Closed Itemset Mining: A Graph Theory Perspective

Fatima Zohra Lebbah[1,2], Moussa Larbani[3,4], Abdellatif Rahmoun[5]
[1]Higher School of Electrical and Energetic Engineering of Oran (ESG2E), Vicinal Road N9,Oran, 31000, Algeria
[2]Laboratory of Research in Computer Science-SBA, Computational Intelligence and Soft Computing Team (CISCO), Higher School of Computer Science, BP 73, Bureau de poste EL WIAM, Sidi-Belabbes, 22016, Algeria
[3]School of Mathematics and Statistics, Carleton University, Ontario, 1125 Colonel By Drive, Ottawa, K1S 5B6, Canada
[4]National Higher School of Statistics and Applied Economics (ENSSEA), Pôle Universitaire de Koléa, Tipaza, Algiers, 42003,Algeria
[5]Laboratory of Research in Computer Science-SBA, Computational Intelligence and Soft Computing Team (CISCO), Higher School of Computer Science, BP 73, Bureau de poste EL WIAM, Sidi-Belabbes, 22016, Algeria
E-mail: fz_lebbah@yahoo.fr, fz.lebbah@esgee-oran.dz, larbani61@hotmail.com, a.rahmoun@esi-sba.dz

Keywords: dataset, closed frequent itemset, labeled graph, maximal clique

Received: November 28, 2023

Data mining is a field that focuses on extracting and analyzing usable data from large databases. This paper specifically concentrates on the problem of finding closed frequent itemsets, which is extensively studied in the field. Previous techniques based on graph theory have used tree structures and recursive algorithms, which have limitations. In this paper, we propose a scalable modeling approach that represents a transaction dataset using an undirected and labeled graph. The labels on the edges are computed and assigned in a clever manner. We also introduce a polynomial and exact algorithm based on the clique notion in graph theory to compute all the closed frequent itemsets. Our initial testing results demonstrate the efficiency of our algorithm in terms of CPU-time and memory usage compared to recent methods in the literature. Additionally, our graph model can be easily extended when the dataset is updated. We utilize linear structures with boolean values to implement our graph and employ an exact algorithm with polynomial complexity. These aspects of our approach provide strong foundations for investigating more challenging issues related to the problem at hand.

Povzetek: Prispevek obravnava skalabilno modeliranje podatkovnih množic z uporabo neusmerjenih, označenih grafov za rudarjenje zaprtih pogostih množic. Predlagan je nov algoritem s polinomsko kompleksnostjo, temelječ na teoriji klik.

1 Introduction

Nowadays, Data Mining is the science that allows us to work with a dataset in line with our perspectives [18, 16, 17, 19]. It involves the process of finding patterns and knowledge from a large amount of data. However, achieving the desired and specific information poses a significant challenge for many researchers. This has led to the development of various competing models and techniques, such as in [27, 3, 2, 4, 7].

Itemset Mining is an appealing field within the realm of Data Mining, primarily because of the need to manage large-scale data. As the size of data increases, traditional methods and techniques used to address Itemsets' problems tend to become slower and less efficient. Researchers are currently focused on discovering and proposing new techniques and algorithms to model and solve problems related to Itemset Mining, including computing frequent, closed, and maximal itemsets.

The process of extracting frequent sets of items bought by customers, known as finding frequent itemsets in a transaction database [23, 15, 3], has been addressed by various techniques. Some of these techniques include Constraints Programming [32, 6, 5, 25, 8], as well as Graph Theory [24, 28, 29, 30, 26].

Currently, graph theory-based approaches have been proposed to find frequent itemsets. However, in this article, we present a new approach to address the problem of Closed Frequent Itemsets CFIs. We show that a transaction database can be effectively modeled using the clique's concept through an undirected and labeled graph.

In this article, we introduce a clique-based algorithm for finding CFIs, which we refer to as CfCi. This approach uses linear structures and a polynomial algorithm. Specifically, we show how our graph model enables the step-by-step computation of all the CFIs.

The article is structured as follows:

Section 2 introduces the basic concepts of the CFIs mining problem, along with a review of related works. Section 3 presents graph theory notions used to transform a transaction database into an undirected and labeled graph. Section 4 describes the equivalency between a (Closed) Frequent Itemsets and a (Maximal) clique and how to deduce a CFI from a maximal clique. Section 5 depicts our CfCi algo-

rithm and the called functions. The complexity analysis and an executed example are also given in this section. In Section 6, we compare our algorithm CfCi to the latest existing techniques in the literature to highlight its contribution. This section also includes the reporting of experimental results on various datasets and their analysis. In Section 7, we discuss, describe, analyze, and interpret our findings. Finally, Section 8 concludes the paper.

2 Problem definition and related works

In this section, we begin by introducing the problem using an example of a market basket. Then, we provide basic concepts and definitions.

For instance, let's consider the market baskets illustrated in Figure 1, where we analyze five shopping baskets. The aim is to study customer shopping habits and identify associations and correlations between different items. In this paper, we focus on computing itemsets that are frequently purchased together by customers. Specifically, we address the problem of $CFIs$.

Figure 1: Market baskets analysis

Table 1 represents the market baskets as transactions and items. Each row corresponds to a customer transaction, and each column represents an item. The assigned letters m, b, e, j, and s represent specific items: milk, bread, egg, jam, and sugar, respectively.

According to the table, the itemset $\{m, j\}$ has a frequency of three, showing that three customers purchase the both articles milk and jam. Similarly, the itemset $\{b, e\}$ has a frequency of two, showing that two customers purchase the both articles bread and egg.

Table 1: The itemset database, as shown on the left side of Figure 1, can be represented using binary matrix notation, as shown on the right side of the figure.

$\mathcal{T}$	$Itemset$
t_1	$\{m, b, e, j\}$
t_2	$\{m, s, e, j\}$
t_3	$\{b, e\}$
t_4	$\{m, j\}$

	m	s	b	e	j
t_1	1	0	1	1	1
t_2	1	1	0	1	1
t_3	0	0	1	1	0
t_4	1	0	0	0	1

Therefore, an itemset database problem is defined by a set of items $\mathcal{I} = \{m, s, b, e, j\}$ and a set of transactions $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$. A boolean matrix $\mathcal{D}(n \times m)$, which is called the transaction database, represents the relationship between these items through the transactions.

Therefore, an itemset database problem is defined by a set of items $\mathcal{I} = \{m, s, b, e, j\}$ and a set of transactions $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$. The relationship between these items through the transactions is represented by a boolean matrix $\mathcal{D}(n \times m)$, which is called the transaction database (see Definition 1). Here, $n = |\mathcal{I}|$ the number of items in $\mathcal{I}$ and $m = |\mathcal{T}|$ represents the number of transactions in $\mathcal{T}$.

Definition 1 (transaction database)**.** *Let $\mathcal{T} = \{t_1, t_2, \cdots t_n\}$ be the set of transactions, and $\mathcal{I} = \{i_1, i_2, \cdots i_m\}$ the set of items. A transaction database is defined through the boolean database matrix $\mathcal{D}(n \times m)$:*

$$\mathcal{D}_{j,k} = \begin{cases} 1 & \textit{if the item } i_j \textit{ occurs in the transaction } t_k \\ 0 & else \end{cases} \quad (1)$$

where:

$n = |\mathcal{T}|$ *is the number of transactions,*

$m = |\mathcal{I}|$ *is the number of items.*

Based on Figure 1 and Table 1, the analyzed shopping results in the following transaction database $\mathcal{D}(4 \times 5)$ as shown in Matrix 2.

$$\mathcal{D} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

2.1 Frequent itemset mining

Before delving into the topic of Closed Frequent Itemset, it is important to introduce the concept of support, as defined in Definition 2.

Definition 2 (support measure)**.** *Let $\mathcal{D}$ be a transaction database, $\mathcal{I}$ be the set of items, and $\mathcal{T}$ be the set of transactions. The support measure of an itemset $I \subseteq \mathcal{I}$ is denoted as $Supp(I)$ and defined as:*

$$Supp(I) = \sum_{i \in I} (\prod_{t \in \mathcal{T}} \mathcal{D}_{ti}) \quad (3)$$

$Supp(I)$ is the number of transactions containing all the items belonging to I.

