A New Algorithm for Mining Frequent Connected Subgraphs based on Adjacency Matrices

Andrés Gago-Alonso[1,2,*] Abel Puentes-Luberta[1,3]
Jesús A. Carrasco-Ochoa[2] José E. Medina-Pagola[1]
José Fco. Martínez-Trinidad[2]

[1] Advanced Technologies Application Center
7a ♯ 21812, Siboney, Playa, CP: 12200, Havana, Cuba
{agago,apuentes,jmedina}@cenatav.co.cu

[2] Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro ♯ 1, Sta. María Tonantzintla, CP: 72840, Puebla, México
{ariel,fmartine}@inaoep.mx

[3] Faculty of Mathematics and Computer Sciences, University of Havana
San Lazaro Avenue, Felipe Poey Building, Vedado, CP: 10400, Havana, Cuba

July 9, 2009

Abstract

Most of the Frequent Connected Subgraph Mining (FCSM) algorithms have been focused on detecting duplicate candidates using canonical form (CF) tests. CF tests have high computational complexity, which affects the efficiency of graph miners. In this paper, we introduce novel properties of the canonical adjacency matrices for reducing the number of CF tests in FCSM. Based on these properties, a new algorithm for frequent connected subgraph mining called grCAM is proposed. The experiments on real world datasets show the impact of the proposed properties in FCSM. Besides, the performance of our algorithm is compared against some other reported algorithms.

Keywords: data mining, graph mining, frequent subgraphs, labeled graphs, canonical adjacency matrices

1 Introduction

Data mining over graph datasets is becoming increasingly important since advances in collecting and storing data have produced an explosive growth in the amount of available structured data. This situation has boosted the necessity

*Corresponding author

of new algorithms to transform this big amount of data in useful information for decision makers. The development of such algorithms requires techniques that usually need long time and a lot of memory. One of these techniques is the frequent connected subgraph mining (FCSM) which is defined as the process of finding connected subgraphs that frequently occur in a collection of graphs.

Recently, FCSM has become in an interesting theme in data mining with wide applications [6], including mining substructures from chemical compound databases, XML documents, citation networks, biological networks, and so forth. Labeled graphs can be used to model relations among data in the aforementioned applications because labels can represent attributes of entities and relations among themselves [8]. As a consequence, several algorithms have been developed for FCSM in collections of labeled graphs.

The first graph mining system, called SUBDUE, was proposed by Holder *et al.* [7]. This system is based on minimum description length and background knowledge. Another system for frequent subgraph mining in chemical compounds datasets was developed using inductive logic programming [13]. However, the first algorithm for finding all frequent subgraphs (connected or unconnected) in a collection of labeled graphs was AGM [10], followed by FSG [12] and AcGM [11], which were designed for mining all frequent connected subgraphs. These algorithms are similar to the original Apriori algorithm [1] for mining frequent itemsets.

Later, pattern growth based algorithms such as gSpan [18], MoFa [3], FFSM [9], Gaston [14] and gRed [5] were developed. Previous comparative studies have shown that the pattern growth based algorithms have better performance than the Apriori based ones [15, 17].

The emergence of duplicate candidates during the enumeration process is one of the major problems in all recent approaches. Duplicate candidates are treated by representing the subgraphs with a unique code called canonical form (CF). The canonical adjacency matrix (CAM) is a CF used in FCSM [9, 10, 11, 12]. Candidate enumeration strategies are commonly defined by means of these representations, trying to avoid non-canonical forms by performing CF tests which have very high computational complexity [2].

In this paper, we propose a non-usefulness property that allows to know the results of some CF tests in constant time; thus, it reduces the size of the candidate space. This property does not remove all the duplicate candidates; therefore, CF tests are required for non-filtered candidates. For this reason, we also propose properties that allow to reduce the number of such expensive tests by reusing previous test results for predicting new results without performing a test. Additionally, this paper introduces a new algorithm called grCAM (graph mining by reducing CAM tests) based on these properties. This algorithm uses the non-usefulness property to reduce the number of candidate graphs and applies the reuse properties for reducing the number of CF tests.

The basic outline of this paper is as follows. Section 2 provides some basic concepts; it also contains the related work. The adjacency matrix properties are introduced, discussed, and tested in section 3; this section also introduces the grCAM algorithm. The experimental results over real world datasets and a

comparison against other FCSM algorithms are presented in section 4. Finally, conclusions of the research and some ideas about future directions are exposed in section 5.

2 Preliminaries

In this paper, we focus on labeled simple undirected graphs. The formal definition of this type of graphs is as follows.

A *labeled graph* is a 4-tuple, $G = \langle V, E, L, l \rangle$, where V is a set whose elements are called *vertices*, $E \subset \{e \,|e \subset V, |e| = 2\}$ is a set whose elements are called *edges*, each edge is a set with exactly two vertices, L is a set of *labels* and $l : V \cup E \to L$ is a *labeling function* for assigning labels to vertices and edges.

Let G_1 and G_2 be two graphs having the same set of labels L and the same labeling function l. We say that G_1 is a *subgraph* of G_2 if $V(G_1) \subseteq V(G_2)$ and $E(G_1) \subseteq E(G_2)$. In this case, we use the notation $G_1 \subseteq G_2$.

For mining over collections of labeled graphs, the frequency of a candidate is calculated using subgraph isomorphism tests. We say that f is an *isomorphism* between G_1 and G_2, if $f : V(G_1) \to V(G_2)$ is a bijective function where $l_{G_1}(v) = l_{G_2}(f(v))$ for each vertex $v \in V(G_1)$, $\{f(u), f(v)\} \in E(G_2)$ and $l_{G_1}(\{u, v\}) = l_{G_2}(\{f(u), f(v)\})$ for all edge $\{u, v\} \in E(G_1)$. A *subgraph isomorphism* from G_1 to G_2 is an isomorphism from G_1 to a subgraph of G_2.

A path in G is a sequence of vertices $v_1, v_2, \ldots, v_k$ with $\{v_i, v_{i+1}\} \in E(G)$ for each $i = 1, \ldots, k-1$. In this case we say that v_1 and v_k are connected. The graph G is *connected* if for all $v_i, v_j \in V(G)$, $i \neq j$, v_i and v_j are connected by at least one path.

Let $D = \{G_1, G_2, \ldots, G_{|D|}\}$ be a collection of labeled graphs and let δ be a predefined threshold of frequency. The *support* of a graph g in D is defined as the number of graphs $G_i \in D$ such that there is a subgraph isomorphism from g to G_i. We use the notation $\sigma(g, D)$ to refer to the support of g in the collection D. A graph g is *frequent* in the collection D if $\sigma(g, D) \geq \delta$. *Frequent connected subgraph mining* (FCSM) is the process of finding all frequent connected subgraphs in a collection of graphs.

2.1 Canonical Adjacency Matrix

A graph can be represented by its canonical adjacency matrix. This kind of canonical representation has been used in several works for graph mining [9, 10, 11, 12].

