A Closed Frequent Subgraph Mining Algorithm in Unique Edge Label Graphs

Nour El Islem Karabadji[1,2], Sabeur Aridhi[3(✉)], and Hassina Seridi[2]

[1] Preparatory School of Science and Technology, P.O. Box 218, 23000 Annaba, Algeria

[2] LabGED Laboratory, Badji Mokhtar University, P.O. Box 12, 23000 Annaba, Algeria

{karabadji,seridi}@labged.net

[3] Aalto University, School of Science, P.O. Box 12200, 00076 Espoo, FI, Finland

sabeur.aridhi@aalto.fi

Abstract. Problems such as closed frequent subset mining, itemset mining, and connected tree mining can be solved in a polynomial delay. However, the problem of mining closed frequent connected subgraphs is a problem that requires an exponential time. In this paper, we present ECE-CloseSG, an algorithm for finding closed frequent unique edge label subgraphs. ECE-CloseSG uses a search space pruning and applies the strong accessibility property that allows to ignore not interesting subgraphs. In this work, graph and subgraph isomorphism problems are reduced to set inclusion and set equivalence relations.

Keywords: Unique edge labels · Closed frequent subgraph · Graph/subgraph isomorphism · Set inclusion/equivalence

1 Introduction

The problem of interesting pattern mining is a main task in pattern mining and has several application domains such as social network analysis [6], bioinformatics [1,13] and Web mining [14]. It consists in finding patterns that satisfy a set of constraints. The frequency is one of the most used constraints [4,5,7,10]. A frequent pattern in a given collection/database $\mathcal{D}$ is a pattern that occurs at least in δ structures of the database where δ is a given support threshold. In general, the size of the set of frequent patterns is too large. This is due to the downward closure property (all generalizations or sub-patterns of a frequent pattern must be frequent). Thus, the enumeration and the analysis of such a big set of frequent patterns is a challenging problem. To overcome this issue, many research works have focused on special types of frequent subgraphs that allow to restore the set of all frequent subgraphs. These particular patterns are closed and maximal frequent subgraphs. In this context, many efficient algorithms for mining closed and maximal patterns have been proposed such as CloseGraph [18], SPIN [9], Margin [16] and ISG [15]. While mining closed and maximal patterns like itemsets,

DOI: 10.1007/978-3-319-41920-6_4

keys and trees needs a polynomial delay [17], [11], and [12], the situation gets more complicated when complex patterns such as graphs are considered. The task of mining graph patterns is called frequent subgraph mining (FSM) and includes two main phases: (1) generation of candidates and (2) frequency test. Frequency test has a variable cost according to pattern complexity. For example, itemsets use set inclusion relation to test frequency, whereas trees use subtree isomorphism. In contrast to these cases that need only a polynomial delay, an exponential delay is required for graph databases (using subgraph isomorphism, which is an *NP-complete* problem).

Recently, there has been an increased interest to identify a practically relevant tractable graph class. Well-behaved outer-planar and unique edge label graphs have been encoded using itemset codes that preserve subgraph isomorphism orders (i.e., comparability). These codes are Block and Bridge Preserving (BBP) [8] and Edges and Converses Edges triplets (ECE) [15] that allow solving the isomorphism problem in a polynomial time.

In this paper, we propose ECE-CloseSG (**E**dges and **C**onverses **E**dges **R**epresentation for **Close**d **S**ub**G**raphs), a novel algorithm that investigates the search space strong accessibility to reduce the number of generated subgraphs and to find closed frequent unique edge label connected subgraphs. Moreover, ECE-CloseSG uses a set-theoretical representation in the candidate generation step and ECE encoding in the frequency closeness computation step. The strong accessibility property allows to jump from a closed subgraph to its immediate closed successors, which allows a significant reduction on the visited candidates. We notice that the set-theoretical representation facilitates the candidate generation task, and the ECE encoding allows graph (respectively subgraph) isomorphism problem to be reduced to set equivalence (respectively set inclusion) where the problem can be solved in polynomial time in the worst case. We compared the performance of ECE-CloseSG to a naive algorithm that tries to visit all the frequent subgraphs and outputs only the closed ones.

This paper is organized as follows. The next section presents the used notations. In Section 3, the proposed approach for closed frequent subgraphs in unique edge label graphs is presented. In Section 4, comparative results on synthetic datasets are reported. In Section 5, a survey of closed and maximal frequent subgraph mining algorithms is presented.

2 Preliminary

In this section, we present some basic notations and definitions.

2.1 Notations

Definition 1. ***(Graph)*** *A graph $G = (V, E)$ is a collection of objects where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges.*

We define a labelled graph by adding a set of labels Ψ and a function $\Gamma : V \cup E \rightarrow \Psi$ that assigns for each vertex and for each edge a label of Ψ.

Definition 2. ***(Labelled graph)*** *A graph $G = (V, E, \Psi, \Gamma)$ is a labelled graph where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, Ψ is a set of labels and Γ is a labelling function for all edges and nodes.*

An undirected graph is a graph in which edges have no orientation. An edge $e = (u, v)$ has two end-points u and v. Two edges are adjacent if they share the same end points. The degree of a vertex v (denoted by $deg_G(v)$) is the number of its incident edges.