Definition 3 (frequent itemset mining)**.** *Let $\mathcal{D}$ be a transaction database and θ be the minimum support threshold, where $\theta > 0$. An itemset I is considered frequent if its support, denoted as $Supp(I)$, is greater than or equal to θ ($\theta > 0$). Conversely, an itemset I is considered infrequent if its support is less than θ ($Supp(I) < \theta$).*

For example, let's consider the transaction database presented in Matrix 2. The support values for $I_1 = \{i_2\}$, $I_2 = \{i_3, i_4\}$ and $I_3 = \{i_1, i_5\}$ are $S1$, 2 and 3, respectively.

Property 1 (downward closure)**.** *Every subset of a frequent itemset is also frequent. In other words,*

$$I \text{ is frequent } \Rightarrow \forall I_i \subset I : I_i \text{ is frequent} \quad (4)$$

or equivalently:

$$I \text{ is infrequent } \Rightarrow \forall I_i \supset I : I_i \text{ is infrequent} \quad (5)$$

Property 1 [2], also known as the *Apriori Principle*, is related to Property 3 and Property 2 [1, 21], which introduce the characteristics of monotonicity and anti-monotonicity of supports and the inclusion of frequent itemsets, respectively.

Property 2 (anti-Monotony)**.** *[20] Let there be two itemsets I_1 and I_2 such that $I_1 \subset I_2$. It follows that $Supp(I_1) \geq Supp(I_2)$.*

$$\forall I_i, I_j \in \mathcal{I}/I_i \subseteq I_j : Supp(I_i) \geq Supp(I_j) \quad (6)$$

Property 3 (monotonicity)**.** *Let I_1 and I_2 be two itemsets such that $I_1 \subseteq I_2$. If $Supp(I_2) \geq \theta$ then $Supp(I_1) \geq \theta$.*

$$\forall I_i, I_j \in \mathcal{I}/I_i \subseteq I_j : Supp(I_j) \geq \theta \Rightarrow Supp(I_i) \geq \theta \quad (7)$$

In other words, the relation *Frequent* (as stated in Property 2) indicates that the subsets of a frequent itemset are also frequent. Additionally, the support measure exhibits monotonicity, as stated in Property 3.

2.2 Closed itemset mining

A closed itemset, denoted as I_c, is a frequent itemset that has a support, denoted as $Supp(I_c)$, equal to or greater than a fixed minimum support threshold, denoted as θ. All the items in I_c must be present in the optimal set of transactions, denoted as T_{I_c}, which has a size equal to $Supp(I_c)$ (as defined in Definition 4).

Definition 4 (closed itemset)**.** *Let I_c be a frequent itemset and $Supp(I_c)$ be its support. The itemset I_c is closed if-if it is not included in a larger frequent itemset I_i that has the same support, $Supp(I_c) = Supp(I_i)$, and appears in all the transactions that include I_c, as well as additional transactions.*

$$I_c \text{ is closed } \Leftrightarrow \nexists I_i \supset I_c : I_i \text{ is frequent} \wedge Supp(I_c) = Supp(I_i) \wedge T_{I_c} \subset T_{I_i}. \quad (8)$$

where, T_{I_c} and T_{I_i} are the sets of the transactions which contain the elements of I_c and I_i, respectively.

Note that our aim is to identify all the closed itemsets. Figure 2 depicts the first part of the Hasse diagram for the database represented in Matrix 2. An edge is drawn from the itemset I_1 to the itemset I_2 if and only if $I_1 \subset I_2$ and $|I_2| = |I_1| + 1$. In this figure, the blue ellipses represent the frequent itemsets, while the green ellipses represent the closed itemsets. The support of each closed itemset is mentioned on the left side of the corresponding ellipse.

For instance, as shown in Figure 2, if we set the minimum support threshold (θ) to 2, the support of $\{e\}$ is 3, while the support of its superset $\{b, e\}$ is 2. Similarly, the support of $\{m, j\}$ is 3. This shows the anti-monotony property (as stated in Property 2), where e is a subset of $\{e\} \subset \{b, e\}$, and their supports are 3 and 2, respectively.

2.3 Related works

So far, several algorithms have been developed for mining Closed Frequent Itemsets, including AprioriTID Close [20], LCM [22], CHARM [9], FP-Close[12], FCILINK [31], NAFCP [10], EMFCI[14] NECLAT [13] and GrAFCI+ [11]. However, most of these methods utilize tree structures and recursive algorithms.

One primary challenge in the field of data mining is to efficiently find all the closed itemsets while optimizing CPU time and memory usage.

To address this challenge, we propose a graph model based on a linear structure that minimizes memory usage. Additionally, we introduce the exact algorithm CfCi to find all the closed frequent itemsets.

To emphasize the contribution of our proposed approach in this work, we will compare the results of the CfCi algorithm with the latest algorithms, namely NAFCP, NECLAT and GrAFCI+. This comparison will be presented in Section 6. We consider not only the novelty of the chosen algorithm but also the availability of corresponding applications and the detailed aspect of the results.

Table 2 presents the key characteristics of the algorithms that are considered in the experiments section.

Table 2: Characteristics of the tested algorithms and their used structures

	Used Structure	Used Algorithms	Proposed Approach
CfCi	linear structure	iterative	exact
NAFCP	tree structure	recursive	approximate
NECLAT	tree structure	recursive	approximate
GrAFCI+	tree structure	recursive	approximate

3 Itemset mining vs undirected graph

Firstly, we propose an undirected and labeled graph to represent a dataset. This model is based on the idea that connecting items that appear in the same transaction forms a clique, which is a sub-graph where every pair of distinct vertices are adjacent. Therefore, each transaction is represented by a clique. However, since a bond may be shared by multiple cliques, we introduce a labeling approach that ensures a bond shared by different cliques (as defined in Definition 9) is accurately represented in the graph model.

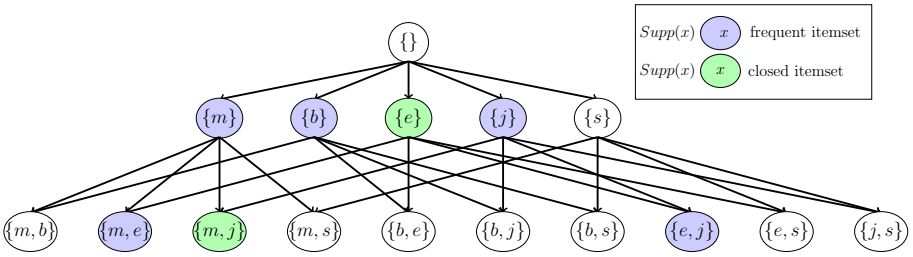


Figure 2: The search space of frequent itemset mining for the database of Table 1 and $\theta = 2$

In our graph model, we assign a binary value, known as a sub-label, to each transaction. Specifically, as defined in Definition 5, the bond (j,k) that represents the occurrence of items i_j and i_k in the same transaction t_p is characterized by the sub-label $\mathcal{L}^p(j,k) = 10^{n-p-1}$, where n is the number of transactions.

Definition 5 (sub-label)**.** *Consider a database $\mathcal{D}(n \times m)$ and a transaction $t_p \in \mathcal{T}$. The occurrence of items i_j and i_k in the same transaction t_p, represented by $\mathcal{D}[p,j] = 1$ and $\mathcal{D}[p,k] = 1$, is modeled by the bond (j,k) labeled with the binary value $\mathcal{L}^p(j,k) = 10^{n-p-1}$. This sub-label is a part of the overall label $\mathcal{L}^p(j,k)$ that represents the relationship between the items i_j and i_k.*

Each time the pair of items i_j and i_k appears in a transaction, a new sub-label is added to the label of the bond (i,j). In other words, the label of the bond (i_j, i_k), denoted by $\mathcal{L}(i_j, i_k)$, is the sum of all the sub-labels associated with that bond, as expressed by Equation 9 in Definition 6.

Definition 6 (bond label)**.** *Let $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ be the graph model of a transactions database. The label of the bond $(i,j) \in U$ is given by:*

$$\mathcal{L}(i,j) = \sum_{t_p \in \mathcal{T}} \mathcal{L}^p(i,j) \tag{9}$$

In Definition 7, we introduce the notion of the label size denoted by $HW(\mathcal{L}(i,j))$, which is the number of the included ones or the included sub-labels in $\mathcal{L}(i,j)$.