Let G be a labeled graph with $|V(G)| = n$ and let $v_1, v_2, \ldots, v_n$ be a permutation of the vertices in $V(G)$, the *adjacency matrix* of G regarding this permutation is a lower triangular matrix $X = (x_{i,j})_{i,j=1}^{n}$ where for each $1 \leq i \leq j \leq n$:

$$x_{i,j} = \begin{cases} l(v_i) & \text{if} \quad i = j & \text{(vertex entry)} \\ l(e) & \text{if} \quad e = \{v_i, v_j\} \in E & \text{(edge entry)} \\ 0 & \text{if} \quad \{v_i, v_j\} \notin E & \text{(non-edge entry)} \end{cases}$$

The adjacency matrix is not unique for G. Since each diagonal entry represents a vertex in the graph, each permutation of the set of vertices corresponds to a different adjacency matrix. There are $O(n!)$ different adjacency matrices for G. An example of two different adjacency matrices for a same graph is shown in Fig. 1.

In order to achieve a unique representation for G, we define a *code* of adjacency matrices which provides a total order among all adjacency matrices. The code of an adjacency matrix X is built concatenating lower triangular rows of X, see (1).

$$code(X) = x_{1,1}x_{2,1}x_{2,2}x_{3,1}x_{3,2}x_{3,3}\ldots x_{n,n} \tag{1}$$

For example, the codes of the matrices shown in Fig. 1 (A) and (B) are `b1a12a001a` and `a1a02a011b` respectively. Codes are strings of labels in $L(G) \cup \{0\}$. An order among the labels in $L(G)$ is assumed. This order is defined according to the real application. The edge absence label 0 is considered the smallest label in the string label order. Therefore, the standard lexicographic order $\prec$ could be used to define a total order among the codes. For example, we have `a1a02a011b` $\prec$ `b1a12a001a` with the codes of the matrices in Fig. 1.

The *canonical form* (CF) of G is usually considered as the maximal code among all its possible codes. Thus, the *canonical adjacency matrix* (CAM) of G is the adjacency matrix of G whose code has the CF. The canonical adjacency matrix is unique for a graph G; that is, two isomorphic graphs have the same CAM.

An adjacency matrix X is an *inner matrix* if it has at least two edge entries in the last row. Otherwise, X is an *outer matrix*. For example, the matrices shown in Fig. 1 (A) and (B) are outer and inner respectively.

The *maximal proper submatrix* of an inner matrix X is defined as the matrix obtained from X by removing the last edge entry (putting 0 in the last edge entry). If X is a outer matrix then its *maximal proper submatrix* is built from X by removing the last row and the last column. The *maximal proper submatrix* of an 1×1 matrix is the empty matrix (0×0 degenerated matrix).

Thus, for each non-empty matrix X there is only one matrix Y such that Y is a maximal proper submatrix of X. Besides, the maximal proper submatrix of a CAM is also a CAM of a connected graph [9]. Therefore, we can build a rooted tree whose nodes are the CAMs of all connected subgraphs of all graphs in a collection D. This tree is called the *CAM tree* of D. The root of the CAM tree is the empty matrix and the parent/child relationship is defined by the maximal proper submatrix as follow. Y is the parent of X if Y is a maximal proper submatrix of X; in this case, X is a child of Y. For example, the CAM tree of the graph collection D showed in Fig. 3 is the tree showed in Fig. 4; it is built using the 60% of the collection size as support threshold.

Another important concept is the *suboptimal CAM*. A matrix X is a suboptimal CAM if its maximal proper submatrix is a CAM. By this definition, if X is a CAM then it is also a suboptimal CAM. When a suboptimal CAM X is not a CAM, we say that X is a *proper suboptimal CAM*. Thus, the suboptimal CAMs of a graph collection D could be organized in a tree in a similar way to

the CAM tree. This tree is called *suboptimal CAM tree*.

2.2 Mining with Canonical Adjacency Matrices

FFSM algorithm [9] is the most efficient algorithm for FCSM among those using CAM for representing graphs [15]. FFSM finds all frequent connected subgraphs (FCS) in a collection D by exploring the CAM tree of D. During the exploration, the CAM tree is built dynamically using two matrix operations called *join* and *extension*. These operations are defined as follows.

The join operation produces one or two graph candidates starting from two graphs whose corresponding adjacency matrices $A = (a_{i,j})_{i,j=1}^{n}$ and $B = (b_{i,j})_{i,j=1}^{m}$ share the same maximal proper submatrix C. This operation is denoted by $join(A, B)$ and it is defined as follows.

Without loss of generality, assume that $n \leq m \leq n+1$. Let $a_{n,h}$ and $b_{m,k}$ be the last edge entries of A and B, respectively. $join(A, B)$ is a set of matrices such that $0 \leq |join(A, B)| \leq 2$ depending on the characteristics of the graphs represented by A and B as states the following conditions.

Condition 1: Suppose that $n = m$, $a_{n,h} = b_{n,k}$ and $h \neq k$. Let $D = (d_{i,j})_{i,j=1}^{n}$ be the matrix obtained directly from A putting $d_{n,k} = b_{n,k}$; that is

$$d_{i,j} = \begin{cases} b_{n,k} & i = n, j = k \\ a_{i,j} & \text{otherwise} \end{cases}.$$

In this case, $D \in join(A, B)$.

Condition 2: Suppose that B is an outer matrix. Let $E = (e_{i,j})_{i,j=1}^{n+1}$ be the matrix where

$$e_{i,j} = \begin{cases} a_{i,j} & 1 \leq i, j \leq n \\ b_{m,k} & i = n+1, j = k \\ b_{m,m} & i = j = n+1 \\ 0 & \text{otherwise} \end{cases}.$$

In this case, $E \in join(A, B)$.

If conditions 1 and 2 are false then $join(A, B) = \emptyset$. If only one of these conditions is true then $join(A, B)$ has only one element. If both conditions are true then $join(A, B)$ has two elements. In the CAM tree of Fig. 5, we can see examples of these three cases.

The extension operation obtains graph candidates from a graph by introducing one edge. New edges are introduced connecting the last node in a CAM with an additional vertex. The set $extension(A)$ of the extensions of a $n \times n$ matrix A is a set of $(n+1) \times (n+1)$ matrices obtained from A by adding a new edge.

The join and extension operations allow enumerating all the nodes in a suboptimal CAM tree [9].

Let X be a CAM, let $C(X)$ denote the children of X in the suboptimal CAM tree of a graph collection D, and suppose that $Y \in C(X)$ is an $n \times n$ matrix.

If $n = 1$, then $C(Y) = extension(Y)$ since 1×1 matrices are not joinable. If $n > 1$, then

$$C(Y) = \left(\bigcup_{B \in C(X)} join(Y, B) \right) \cup extension(Y), \tag{2}$$

the elements of $C(Y)$ are obtained by the join and extension operations. The general description of the FFSM algorithm is as follows:

Procedure `FFSM`(D, δ, S)

Input: D - graph collection, δ - support threshold
Output: S - mining results

1 $S^{(1)} \leftarrow$ all CAMs of frequent vertices in D (1×1 matrices);
2 $S^{(2)} \leftarrow$ all CAMs of frequent edges in D (2×2 matrices);
3 $S \leftarrow S^{(1)} \cup S^{(2)}$;
4 `FFSM-Explore`(D, δ, $S^{(2)}$, S);

Procedure `FFSM-Explore`(D, δ, C, S)

Input: D - graph collection, δ - support threshold, C - a suboptimal CAM list
Output: S - mining results

1 **forall** $X \in C$ **do**
2 **if** `isCAMTest`(X) **then**
3 $S \leftarrow S \cup \{X\}$;
4 $C(X) \leftarrow$ the children of X which are calculated by means of (2) using C and D;
5 Remove from $C(X)$ the non-frequent CAMs;
6 `FFSM-Explore`(D, δ, $C(X)$, S);
7 **end**
8 **end**

First, the procedure `FFSM` calculates frequent vertices and edges. Frequent edges are the starting point for traversing the suboptimal CAM tree of a graph collection D using the procedure `FFSM-Explore`.