Definition 3. ***(Unique edge label graph)*** *A graph $G = (V, E, \Psi, \Gamma)$ is a unique edge label graph if and only if each edge label occurs at most once.*

Example 1. Figure 1 shows some examples of unique edge label graphs. The graph G_4 is not a unique edge label graph because edges (a, b) and (a, c) share the same label.

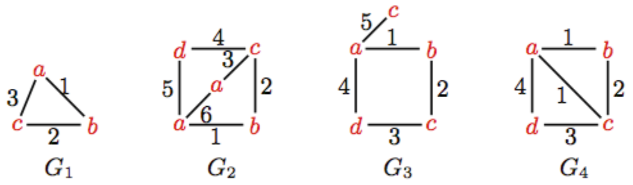

Fig. 1. Example of undirected labelled connected graphs.

Definition 4. ***(Isomorphism)*** *A graph isomorphism from $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ is a bijective function φ from E_1 to E_2 such that: $\forall\ u, v \in V_1 \iff (\varphi(u), \varphi(v)) \in E_2$.*

Definition 5. ***(Subgraph isomorphism)*** *$G_1 = (V_1, E_1)$ is a subgraph isomorphism into $G_2 = (V_2, E_2)$ if there is a bijection function φ from G_1 to G'_2 where G'_2 is a subgraph ($\subseteq_{sgi}$) of G_2.*

2.2 Closure Operator and Set System

Let $\mathcal{P}$ be a poset system (a set of subsets). A mapping $\sigma : \mathcal{P} \to \mathcal{P}$ is called a closure operator if it satisfies for all $X, Y \in \mathcal{P}$ that

- $X \subseteq \sigma(X)$(extensivity);
- $X \subseteq Y \to \sigma(X) \subseteq \sigma(Y)$ (monotonicity);
- $\sigma(X) = \sigma(\sigma(X))$(idempotence).

In data mining, a different closeness definition is used. The support-closed pattern of a dataset is defined as follows. Given a transactional database, a pattern is closed if there is no specific pattern that has the same support of the original pattern.

In the following, we give a definition of a set system and we present some of its properties.

Definition 6. *(Set system). A set system is an ordered pair* $(R, \mathcal{P})$*, where* R *is the ground finite set and* $\mathcal{P}$ *is a non-empty subset of the power set of* R*,* $\mathcal{P} \subseteq 2^R$*. A non-empty set system* $(R, \mathcal{P})$ *is:*

- **a closed system** if $R \in \mathcal{P}$, and $X, Y \in \mathcal{P}$ implies $X \cap Y \in \mathcal{P}$;
- **accessible** if for all $X \in \mathcal{P}\backslash\{\emptyset\}$ there is an $e \in X$ such that $X\backslash\{e\} \in \mathcal{P}$;
- **strongly accessible** if for every $X, Y \in \mathcal{P}$ satisfying $X \subset Y$, there is an $e \in Y\backslash X$ such that $X \cup \{e\} \in \mathcal{P}$;
- **independent** if $Y \in \mathcal{P}$ and $\forall X \subseteq Y \rightarrow X \in \mathcal{P}$;
- **a confluent** if $\forall I, X, Y \in R$ with $\emptyset \neq I \subseteq X$ and $I \subseteq Y$ it holds that $X \cup Y \in \mathcal{P}$.

3 Closed Frequent Subgraph Mining in Unique Edge Label Graphs

In this section, we present ECE-CloseSG, an algorithm for closed frequent subgraph mining in unique edge label graphs. We first present the subgraph system, the ECE representation and the support closure operation. Then, we present the basic steps of the ECE-CloseSG algorithm.

3.1 Connected Subgraph System

The main objective of the proposed approach is to restrict the search space. To this end, the strong accessibility of the subgraph system, and non-redundancy of the data $\mathcal{D}$ are required. A subgraph system considered in this work is the family of edge sets that induce connected subgraphs of $G_i \in \mathcal{D}$. A subgraph system $\mathcal{P}_i$ of a given graph $G_i(V_i, E_i) \in \mathcal{D}$ satisfies $\mathcal{P}_i \subseteq \mathcal{P}(E_i)$, where $\mathcal{P}(E_i)$ denotes the power set of the set of edges E_i. Here, we study the strong accessibility of a connected subgraph system $\mathcal{P}_i$. A subgraph system $\mathcal{P}_i$ is strongly accessible means that every $X \in \mathcal{P}_i$ can be reached from all $Y \subset X$ with $Y \in \mathcal{P}_i$ via extension with single edge inside $\mathcal{P}_i$.

Theorem 1. *A unique edge label subgraph system* $\mathcal{P}_i$ *is strongly accessible.*

Proof. Let X be a connected subgraph where $X \in \mathcal{P}_i$, if X is a graph that contains a cycle then we just drop an edge e from the cycle and the result $X\backslash e \in \mathcal{P}_i$. Otherwise, if X does not contain a cycle (a tree), we just drop one of its leaves. Given two graph $X_1, X_2 \in \mathcal{P}_i$ with $X_2 \subset X_1$. Assume that there is no edge $e \in X_1\backslash X_2$ such as $X_2 \cup e \in \mathcal{P}_i$. So, X_2 and $X_1\backslash X_2$ are two disconnected components in $G[X_1]$ which contradicting the choice of X_2.

