Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

Tipo de artículo: Artículo original
Temática: Matemática computacional
Recibido: 19/09/2014 | Aceptado: 15/06/2015

Connected Permutations of Vertices for Canonical Form Detection in Graph Mining

Permutaciones conexas de vértices para la detección de formas canónicas en la minería de grafos

Andrés Gago-Alonso [1*]

[1] Centro de Aplicaciones de Tecnologías Avanzadas. La Habana, Cuba.

[*] Autor para correspondencia: agago@cenatav.co.com

Abstract

Checking redundancies is one of the most significant tasks in graph mining. Canonical forms of graphs are widely used to guarantee and speed up this kind of task. In general, canonical form calculation requires to orderly check partial or complete prefixes of vertex permutations for picking up the codification to unambiguously represent a graph. In this paper, novel theoretical results are introduced for reducing the number of candidate prefixes to a specific subset associated with connected permutations. Furthermore, several interesting mathematical properties are also described and proved, including strong linkages among graph mining, discrete mathematics, and different kinds of canonical forms. Although this paper does not declare a scheme for directly reducing the complexity of finding canonical descriptions, our contributions can open novel opportunities for future improvements in graph mining by interrelating concepts from different existing approaches.

Key words: canonical form, graph isomorphism, connected permutation, adjacency matrix, spanning tree.

Resumen

La verificación de redundancias es una de las taras más influyentes en la minería de grafos. Las formas canónicas son ampliamente usadas para garantizar y acelerar este tipo de tarea. En general, el cómputo de una forma canónica requiere la verificación parcial o completa de todos los prefijos de permutaciones de vértices, para seleccionar aquellas que representa sin ambigüedad al grafo. En este artículo, se introducen nuevos resultados teóricos enfocados a reducir el número de candidatos prefijos a un subconjunto específico con las permutaciones conexas. Adicionalmente, varias

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

propiedades son también descritas y probadas, incluyendo fuertes vínculos entre minería de grafos, matemática discreta, y diferentes tipos de formas canónicas. Aunque este artículo no declara un esquema para reducir directamente la complejidad computacional para detectar formas canónicas, estas contribuciones pueden abrir nuevas oportunidades para obtener futuras mejoras en la minería de grafos, interrelacionando conceptos provenientes de diferentes enfoques que hasta ahora han sido propuestos de manera aislada.

Palabras clave: *formas canónicas, isomorfismo de grafos, permutaciones conexas, matriz de adyacencia, árbol de cobertura*

Introduction

Graph mining is gaining more attention and significance, since advances in collecting and storing data have produced an explosive growth in the amount of available structured data (JIANG, 2013; MANSO, 2014; LI, 2015; VO, 2015). This situation has boosted the necessity to develop new algorithms, called graph miners, to transform this big amount of data into useful information for decision makers. The main idea of several graph miners is to grow subgraphs into the graph collection, adding a new edge or perhaps a new vertex at each step, calculating the quality of each grown subgraph, and rejecting those with low scores. Thus, the development of these miners requires techniques for dealing with the redundancy of candidates during mining process, since the same subgraph can be grown in several ways, adding vertices and edges in different orders. This redundancy can significantly increase the execution times in graph mining (GAGO-ALONSO, 2010a; VO, 2015).

One of the most widely used techniques, to avoid redundant search, consists in defining a canonical form of a graph and using it for representing subgraphs during the mining process (BORGELT, 2006). Some kinds of canonical forms have been defined as strings of labels, which are built by concatenating rows or columns of an adjacency matrix of a graph (INOKUCHI, 2000; KURAMOCHI, 2001; HUAN, 2003). Others are defined as codes of tuples, which are obtained from a spanning tree of a graph (YAN, 2002; NIJSSEN, 2004; BORGELT, 2006; LI, 2015, VO, 2015). All of these approaches are focused on calculating the canonical form of a graph, by traversing the set of vertex permutations.

In this paper, novel theoretical results for enacting the significance of a specific subset of vertex permutations, called connected ones, in canonical form calculation tasks are introduced. In fact, a theorem ensuring that only connected permutations need to be checked during these tasks is mathematically proved. In this sense, other propositions characterizing the cardinality of the connected permutation set of specific kind of graphs are presented. These results give distinction to the reduction achieved by this subset regarding the whole set. Thus, a basic framework for future

improvements in graph mining is stated. Additionally, a linkage between graph mining and discrete mathematics is described, in one of these new properties.