The procedure `FFSM-Explore` uses the join and extension operations (see line 4) for candidate enumeration. The proper suboptimal CAMs are duplicated candidates but they are needed for the join operations. Proper suboptimal CAMs are ignored after the corresponding join operations by performing a CF tests (see line 2). During the traversal, FFSM maintains data structures (embedding structures [9]) for indexing candidate occurrences in the collection D. These structures are used to optimize the extension operations and they are also used for calculating the frequency of the candidates. Thus, non-frequent CAMs are pruned from the suboptimal CAM tree (see line 5) since they cannot generate useful candidates.

3 Frequent connected subgraph mining

In this section, new CAM properties and a novel approach for FCSM using these properties to reduce the number of CAM tests are introduced.

3.1 Novel Properties of Adjacency Matrices

Let X be an $n \times n$ CAM. The elements of $C(X)$ obtained by the join condition 1 are $n \times n$ matrices and the other elements (obtained by the join condition 2 or extension) are $(n+1) \times (n+1)$ matrices. Thus, the set $C(X)$ can be partitioned into two sets $C^{(n)}(X)$ and $C^{(n+1)}(X)$ containing the $n \times n$ and $(n+1) \times (n+1)$ matrices respectively; that is, $C(X) = C^{(n)}(X) \cup C^{(n+1)}(X)$.

The set $C^{(n)}(X)$ can be partitioned into several sets

$$C^{(n)}(X) = \bigcup_{j=1}^{n-1} C_j^{(n)}(X), \tag{3}$$

where $C_j^{(n)}(X)$ contains the children of X whose last edge entry lies in the j-th column.

On the other hand, $C^{(n+1)}(X)$ can also be partitioned as follow

$$C^{(n+1)}(X) = \bigcup_{v \in L} \bigcup_{j=1}^{n} C_{v,j}^{(n+1)}(X), \tag{4}$$

where L is the set of labels in the last vertices of the matrices in $C^{(n+1)}(X)$, and $C_{v,j}^{(n+1)}(X)$ contains the children of X with last vertex label equal to v and last edge entry in the j-th column.

For example, Fig. 6 shows some elements of $C_{a,1}^{(n+1)}(X)$, Fig. 7 shows elements of $C_{b,2}^{(n+1)}(X)$, and Fig. 8 shows some elements in the sets $C_{a,2}^{(n+1)}(X)$, $C_{b,2}^{(n+1)}(X)$, $C_{c,2}^{(n+1)}(X)$, and $C_{d,2}^{(n+1)}(X)$.

The first properties presented in this section are called reuse properties since they allow to reuse previous CF test results for predicting new results without performing a test. Theorems 1 and 2 are called the *reuse properties for the last edge entry*.

Theorem 1 works when the result of a CF test in a matrix Y is true. It allows to reuse this fact for predicting the result of CF tests in other matrices whose difference with Y is only one edge entry.

Theorem 1 *Let $Y = (y_{i,j})_{i,j=1}^{n}$ be a CAM where its last edge entry is $y_{n,b}$ ($y_{n,b} > 0$). Suppose that $Z = (z_{i,j})_{i,j=1}^{n}$ is an adjacency matrix where $z_{i,j} = y_{i,j}$ for all $i \neq n$ and $j \neq b$. If $0 < z_{n,b} < y_{n,b}$ then Z also is a CAM.*

Proof. Suppose that Z is not a CAM; then, there is a permutation P among the vertices of Z such that we obtain a new matrix[1] $P(Z)$ where $code(Z) \prec$

[1] In general, we will use the notation $P(A)$ for referring to the matrix obtained from the matrix A making the permutation P.

$code(P(Z))$. Assume that

$$\begin{array}{rcl} code(P(Z)) & = & z'_1 z'_2 \dots z'_t \\ code(Z) & = & z_1 z_2 \dots z_t \end{array},$$

where $t = n(n+1)/2$. Let h be the minimum integer $1 \leq h \leq t$ such that $z'_h > z_h$, let k and k' be the positions of $z_{n,b}$ in the codes $code(Z)$ and $code(P(Z))$ respectively. Thus, we have two cases:

- If $h < k'$ then the entry $z_{n,b}$ in Z does not determine that $code(P(Z)) > code(Z)$. Therefore, $code(P(Y)) > code(Y)$ since Y and Z are the same matrix except for the entry in the (n, b) position. It is a contradiction with the hypothesis that Y is a CAM.
- If $h > k'$ then $z_{k'} = z'_{k'} = z_{n,b} < y_{n,b}$. In this case, if we apply P over Y we obtain that $code(P(Y)) > code(Y)$ since $y_{k'} = z_{k'} < y_{n,b} = y'_{k'}$ (where $y_1 y_2 \dots y_t$ and $y'_1 y'_2 \dots y'_t$ are the codes of Y and $P(Y)$ respectively). It is a contradiction with the hypothesis that Y is a CAM.
- If $h = k'$ then $z_h < z_{n,b} < y_{n,b}$. In this case, if we apply P over Y we obtain that $code(P(Y)) > code(Y)$ since $y_h = z_h < y_{n,b} = y'_h$. It is a contradiction with the hypothesis that Y is a CAM.

Therefore, Z is a CAM. □

Similar results are showed in theorem 2 which works when the result of a CF test in a matrix Y is false.

Theorem 2 *Let $Y = (y_{i,j})_{i,j=1}^n$ be a proper suboptimal CAM and its last edge entry is $y_{n,b}$ ($y_{n,b} > 0$). Suppose that $Z = (z_{i,j})_{i,j=1}^n$ is an adjacency matrix where $z_{i,j} = y_{i,j}$ for all $i \neq n$ and $j \neq b$. If $z_{n,b} > y_{n,b}$ then Z is not a CAM.*

Proof. Suppose that Z is a CAM; then, by theorem 1 the matrix Y is a CAM and it is a contradiction. Therefore, Z is not a CAM and the proof concludes. □

The theorems 1 and 2 are applicable to matrices Y and Z which lie in the same partition subset $C_j^{(n)}(X)$ or $C_{v,j}^{(n+1)}(X)$ where X is the maximal proper submatrix of Y and Z (see Fig. 6 and Fig. 7).

The theorems 3 and 4 are called the *reuse properties for the last vertex entry.* They allow to reuse previous results when the matrices differ in only one vextex entry.

