjectMiner* and *STreeDC-Miner* for datasets: a) D_0 with $Percent(M) = 100$, b) D_1 with $Percent(M) < 75$, c) D_2 with $Percent(M) < 50$ and d) D_3 datasets with $Percent(M) < 25$.

For the next experiment, we create $\mathcal{D}_1 \in O(10000)A(10)D(100)^*$ with $Percent(M) < 75\%$, $\mathcal{D}_2 \in O(10000)A(10)D(100)^{**}$ with $Percent(M) < 50\%$ and $\mathcal{D}_3 \in O(10000)A(10)D(100)^{***}$ with $Percent(M) < 25\%$, for the minimum frequency threshold {10%, 30%, 50%, 70%, 90%}.

To obtain $\mathcal{D}_1$, it was only necessary to generate less than 20 times the random dataset. This fact is justified because the probability distribution from which the Z values were generated, guarantees that the probability of $FCL(T) \neq T$ is high and *Percent* has a high probability of decrease. Like the previous case, a proof of this is not relevant and it is sufficient to say that $\mathcal{D}_1$ was obtained with those characteristics.

$\mathcal{D}_2 \in O(10000)A(10)D(100)^{**}$ with $Percent(M) < 50\%$ and $\mathcal{D}_3 \in O(10000)A(10)D(100)^{***}$ with $Percent(M) < 25\%$ were obtained analogously.

Note that the runtime of *CFSP-Miner* in Fig. 6 is the best for datasets with percent less than 80 (i.e., D_1, D_2 and D_3) of frequent similar patterns that are closed. This confirms the result of the

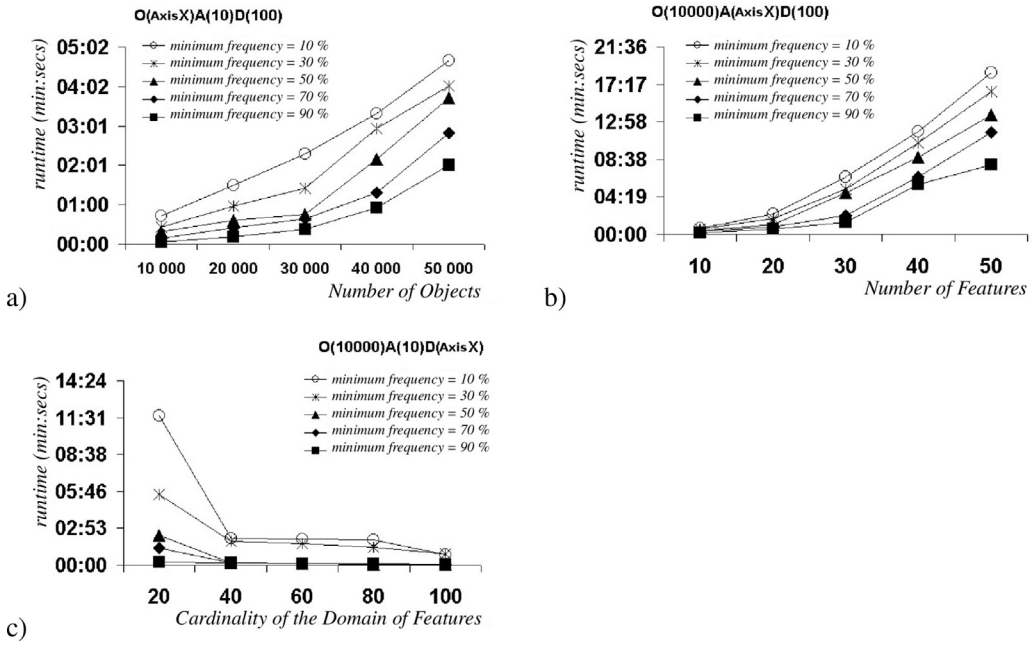

Fig. 7. Scalability of *CFSP-Miner* algorithm. a) w.r.t objects. b) w.r.t features. c) w.r.t domains.

previous experiment: *CFSP-Miner* outperforms the runtimes of *STreeDC-Miner* and *ObjectMiner* when less than 80% percent of frequent similar patterns are closed.

For all minimum frequency thresholds, *CFSP-Miner* also improves its runtime performance gradually from D_0 to D_3 (i.e., *Percent* decreases).

6.2. *Scalability of* CFSP-Miner *algorithm*

To show the scalability of *CFSP-Miner*, several experiments were conducted varying the dimensions of the datasets. Random data sets with $Percent(M) = 100\%$ were automatically generated. These datasets were selected because it represents the worst case for *CFSP-Miner* algorithm, as was shown in Section 6.1.

Fig. 7 shows the scalability of *CFSP-Miner* algorithm with respect to a) objects, b) features, and c) domains.