Additionally, a new kind of code of tuples, called underlying code, is defined. This concept is strongly linked with adjacency matrices and spanning trees, by means of another theorem introduced and proved in this paper. Thus, this linkage opens new research chances in graph mining by mixing the skilled features of the above mentioned kind of canonical forms.

The rest of this paper is organized as follows. Firstly, the necessary background for understanding the proposed work is described, including basic graph definitions, previously reported propositions, and examples of canonical forms for graphs. Next, the novel framework for characterizing canonical forms using vertex connected permutations is presented, including the description of the underlying code and its relationship with other canonical forms. Finally, conclusions and future work are given.

Methodology

In this section, the necessary background (coming from the literature) for understanding the proposed theoretical framework and the rest of the paper is presented. Examples of canonical forms for labeled graphs are also included.

Graph definitions

This paper is focused on labeled simple undirected graphs. The formal definition of this kind of graph is a classical concept in graph theory, labeled graph (HARARY, 1969), and it is also given below.

The *universe* of labels is defined as a finite subset, $\Lambda = \{1,2,\dots,\lambda\}$, of positive integer numbers, called *labels*. Thus, 1 and λ are the lowest and highest elements in the universe of labels, respectively.

A *labeled graph* is a 4-tuple, $G =< V, E, L, l >$, 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* (undirected edges are implicitly assumed), each edge is a set with exactly two vertices, L is a set of *labels*, $L \subseteq \Lambda$, and $l: V \cup E \rightarrow L$ is a *labeling function* for assigning labels to vertices and edges. A vertex $v \in V$ such that $v \notin e$, for all edge $e \in E$, is an *isolated vertex*. If for each pair of vertices $u \in V$ and $v \in V$ there is $\{u, v\} \in E$ then G is named as *complete graph*.

Let $G_1 =< V_1, E_1, L_1, l_1 >$ and $G_2 =< V_2, E_2, L_2, l_2 >$ be two graphs. It is said that G_1 is a *subgraph* of G_2 if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $L_1 \subseteq L_2$, and the function l_1 is a restriction of l_2 to L_1. In this case, the notation $G_1 \subseteq G_2$ is used.

A function f is an *isomorphism* between $G_1 =< V_1, E_1, L_1, l_1 >$ and $G_2 =< V_2, E_2, L_2, l_2 >$, if $f: V_1 \rightarrow V_2$ is a bijective function where $l_1(v) = l_2(f(v))$ for each vertex $v \in V_1$, $\{f(u), f(v)\} \in E_2$ and $l_1(\{u, v\}) = l_2(\{f(u), f(v)\})$ for all

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

edge $\{u, v\} \in E_1$. A *subgraph isomorphism* from G_1 to G_2 is an isomorphism from G_1 to a subgraph of G_2; in such case, the notation $G_1 \sqsubseteq G_2$ is used.

A *path* in G is a sequence of vertices $P = (v_1, v_2, \ldots, v_k)$ with $\{v_i, v_{i+1}\} \in E$ for each $i = 1, \ldots, k-1$; in this case, it is said that v_1 and v_k are connected. When $v_1 = v_k$, it is said that the path P is a *cycle*. The graph G is *connected* if for all $v_i, v_j \in V$, $i \neq j$, v_i and v_j are connected by at least one path. The proposition 1 offers a good characterization, already reported in the literature, for connected graphs.

Proposition 1. For each graph $G =< V, E, L, l >$ with $|V| = n$, the following statements are mutually equivalent:

1. G is a connected graph.
2. There is a permutation $P = (v_1, v_2, \ldots, v_n)$ of the vertices in V, such that for each v_i, $2 \leq i \leq n$, there is at least one $v_j \in \{v_1, v_2, \ldots, v_{i-1}\}$ where $\{v_i, v_j\} \in E$.

Proof. The proof of theses equivalences can be found in a book of graph theory (DIESTEL, 2000).

Vertex permutations fulfilling the statement 1 of proposition 1 are called in the scope of this paper as *connected permutation*. A connected graph without cycles is known as *simple tree*. The proposition 2 provides relationships, already reported in the literature, among the above mentioned concepts. Moreover, it also supports the most commonly used canonical form definitions.