Theorem 3 *Let $Y = (y_{i,j})_{i,j=1}^n$ be a CAM and let $Z = (z_{i,j})_{i,j=1}^n$ be a adjacency matrix where $z_{i,j} = y_{i,j}$ for all $i \neq n$ and $j \neq n$. If $0 < z_{n,n} < y_{n,n}$ then Z also is a CAM.*

Proof. Suppose that Z is not a CAM; then, there is a permutation P among the vertices of Z such that $code(Z) \prec code(P(Z))$. Assume that

$$\begin{array}{rcl} code(P(Z)) & = & z'_1 z'_2 \dots z'_t \\ code(Z) & = & z_1 z_2 \dots z_t \end{array},$$

where $t = n(n+1)/2$. Let h be the minimum integer $1 \leq h \leq t$ such that $z'_h > z_h$, let k' be the position of $z_{n,n}$ in the code $code(P(Z))$ after the permutations. Thus, we have two cases:

- If $h < k'$ then the entry $z_{n,b}$ in Z does not determine that $code(P(Z)) > code(Z)$. Therefore, $code(P(Y)) > code(Y)$ since Y and Z are the same matrix except for the entry in the (n, n) position. It is a contradiction with the hypothesis that Y is a CAM.
- If $h > k'$ then $z_{k'} = z'_{k'} = z_{n,n} < y_{n,n}$. In this case, if we apply P over Y we obtain that $code(P(Y)) > code(Y)$ since $y_{k'} = z_{k'} < y_{n,n} = y'_{k'}$ (where $y_1 y_2 \dots y_t$ and $y'_1 y'_2 \dots y'_t$ are the codes of Y and $P(Y)$ respectively). It is a contradiction with the hypothesis that Y is a CAM.
- If $h = k'$ then $z_h < z_{n,n} < y_{n,n}$. In this case, if we apply P over Y we obtain that $code(P(Y)) > code(Y)$ since $y_h = z_h < y_{n,n} = y'_h$. It is a contradiction with the hypothesis that Y is a CAM.

Therefore, Z is a CAM. □

Theorem 4 *Let $Y = (y_{i,j})_{i,j=1}^{n}$ be a proper suboptimal CAM. Suppose that $Z = (z_{i,j})_{i,j=1}^{n}$ is an adjacency matrix where $z_{i,j} = y_{i,j}$ for all $i \neq n$ and $j \neq n$. If $z_{n,n} > y_{n,n}$ then Z is not a CAM.*

Proof. Suppose that Z is a CAM; then, by theorem 3 the matrix Y is a CAM and it is a contradiction. Therefore, Z is not a CAM and the proof concludes. □

The theorems 3 and 4 are applicable to matrices Y and Z which lie in different subsets $C^{(n+1)}_{(y_{n,n}),j}(X)$ and $C^{(n+1)}_{(z_{n,n}),j}(X)$ with the same last edge entry position j, where X is the maximal proper submatrix of Y and Z (see Fig. 8).

Another interesting property of the adjacency matrices called *non-usefulness property* is showed in the theorem 5. It is an example of a sufficient condition for guaranteeing that a suboptimal CAM is proper suboptimal CAM.

Theorem 5 *Let $X = (x_{i,j})_{i,j=1}^{n}$ be a CAM where $n > 1$ and let $Y = (y_{i,j})_{i,j=1}^{n+1}$ be an adjacency matrix such that:*

$$y_{i,j} = \begin{cases} x_{i,j} & 1 \leq i, j \leq n \\ e & i = n+1, j = a \\ v & i = j = n+1 \\ 0 & otherwise \end{cases}$$

where $e > 0$ and $v > 0$ are labels. If $x_{1,1}x_{2,1} \prec ve$ then Y is not a CAM and any join or extension involving Y does not produce CAM matrices.

Proof. From a permutation of the vertices of Y starting with the current vertices $n+1$ and a in this order, we obtain an adjacency matrix Y' such that $code(Y') > code(Y)$. Thus, Y is not a CAM. Finally, the same arguments ensure that any matrix obtained from Y will be not a CAM. □

The non-usefulness property allow to reduce the size of suboptimal CAM trees during FCSM.

3.2 The grCAM algorithm

The FFSM algorithm detects duplicate candidates by performing a CF test for each frequent and sub-optimal matrix candidate (see line 2 of the `FFSM-Explore` procedure in section 2.2). In this section, the algorithm grCAM (graph mining by reducing CAM tests) is introduced, including a novel strategy to reduce the number of such tests using the properties proposed in section 3.1.

In the `FFSM-Explore` procedure, the list C contains candidates to be FCS and a CF test is performed for each one of them. The first time, C contains 2×2 adjacency matrices representing frequent edges. After, C contains N adjacency matrices with the same $n \times n$ maximal proper submatrix X (that is $C = C(X)$); therefore, we could modify the FFSM-Explore procedure using the proposed properties to reduce the number of CF tests.

Taking into account that the non-usefulness property can be verified in constant time, it could be used inside the pattern growth process (see line 4 of the `FFSM-Explore`). Let $\bar{C}(X) \subset C(X)$ be the set containing the useful candidates after applying the non-usefulness property and suppose that $\bar{N} = |\bar{C}(X)|$ (notice that $\bar{N} \leq N$). This process is called *pre-filtering*.

The reuse properties could be used to reduce the number of CF tests starting from the useful candidate set $\bar{C}(X)$ in a more efficient way than the one used in FFSM. This procedure called *post-calculation* is as follows.

In post-calculation, the result of the corresponding CF test for each $Y \in \bar{C}(X)$ is calculated and stored. The reuse properties use those results for avoiding unnecessary tests.

The set $\bar{C}(X)$ can be partitioned according to (3) and (4) as $\bar{C}(X) = \bar{C}^{(n)}(X) \cup \bar{C}^{(n+1)}(X)$ where:

$$\bar{C}^{(n)}(X) = \bigcup_{j=1}^{n-1} \bar{C}_j^{(n)}(X),$$

and

$$\bar{C}^{(n+1)}(X) = \bigcup_{v \in L} \bigcup_{j=1}^{n} \bar{C}_{v,j}^{(n+1)}(X).$$

Let M be one of the sets $\bar{C}_j^{(n)}(X)$ or $\bar{C}_{v,j}^{(n+1)}(X)$ and suppose that M is sorted according to the lexicographic order $\prec$ (see section 2.1). From the reuse properties for the last edge entry (theorems 1 and 2), we can conclude the following two statements:

1. if there are CAMs in M then they are at the beginning of M;
2. if there are non CAMs in M then they are at the end of M.

Thus, the boundary between CAMs and non CAMs can be detected using a binary search. Therefore, the number of CF tests needed to detect this boundary is less than $log_2(|M|) + 1$. This process for reducing the number of CF tests is

called *internal reusing process* since it allows to reuse previous CF test results inside each $\bar{C}_j^{(n)}(X)$ or $\bar{C}_{v,j}^{(n+1)}(X)$.

Let j be an integer such that $1 \leq j \leq n$ and consider the sets

$$\bar{C}_{v_1,j}^{(n+1)}(X), \bar{C}_{v_2,j}^{(n+1)}(X), \ldots, \bar{C}_{v_h,j}^{(n+1)}(X)$$

which are used for partitioning $\bar{C}^{(n+1)}(X)$ with the same last edge entry j and different last vertex entry labels $v_1 < v_2 < \ldots < v_h$.

First of all, the internal reusing process is performed for $\bar{C}_{v_1,j}^{(n+1)}(X)$ and $\bar{C}_{v_h,j}^{(n+1)}(X)$ (the first and the last sets). Let α_1 be the highest last edge entry label in $\bar{C}_{v_1,j}^{(n+1)}(X)$ such that its corresponding matrix is a CAM and let β_h be the lowest last edge entry label in $\bar{C}_{v_h,j}^{(n+1)}(X)$ such that its corresponding matrix is not a CAM. The theorem 3 ensures that any matrix in the considered sets with last edge entry label less than α_1 is a CAM. The theorem 4 ensures that any matrix in the considered sets with last edge entry label greater than β_h is not a CAM. Thus, the reuse properties for the vertex entry can be used to reduce the number of CF tests.