This property improves the enumeration process because any closed frequent subgraph can be reached from anyone that has been already found by one edge augmentation [2].

Table 1. ECE 3-edge graph encoding

Graph	ECE Items	COMMONID	TYPEID
G_1	$(a,1,a),(a,2,a),(a,3,a),(1,a,2),(2,a,3),(1,a,3)$	(1,2,3)	((1,2,3),150)
G_2	$(a,1,a),(a,2,a),(a,3,a),(1,a,2),(2,a,3),(1,a,3)$	(1,2,3)	((1,2,3),200)
G_3	$(a,1,a),(a,2,a),(a,3,a),(1,a,2),(2,a,3)$	(1,2,3)	((1,2,3),100)

3.2 Edges and Converse Edges (ECE) Representation

Graph representation has a significant influence on memory usage and execution time. In this work, ECE-CloseSG uses two different graph representations, where the algorithm swings between: (1) a set-theoretical (vertices, and edges sets) to facilitate the augmentation and the connectivity tasks, and (2) an ECE codes to simplify the frequency tests.

The ECE encoding is a mapping of different parts of graphs to set of items. Every edge $e = (u, v)$ is represented by a 3-tuple $(et_u,\ et_e,\ et_v)$, where et_u, et_v are the labels of the two vertices connected by e; et_e is the label of e. Moreover, each 3-edge connected substructure is represented by two sign items (i.e., items that show the 3-edge connectivity and their structural form: a linear chain, a triangle, or a spike). Thus, to ensure unambiguous decoding of an ECE code to the right graph. Sign items can be presented by a first unique item that is assigned to each three connected edges noted the COMMONID item, and a second item noted TYPEID item to precise the type of the three edges code (i.e., triangle, spike and linear chain). Table 1 presents ECE codes of the three edge graphs (G_1, G_2, G_3) illustrated in Figure 2.

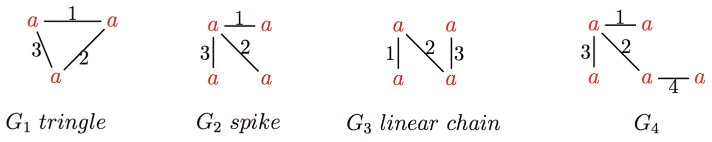

Fig. 2. Example of connected graphs.

Table 1 shows an identical ECE code of two different graphs G_1 and G_2 where the codes of G_1 and G_2 include the ECE code of G_3. However, this encoding allows preserving graphs' equivalence relation. Additionally, it is trivial to note that a graph ECE code is the union of the ECE codes of its 3-edge connected subgraphs.

As mentioned earlier, we propose to use ECE codes to apply subset relation that replaces the subgraph isomorphism test. This will reduce the frequency test complexity. Unfortunately, there is a failure case, and ECE encoding does not preserve incompatibilities and affects frequency test's accuracy. Therefore, two incomparable graphs can have two comparable ECE codes. This occurs in graphs having more than two linear chains' subgraphs. Figure 3 presents two connected incomparable graphs g_1(i.e., g), and g_2(i.g., g') and their 3-edge connected subgraphs (parts (a) $\mathcal{E}(g_1)$ and (b) $\mathcal{E}(g_2)$). The two graphs are not isomorphic, but g_2 contains all 3-edge subgraphs of g_1 (i.e., $\mathcal{E}(g_1) \subset \mathcal{E}(g_2)$). Thus, g_1 ECE code is included in g_2 ECE code. According to this ECE failure, g_1 is a subgraph of all graphs G_i that include g_2, but this is not the case when subgraph isomorphism is used. Assume that the database $\mathcal{D}$ contains two graphs G_1 and G_2. The two graphs contain g_1 and g_2 respectively. For a frequency threshold =2, g_1 is listed frequent, but it occurs only once at the first graph G_1.

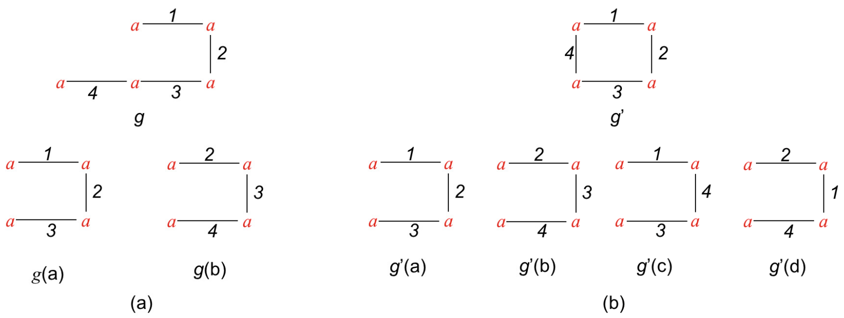

Fig. 3. Encoding failure example.