Definition 7 (label size)**.** *Let $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ be the graph model of a transaction database. The size of the label $\mathcal{L}(i,j)$, denoted as $HW(\mathcal{L}(i,j))$, represents the Hamming Weight of$\mathcal{L}(i,j)$, which is the number of included sub-labels.*

Definition 8 introduces the concept that the items appearing in the transaction t are well represented by a clique. To differentiate one transaction or clique from another, we assign the corresponding sub-label $\mathscr{L}^p(i,j)$ to each bond (i,j).

Definition 8 (transaction VS clique)**.** *Let $\mathcal{D}(n \times m)$ be a transaction database defined on the set $\mathcal{T}$ of transactions and the set $\mathcal{I}$ of items. Suppose that the transaction $t_p \in \mathcal{T}$ includes all the items of the itemset $I \subset \mathcal{I}$. The transaction t_p is represented by the clique C_p, where:*

$$\mathcal{C}_p = \left\{ i_j, i_k \in I_i, \mathcal{L}^p(i_j, i_k) = 10^{(n-p-1)} \right\} \tag{10}$$

In Definition 9, we depict how we model each transaction and its included items by a labeled clique in the corresponding undirected and labeled graph, called dataset graph.

Definition 9 (dataset VS Labeled graph)**.** *Let $\mathcal{D}(n \times m)$ be a transaction database, where $n = |\mathcal{T}|$ represents the number of transactions and $m = |\mathcal{I}|$ represents the number of items. We associate with $\mathcal{D}$ the undirected and labeled graph $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ where:*

X: set of vertices that represents the set of items, with

$$|X| = |\mathcal{I}| = m$$

U: set of bonds which are defined as follows:

$$\begin{cases} (i_j, i_k) \in U_{\mathcal{D}} & \text{if } \exists t_p \in \mathcal{T} : \mathcal{D}[p,j] = 1 \\ & \wedge \mathcal{D}[p,k] = 1 \\ (i_j, i_k) \notin U & else \end{cases} \tag{11}$$

and

$$|U| = m + \sum_{i=1..n} \sum_{j=1..m} \mathcal{D}[i,j] - n$$

$\mathcal{L}_{ij}$: binary value assigned to the bond $(i,j) \in U$, which is defined as follows:

$$\mathcal{L}_{i_j, i_k} = \sum_{p=1..n} \mathcal{D}[p,j] * \mathcal{D}[p,k] * 10^{n-p-1} \tag{12}$$

For instance, let's consider the itemset mining problem [6] described below:

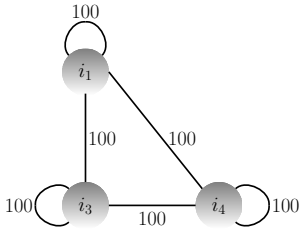
	i_1	i_2	i_3	i_4
t_1	1	0	1	1
t_2	1	1	0	1
t_3	0	0	1	1

In Figures 3a, 3b, and 3c, we depict the cliques generated from the transactions t_1, t_2, and t_3, respectively.

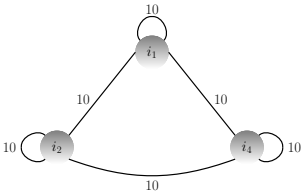
Therefore, our graph model $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ is formed by combining all the cliques extracted from D, and the corresponding adjacency matrix $\mathcal{A}$ will be a binary symmetric matrix, as defined in Definition 10.

Definition 10 (items adjacency matrix)**.** *Let $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ be the graph modeling the database $\mathcal{D}(n \times m)$. The corresponding adjacency matrix $\mathcal{A}$ to $\mathcal{G}$ is defined as follows:*

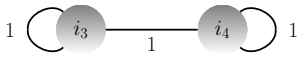
$$a_{jk} = \begin{cases} \mathscr{L}(i_j, i_k) \text{ if } (i_j, i_k) \in U \\ 0 \text{ else} \end{cases} \tag{13}$$



(a) t_1 VS C_1



(b) t_1 VS C_1



(c) t_2 VS C_2

Figure 3: Transactions VS Cliques

where:
$\mathscr{L}(i_j, i_k) = \sum_{p=1}^{n} \mathscr{L}^p(i_j, i_k)$

We illustrate through Figures 4, 5 and 6, the sub-graphs $\mathcal{G}^1 = \langle X, U, \mathcal{L}^1 \rangle$, $\mathcal{G}^2 = \langle X, U, \mathcal{L}^2 \rangle$ and $\mathcal{G}^3 = \langle X, U, \mathcal{L}^3 \rangle$ which model the transactions t_1, t_2 and t_3, respectively. On the right of each sub-graph $\mathcal{G}^p$, we introduce the corresponding adjacency matrix $\mathcal{L}^p$ containing the bonds sub-labels.

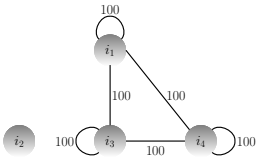


Figure 4: $\mathcal{G}^1 = \langle X, U, \mathcal{L}^1 \rangle$ models the transaction t_1

$$\mathcal{L}^1 = \begin{bmatrix} 100 & 000 & 100 & 100 \\ 000 & 000 & 000 & 000 \\ 100 & 000 & 100 & 100 \\ 100 & 000 & 100 & 100 \end{bmatrix}$$

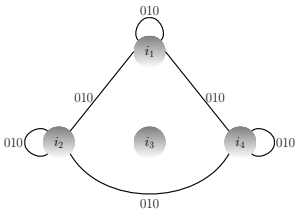


Figure 5: $\mathcal{G}^2 = \langle X, U, \mathcal{L}^2 \rangle$ models the transaction t_2

$$\mathcal{L}^2 = \begin{bmatrix} 010 & 010 & 000 & 010 \\ 010 & 010 & 000 & 010 \\ 000 & 000 & 000 & 000 \\ 010 & 010 & 000 & 010 \end{bmatrix}$$

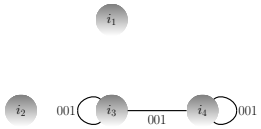


Figure 6: $\mathcal{G}^3 = \langle X, U, \mathcal{L}^3 \rangle$ models the transaction t_3

$$\mathcal{L}^3 = \begin{bmatrix} 000 & 000 & 000 & 000 \\ 000 & 000 & 000 & 000 \\ 000 & 000 & 001 & 001 \\ 000 & 000 & 001 & 001 \end{bmatrix}$$

As illustrated in Figure 7, according to Definition 10, the combination of the sub-graphs $\mathcal{G}^1$, $\mathcal{G}^2$ and $\mathcal{G}^3$, provides the labeled and undirected graph $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$. The corresponding adjacency matrix $\mathcal{A} = \mathcal{L}^1 + \mathcal{L}^2 + \mathcal{L}^3$ is given on the right side of the figure.

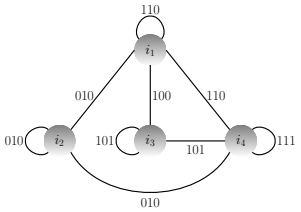


Figure 7: $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ models the database transactions t_1, t_2 and t_3

$$\mathcal{A}_{\mathcal{D}} = \begin{bmatrix} 110 & 010 & 100 & 110 \\ 010 & 010 & 000 & 010 \\ 100 & 000 & 101 & 101 \\ 110 & 010 & 101 & 111 \end{bmatrix}$$

4 Frequent itemset vs clique

In this section, we present the utilization of the maximal clique principle in our approach to find the closed itemsets.

The concept of a frequent itemset is defined in Definition 11, using the concept of a clique in graph theory. In other words, a clique in the dataset corresponding graph $\mathcal{G} = \langle X, Y, \mathcal{L} \rangle$ models a frequent itemset, where the contained vertices correspond to items.