Next, the internal reusing process is performed for $\bar{C}_{v_2,j}^{(n+1)}(X)$ and $\bar{C}_{v_{h-1},j}^{(n+1)}(X)$ (the second and the penultimate sets) considering only the matrices whose last edge entry label is less than α_1 and greater than β_h. Thus, new values α_2 and β_{h-1} are calculated in a similar way. These steps are repeated until all the sets are analyzed. This process is called *external reusing process* since it allows to reuse previous CT test results among different sets $\bar{C}_{v,j}^{(n+1)}(X)$. The external reusing process must be performed for each $1 \leq j \leq n$.

The internal and external reusing processes integrate the post-calculation procedure.

The following pseudo-code contains the general description of the grCAM algorithm and it shows the use of pre-filtering and post-calculation procedures to reduce the number of CF tests.

Procedure `grCAM(`D, δ, S`)`

Input: D - graph collection, δ - support threshold
Output: S - mining results

1 $S^{(1)} \leftarrow$ all CAMs of frequent vertices in D (1×1 matrices);
2 $S^{(2)} \leftarrow$ all CAMs of frequent edges in D (2×2 matrices);
3 $S \leftarrow S^{(1)} \cup S^{(2)}$;
4 `grCAM-Explore(`D, δ, $S^{(2)}$, S`)`;

Procedure `grCAM-Explore`(D, δ, C, S)

Input: D - graph collection, δ - support threshold, C - a suboptimal CAM list

Output: S - mining results

```
1 forall X ∈ C do
2     if X.isCAM then
3         S ← S ∪ {X};
4         C̄(X) ← the children of X filtered by non-usefulness property
          (pre-filtering);
5         Remove from C(X) the non-frequent CAMs;
6         Apply the post-calculation procedure to calculate Z.isCAM for
          each Z ∈ C(X);
7         grCAM-Explore(D, δ, C(X), S);
8     end
9 end
```

The procedure `grCAM` calculates frequent vertices and edges. Frequent edges are the starting point for traversing the suboptimal CAM tree of a graph collection D using the procedure `grCAM-Explore`.

The procedure `grCAM-Explore` uses the CAM matrix operations for candidate enumeration and it includes the above described pre-filtering and post-calculation procedures for reducing the number of CAM tests. Proper suboptimal CAMs are not considered in the mining results since they represent duplicate candidates. In grCAM, some non-useful candidates are filtered during the pattern growth process (see line 4) using the above mentioned pre-filtering. During the traversal, embedding structures are maintained for each candidate and they are used for indexing the candidate occurrences in the collection D as in FFSM [9]. Non-frequent CAMs are pruned from the suboptimal CAM tree (see line 5) since they cannot generate useful candidates. The post-calculation procedure is used to determine if each candidate matrix is a CAM or not, which reduces the number of CF tests.

It is important to highlight that the pre-filtering and post-calculation procedures introduced in grCAM can be also introduced in any other FCSM algorithm based on adjacency matrices.

3.3 General considerations of grCAM

Let N and $\overline{N}$ be the number of candidates of the list $C(X)$ during an execution of the procedures `FFSM-Explore` and `grCAM-Explore` respectively. In the previous section, we have shown that $\overline{N} \leq N$ since the pre-filtering stage in grCAM could reduce the number of candidates.

The internal reusing process without considering the external reusing process allows to measure the expected number of CF tests needed to process the list C in the worst case. That is, when the external reusing process does not reduce

the number of CF tests. Thus, the number of CF tests is less than:

$$\hat{K} = \sum_{j=1}^{n-1}(\log_2 |\bar{C}_j^{(n)}(X)| + 1) + \sum_{v \in L}\sum_{j=1}^{n}(\log_2 |\bar{C}_{v,j}^{(n+1)}(X)| + 1). \quad (5)$$

Since $log_2(x) : [1, +\infty) \rightarrow [0, +\infty)$ is a convex function, (5) can be transformed in:

$$\hat{K} \leq \hat{P} \log_2 \left[\left(\sum_{j=1}^{n-1} |\bar{C}_j^{(n)}(X)| + \sum_{v \in L}\sum_{j=1}^{n} |\bar{C}_{v,j}^{(n+1)}(X)| \right) / \hat{P} \right] + \hat{P}. \quad (6)$$

where $\hat{P} = n - 1 + n|L|$. Since the sum of the cardinalities of these sets is the cardinality of $C(X)$, then (6) can be transformed in:

$$\hat{K} \leq \hat{P} \log_2(\bar{N}/\hat{P}) + \hat{P}. \quad (7)$$

In the best case, the external reusing process allows to remove all CF tests after the first step. Since in the first step only two sets are checked, we can conclude that in the best case the number of CF tests is less than:

$$\check{K} = \sum_{j=1}^{n-1}(\log_2 |\bar{C}_j^{(n)}(X)| + 1) + \sum_{j=1}^{n}(\log_2 |\bar{C}_{v_1,j}^{(n+1)}(X)| + \log_2 |\bar{C}_{v_l,j}^{(n+1)}(X)| + 2).$$

In a similar way, we obtain:

$$\check{K} \leq \check{P} \log_2(\bar{N}/\check{P}) + \check{P}. \quad (8)$$

where $\check{P} = 3n - 1$.

The inequalities (7) and (8) state an upper bound for the number of CF tests required to process the list $C(X)$ in the worst and best cases respectively. In the worst case, this bound depends on three parameters the number of extensions $\bar{N}$ (this number depends on the graph collection features such as density of edges, number of labels, etc.), n the size of the matrix X, and $|L|$ (this number depends on the vertex label distribution). In the best case, this bound depends only on N and n.

4 Experimental results

In order to evaluate the usefulness of the proposed properties to reduce the number of canonical form tests, we compared grCAM against FFSM regarding the number of duplicates and the number of canonical form tests performed during the mining.

Additionally, we include a comparison of grCAM against FFSM, gRed, gSpan, and Gaston regarding their execution times. The gRed algorithm is one of the most recent algorithm for FCSM and the other algorithms are the most commonly referenced and the most successful in previous comparative studies [15, 17]. They were implemented in the common Java framework [17] which

is distributed under GNU license. The implementation of gRed and grCAM was developed using this framework.

All the experiments were done using an Intel Core 2 Duo PC at 2.2 GHz with 4 GB of RAM using a 64-bits Debian GNU/Linux distribution. The IBM Java Virtual Machine (JVM) Version 6 was used to run the algorithms. The maximum heap memory space of JVM was assigned in 3.2GB. Thus, we remove the influence of operative system swap operations during the executions.

4.1 Collections of graphs used for the experimentation

The biochemical data collections, specifically the molecular datasets, constitute one of the main application field for graph mining. Therefore, this kind of collections has been commonly used to evaluate the performance of the algorithms for FCSM. The collections used in our experiments are shown in Table 1.

The PTE collection is the smallest dataset (according to the number of graphs) considered in this work; it contains only 337 graphs representing molecules used in the predictive toxicologic evaluation challenge [16]. In spite of its small size, PTE has a big amount of frequent connected subgraphs; for example, it has 136981 frequent connected subgraphs using the 2% of the collection size as support threshold.