For two incomparable graphs g and g', we have an ECE encoding failure when subgraphs $\mathcal{E}(g')$ of g' contain all subgraphs $\mathcal{E}(g)$ of g. Specifically, there are two equal sized subgraphs $s \subseteq_{sgi} g$ and $s' \subseteq_{sgi} g'$ having the same edges and $\mathcal{E}(s) \subset \mathcal{E}(s')$, but s and s' are incomparable graphs. The first subgraph (i.e., s) is a simple path (sequence of linear chains) that begins and ends with two distinct vertices with an identical label. The second one (i.e., s') is a circuit that shares the same edges of s in the same order, but begins and ends with the same vertex. Closing the last edge of a chain toward the initial vertex produces two new linear chains' subgraphs. Therefore, to detect a failure with respect to two graphs g and g', we first seek couples of suspicious edges. Then, the suspected pairs of edges are tested. These edges are identified following characteristics: (1) two edges (e_1, and e_2) are not connected if $ECE(g)$ does not admit a converse edge that represents a connection between e_1 and e_2, and (2) in $ECE(g)$, there is no COMMONID item that includes the two edges. Additionally, any edge expansion requires adding: (a) one edge code item, (b) some edge converse codes (according to the degrees of the vertices), and (c) some corresponding sign items. For instance, the graph G_4 in Figure 2 is an augmentation from G_2 by adding the edge $e = (a, 4, a)$. The code of G_4 ($ECE(G_4)$) is constructed by appending $ECE(G_2$, the code of the edge e, one converse code and two sign items ($ECE(G_4)$=$ECE(G_2) \cup \{(a, 4, a), (2, a, 4), (1, 2, 4), ((1, 2, 4), 100), (2, 3, 4), ((2, 3, 4), 100)\}$).

Moreover, the support set of X with respect to $\mathcal{D}$, noted $\mathcal{D}[X]$, is a set of transactions containing X set, where the support-closure operator is defined by $\sigma(X) = \cap\mathcal{D}[X]$. According to this support-closure operator, the result $ECE(g)$ can be ambiguous, when it contains a 3-edge COMMONID item without the presence of its TYPEID item. For example, let $\mathcal{D}' = \{ECE(G_1), ECE(G_2), ECE(G_3)\}$, $\sigma((a,1,a)) = \{(a,1,a),(a,2,a),(a,3,a),(1,a,2),(2,a,3),(1,a,3),(1,2,3)\}$, the three item sets share the three edge triplets $(a,1,a)$, $(a,2,a),(a,3,a)$, the converse edge triplets $(1,a,2),(2,a,3)$, and the 3-edge COMMONID item $((1,2,3))$, but they do not share any of the TYPEID items. The $\cap\mathcal{D}[(a,1,a)]$ would form an ambiguous itemset which would form more than one possible graph. A postprocessing phase is required for finding the maximal frequent itemsets that form closed no conflicting itemsets. This phase begins by forming an initial set of components based on the couple of edges that belong to each converse edge triplet, and recursively extending these components until they cannot be extended. According to our example, the components are $\{(a,1,a),(a,2,a),(1,a,2)\}$ and $\{(a,2,a),(a,3,a),(2,a,3)\}$.

Proposition 1. *Let $\mathcal{D}$ and $\mathcal{D}'$ two graphs, $ECE(\mathcal{D})$ a set of sets, $G_i \in \mathcal{D}$ a connected graph, g a subgraph of G_i such that $g \in \mathcal{P}_i$. If $y = \sigma(ECE(g))$ is an ambiguous itemset then it contains at least two maximal non-ambiguous itemsets t_0 and t_1 such that $\mathcal{D}'[i] = \mathcal{D}'[t_0] = \mathcal{D}'[t_1]$.*

Proof. Clearly for an unambiguous itemset y there exist at least one 3-edge COMMONID item without its TYPEID item, and this COMMONID item refers to at least two converse edges (in the case of a linear chain). This implies to start the postprocessing by two components, which are recursively extended to generate two maximal non-ambiguous itemsets t_0, t_1 such that $\mathcal{D}'[y] = \mathcal{D}'[t_0] = \mathcal{D}'[t_1]$.

Assuming the proposition 1, and the fact that ECE encoding with respect to the failure detection test is a bijective function from $\mathcal{P}_i \rightarrow ECE(\mathcal{P}_i)$, we deduce that for any connected graph $g \in \mathcal{P}_i$, the set of closed connected graphs that contain g is $C_g = \{ECE^{-1}(\sigma(ECE(g)))\}$.