It can be seen in Fig. 7 a) that the runtime increases when the number of objects increases, as it was expected. The worst runtime is still in an acceptable range compared to results in the previous section.

Something similar happens in Fig. 7 b) when the number of features is increased. Note that increasing the number of features produces an exponential increase of the search space of closed frequent similar patterns, which slows runtime. Despite this fact, *CFSP-Miner* still achieves acceptable runtimes compared to results in the previous section. This demonstrates good scalability for the range of number of objects and features shown in this experiment.

Fig. 7c) shows that runtime decreases when the size of the domain increases. This increase in size of the domain causes a decrease in the probability of having repeated values due to using a uniform distribution. The number of frequent patterns decreases and reducing the execution time.

In Summary, we can see from the Fig. 7 that the number of features is the parameter that most negatively impacts the scalability of our algorithm. This is in concordance with the complexity analysis presented in section 5.1.

7. Conclusions

In this paper we proposed the concept of *closed frequent similar pattern mining* for discovering a reduced set of frequent similar patterns without information loss. We also proposed a novel *closed frequent similar pattern mining* algorithm, named *CFSP-Miner*, that uses boolean monotonic similarity functions to find all the closed frequent similar patterns.

The results show that the proposed *CFSP-Miner* algorithm is more efficient than other frequent similar pattern mining algorithms from the literature, except in the case when the number of frequent similar patterns and the number of closed frequent similar patterns are almost equal. Another strength of *CFSP-Miner* is its ability to find the "closed" similar patterns without information loss. In most of the analysed data sets, *CFSP-Miner* reduced amount of discovered frequent similar patterns by approximately 50%. *CFSP-Miner* can also be scaled for use in high-dimensional data sets because the experimental results have shown that increases in the number of objects, the number of features, or the size of each feature domain maintains the runtime in an acceptable range.

For future work, we visualize improving the efficiency of *CFSP-Miner*, exploring the ideas proposed in the most recently works to improve the efficiency of traditional closed frequent itemset mining algorithms and studying the feasibility of extending these results to closed frequent similar pattern mining. Extending closed frequent similar patterns mining and association rules mining for non-boolean similarity functions and non-monotonic similarity functions is another interesting future work.

Acknowledgments

Thanks to Douglas David Crockett for his editing help as a native English speaker.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record: 22* (pp. 207–216). ACM.

Agrawal, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, vldb: 1215* (pp. 487–499).

Alatas, B., Akin, E., & Karci, A. (2008). Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing, 8*(1), 646–656.

Baeza-Yates, R., et al. (1999). *Modern information retrieval*: 463. ACM press New York.

Beil, F., Ester, M., & Xu, X. (2002). Frequent term-based text clustering. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 436–442). ACM.

Burdick, D., Calimlim, M., & Gehrke, J. (2001). Mafia: A maximal frequent itemset algorithm for transactional databases. In *Data engineering, 2001. proceedings. 17th international conference on* (pp. 443–452). IEEE.

Chiu, C.-Y., Yeh, C.-T., & Lee, Y.-J. (2013). Frequent pattern based user behavior anomaly detection for cloud system. In *2013 conference on technologies and applications of artificial intelligence* (pp. 61–66). IEEE.

Danger, R., Ruiz-Shulcloper, J., & Llavori, R. B. (2004). Objectminer: A new approach for mining complex objects.. In *Iceis (2)* (pp. 42–47). Citeseer.

Fan, Y., Ye, Y., & Chen, L. (2016). Malicious sequential pattern mining for automatic malware detection. *Expert Systems with Applications, 52*, 16–25.

Gómez-Herrera, J., et al. (1994). Prognostic of gas-oil deposits in the cuban ophiological association. *Applying Mathematical Modeling, Geofisica Internacional, 33*(3), 447467.

Han, H. L. J., & Shao, D. X. Z. (2006). Mining frequent patterns from very high dimensional data: A top-down row enumeration approach. In *Proceedings of the sixth siam international conference on data mining: 124* (p. 282). SIAM.

Hashem, T., Karim, M. R., Samiullah, M., & Ahmed, C. F. (2017). An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure. *Expert Systems with Applications, 67*, 252–271. doi:10.1016/j.eswa.2016.09.023.

Hernández-León, R., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., & Hernández-Palancar, J. (2012). Classification based on specific rules and inexact coverage. *Expert Systems with Applications, 39*(12), 11203–11211.

Hu, T., Sung, S. Y., Xiong, H., & Fu, Q. (2008). Discovery of maximum length frequent itemsets. *Information Sciences, 178*(1), 69–87.