Proposition 2. For each graph $G =< V, E, L, l >$ with $|V| = n$, the following statements are mutually equivalent:

1. G is a simple tree.
2. There is a permutation $P = (v_1, v_2, \ldots, v_n)$ of the vertices in V, such that for each v_i, $2 \leq i \leq n$, there is only one $v_j \in \{v_1, v_2, \ldots, v_{i-1}\}$ where $\{v_i, v_j\} \in E$.
3. G is connected with $n - 1$ edges.

Proof. The proof of theses equivalences can be found in a book of graph theory (DIESTEL, 2000).

The graph $T =< V_T, E_T, L_T, l_T >$ is a *spanning tree* of G if $T \subseteq G$, T is a simple tree, and $|V_T| = |V|$. Taking it for granted, let $P = \{v_1, v_2, \ldots, v_n\}$ be a permutation of V_T according to the statement 2 of proposition 2.

Let us suppose that the vertices u and v have indices i and j, respectively, according to the permutation P. Let $l_i = l(u)$, $l_j = l(v)$ and $l_{(i,j)} = l_{(j,i)} = l(e)$ be the labels of u, v and $e = \{u, v\}$, respectively. Without loss of generality, it can be assumed that $i < j$. The *tuple* of e regarding T is calculated as in (1).

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

$$\tau(e,T) = \begin{cases} (i,j,l_i,l_{(i,j)},l_j) & , \quad e \in E_T, \\ (j,i,l_j,l_{(j,i)},l_i) & , \quad e \notin E_T. \end{cases} \tag{1}$$

Thus, each edge $e \in E$ can be coded as a tuple, $\tau(e,T) \in W_n = K_n^2 \times \Lambda^3$, where $K_n = \{1,2,\dots,n\}$. The set W_n is the *vocabulary* and it contains the available tuples in the graph G.

Let $s_1 = (a_1, a_2, \dots, a_m)$ and $s_2 = (b_1, b_2, \dots, b_n)$ be two tuple sequences, where $a_i, b_j \in K_n^2 \times L^3$ for $1 \leq i \leq m$ and $1 \leq j \leq n$ and $\prec$ be a total order in W_n. It is said that $s_1 < s_2$ according $\prec$ if one of the following conditions is true

$$\exists t, \forall k < t, a_k = b_k, \text{ and } a_t \prec b_t; \tag{2}$$

$$m < n \text{ and } \forall k \leq m, a_k = b_k. \tag{3}$$

Canonical form based on string of labels

A graph can be represented by its canonical adjacency matrix. This kind of representation has been used in previously reported works for graph mining (INOKUCHI, 2000; KURAMOCHI, 2001; HUAN, 2001; LI, 2015). In this section, the string of labels is defined in a slightly different way regarding previously published works, see (4), giving priority to vertex labels over edge ones.

Let $G =< V, E, L, l >$ be a labeled graph with $|V| = n$ and let $P = (v_1, v_2, \dots, v_n)$ be a permutation of the vertices in V. The *adjacency matrix* of G regarding P is a lower triangular matrix $X(G,P) = (x_{i,j})_{i,j=1}^{n}$ 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.

The *string of labels* of an adjacency matrix $X = X(G,P)$ is built concatenating lower triangular rows of X, see (4). This string is made up by labels in $\Lambda \cup \{0\}$.

$$\text{string}(X) = x_{1,1}x_{2,2}x_{2,1}x_{3,3}x_{3,1}x_{3,2} \dots x_{n,n}x_{n,1} \dots x_{n,n-1} \tag{4}$$

A standard lexicographic order $\prec$ could be used to define a total order among strings, considering that their labels are sorted as integer numbers. Thus, a *canonical string* of G is usually considered as the maximal string among all its possible strings. The adjacency matrix where such maximal string is attached is called the *canonical adjacency matrix* of G. Additionally, the vertex permutation associated with this matrix is also called the *canonical permutation* of G.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

Canonical form based on code of tuples

A labeled graph can be represented by a unique sequence of edges called minimum depth-first search (DFS) code. This kind of canonical representation is based on DFS graph traversals and it was introduced by gSpan (YAN, 2002).