In this work, we used two medium size collections CAN2DA99[2] and HIV[3]. CAN2DA99 collection contains the graph representation of 32557 molecules discovered in carcinogenic tumors and HIV collection contains the graph representation of 42689 molecular structures of the human immunodeficiency virus. Additionally, we also use the collections HIV-CM and HIV-CA; which are small parts of the HIV dataset containing only the confirmed moderately active molecules (HIV-CM) and the confirmed active molecules (HIV-CA). The biggest collection used in our experiments was NCI[4], which contains the graph representation of molecules from several sources.

4.2 Experiments

In our experiments we used low support thresholds to evaluate the performance of the algorithms. These thresholds are very important in data mining applications [4, 8]. For example, there are some applications like classification and clustering where frequent complex graph structures are needed [8], and these complex structures only can be found with low support thresholds. Additionally, high thresholds are commonly fulfilled by connected subgraphs with small size regarding the number of vertices, edges, or cycles. For example, Table 2 shows the number and size of frequent connected subgraphs found in the HIV-CA collection using high and low support thresholds. High values of these thresholds only produce a few small patterns unlike low values which produce more patterns with large size. Moreover, almost all recent algorithms achieve short

[2]http://dtp.nci.nih.gov/docs/cancer/cancer_data.html
[3]http://dtp.nci.nih.gov/docs/aids/aids_data.html
[4]http://cactus.nci.nih.gov/ncidb2/download.html

runtimes for high support thresholds, therefore it is more important to propose fast algorithms for low support thresholds. Finally, it is important to highlight that, like in previous comparative studies [17, 15], in this work the support thresholds are defined as a percentage of the collection size.

The first experiment of this paper was conceived for evaluating the impact of the non-usefulness property in the pre-filtering stage of grCAM. The algorithms grCAM and FFSM use adjacency matrices to represent graph candidates during the mining process, unlike gRed, gSpan and Gaston which use other different approaches. Therefore, in this experiment grCAM and FFSM are compared regarding the number of duplicate candidates that were considered in their mining strategies. As we can see in Table 3, the number of duplicates in all cases was reduced by grCAM. For example, in CAN2DA99 for support thresholds between 5% and 7%, grCAM reduces almost 25% of duplicates regarding FFSM.

The second experiment evaluated the impact of the reuse properties in the post-calculation stage of grCAM. The algorithms grCAM and FFSM were compared regarding the number of exhaustive canonical form tests. As it can be seen in Table 4, the number of such expensive tests were reduced by grCAM in all the cases. For example, in NCI for support thresholds between 5% and 7%, grCAM reduces almost 30% of CF tests regarding FFSM.

In this two experiments, even though our algorithm has the best performance, the improvement achieved by grCAM in PTE (for low support thresholds) is small (the attained reduction is less than the 10%) due to the label distribution in this dataset. In PTE, frequent connected subgraphs (for the lowest support threshold 2%) contain only three edge labels and only one of this label prevails. Therefore, it is difficult to use candidates in the post-calculation procedure to reuse previous CF test results. Moreover, it is also difficult to filter non-useful candidates during pre-filtering using the non-usefulness property. Thus, the impact of the novel properties for this collection is lesser. Nevertheless in general, the new properties help to reduce runtimes of grCAM.

Additionally, we included a performance comparison involving grCAM, FFSM, gRed, gSpan, and Gaston. This comparison includes the evaluation of the runtimes. The runtime for the algorithms was recorded varying the support threshold in the six datasets. As we can see in Fig. 9, grCAM beats FFSM in all the tests.

Gaston was unable to complete the execution for some low support thresholds (less than 3% in NCI) due to its high memory requirements. However, in the smallest collection (PTE), the best runtimes were achieved by Gaston and gRed. The worst runtimes were achieved by FFSM and gSpan while the best runtimes on the large collections were obtained by grCAM for the evaluated support thresholds.

5 Conclusions

In this paper, a non-usefulness property and four reuse properties of adjacency matrices, which are useful for graph mining were introduced. The non-use-

fulness property allows the reduction of the number of candidates. The reuse properties enable to reduce the number of CF tests by reusing previous test results. Besides, the reuse properties allow to define boundaries between CAMs and non CAMs in the candidate space (sub-optimal CAM tree).

Additionally, a new algorithm (grCAM) for FCSM using the proposed properties, was introduced. Theoretical analysis and experimental results shown the good performance of our proposal.

We compared grCAM against FFSM and other state of the art algorithms. The usefulness of the novel adjacency matrix properties for graph mining was corroborated in the experimentation, showing that these properties allow to reduce the number of duplicate candidates (this reduction could be more than 25%), as well as the number of canonical form tests (this reduction could be more than 30%). Moreover, grCAM achieved better runtimes than the other tested algorithms for low supports, when graph collections were large.

In this research, we have shown that canonical forms have not been sufficiently studied and new properties can be found to improve the mining process. Our proposal showed that it is possible to reduced the time spent in duplicate candidate detection during the mining. Moreover, the novel adjacency matrix properties can be used in any proposal for FCSM that uses candidates represented by adjacency matrices.

As future work, we are going to develop novel approaches using the grCAM ideas for finding other frequent graph based patterns like closed and maximal.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, 1994.

[2] C. Borgelt. Canonical Forms for Frequent Graph Mining. In *Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V.*, pages 8–10, Freie Universitat Berlin, 2006.

[3] C. Borgelt and M. R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In *Proceedings of the 2002 International Conference on Data Mining (ICDM'02)*, pages 211–218, Maebashi, Japan, 2002.

[4] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. Yu, and O. Verscheure. Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 230–238, Nevada, USA, 2008.

[5] A. Gago-Alonso, J. E. Medina-Pagola, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. Mining Frequent Subgrahps Reducing the Number of Candidates. In *Proceedings of the European Conference on Machine*

Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'08), Antwerp, Belgium, 2008.

[6] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent Pattern Mining: Current Status and Future Directions. *Data Mining and Knowledge Discovery, 10th Anniversary Issue*, 15(1):55–86, 2007.

[7] L.B. Holder, D.J. Cook, and S. Djoko. Substructure discovery in the subdue system. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases (KDD'94)*, page 169180, Seattle, WA, 1994.

[8] M.S. Hossain and R.A. Angryk. GDClust: A Graph-based Document Clustering Technique. In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 417–422, Omaha, NE, 2007.

[9] J. Huan, W. Wang, and J. Prins. Efficient Mining of Frequent Subgraph in the Presence of Isomorphism. In *Proceedings of the 2003 International Conference on Data Mining (ICDM'03)*, pages 549–552, Melbourne, FL, 2003.

[10] A. Inokuchi, Washio, and H. T., Motoda. An Apriori based Algorithm for Mining Frequent Substructures from Graph Data. In *Proceedings of the 2000 European Symposium on the Principle of Data Mining and Knowledge Discovery (PKDD'00)*, pages 13–23, Lyon, France, 2000.

[11] A. Inokuchi, T. Washio, Nishimura K., and H. Motoda. A Fast Algorithm for Mining Frequent Connected Subgraphs. Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.

[12] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. In *Proceedings of the 2001 International Conference on Data Mining (ICDM'01)*, pages 313–320, San Jose, CA, 2001.