3.3 Support Closure Operation

While the strong accessibility guarantees the existence of a chain between each consecutive closed connected subgraphs pair, calculating closure allows to jump directly from an immediate successor of a closed one g to its closed subgraph successor g^* [2]. We notice that the itemsets closure of an itemset I is the intersection of the transactions containing I as a subset [17]. Unfortunately, for connected subgraphs, the result of a subgraph closure is not unique and can be a disconnected graph that does not even belong to the system. Therefore, we conclude that there is no closure operator in P_i, and we define the closure support operation within $\mathcal{P}_i$ with respect to $\mathcal{D}$ as follows:

Definition 7. *Let P_i be a subgraph system and $\mathcal{D}$ be a dataset. The support closure of $\mathcal{P}_i$ with respect to $\mathcal{D}$ is defined by:*

$$\sigma(i) = max\ \Sigma(i)$$

Algorithm 1. | ECE-CloseSG

```
Input: a graph $G_j \in \mathcal{D}$, support closure operator $\sigma$.
Output: $\sigma(\mathcal{F}_j)$: closed frequent subgraphs of $G_j$.
 1: $E_F \leftarrow$ All frequent 1-edge graphs in $G_j$
 2: $i \leftarrow 0$
 3: $C_i \leftarrow \emptyset$
 4: $\sigma(\mathcal{F}_j) \leftarrow C_0$
 5: $IN_i \leftarrow$ GET-IND-GENERATOR $(C_i, E_F, G_j)$
 6: while $IN_i \neq \emptyset$ do
 7:     $C_{i+1} \leftarrow$ GET-CLOSED $(IN_i, E_F, G_j)$
 8:     $\sigma(\mathcal{F}_j) \leftarrow \sigma(\mathcal{F}_j) \cup C_{i+1}$
 9:     $IN_{i+1} \leftarrow$ GET-IND-GENERATOR $(C_{i+1}, E_F, G_j)$
10:     $i \leftarrow i + 1$
11: end while
        return $\sigma(\mathcal{F}_j)$
```

where $\Sigma(i) = \{i' \in \mathcal{P}_i : i \subseteq i' \; et \; \mathcal{D}[i] = \mathcal{D}[i']\}$

i.e., the set of subgraphs $\Sigma(i)$ *may have more than one maximal.*

From this definition, we clearly notice that the closure of a connected graph $g \in \mathcal{P}_i$ consists of all connected subgraphs of maximal size in $\mathcal{P}_i$ and having g as predecessors and similar occurrences in $\mathcal{D}$.

The set-theoretical representation simplifies the augmentation and connectivity test operations. However, the main motivation to use it is the transformation facility of graphs to ECE codes; that reduces the graph/subgraph isomorphism complexity to equivalence sets and inclusion sets tests. We recall that the closure of a graph g consists of a set of ECE codes. These ECE codes are the result of a post-processing of an ambiguous ECE code X calculated by intersecting graphs $G_i \in \mathcal{D}$ ECE codes that contain the graph inductive generator g code (i.e., $X = \cap\mathcal{D}[ECE(G_i)]$ $s.t$ $ECE(g) \subseteq ECE(G_i)$). The post-processing phase consists in replacing an ambiguous ECE code by the largest connected and not ambiguous itemsets I included in $ECE(g)$. Moreover, the code X may correspond to an unconnected subgraph $g' \notin G_i$. To avoid this latter problem, only the sub-code $I \subseteq X$ that contains $ECE(g)$ is considered. We note for the best case, where there is a unique closed subgraph Y (i.e., an unambiguous ECE code), calculating closure needs at least $|\mathcal{D}|$ inclusion tests and generation operations.

3.4 The ECE-CloseSG Algorithm

The main objective of our proposed subgraph mining algorithm is to restrict the number of visited subgraphs and to reduce the complexity of graph/subgraph isomorphism tests. Algorithms 1, 2 and 3 illustrate our algorithm.

Algorithm 2. | GET-IND-GENERATOR

Input: a graph $G_j \in \mathcal{D}$, a set of closed graphs C_i, and a set of frequent edges E_F.
Output: IN: a set of ECE codes of inductive graph generators.

```
1: $IN \leftarrow \emptyset$
2: for each $c \in C_i$ do
3:     for each $e \in E_F$ do
4:         if $c \cup e$ is connected then
5:             $g' \leftarrow$AUGMENTATION $(c, e)$
6:             $ECE(g') \leftarrow$ ENCODE $(g')$
7:             if IS-FREQUENT $(ECE(g'), \mathcal{D}')$ then
8:                 $IN \leftarrow IN \cup ECE(g')$
9:             end if
10:        end if
11:    end for
12: end for

        return $IN$
```

Starting from the closed set C_0 that contains only an empty subgraph (i.e., $\emptyset$) that still considered closed frequent when the data $\mathcal{D}$ is non-redundant [2]. Then, for each set of closed frequent subgraphs C_i, each closed $c \in C_i$ is extended to generate all possible inductive generator subgraphs (i.e., using GET-IND-GENERATOR function). Thus, a level IN_i of inductive generator subgraphs is generated from each level of closed frequent subgraphs C_i. Next, closure of each subgraph in IN_i is calculated to generate the next level of the closed frequent subgraphs C_{i+1} (i.e., using GET-CLOSED function). In order to avoid redundancy, all closed and inductive subgraphs already visited at previous levels are eliminated. Finally, these steps are repeated until no new generator inductive subgraph is generated (i.e., $IN_i = \emptyset$).