Kalpana, B., & Nadarajan, R. (2008). Incorporating heuristics for efficient search space pruning in frequent itemset mining strategies. *Current science, 94*(1), 97–101.

Le, T., & Vo, B. (2015). An n-list-based algorithm for mining frequent closed patterns. *Expert Systems with Applications, 42*(19), 6648–6657. doi:10.1016/j.eswa.2015.04.048.

Lee, A. J., Wang, C.-S., Weng, W.-Y., Chen, Y.-A., & Wu, H.-W. (2008). An efficient algorithm for mining closed inter-transaction itemsets. *Data & Knowledge Engineering, 66*(1), 68–91.

Li, J., Fu, A. W.-c., & Fahey, P. (2009). Efficient discovery of risk patterns in medical data. *Artificial intelligence in medicine, 45*(1), 77–89.

Li, J., et al. (2005). Mining risk patterns in medical data. In *Proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining* (pp. 770–775). ACM.

Liu, H., et al. (2009). Top-down mining of frequent closed patterns from very high dimensional data. *Information Sciences, 179*(7), 899–924.

Lopez, F. J., Blanco, A., Garcia, F., Cano, C., & Marin, A. (2008). Fuzzy association rules for biological data analysis: a case study on yeast. *BMC bioinformatics, 9*(1), 1.

Moonesinghe, H., Fodeh, S., & Tan, P.-N. (2006). Frequent closed itemset mining using prefix graphs with an efficient flow-based pruning strategy. In *Sixth international conference on data mining (icdm'06)* (pp. 426–435). IEEE.

Nahar, J., Imam, T., Tickle, K. S., & Chen, Y.-P. P. (2013). Association rule mining to detect factors which contribute to heart disease in males and females. *Expert Systems with Applications, 40*(4), 1086–1093.

Nair, B., & Tripathy, A. K. (2011). Accelerating closed frequent itemset mining by elimination of null transactions. *Journal of Emerging Trends in Computing and Information Sciences, 2*(7), 317–324.

Nezhad, J. T., & Sadreddini, M. (2007). Ptclose: A novel algorithm for generation of closed frequent itemsets from dense and sparse datasets. In *Proceedings of the world congress on engineering: 1*.

Nguyen, L. T., & Nguyen, N. T. (2015). An improved algorithm for mining class association rules using the difference of obidsets. *Expert Systems with Applications, 42*(9), 4361–4369.

Ortiz-Posadas, M. R., Vega-Alvarado, L., & Toni, B. (2009). A mathematical function to evaluate surgical complexity of cleft lip and palate. *Computer methods and programs in biomedicine, 94*(3), 232–238.

Pan, F., Tung, A. K., Cong, G., & Xu, X. (2004). Cobbler: combining column and row enumeration for closed pattern discovery. In *Scientific and statistical database management, 2004. proceedings. 16th international conference on* (pp. 21–30). IEEE.

Pei, J., et al. (2000). Closet: An efficient algorithm for mining frequent closed itemsets.. In *Acm sigmod workshop on research issues in data mining and knowledge discovery: 4* (pp. 21–30).

Prabha, S., Shanmugapriya, S., & Duraiswamy, K. (2013). A survey on closed frequent pattern mining. *International Journal of Computer Applications, 63*(14).

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2008). Mining frequent similar patterns on mixed data. In *Iberoamerican congress on pattern recognition* (pp. 136–144). Springer.

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2011). Rp-miner: A relaxed prune algorithm for frequent similar pattern mining. *Knowledge and information systems, 27*(3), 451–471.

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2013). Mining frequent patterns and association rules using similarities. *Expert Systems with Applications, 40*(17), 6823–6836.

Ruiz-Shulcloper, J., & Fuentes-Rodríguez, A. (1981). A cybernetic model to analyze juvenile delinquency. *Revista Ciencias Matemáticas, 2*(1), 123–153.

Uno, T., Asai, T., Uchida, Y., & Arimura, H. (2003). Lcm: An efficient algorithm for enumerating frequent closed item sets.. *Fimi*: 90. Citeseer.

Vo, B., Hong, T.-P., & Le, B. (2012). Dbv-miner: A dynamic bit-vector approach for fast mining frequent closed itemsets. *Expert Systems with Applications, 39*(8), 7196–7206. doi:10.1016/j.eswa.2012.01.062.

Weisstein, E. W. (2002). *CRC concise encyclopedia of mathematics*. CRC press.

Wen, J., Zhong, M., & Wang, Z. (2015). Activity recognition with weighted frequent patterns mining in smart environments. *Expert Systems with Applications, 42*(17), 6423–6432.

Zaki, M. J., & Hsiao, C.-J. (2002). Charm: An efficient algorithm for closed itemset mining.. In *Sdm: 2* (pp. 457–473). SIAM.