[13] Dehaspe L., Toivonen H., and King R. Finding frequent substructures in chemical compounds. In *Proceedings of the 1998 International Conference on Knowledge Discovery and Data Mining (KDD'98)*, page 3036, New York, NY, 1998.

[14] S. Nijssen and J. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. In *Proceedings of the 2004 ACM SIGKDD International Conference on Kowledge Discovery in Databases (KDD'04)*, pages 647–352, Seattle, WA, 2004.

[15] S. Nijssen and J. Kok. Frequent Subgraph Miners: Runtimes Don't Say Everything. In *Proceedings Mining and Learning with Graphs (MLG'06), workshop held with ECML-PKDD'06*, pages 173–180, Berlin, Germany, 2006.

[16] A. Srinivasan, R.D. King, S.H. Muggleton, and M. Sternberg. The Predictive Toxicologic Evaluation Challenge. In *Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI'97), Morgan-Kaufmann*, pages 1–6, 1997.

[17] M. Wörlein, T. Meinl, I. Fischer, and M. Philippsen. A Quantitative Comparison of the Subgraph Miners Mofa, gSpan, FFSM, and Gaston. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, pages 392–403, Porto, Portugal, 2005.

[18] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. In *Proceedings of the 2002 International Conference on Data Mining (ICDM'02)*, pages 721–724, Maebashi, Japan, 2002.

Table 1: Collections of graphs.

	PTE	CAN2DA99	HIV-CA	HIV-CM	HIV	NCI
Number of graphs	340	32 557	423	1 081	42 689	237 771
Number of edge labels	4	4	3	3	3	4
Average of edges by graph	27	28	42	34	28	22
Number of vertex labels	66	69	21	27	63	78
Average of vertices by graph	27	26	40	32	26	21

Table 2: Number and size of graphs found for high and low thresholds in the HIV-CA dataset.

High support thresholds			Low support thresholds		
threshold	Number of frequent connected subgraphs	Size of the bigest found subgraph	threshold	Number of frequent connected subgraphs	Size of bigest found subgraph
30%	277	9	3%	8 838 698	44
40%	105	7	4%	1 230 027	37
50%	61	7	5%	272 943	33

Table 3: Number of duplicate candidates found in the six datasets varying the support threshold for the algorithms grCAM and FFSM.

Alg.\ th.	2%	3%	4%	5%	6%	7%
PTE						
grCAM	521 014	56 023	15 217	8 236	4 290	3 451
FFSM	575 302	62 216	16 866	9 652	5 462	4 516
HIV-CM						
grCAM	1 709 165	381 833	51 671	17 660	10 060	6 902
FFSM	2 171 660	434 332	62 951	22 409	12 901	9 067
HIV-CA						
grCAM	548 735 833	40 348 638	4 299 527	923 809	379 068	174 266
FFSM	658 483 021	43 309 620	4 809 405	989 306	416 344	196 931
HIV						
grCAM	49 175	20 206	10 950	6 965	4 754	3 463
FFSM	59 550	25 024	13 706	8 822	6 122	4 537
CAN2DA99						
grCAM	37 918	11 523	5 835	3 660	2 302	1 632
FFSM	45 924	14 545	7 548	4 839	3 157	2 237
NCI						
grCAM	12 818	6 266	4 168	1 408	940	627
FFSM	15 331	7 586	5 073	1 911	1 198	788

Table 4: Number of canonical form tests performed by grCAM and FFSM in the six datasets varying the support threshold.

Alg.\ th.	2%	3%	4%	5%	6%	7%
PTE						
grCAM	643 121	77 156	20 746	11 624	6 344	5 165
FFSM	712 283	80 362	22 821	13 279	7 600	6 302
HIV-CM						
grCAM	2 149 015	482 159	71 544	25 973	15 118	10 578
FFSM	2 730 520	546 104	85 557	32 066	18 799	13 316
HIV-CA						
grCAM	568 576 794	43 332 196	5 376 833	1 152 101	477 769	224 373
FFSM	790 179 611	52 148 318	6 039 432	1 262 249	539 634	259 179
HIV						
grCAM	69 266	29 425	16 360	10 625	7 388	5 457
FFSM	84 677	36 515	20 349	13 244	9 272	6 904
CAN2DA99						
grCAM	52 754	16 891	8 822	5 651	3 631	2 577
FFSM	63 015	20 822	11 055	7 175	4 728	3 365
NCI						
grCAM	28 121	12 831	6 371	2 338	1 641	1 111
FFSM	33 105	16 114	7 720	3 140	2 231	1 761

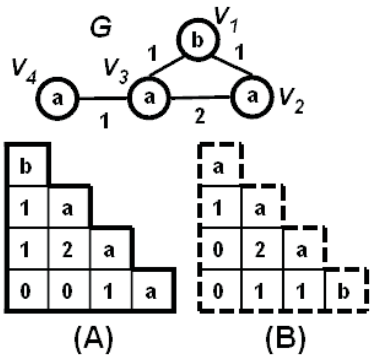

Fig. 1: Example of adjacency matrices of a graph G. The first matrix (A) is the CAM of G according with with the permutation of the vertices $\{v_1, v_2, v_3, v_4\}$; the other one (B) is a not CAM adjacency matrix of G.

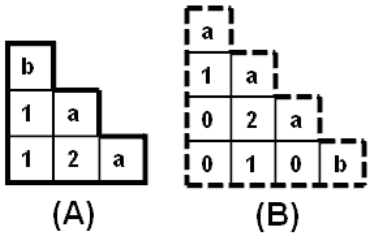

Fig. 2: (A) and (B) are examples of the maximal proper submatrices of the matrices shown in Fig. 1 (A) and (B) respectively.

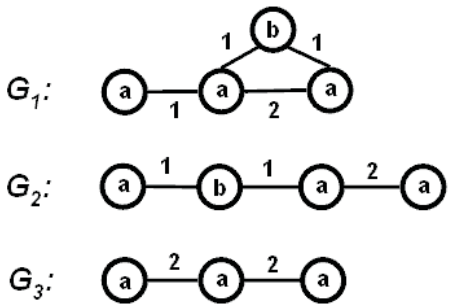

Fig. 3: Example of collection of graphs D.

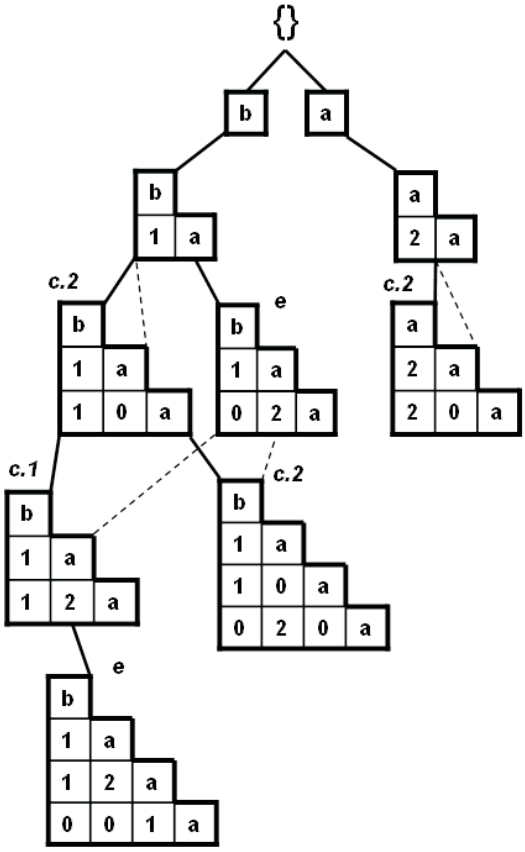

Fig. 4: Example of a CAM Tree. Solid lines represent the parent/child relationship among the nodes and dotted lines are used for pointing out the second parameter matrix in join operations. The additional labels 'c.1', 'c.2', or 'e' point out the way that the matrices were generated: condition 1 or 2 of join, or extension respectively.