Definition 11 (frequent itemset VS Clique)**.** *Let's consider the adjacency matrix $\mathcal{A}(n \times n)$, the set of items $\mathcal{I}$, the set of transactions $\mathcal{T}$ and the min-support θ. An itemset I is frequent if and only if its items are vertices of a clique $\mathcal{C}$ in $\mathcal{G}$, such that all the labels of the corresponding bonds share at least θ sub-labels.*

$$\forall i, j \in I : | \cap_{\mathcal{SL}(i,j)} | \geqslant \theta \tag{14}$$

where $\mathcal{SL}(i, j)$ is the set of the sub-labels whose sum equals $\mathcal{A}[i, j]$

	i_1	i_2	i_3	i_4	i_5
t_1	1	0	1	1	0
t_2	1	1	0	1	0
t_3	0	0	1	1	0
t_4	0	0	0	0	1

(a)

$$\mathcal{D} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(b)

Figure 8: Example of itemset mining problem including 5 items and 4 transactions

To compute the CFIs, the infrequent items (see Property 1) and empty transactions should be removed. For instance, consider the dataset given in Figure 8.

As illustrated in Figure 9, if the min-support is set to $\theta = 2$, we notice that i_2 and i_5 are infrequent and should be removed. Thus, the dataset $\mathcal{D}$ is replaced by $\mathcal{D}'$ (see sub-Figure 9a), which includes the empty transaction t_4. At this level, t_4 is deleted to move to $\mathcal{D}''$ (see sub-Figure 9b).

	i_1	i_3	i_4
t_1	1	1	1
t_2	1	0	1
t_3	0	1	1
t_4	0	0	0

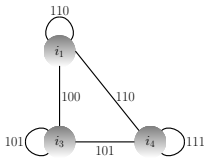
$$\mathcal{D}' = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(a)

	i_1	i_3	i_4
t_1	1	1	1
t_2	1	0	1
t_3	0	1	1

$$\mathcal{D}'' = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

(b)



$\mathcal{G}_{\mathcal{D}''} = \langle X_{\mathcal{D}''}, U_{\mathcal{D}''}, \mathcal{L}_{\mathcal{D}} \rangle$

(c)

Figure 9: Removing infrequent items and empty transactions

The graph corresponding to $\mathcal{D}''$, called $\mathcal{G}'' = \langle X'', U'', \mathcal{L} \rangle$ (see sub-Figure 9c), will be the input for our proposed algorithms and will be noted as $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$.

Every pair $i_j, i_k \in I_c$, where l_c is a closed itemset, is frequent, as defined in Definition 4. Therefore, if we project this concept onto the graph model $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$, the label of each bond $(i, j) \in U$ must contain at least θ ones. Stated differently, a bond in our graph needs to be consistent (Definition 12).

Definition 12 (consistent bond)**.** *Let $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ be the graph model of the dataset $\mathcal{D}''$. Suppose that $\mathcal{L}(i, j)$ is the corresponding label to the bond (i, j), and $HW(\mathcal{L}(i, j))$ is the number of the ones included in $\mathcal{L}(i, j)$. If $HW(\mathcal{L}(i, j)) \geqslant \theta$ then (i, j) is consistent and (i, j) is inconsistent otherwise.*

Consequently, as given in Definition 13, if all the bonds of the graph are consistent,the graph is consistent.

Definition 13 (Consistent data mining graph)**.** *Let $\mathcal{G}_c = \langle X, U_c, \mathcal{L} \rangle$ be the dataset graph. $\mathcal{G}_c$ is consistent if-if all its bonds are consistent:*

$$\mathcal{G}_c = \langle X_c, U_c, \mathcal{L} \rangle \text{ is consistent } \Leftrightarrow \forall (i, j) \in U : (i, j) \text{ is consistent.} \quad (15)$$

Thus, the input graph of our algorithm should be a consistent graph whose inconsistent bonds are omitted. As illustrated in Figure 10, $\mathcal{G}_c = \langle X_c, U_c, \mathcal{L}_c \rangle$ is the consistent version of $\mathcal{G}$ (see Figure 7). The corresponding adjacency matrix $\mathcal{A}(3 \times 3)$ is shown on the right side of the figure.

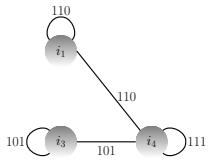


Figure 10: The consistent graph $\mathcal{G}_c = \langle X_c, U_c, \mathcal{L}_c \rangle$ (see sub-Figure 9c)

$$\mathcal{A} = \begin{bmatrix} 110 & 0 & 110 \\ 0 & 101 & 101 \\ 110 & 101 & 111 \end{bmatrix}$$

5 CfCi exact algorithm for complete enumeration

In Definition 15, we introduce the closed itemset notion in terms of the graph theory. A CFI corresponds to a maximal clique (see Definition 14), whose at least θ sub-labels are shared by the bonds.

Definition 14 (maximal clique)**.** *Let $\mathcal{G} = \langle X, U \rangle$ be an unditrected graph. A clique C is maximal if it cannot be extended into a larger clique. In other words, C is not a subset of a larger clique.*

Definition 15 (closed itemset VS dataset graph)**.** *Let $\mathcal{A}(n \times n)$ be the adjacency matrix of the consistent graph $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$. An itemset I is closed if its items are the vertices of a maximal clique $C\mathcal{Q}$ such that all the labels of the corresponding bonds share at least θ sub-labels.*

In graph theory, finding maximal cliques is a hard problem. It is difficult to find a polynomial algorithm that provides this kind of clique. However, in this paper, we propose Algorithm 2, called CfCi. This algorithm, based on the bonds' sub-labels, is exact and has polynomial complexity.

Considering, for example, the graph given in Figure 10. The corresponding structures are given below.

$$\mathcal{D} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \mathcal{A} = \begin{bmatrix} 110 & 0 & 110 \\ 0 & 101 & 101 \\ 110 & 101 & 111 \end{bmatrix}$$

$$Labs = [110, 101, 111]$$

$$supp = [2, 2, 3]$$

$$IndVect = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Algorithm 1, having the polynomial complexity $O(\frac{nbcons \times (nbcons-1)}{2})$ (see Proposition 1), provides the CFIs corresponding to the initial labels that belong to $Labs$.

Proposition 1. *Algorithm 1 has the worst case complexity* $\approx O(\frac{nbcons^2 - nbcons}{2})$*, where* $nbcons$ *is the number of the consistent vertices.*

Proof. In this algorithm, we have three nested loops. At most, at the outer one $nbcons$ vertices will be treated and at the inner ones $\frac{nbcons-1}{2}$ will be processed. Thus, the complexity of $BasicCFI$'s $\approx O(\frac{nbcons^2 - nbcons}{2})$. □

Let's consider, for example, the graph $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ illustrated in Figure 11.

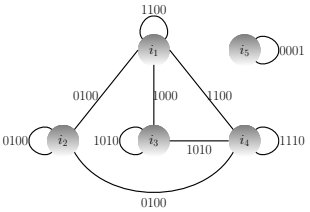


Figure 11: $\mathcal{G} = \langle X, U, \mathcal{L} \rangle$ models the dataset given in Table 1

We suppose that $\theta = 1$, which makes the graph consistent (see 13) and the correct input of our algorithm.

Figure 12 shows the different steps of Algorithm 1 applied to the graph. The sub-figures contain arrows in green, blue and red, that correspond to the fact that the intersection between the first label and the second generates a label equal to the first one, a label with support$\geqslant \theta$ and a label with support$< \theta$, respectively.

The algorithm 2, called CfCi, is the proposed algorithm of the polynomial complexity $\approx O(\frac{l^3 - l^2}{2}(v-1))$, where l is the labels' number and v is the vertices' number (see Proposition 2). It provides all the CFIs of the tackled dataset since the second loop of the algorithm computes all the possible labels' intersections. In other words, each new generated label whose$supp \geqslant \theta$ should lead to a new CFI.