Let $G =< V, E, L, l >$ be a connected graph and let us suppose that a DFS traversal in G is performed. A *DFS tree* $T = < V_T, E_T, L_T, l_T >$ of G is the rooted tree built as follow: the starting vertex in the traversal is the root of T, T is a spanning tree of G ($V_T = V$)) and T contains the edges of G that were used for the DFS traversal ($E_T \subseteq E$).

The graph G can have many different DFS trees because there is more than one DFS traversal. Each DFS tree T defines a unique order among all the vertices in V. Therefore, each vertex could be numbered according to this *DFS order*. Thus, a permutation of V_T according to statement 2 of proposition 2 is given. Assuming $n = |V|$, the root of T is numbered with index 1 and the last vertex in the DFS traversal is numbered with index n. The last vertex is also called *rightmost vertex* of T.

Each edge $e = \{u, v\} \in E$ is coded as a tuple $\tau(e, T)$ according to the DFS tree T, see (1). In addition, a linear order $\prec_e$ among the vocabulary W_n could be defined as follows. Let $t_1 = (i_1, j_1, \dots)$ and $t_2 = (i_2, j_2, \dots)$ be two tuples, it is said that $t_1 \prec_e t_2$ if and only if one of the following statements is true:

- $i_1 < j_1 \wedge i_2 < j_2 \wedge (j_1 < j_2 \vee (j_1 = j_2 \wedge i_1 > i_2))$,
- $i_1 \geq j_1 \wedge i_2 \geq j_2 \wedge (i_1 < i_2 \vee (i_1 = i_2 \wedge j_1 < j_2))$,
- $i_1 \geq j_1 \wedge i_2 < j_2 \wedge i_1 < j_2$,
- $i_1 < j_1 \wedge i_2 \geq j_2 \wedge j_1 \leq i_2$,
- $i_1 = i_2 \wedge j_1 = j_2 \wedge t_1 \prec_l t_2$.

The lexicographic order $\prec_l$ is used to compare the tuples t_1 and t_2 regarding the last three components in each tuple. This order is determined by comparing the third component as first priority, next the fourth one, and finally the fifth one.

The *DFS code* of the graph G regarding the DFS tree T is a sequence in W_n built using $\prec_e$. All the tuples obtained from the edges in E are sorted using $\prec_e$ to build this sequence. Thus, a graph G can be coded as a sequence of tuples, denoted as $\text{code}(G, T)$, using one of its DFS trees. A *canonical form code* of a graph G is defined as the minimum tuple sequence according to $\prec_e$ among all DFS codes of G.

Other example of ways for building a canonical form codes based on tuples were presented by (BORGELT, 2006), using BFS trees instead of DFS ones, and the proposal of (NIJSSEN, 2004), using graph backbone paths.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

Results and discussion

In this section, results of our research are presented, including the novel framework for characterizing canonical forms using vertex connected permutations, the description of the underlying code and its relationship with other canonical forms.

Novel properties for canonical adjacency matrix

It is a fact that the number of vertex permutation to be checked does not determine the efficiency of canonical form calculations, since the most efficient algorithms, for example (MCKAY, 1981), employ topological properties and label occurrences for pruning partial permutation prefixes. However, there is a worst case where such algorithms require checking the $n!$ Vertex permutations.

An interesting property for describing the set of permutations to be checked is stated in theorem 3. This statement only uses topological properties of graphs for giving distinction to the canonical permutation. Although the cardinality of this set can be irrelevant for graph mining, this property could be used in the future for enriching the already mentioned pruning strategies and speeding up canonical form calculations.

Theorem 3. The canonical permutation of a connected graph is a connected permutation.