More specifically, ECE-CloseSG algorithm consists of four steps:

1. the graph data $\mathcal{D}$ is encoded to a transactional database $\mathcal{D}'$ containing a list of itemsets.
2. starting at the closed set C_0 which contains only an empty subgraph $\emptyset$, all codes of inductive generator subgraphs are generated from C_0. Algorithm 2 illustrates the pseudo-code of the GET-IND-GENERATOR function that allows generating inductive generator subgraphs. This function receives as input a set of closed graphs C_i, the frequent edges E_F and the graph G_i, and generates all the codes of the inductive generator graphs IN. For each closed subgraph $g \in C_i$, all extensions that generate connected subgraphs are applied and encoded as an ECE code. The frequent ones are added to the set IN.
3. for each set of ECE codes IN_i, the closure is computed. Algorithm 3 shows the pseudo-code of computing the closure function (noted GET-CLOSED). This function receives as input a set of ECE codes, the frequent edges E_F and the graph G_j. It generates the set of closed connected graphs C_{i+1}. For each of the ECE codes ($ECE(X) \in IN_i$), the intersection of all codes in $\mathcal{D}'$ that

Algorithm 3. | GET-CLOSED

Input: a graph $G_j \in \mathcal{D}$, a set of ECE codes IN_i, and a set of frequent edges E_F.
Output: C_i: a set of closed connected subgraphs.

```
1:  for each $ECE(X) \in IN_i$ do
2:      $ECE(Y) \leftarrow \cap\mathcal{D}'[ECE(X)]$
3:      if $ECE(Y)$ is an ambiguous code then
4:          $MaxECE \leftarrow$ GET-SUB-CODES $(ECE(Y))$
5:          for each $ECE(Z) \in MaxECE$ do
6:              $g'' \leftarrow$ RECONSTRUCT $(ECE(Z), G_j)$
7:              $C_i \leftarrow C_i \cup g''$
8:          end for
9:      else
10:         $g'' \leftarrow$ RECONSTRUCT $(ECE(Y), G_j)$
11:         $C_i \leftarrow C_i \cup g''$
12:     end if
13: end for

        return $C_i$
```

contain the code $ECE(X)$ with respect to a negative test of an encoding failure test. Further, for each code $ECE(X)$ a new code $ECE(Y)$ is generated ($ECE(Y) = \cap\mathcal{D}[ECE(X)]$). This code may correspond to an ambiguous and/or disconnected graph. Then, the ambiguity and the connectivity of each newly generated code $ECE(X)$ are checked. If it is an ambiguous code, $ECE(X)$ will be replaced by a set of unambiguous sub-codes $MaxECE$, and for each of these codes a graph g'' is reconstructed. If the code is unambiguous, then the corresponding graph g'' is generated. Finally, each graph is reconstructed and added to the set of closed subgraphs C_i.

4. each set of closed subgraphs C_i is added to the global closed set $\sigma(\mathcal{F}_j)$ and both step 2 and step 3 are repeated until no new inductive subgraph generator is generated. The global closed set $\sigma(\mathcal{F}_j)$ is returned.

4 Experimental Study

The proposed closed frequent subgraph mining algorithm is implemented in Java and tested on synthetic datasets, which are generated by GraphGen [3], a graph generator that generates graph data based on five parameters: T (the number of graphs), S the size (i.e., the number of edges) of each graph. The size is defined as a normal distribution with the input as the mean and five as the variance. ETE (respectively ETV) represents the number of distinct edges (respectively the number of the labels of the vertices), L represents the density of each graph. The latter is defined as the number of edges in the graph divided by the number of edges in a complete graph, i.e., $L = |E|/(|V|(|V|-1)/2)$. After data generating, a post-processing step is invoked, the edge labels of each graph are randomly modified such that graph satisfies the constraint of unique edge labels.

Table 2. Experimental results with frequency support=2% and ETV=20.

Dataset						Runtime (s)	
$\vert\mathcal{D}\vert$	S	ETE	L	$\vert\mathcal{F}\vert$	$\vert\mathcal{C}\vert$	ECE-CloseSG	NAIVE
	20	20	0.1	2867	1420	1.76	10.32
$T = 100$	20	20	0.4	5032	1660	4.86	20.59
	20	20	0.7	6058	1059	12.44	28.31
	20	20	0.1	4961	2004	5.43	62.07
$T = 200$	20	20	0.4	6342	2207	9.05	43.97
	20	20	0.7	7224	1897	15.08	63.38
	30	40	0.1	5690	3364	29.37	199.86
$T = 300$	30	40	0.4	7351	4144	60.87	338.92
	30	40	0.7	7731	3954	124.12	439.60
	30	40	0.1	5692	3590	77.75	1129.30
$T = 400$	30	40	0.4	8395	5585	183.29	1199.69
	30	40	0.7	8909	5149	440.35	1291.04
	30	40	0.1	6045	4243	153.05	3488.83
$T = 500$	30	40	0.4	9234	6721	487.61	2481.30
	30	40	0.7	9693	5982	944.77	3591.94

We run our experiments on a 2.9 GHz Intel i7 PC with 8 GB of RAM. Running times of ECE-CloseSG are analyzed for frequency support value of 2%. The set of graphs are generated by varying the parameters of the graph generator as follows: (1) the number of graphs T from 100 to 500; (2) S as 20 and 30 that overlap with a high probability the intervals [10,30] and [15,45] respectively; (3) Vertex and edge labels: ETV 20, ETE 20 and 40; (4) density ranges from 0.1 to 0.7. This variation of parameters aims to prove a picture of ECE-CloseSG behaviour over different graph datasets.