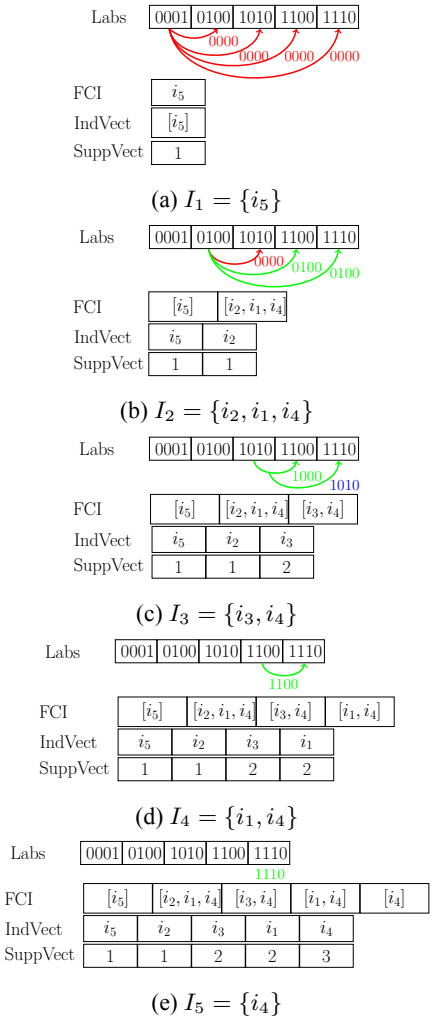


Figure 12: Generated CFIs by Algorithm 1. $IndVect$ and $SuppVect$ contain the vertices who are at the origin of the corresponding labels and their supports, respectively.

To find a new maximal clique, our algorithm calls the function $NewClique$ (see Algorithm 3). $NewClique(r)$ computes the incident vertices to $IndVect\,[r]$, that share at least θ sub-labels contained in $Labs\,[r]$.

Proposition 2. *Algorithm 2 has the worst case complexity* $\approx O(\frac{l^3 - l^2}{2}(v-1))$*, where* l *is the labels' number and* v *is vertices' number.*

Proof. In this algorithm, we have three nested loops. Consider l the labels' number and v vertices' number. At most, at the outer one l labels will be treated, and at the first and the second inner ones l and $\frac{l-1}{2}$ iterations will be processed, respectively. In addition, at most $v-1$ successors are handled in the function $NewClique$. Thus, the CfCi's complexity $\approx O(\frac{l^3 - l^2}{2} \times v - 1)$. □

In Figure 13, we illustrate the variation of the used structures: $Labs$, FCI, $IndVect$ and $SuppVect$. As given in the proof of Proposition 2, Algorithm 2 executes in $nbcons$ outer loops, where $nbcons$ is the number of the consistent

Algorithm 1 BasicCFI(M)

Require: θ: threshold, M: adjacency matrix of the cleaned graph, $nbcons$: number of consistent vertices; $ConsItem$: vector of consistent vertices;

$VertexSet \leftarrow \emptyset$;
for $i = 0..nbcons$ **do** ▷ the vertices are covered, according to their labels, from the smallest to the biggest one
  $VertexSet \leftarrow VertexSet \cup \{i\}$;
  $IndVect \leftarrow IndVect \cup \{i\}$;$SuppVect \leftarrow SuppVect \cup \{supp\,[i]\}$
  for $j = i + 1..nbcons$ **do**
    if $DistLabs(i, j) = supp\,[j]$ **then** ▷ return the distance between the labels i and j
      $VertexSet \leftarrow VertexSet \cup \{j\}$
    end if
  end for
  if $VertexSet \neq \emptyset$ **then**
    $FCI \leftarrow FCI \cup \{VertexSet\}$
  end if
  $VertexSet \leftarrow \emptyset$
end for
return FCI

Algorithm 2 CfCi()

Require: θ: threshold, M: adjacency matrix of the cleaned graph, $nbcons$: number of consistent vertices; $ConsItem$: vector of consistent vertices; $Labs$: set of the generated labels by $BasicCFI$ algorithm

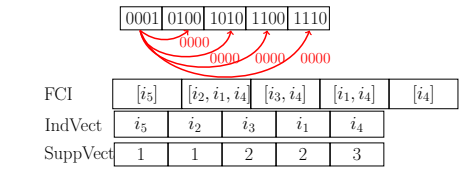
$beg \leftarrow 0$; $end \leftarrow |Labs|$; $e \leftarrow |Labs|$
for $k = 0 \cdots |Labs|$ **do**
  $bg = beg[k]$; $ed = end[k]$
  for $i = bg \cdots ed$ **do**
    $b \leftarrow 0$
    for $j = i + 1 \cdots ed$ **do**
      $cl \leftarrow 0$;$dst \leftarrow DistLabs(Labs[i], Labs[j])$
      if $dst \neq SuppVect[i] \wedge dst \geq \theta$ **then**
        $cl = comp2Labs(Labs[i], Labs[j])$ ▷ return to cl the intersection of the labels i and j
        if $cl \notin Labs$ **then**
          $Labs \leftarrow Labs \cup \{cl\}$; $IndVect \leftarrow IndVect \cup \{i\}$
          $SuppVect \leftarrow SuppVect \cup \{dst\}$
          $VertexSet \leftarrow NewClique(|Labs| - 1)$
          if $VertexSet \neq \emptyset$ **then**
            $VertexSet \leftarrow VertexSet \cup \{ind\}$;$FCI \leftarrow FCI \cup \{VertexSet\}$
            $VertexSet \leftarrow \emptyset$
          end if
          $b \leftarrow b + 1$
        end if
      end if
    end for
    if $b \geq 1$ **then**
      $beg \leftarrow beg + e$; $end \leftarrow end + ed + b$; $e \leftarrow e + b$
    end if
  end for
end for

vertices. Thus, since $nbcons = 5$, our algorithm executes 5 outer loops. Through sub-Figures 13a, 13b, 13c, 13d and 13e, we illustrate the different loops' details. We used the green arrow to express the novelty and consistency of the generated label, the blue to show that the generated label is already implemented, and the red when the support of the generated label is less than θ.

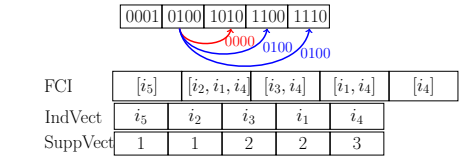
Thus, Algorithm 2, applied to the graph, provides six

Algorithm 3 *NewClique*

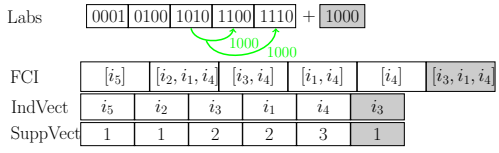
```
1:  function NewClique(r)
2:      VertexSet ← ∅
3:      ind ← IndVect[r]
4:      while v ∈ succ[ind] do
5:          dst = DistLabs(r, v)
6:          if dst = SuppVect[r] then
7:              VertexSet ← VertexSet ∪ {v}
8:          end if
9:      end while
10:     return VertexSet
11: end function
```



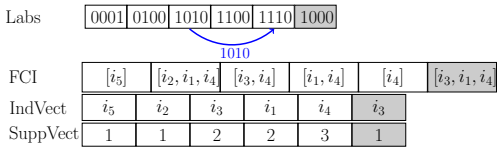
(a) No new label is generated.



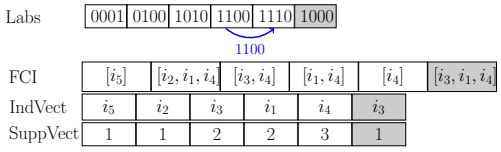
(b) No new label is generated.



(c) $I_6 = \{i_3, i_4, i_4\}$ is generated.



(d) No new label is generated.



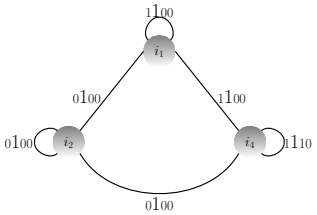
(e) No new label is generated.

Figure 13: Generated CFIs by Algorithm CfCi. $IndVect$ and $SuppVect$ contain the vertices who are at the origin of the corresponding labels and their supports, respectively.

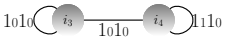
maximal cliques as given in Figure 14. Each clique $\mathcal{C}_i$ induces a closed itemset I_i.

The maximal cliques resulted from our algorithm $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$, $\mathcal{C}_5$ and $\mathcal{C}_6$ are illustrated via sub-Figures 14a, 14b,14c, 14d, 14e and 14f, respectively.