Proof. Let us suppose that $P = (v_1, v_2, \dots, v_n)$ is the canonical permutation of a connected graph G and P is non-connected. Thence, there is i, $2 \leq i \leq n$, such that $\{v_i, v_j\} \notin E$ for all $v_j \in \{v_1, v_2, \dots, v_{i-1}\}$. It is easy to verify that $i \neq n$, since G is connected and v_n cannot be an isolated vertex; thus, $2 \leq i \leq n-1$. Moreover, there is k, $i+1 \leq k \leq n$, there is at least one $v_j \in \{v_1, v_2, \dots, v_{i-1}\}$ where $\{v_k, v_j\} \in E$, since G is connected. Let $P' = (v_1, v_2, \dots, v_{i-1}, v_k, \dots, v_{k-1}, v_i, \dots, v_n)$ be the permutation obtained from P by swapping v_i and v_k. It is easy to prove that $\text{string}(X(G,P)) \prec \text{string}(X(G,P'))$. In fact, the substring corresponding to the i-th row $(x_{i,i}x_{i,1}x_{i,1} \dots x_{i,i-1})$ in $X(G,P)$ is lexicographically lesser than the one in $X(G,P')$, since this substring in $X(G,P)$ is $l00 \dots 0$, where $l = l(v_i)$, whereas in $X(G,P')$ there is at least a non-zero element on the corresponding substring. Therefore, P cannot be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by reductio ad absurdum.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

The following propositions illustrate the number of connected permutations in specifics kind of graphs. Proposition 4 describes the behavior of this number in paths, showing a strong and interesting linkage among graph mining, graph theory, and some special numbers (CONWAY, 1996) coming from discrete mathematics.

Proposition 4. Let $G =< V, E, L, l >$ be a graph representing a path; that is, $V = \{v_1, v_2, \dots, v_n\}$, $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}\}$, and $n \geq 2$. Then, the number of connected permutations of G is 2^{n-1}.

Proof. Let $N(v_i)$ be the number of connected permutations of G starting from v_i.

First of all, an interesting function sequence which will be used for counting the number of connected permutations in the graph G is defined. This sequence represents a strong linkage between graph theory and some special numbers.
Let $T_k: N \to N$, $k \geq 1$, be the sequence of functions defined for each $n \in N$, according the following recurrence formula: $T_1(n) = 1$ for $k = 1$, and $T_k(n) = \sum_{m=1}^{n} T_{k-1}(m)$ for $k > 1$. In (5), the above mentioned function sequence is shown in expanded way:

$$\begin{array}{lcll}
T_1(n) & = & 1 & = 1 \quad \text{(ones)} \\
T_2(n) & = & 1 + 1 + 1 + \cdots + 1 & = n \quad \text{(counting numbers)} \\
T_3(n) & = & 1 + 2 + 3 + \cdots + n & = n(\frac{n+1}{2}) \quad \text{(triangular numbers)} \\
T_4(n) & = & 1 + 3 + 6 + \cdots + n(\frac{n+1}{2}) & = n(\frac{n+1}{2})(\frac{n+2}{3}) \quad \text{(tetrahedral numbers)} \\
T_5(n) & = & 1 + 4 + 10 + \cdots + n(\frac{n+1}{2})(\frac{n+2}{3}) & = n(\frac{n+1}{2})(\frac{n+2}{3})(\frac{n+3}{4}) \quad \text{(pentatope numbers)} \\
\vdots & & \vdots & \vdots \\
T_k(n) & = & 1 + (k-1) + \cdots & = C_{k-1}^{n+k-2} \quad (k - \text{tope numbers}) \\
\vdots & & \vdots & \vdots
\end{array} \tag{5}$$

where C_r^n represents a binomial coefficient or the number of combinations of r items that can be selected from a set of n items. Next, the following property of binomial coefficients is underlined:

$$\sum_{i=0}^{n} C_k^{k+i} = C_{k+1}^{k+n+1} \tag{6}$$

which can be proved by mathematical induction. For the base case $n = 1$, it is verified that $C_k^k + C_k^{k+1} = k + 2 = C_{k+2}^{k+1}$. The inductive step is also achieved since $\sum_{i=0}^{n+1} C_k^{k+i} = \sum_{i=0}^{n} C_k^{k+i} + C_k^{k+n+1} = C_{k+1}^{k+n+1} + C_k^{k+n+1} = C_k^{k+n+2}$.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

After that, it is easy to prove that $T_k(n) = C_{k-1}^{n+k-2}$ using mathematical induction. For the base cases $n = 1$, $n = 2$, $n = 3$, $n = 4$, and $n = 5$, the fact is already known (CONWAY, 1996), and it can be verified in (5). The inductive step is also checked, using (6), since $T_k(n) = \sum_{m=1}^{n} T_{k-1}(m) = \sum_{m=0}^{n-1} C_{k-2}^{m+k-3} = C_{k-1}^{n+k-2}$.