Table 2 lists the results of the proposed algorithm on synthetic datasets. It presents:

1. the number of frequent connected subgraphs generated by NAIVE, a naive algorithm that explores all the search space to find all frequent ones first, and then filters them to list only the closed ones.
2. the runtime and the number of closed frequent connected subgraphs listed by ECE-CloseSG.
3. the runtime of the NAIVE algorithm.

The presented results show a significant difference in the size of the set of frequent subgraphs $\mathcal{F}$ compared to the closed frequent ones $\mathcal{C}$. Moreover, Table 2 illustrates a total domination of the proposed algorithm compared to the NAIVE algorithm. ECE-CloseSG performs 10 times faster than the NAIVE algorithm. In addition, we observe that increasing the graph size and the densities allow to increase the runtime of ECE-CloseSG to list the closed subgraphs. Finally, ECE-CloseSG requires more important runtime with respect to the value of the density, which increases the number of triples in a graph, where the encoding and decoding, as well as inclusion and equivalence tests, will be more complex.

5 Related Works

Mining only closed and maximal frequent subgraphs is the common way to reduce the huge number of frequent subgraphs. A typical closed and maximal frequent subgraph mining task consists of two steps. The first step lies in finding all frequent subgraphs $\mathcal{F}$. The second step consists in filtering the frequent subgraphs in order to keep only the closed and maximal ones. This approach seems to be not efficient due to the huge number of subgraphs to be visited in order to find the maximal frequent ones. To avoid exploring all frequent subgraphs, existing closed and maximal frequent subgraph mining algorithms use pruning techniques to reduce the search space. In this section, we discuss most popular algorithms that were proposed to solve this problem.

CloseGraph [18] is a closed frequent subgraph mining algorithm that uses adjacency lists to store graphs and to test frequency, whereas an ordered sequence's edges code (called DFS lexicographic order) is used to generate candidates and to detect redundancy. CloseGraph adopts a pruning technique to restrict the search space. This pruning technique uses an equivalent occurrences property based on an early termination condition. This property allows to decide whether a descendant super-graph of a given graph is a closed one or not. The early termination condition means that the search process will be completely stopped for some descendant branches, which effectively reduces the search space.

In the case of maximal frequent subgraph mining, several algorithms such as SPIN [9], Margin [16] and ISG [15] have been proposed.

SPIN [9] is based on the fact that most of the frequent subgraphs are trees (acyclic graphs). In order to reduce the computation cost, SPIN mines all frequent trees from a graph database and then built groups of frequent subgraphs from the mined trees. Each group of frequent subgraphs consists in an equivalence class, where each class is composed of subgraphs that share the same canonical spanning tree. We mention that this grouping step is not efficient because we still need to list all frequent subgraphs to construct maximal and frequent ones. To avoid this problem, some optimization techniques (i.e., Bottom-Up Pruning, Tail Shrink, and External- Edge Pruning) could be integrated to speed up the mining process.

Margin [16] is a subgraph mining algorithm that mines only maximal frequent subgraphs. Margin is based on the fact that maximal frequent subgraphs lie in the middle of the search space, which implies a high computational cost. To overcome this problem, Margin restricts the search space by visiting only the set of promising subgraphs $\widehat{\mathcal{F}}$ that encloses the frequent and infrequent subgraphs lie on the border. For each graph $G_i \in \mathcal{D}$, Margin works as follows. First, it explores the search space (the subgraph system) in a depth-first way to find the representative subgraph R_i. Then, it applies a method called ExpandCut on the cut CR_i and R_i where CR_i is a super infrequent graph of R_i. ExpandCut finds the nearby cuts and recursively calls itself for each newly found cut, until no new cut can be found. Given a set of cuts $(C|P)$ in which ExpandCut is applied, the frequent subgraphs P for each cut is reported as promising frequent subgraphs $\widehat{\mathcal{F}}$, and then only maximal local frequent subgraphs $\mathcal{ML}$ are kept from the promising ones ($\widehat{\mathcal{F}}$). Finally, the maximal global frequent subgraphs $\mathcal{M}$ in the database $\mathcal{D}$ are listed by removing the set of graphs in $\mathcal{ML}$, which are proper subgraphs of other frequent graphs in $\mathcal{ML}$.