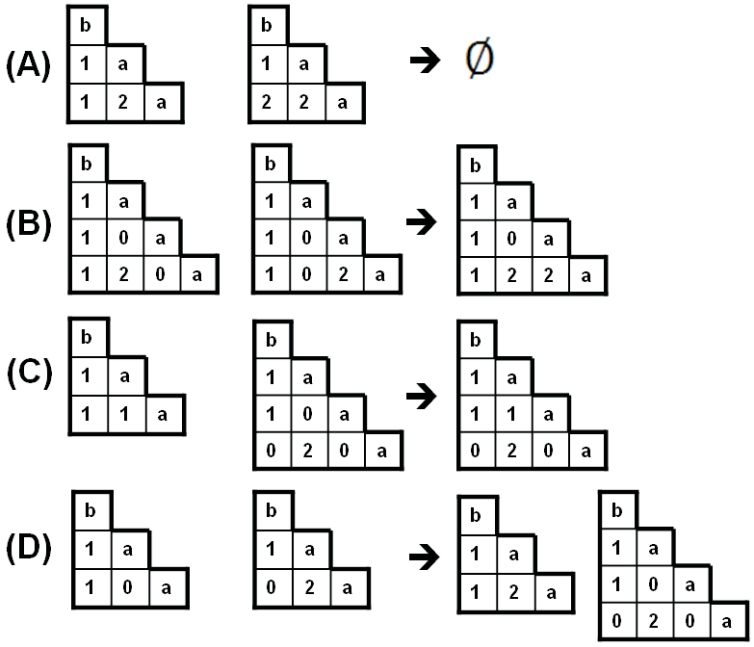

Fig. 5: Examples of cases where the result of the join operation has (A) zero elements, (B,C) one element, and (D) two elements.

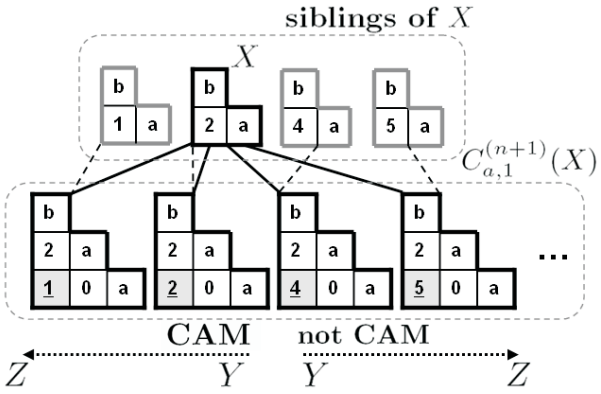

Fig. 6: Example of the usefulness of the reuse properties for the last edge entry in children obtained by join operations. Solid lines represent the parent/child relationship among the nodes in a portion of suboptimal CAM tree. Dotted lines between matrices are used for pointing out the second parameter matrix in join operations.

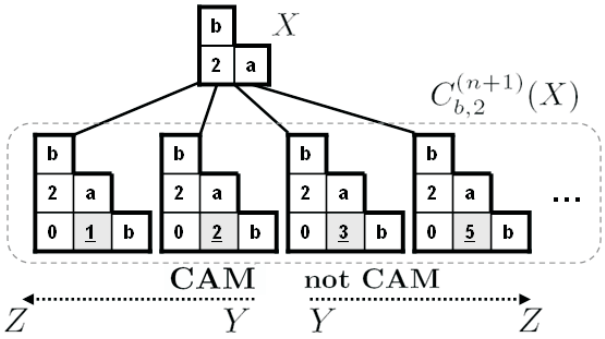

Fig. 7: Example of the usefulness of the reuse properties for the last edge entry in children obtained by extension operations with the same last vertex label.

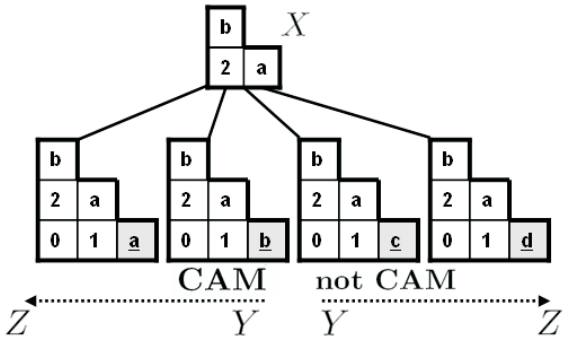

Fig. 8: Example of the usefulness of the reuse properties for the last vertex entry.

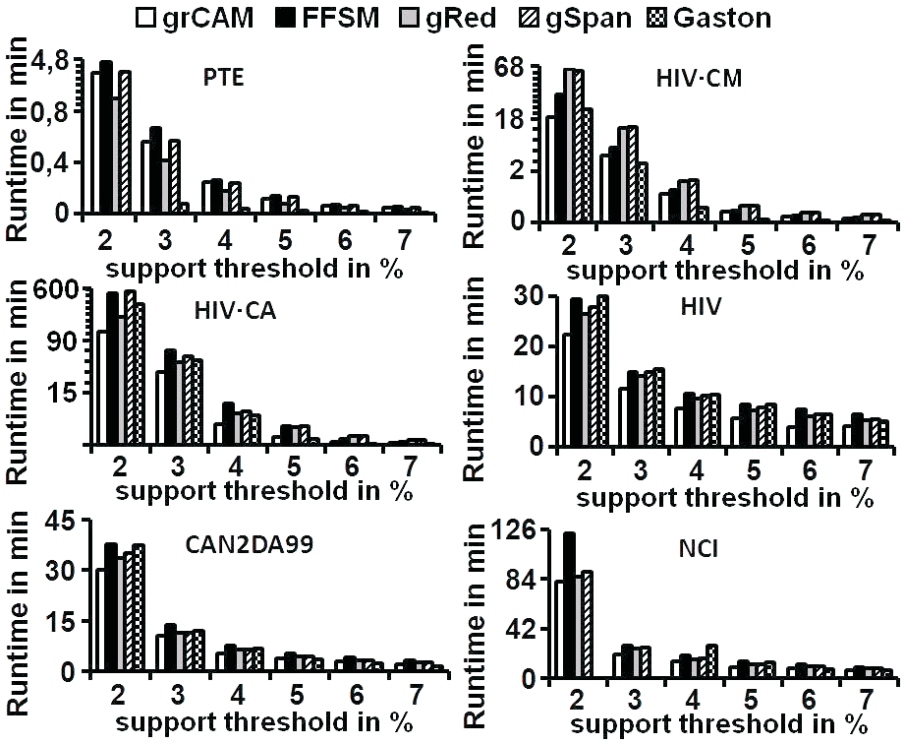

Fig. 9: Runtimes for the six datasets varying the support threshold. Gaston was unable to complete the execution for some low support thresholds in NCI due to its high memory requirements.