Returning to the graph G, it can be seen that there is only one connected permutations starting from v_1, since v_2 must be the second permutation element, and so on. By symmetry, this fact is also true for v_n. For $n \geq 3$, it can be checked manually that there are $n-1$ connected permutations starting from v_2; this fact is also true for v_{n-1} by symmetry. Thus, the symmetry $N(v_i) = N(v_{n-i+1})$, for each $1 \leq i \leq n$, can be proven easily. Besides, it is easy to prove that $N(v_i) = T_i(n-i+1)$, for $i \leq \lceil n/2 \rceil$, and it can be calculated, by symmetry, for the remaining vertices. Thus, $N(v_i) = C_{i-1}^{n-1}$, for each $1 \leq i \leq \lceil n/2 \rceil$. Finally, the number of connected permutations of G is 2^{n-1}, using properties of binomial coefficients.

The Proposition 4 is entirely irrelevant for graph mining, since there are $O(1)$ strategies (NIJSSEN, 2004) for detecting path canonical forms, taking into account the string of labels. Nevertheless, it is presented for illustrating the contrast between connected and non-connected permutation sets in a family of graphs ($2^{n-1} << n!$), without considering labels.

This fact emphasizes the usefulness of theorem 3 for distinguishing the canonical permutation in paths. Similar results can be stated for cycles, see proposition 5.

Proposition 5. Let $G =< V, E, L, l >$ be a graph representing a cycle; that is, $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$, and $n \geq 3$. Then, the number of connected permutations of G is $n2^{n-2}$.

Proof. For the first position in a permutation, there are n possibilities. For the subsequent $n-2$ positions, there are only two possibilities that guarantee a connected permutation. For the last position, there is only one option for the last unselected vertex. Thus, the number of connected permutations of G is $n2^{n-2}$.

Until now, analytical formulae for more topologically complex graphs are not given. For example in complete graphs, every permutation is connected. However, even in barely complete graphs, a remarkable number of non-connected permutations (see proposition 6) can be detected.

Proposition 6. Let $G =< V, E, L, l >$ be a complete graph with n vertices. The number of connected permutations of G and the number of vertex permutations, $n!$, are the same. Let $e \in E$ be an edge of G; then, the graph obtained from G by removing e has $n! - 2(n-2)!$ connected permutations.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

Proof. The first statement is easy to check since any vertex permutation of G is connected due to completeness. Let us suppose that $e = \{u, v\}$. Permutations starting with u and v are non-connected ones. There are $2(n-2)!$ permutations in this case. The remaining ones are connected. Therefore, the proof was concluded.

In this way, theorem 3 could be used, in the future, for speeding up algorithms for canonical form calculation. They only need to check connected permutation prefixes, diminishing somehow the number of iterations.

A linkage between Adjacency Matrices and Spanning Trees

The question of establishing connections between adjacency matrices and spanning trees has already been treated. In fact, several variants of constructing a code from an adjacency matrix preserving the equivalence to a spanning tree can be described (BAPAT, 1996). This section contains an example for illustrating the connection with a kind of code only based on tuples describing edges and structurally similar to the already known DFS code.

Let $G =< V, E, L, l >$ be a connected graph with n vertices and $P = \{v_1, v_2, \ldots, v_n\}$ be a connected permutation of V. The *first edge* of v_i, $1 < i \leq n$, in P is defined by us as the edge $\{v_j, v_i\} \in E$ such that $1 \leq j < i$ and $\{v_k, v_i\} \notin E$ for all k, $1 \leq k < j$. The spanning tree of G, made up by the first edges of any vertex of V in P, is called by us the *underlying spanning tree* of G in P.