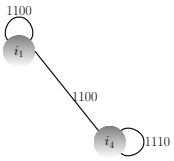
(a)
$\mathcal{C}_1$ modeling the CFI $I_1 = \{i_5\}$



(b) $\mathcal{C}_2$ modeling the CFI $I_2 = \{i_2, i_1, i_4\}$

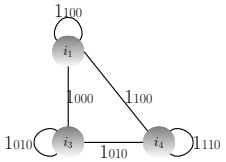


(c) $\mathcal{C}_3$ modeling the CFI $I_3 = \{i_3, i_4\}$



(d) $\mathcal{C}_4$ modeling the CFI $I_4 = \{i_1, i_4\}$

(e) $\mathcal{C}_5$ modeling the CFI $I_5 = \{i_4\}$



(f) $\mathcal{C}_6$ modeling the CFI $I_6 = \{i_3, i_1, i_4\}$

Figure 14:
Maximal cliques VS CFIs

6 Experiments

As stated in Section 2.3, besides CfCi, we consider in our experiments the algorithms NAFCP [10], NECLAT [13] and GrAFCI+ [11]. In addition, to have credible results, we chose the datasets according to their varied densities. More precisely, the chosen datasets : Mushroom, Lymph and Soybean datasets [1] have respectively 18%, 40% and 32% (see Table 3).

In Table 3, we introduce the characteristics of each tested dataset $DName(nbItem, nbTr, dns)$, where $DName$ is the dataset's name, and $nbItem$, $nbTr$ and dns are, respectively, the corresponding items' number, transactions' number and density. The datasets' information given in the table are:

- $freq$: the frequency adopted by the algorithm,
- θ: the corresponding support to $freq$,
- dns: the density of the dataset according to $freq$,

[1] The tested datasets are downloaded from `https://dtai.cs.kuleuven.be/CP4IM/datasets/`

– nvr: the vertices' number of the consistent dataset's graph,

– nbd: the bonds' number of the consistent dataset's graph.

Table 3: Characteristics of the tested dataset: Mushroom, Lymph and SoyBean.

Mushroom(119,8124,18%)				
$freq$	θ	dns	nvr	nbd
5%	406	30%	67	1168
25%	2031	51%	31	229
50%	4062	74%	12	52
80%	6499	94%	5	14
Lymph(148,68,40%)				
$freq$	θ	dns	nvr	nbd
5%	7	45%	59	1369
25%	37	60%	40	549
50%	74	76%	24	193
80%	118	91%	11	58
Soybean(50,630,32%)				
$freq$	θ	dns	nvr	nbd
5%	32	36%	44	601
25%	158	56%	23	157
50%	315	75%	12	55
80%	504	93%	5	13

The results provided by the tested algorithms are analyzed regarding the CPU-time, the number of computed CFIs (called ncl) and the memory usage (called Mem).

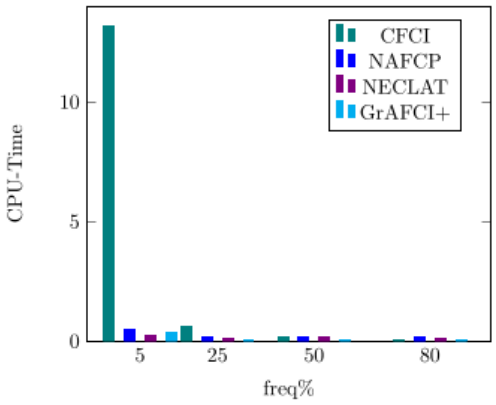
The used computer has the following configuration: Linux-20.04.1 on a notebook with Intel Core i7 and 8 GB/RAM. The experiments are performed in the Java language.

In sub-Figures 15a, 15b and 15c, we present the behavior of CfCi, NAFCP, NECLAT and GrAFCI+ applied to Mushroom, in terms of the gotten CPU-time, ncl (the number of closed itemsets) and mem (the memory usage), respectively.

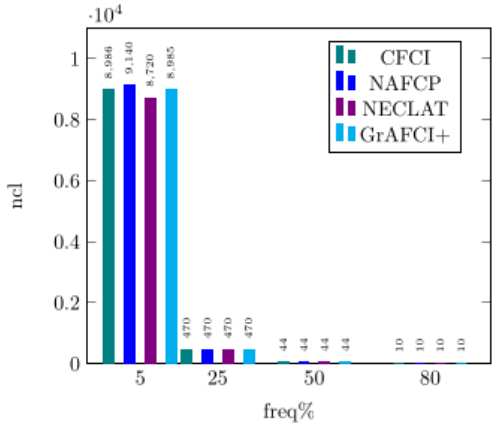
We notice, via sub-Figure 12a, that the applied methods have approximately the same CPU time until the frequency of 25%, where CfCi and GrAFCI+ become slower. At a frequency of 5%, CfCi and NECLAT are the fastest ones, whereas NAFCP is considerably the slowest one. Sub-Figure 15b shows that all the methods, at frequencies of 80%, 50% and 25%, provide the same number of CFIs. Whereas, at a frequency of 5%, our algorithm provides more CFIs than NECLAT and GrAFCI+, and less than NAFCP.

In terms of memory usage, sub-Figure 15c highlights the efficiency of our algorithm compared to the others.

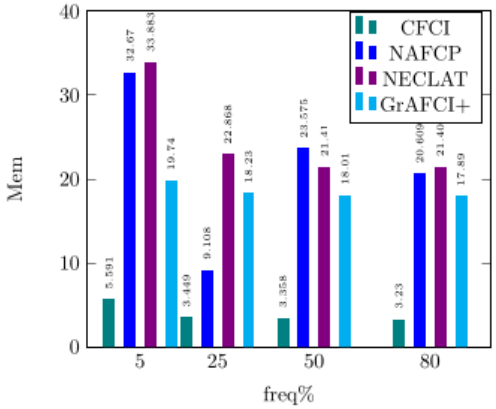
In Figure 16, we introduce three diagrams that express the behavior of the adopted methods, applied to the dataset



(a) CPU-time



(b) Number of CFIs

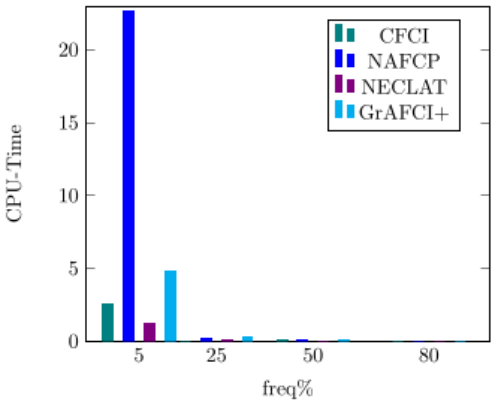


(c) Memory usage

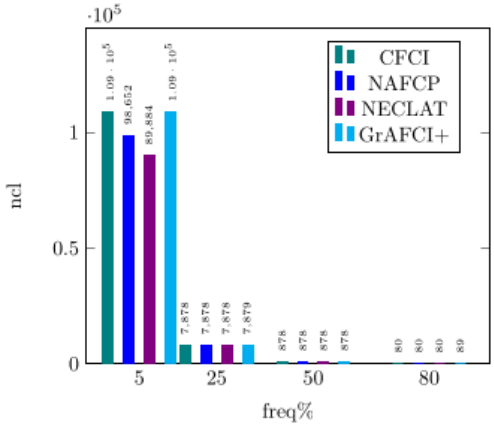
Figure 15: MUSHROOM dataset

$Lymph$, in terms of CPU-time, number of CFIs and memory usage.

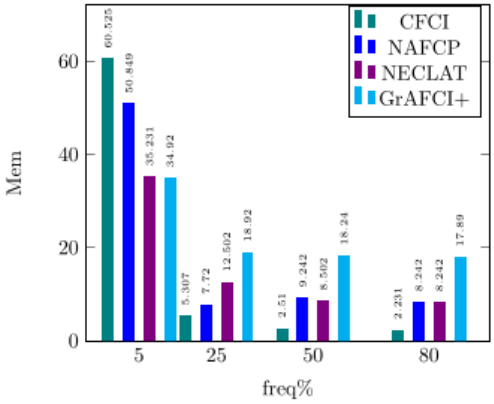
As shown in sub-Figure 16a, at frequencies 80% and 50%, all the methods have the same level of fastness, whereas CfCi becomes faster at frequency 25%. In addition, we notice that at frequency 5%, NAFCP becomes the slowest, NECLAT the fastest and CfCi becomes close to NECLAT.