ISG [15] is an algorithm for mining maximal frequent subgraphs over a graph database with unique edge labels. In order to list these maximal frequent subgraphs, an itemset mining technique is used. The idea of ISG is to transform the problem of maximal graph mining to maximal itemset mining. First, the graph database $\mathcal{D}$ is transformed to a list of itemsets $\mathcal{D}'$, where each graph in $\mathcal{D}$ is encoded as a set of items. These items represent the edges and the converse edges of the graph and 3-edge substructure that are contained in the graph (i.e., triangle, spike, and linear chain). These 3-edge blocks are called secondary structures and are added to avoid the problem of edge triplets and converse edge triplets conversion to a graph. The edge triplets and converse edge triplets together do not guarantee that the given maximal frequent itemset can be converted into a graph. Each secondary structure is assigned to a unique item identifier in addition to the unique identifiers of the edge and the converse edge triplets that compose the 3-edge block. Second, the transaction database $\mathcal{D}'$ is used as input to a maximal itemset mining algorithm. The set of the maximal code itemsets $\mathcal{M}_I$ is enumerated. Generally, there is no guarantee that all these codes correspond to unique connected subgraphs (i.e., there is no bijection between $\mathcal{M}_I$ and a subgraph in $\mathcal{P}$). The main problem over $\mathcal{M}_I$ is that there are codes for which the conversion generates disconnected graphs. To avoid these problems, a post-processing step after the conversion of the codes that can not be converted unambiguously is required. This step consists in converting a disconnected code to a set of its connected graphs, and for the case of conflicting maximal frequent itemset, it is broken to form non-conflicting subsets. After this post-processing step, the set of generated subgraphs is filtered, and the subgraphs that are contained in other subgraphs are pruned. Finally, the set of maximal connected unique edge label subgraphs is mined. We mention that the ECE encoding does not preserve the incomparability of subgraphs in a particular case. Thus, we have noticed a failure that affects the frequency test, which explains the incomplete output. This phenomenon leads to produce false frequency results when two incomparable graphs encoding can be comparable.

Besides the fact that subgraph mining related tasks are complex, and graph/subgraph isomorphism test is a hard problem; current works do not show up a significant optimization in the size of visited subgraphs during the mining process.

6 Conclusion

In this paper, we have illustrated that the pattern search space system and the complexity of frequency test affect the enumeration process of closed and maximal frequent pattern mining algorithms. We proposed ECE-CloseSG, an algorithm that mines the unique edge label connected subgraphs based on an encoding of the input graphs into a set of ECE items. This encoding allowed us to reduce graph and subgraph isomorphism problems to set-equivalence and set-inclusion tests, respectively. We investigated the strong accessibility property in order to restrict the search space. The efficiency of the ECE-CloseSG algorithm depends on the position of the border between infrequent and frequent nodes of the search space. For a dense (respectively sparse) search space, the border lies in the higher (respectively lower) levels.

In future works, we plan to do a comparative study of our approach with existing closed frequent subgraph mining algorithms.

References

1. Aridhi, S., Sghaier, H., Zoghlami, M., Maddouri, M., Nguifo, E.M.: Prediction of ionizing radiation resistance in bacteria using a multiple instance learning model. Journal of Computational Biology **23**(1), 10–20 (2016)
2. Boley, M., Horváth, T., Poigné, A., Wrobel, S.: Listing closed sets of strongly accessible set systems with applications to data mining. Theoretical Computer Science **411**(3), 691–700 (2010)
3. Cheng, J., Ke, Y., Ng, W.: Graphgen: A graph synthetic generator (2006)
4. Chi, Y., Muntz, R.R., Nijssen, S., Kok, J.N.: Frequent subtree mining-an overview. Fundamenta Informaticae **66**(1–2), 161–198 (2005)
5. Ganter, B., Reuter, K.: Finding all closed sets: A general approach. Order **8**(3), 283–290 (1991)
6. Giatsidis, C., Thilikos, D., Vazirgiannis, M.: Evaluating cooperation in communities with the k-core structure. In: 2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 87–93 (2011)
7. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharma, R.S.: Discovering all most specific sentences. ACM Transactions on Database Systems (TODS) **28**(2), 140–174 (2003)
8. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. Data Mining and Knowledge Discovery **21**(3), 472–508 (2010)
9. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 581–586. ACM (2004)
10. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery **1**(3), 241–258 (1997)
11. Nourine, L., Petit, J.M.: Extending set-based dualization: Application to pattern mining. ECAI **242**, 630–635 (2012)
12. Nourine, L., Petit, J.M.: Dualization on partially ordered sets: Preliminary results. In: Choong, Y.W., Kotzinos, D., Spyratos, N., Tanaka, Y. (eds.) ISIP 2014. CCIS, vol. 497, pp. 23–34. Springer, Heidelberg (2016)
13. Saidi, R., Aridhi, S., Nguifo, E.M., Maddouri, M.: Feature extraction in protein sequences classification: A new stability measure. In: Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB 2012, pp. 683–689. ACM, New York (2012)
14. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web usage mining: Discovery and applications of usage patterns from web data. SIGKDD Explor. Newsl. **1**(2), 12–23 (2000)

15. Thomas, L., Valluri, S., Karlapalem, K.: Isg: Itemset based subgraph mining. Tech. rep., Technical Report, IIIT, Hyderabad, December 2009
16. Thomas, L.T., Valluri, S.R., Karlapalem, K.: Margin: Maximal frequent subgraph mining. ACM Transactions on Knowledge Discovery from Data (TKDD) **4**(3), 10 (2010)
17. Uno, T., Kiyomi, M., Arimura, H.: Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: FIMI, vol. 126 (2004)
18. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 286–295. ACM (2003)