(a) CPU-time



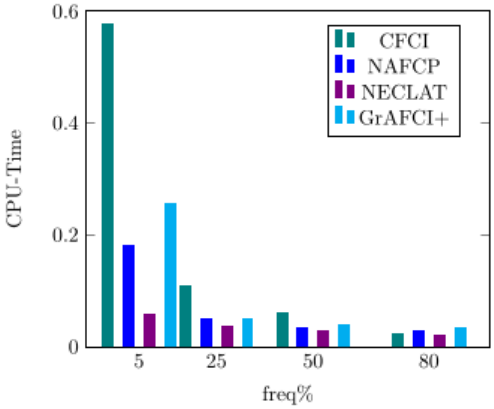
(b) Number of CFIs



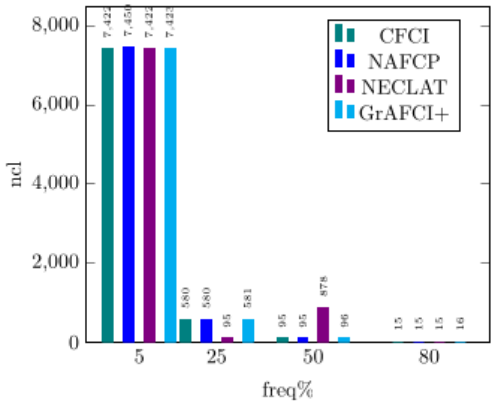
(c) Memory usage

Figure 16: LYMPH dataset

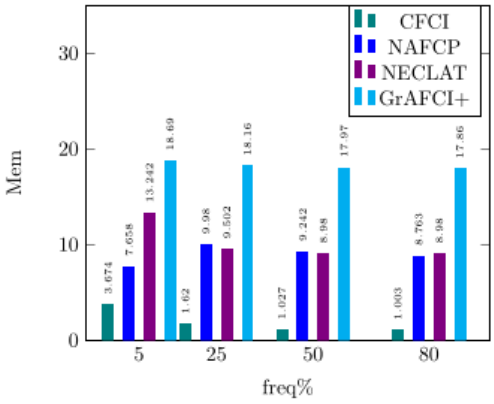
Sub-Figure 16b shows, in terms of the CFIs number, that at frequencies 80%, 50% and 25%, all the methods provide the same results except GrAFCI+, which gives more CFIs at frequency 80%. Whereas, at frequency 5%, the provided number of CFIs differs from algorithm to algorithm, and our algorithm provides more CFIs than NAFCP and NECLAT and less than GrAFCI+. Figure 17 contains SoyBean's diagrams, that reveal the behavior of the tested methods in terms of CPU-time, number of CFIs and memory usage.



(a) CPU-time



(b) Number of CFIs



(c) Memory usage

Figure 17: SOYBEAN dataset

We notice through sub-Figure 17a that CfCi and NECLAT start the fastest ones at $freq = 80\%$, then NECLAT remains the most efficient. On the other side, as the frequency decreases, the fastness of CfCi and GrAFCI+ decreases.

Regarding the number of CFIs, all the methods pro-

vide the same number of CFIs at each frequency, except GrAFCI+, which gives one more CFI.

We notice through sub-Figure 17a that CfCi and NECLAT start the fastest ones at $freq = 80\%$, then NECLAT remains the most efficient, and as the frequency decreases, CfCi and GrAFCI+.

Regarding the number of CFIs, all the methods provide the same number of closed frequent itemsets at each frequency, except GrAFCI+ which gives one more CFIs.

Sub-Figure 17c shows the efficiency of our algorithm in regards to memory usage. Contrary to GrAFCI+ and NECLAT, which are really costly.

As stated in Section 5, our algorithm is a complete approach that is designed to provide the exact number of CFIs. We observed through the diagrams given above, that NAFCP, GrAFCI+ and NECLAT may yield less or more CFIs in certain situations. In addition, our algorithm is the most efficient in terms of memory usage for all datasets, except Lymph with $freq = 5\%$. Thus, CfCi has the average CPU-time results and the best results in terms of the number of CFIs and memory usage.

We have to precise that the case where CfCi, applied to Lymph, provides the worst results in terms of memory usage. This result can be justified by the size of the corresponding graph. According to Table 3, the graph that models the Lymph dataset contains the biggest values, namely 118 vertices and 91 bonds, compared to the others.

7 Discussion

There are advantages and disadvantages to each of the previous data mining techniques. Most of them are based on tree structures (see Table 2), namely NList, Gr-tree and FP-tree in NAFCP, GrAFCI+ and NECLAT, respectively. This is because of the recursive relationship indirectly expressed in Property 1.

In addition, the authors of the tested techniques, proposed approximate approaches, and employed recursive algorithms to construct and to explore the corresponding tree structure.

Using recursive structures and approximated algorithms in most SOTA approaches means that memory usage is costly and finding all closed itemsets is not guaranteed. In addition, updating tree structures, when the corresponding dataset is modified, is generally a complex task.

In this article, we developed a new boolean and linear structure that can be easily updated when necessary, and iterative algorithms to optimize memory usage. Since our algorithm CfCi is exact and polynomial, it provides all the CFIs and shows good CPU-time.

In certain instances, when the frequency is 5%, CfCi slows down as compared to NAFCP, GrAFCI+ and NECLAT. This is because of the step that verifies if the computed CFI does not belong to the current set of CFIs.

Most times, the number of the CFIs ncl provided by our algorithm is the same as the other techniques. Sometimes, we get less or more CFIs, particularly when $freq = 5\%$. Our algorithm comprises calculating all the intersections between the bonds' labels and preventing duplicated CFIs. Thus, we have the incompleteness of the algorithms NAFCP, NECLAT and GrAFCI+ compared to CfCi.

Our approach is the best choice for optimizing memory usage, except in the Lymph dataset when $freq = 5\%$.

8 Conclusion and future works

This paper has introduced new modeling and solving approaches to find all the CFIs. We have presented an efficient graph modeling approach that is based on the clique notion in graph theory. Implementing our model by using a linear structure makes our graph model more flexible to be updated when transactions or items are added or removed. Our proposed model is based on the fact that a frequent itemset is considered a clique in an undirected graph. Thus, CfCi is a new algorithm, based on the maximal clique's principle, has been introduced to tackle the closed itemsets mining problem.

Our first experiments have shown the efficiency of CfCi in finding all the CFIs compared to the recent algorithms existing in the literature. This is because the proposed algorithm uses boolean structures and is basically conceived to find all CFIs, through systematic searches. Thus, our algorithm is more efficient than the tested methods in terms of the number of CFIs and memory usage, and approximately more efficient in terms of CPU time.

To summarize, the structure used in our approach is linear, and the proposed algorithm is iterative, exact, and polynomial. That is the origin of the promising results exposed and analyzed in this article.

However, there are some cases where our approach is less efficient than the newest methods, particularly in terms of CPU time. This is because of the CfCi's step, which tackles the duplicate CFIs cases.

As with any research work, a new version that inherits the advantages and avoids the disadvantages of our approach can follow our approach. Exploring opportunities for the practical implementation of the proposed algorithm will serve as the future direction for our research. We consider adapting our dataset graph model and CfCi algorithm to tackle classification and clustering problems.

References

[1] Fournier-Viger, Philippe and Chun-Wei Lin, Jerry and Truong-Chi, Tin and Nkambou, Roger (2019) A Survey of High Utility Itemset Mining, *High-Utility Pattern Mining: Theory, Algorithms and Applications*, Springer International Publishing, pp. 1–45. https://doi.org/10.1007/978-3-030-04921-8_1

[2] Charu C. Aggarwal (2015) *Data Mining-The Textbook*, Springer International Publishing. https://doi.org/10.1007/978-3-319-14142-8

[3] Han, Jiawei and Kamber, Micheline and Pei, Jian (2011) *Data Mining: Concepts and Techniques*, Morgan Kaufmann. http://www.sciencedirect.com/science/book/9780123814791

[4] Truong-Chi, Tin and Fournier-Viger, Philippe (2019) A Survey of High Utility Sequential Pattern Mining*High-Utility Pattern Mining*, Springer International Publishing, pp. 97–129. http://dx.doi.org/10.1007/978-3-030-04921-8_4

[5] Carson Kai-Sang Leung (2009)Frequent Itemset Mining with Constraints, *Encyclopedia of Database Systems*, Springer International Publishing, pp. 1179–1183. https://doi.org/10.1007/978-0-387-39940-9_170

[6] Tias Guns and Siegfried Nijssen and Luc De Raedt (2011) Itemset mining: A constraint programming perspective, *Artificial Intelligence*, Elsevier BV, 175(12-13), pp. 1951–1983. https://doi.org/10.1016/j.artint.2011.05.002

[7] Wensheng Gan and Jerry Chun-Wei Lin and Philippe Fournier-Viger and Han-Chieh Chao and Justin Zhan (2017) Mining of frequent patterns with multiple minimum supports, *Engineering Applications of Artificial Intelligence*, Elsevier BV, 60(C), pp. 83–96. https://doi.org/10.1016/j.engappai.2017.01.009

[8] Fatima-Zahra El Mazouri and Said Jabbour and Badran Raddaoui and Lakhdar Sais and Mohammed Chaouki Abounaima and Khalid Zenkouar (2019) Breaking Symmetries in Association Rules, *Procedia Computer Science*, Elsevier BV, 148(C), pp. 283–290. https://www.sciencedirect.com/science/article/pii/S1877050919300523

[9] Zaki, M.J. and Hsiao, C.-J. (2005) Efficient algorithms for mining closed itemsets and their lattice structure, *Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), 17(4), pp. 462–478. http://dx.doi.org/10.1109/tkde.2005.60

[10] Tuong Le and Bay Vo (2015) An N-list-based algorithm for mining frequent closed patterns,*Expert Systems with Applications*, Elsevier BV, 42(19), pp. 6648-6657. https://doi.org/10.1016/j.eswa.2015.04.048

[11] Ledmi, Makhlouf and Zidat, Samir and Hamdi-Cherif, Aboubekeur (2021)GrAFCI+ A Fast Generator-Based Algorithm for Mining Frequent Closed Itemsets, *Knowledge and Information Systems*, Springer Science and Business Media LLC, 63(7), pp. 1873–1908. https://doi.org/10.1007/s10115-021-01575-3

[12] Grahne, G. and Zhu, J. (2005) Fast algorithms for frequent itemset mining using FP-trees, *Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), 17(10), pp. 1347–1362. http://dx.doi.org/10.1109/tkde.2005.166

[13] Nader Aryabarzan and Behrouz Minaei-Bidgoli (2021) NEclatClosed: A vertical algorithm for mining frequent closed itemsets, *Expert Systems with Applications*, Elsevier BV, 174(C), pp. 114738. https://doi.org/10.1016/j.eswa.2021.114738

[14] Gang Fang and Yue Wu and Ming Li and Jia Chen (2015) An Efficient Algorithm for Mining Frequent Closed Itemsets, *Informatica (Slovenia)*, Slovenian Society Informatika, 39(1), pp. 87–98. https://api.semanticscholar.org/CorpusID:214751872

[15] Renata Iváncsy and István Vajk (2005) Fast Discovery of Frequent Itemsets: a Cubic Structure-Based Approach, *Informatica (Slovenia)*, Slovenian Society Informatika, 29(1), pp. 71–78. http://www.informatica.si/index.php/informatica/article/view/19

[16] Alkenani, Jawad and Kheerallah, Yousif Abdulwahab (2023) A New Method Based on Machine Learning to Increase Efficiency in Wireless Sensor Networks, *Informatica (Slovenia)*, Slovenian Society Informatika, 46(9), pp. 45–52. http://dx.doi.org/10.31449/inf.v46i9.4396

[17] Al-Jammali, Karrar (2023) Prediction of Heart Diseases Using Data Mining Algorithms, *Informatica (Slovenia)*, Slovenian Association Informatika, 47(5), http://dx.doi.org/10.31449/inf.v47i5.4467

[18] Karna, Hrvoje (2020) Data Mining Approach to Effort Modeling On Agile Software Projects, *Informatica (Slovenia)*, Slovenian Association Informatika, 44(2), pp. 231–139. http://dx.doi.org/10.31449/inf.v44i2.2759

[19] Awad, Fouad Hammadi and Hamad, Murtadha M. (2023) Big Data Clustering Techniques Challenged and Perspectives: Review, *Informatica (Slovenia)*, Slovenian Association Informatika, 47(6), pp. 203–218. http://dx.doi.org/10.31449/inf.v47i6.4445

[20] Agrawal, Rakesh and Srikant, Ramakrishnan (1994) Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of the 20th International Conference on Very Large Data Bases*,Morgan

Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 487–499. https://dl.acm.org/doi/10.5555/645920.672836

[21] Saïd Jabbour and Mehdi Khiari and Lakhdar Sais and Yakoub Salhi and Karim Tabia (2013) Symmetry-Based Pruning in Itemset Mining, *25th IEEE International Conference on Tools with Artificial Intelligence -ICTAI*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 483–490. https://doi.org/10.1109/ICTAI.2013.78

[22] Takeaki Uno and Masashi Kiyomi and Hiroki Arimura (2004) LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets, *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, CEUR-WS.org, Brighton, UK, pp. 1–11. https://ceur-ws.org/Vol-126/uno.pdf

[23] Agrawal, Rakesh and Imieliński, Tomasz and Swami, Arun (1993) Mining association rules between sets of items in large databases, *International Conference on Management of Data*, Association for Computing Machinery, Washington, D.C., USA, pp. 207–216. https://doi.org/10.1145/170035.170072,

[24] Yun Chi and Haixun Wang and Yu, P.S. and Muntz, R.R. (2004) Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window, *Fourth International Conference on Data Mining (ICDM'04)*, IEEE, Brighton, UK, pp. 59–66. http://dx.doi.org/10.1109/icdm.2004.10084

[25] Belaid, Mohamed-Bachir and Bessiere, Christian and Lazaar, Nadjib (2019) Constraint Programming for Mining Borders of Frequent Itemsets,*Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Macao, China, pp. 1064–1070. https://doi.org/10.24963/ijcai.2019/149

[26] AlZoubi, Wael (2015) An Improved Graph Based Method for Extracting Association Rules, *International Journal of Software Engineering and Applications*, Academy and Industry Research Collaboration Center (AIRCC), pp. 1–10. http://dx.doi.org/10.5121/ijsea.2015.6301

[27] Raedt, Luc De and Guns, Tias and Nijssen, Siegfried (2010) Constraint programming for data mining and machine learning, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI Press, Atlanta, Georgia, pp. 1671–-1675. https://dl.acm.org/doi/abs/10.5555/2898607.2898874

[28] Schlegel, Benjamin and Gemulla, Rainer and Lehner, Wolfgang (2011) Memory-efficient frequent-itemset mining, *Proceedings of the 14th International Conference on Extending Database Technology*, Association for Computing Machinery, Uppsala, Sweden, pp. 461–472. https://doi.org/10.1145/1951365.1951420

[29] Tiwari, Vivek and Tiwari, Vipin and Gupta, Shailendra and Tiwari, Renu (2010) Association rule mining: A graph based approach for mining frequent itemsets, *2010 International Conference on Networking and Information Technology*, IEEE, Manila, Philippines, pp. 309–313. http://dx.doi.org/10.1109/icnit.2010.5508505

[30] Gouda, K. and Zaki, M.J.(2001) Efficiently mining maximal frequent itemsets, *Proceedings 2001 IEEE International Conference on Data Mining*, IEEE, San Jose, CA, USA, pp. 163–170. http://dx.doi.org/10.1109/icdm.2001.989514

[31] Han, Kyong Rok and Kim, Jae Yearn (2005) FCILINK: Mining Frequent Closed Itemsets Based on a Link Structure between Transactions, *Journal of Information & Knowledge Management*, World Scientific Pub Co Pte Lt, pp. 257–267. https://doi.org/10.1142/S0219649205001213

[32] De Raedt, Luc and Guns, Tias and Nijssen, Siegfried (2008) Constraint Programming for Itemset Mining, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Las Vegas, Nevada, USA, pp. 204–212. https://doi.org/10.1145/1401890.1401919