Let T be the underlying spanning tree of G in P, coding each edge of G by means of $\tau(e, T)$, see (1). Now, a total order in W_n using P is defined. Let, $t_1 = (i_1, j_1, a_1, b_1, c_1)$ and $t_2 = (i_2, j_2, a_2, b_2, c_2)$ be two tuples. It is said that $t_1 \prec_u t_2$ if and only if one of the following statements is true:

- $i_1 < j_1 \wedge i_2 < j_2 \wedge (j_1 < j_2 \vee (j_1 = j_2 \wedge i_1 < i_2))$,
- $i_1 \geq j_1 \wedge i_2 \geq j_2 \wedge (i_1 < i_2 \vee (i_1 = i_2 \wedge j_1 < j_2))$,
- $i_1 \geq j_1 \wedge i_2 < j_2 \wedge i_1 < j_2$,
- $i_1 < j_1 \wedge i_2 \geq j_2 \wedge j_1 \leq i_2$,
- $i_1 = i_2 \wedge j_1 = j_2 \wedge t_1 \prec_l t_2$.

The lexicographic order $\prec_l$ is used to compare the tuples t_1 and t_2 regarding the last three components in each tuple. This order is determined comparing the third component as first priority, next the fifth one, and finally the fourth one.

The *underlying code* of G given P is defined as a sequence in W_n constructed using $\prec_u$. All of the tuples $\tau(e, T)$ obtained from the edges $e \in E$ are sorted using $\prec_u$ to build this sequence. Thus, a graph G can be coded asa sequence of tuples, denoted as u-code(G, P), using one of its DFS trees. A *canonical underlying code* of a graph G is defined as the minimum underlying code according to $\prec_u$ among all vertex permutations of G.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

Underlying code becomes a novel kind of canonical form, preserving a semantics coming from adjacency matrices and showing syntax based on tuples like DFS codes. The following theorem boosts such affirmation.

Theorem 7. Let us suppose that P is the canonical permutation of G. Let $G' =< V, E, L', l' >$ be a graph obtained from G by relabeling vertices and edges, according to $l'(x) = \lambda - l(x)$, for each $x \in V \cup E$ and $L' = \{l'(x) | x \in V \cup E\}$, where λ is the highest element in the universe of labels Λ. The tuple sequence u-code(G', P) is the canonical underlying code of G'.

Proof. Let us suppose that u-code$(G', P) = (b_1, b_2, \dots, b_m)$ is a non-canonical underlying code of G' and m is the number of edges in G. Thence, there is a permutation P' such that u-code$(G', P') = (a_1, a_2, \dots, a_m) < (b_1, b_2, \dots, b_m)$, according to $\prec_u$. Thus, there is natural number t, $1 \leq t \leq m$, such that $a_k = b_k$, for all k, $k < t$, and $a_t \prec_u b_t$. Let us denote $a_t = (i_1, j_1, \alpha_1, \beta_1, \gamma_1)$ and $b_t = (i_2, j_2, \alpha_2, \beta_2, \gamma_2)$.

If $t = 1$ then $a_1 = (1,2, \alpha_1, \beta_1, \gamma_1) \prec_l (1,2, \alpha_2, \beta_2, \gamma_2) = b_t$. In this case, the string of labels of the matrices $X(G, P)$ and $X(G, P')$ starts with the labels of L, where $(\lambda - \alpha_1, \lambda - \gamma_1, \lambda - \beta_1, \dots) > (\lambda - \alpha_2, \lambda - \gamma_2, \lambda - \beta_2, \dots)$. Therefore, P cannot be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by *reductio ad absurdum* in this case.

Now, let us assume that $t > 1$, and $i = \min\{\max\{i_1, j_1\}, \max\{i_2, j_2\}\}$. Then, it is not difficult to check that the matrices $X(G, P)$ and $X(G, P')$ have the same $(i-1)$-main minor, and the first difference between them takes place at the i-th row.

Taking into account the definition of $\prec_u$, five cases where $a_t \prec_u b_t$ are given. Each one of these cases will be individually analyzed.

The first subcase of the first case, $i_1 < j_1$, $i_2 < j_2$ and $j_1 < j_2$, never takes place in connected permutations of the same graph. In fact, the tuple of the first edge of v_{j_1} in P is always located between b_{t-1} and b_t.

Let us suppose the first case, but in the second subcase $i_1 < j_1$, $i_2 < j_2$, $j_1 = j_2$ and $i_1 < i_2$. In this case, $i = j_1 = j_2$ and the i-th row of $X(G, P')$ has a non-zero element in a position lesser than the one the i-th row of $X(G, P)$. Therefore, P can not be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by reductio ad absurdum in this case.

Now, let us assume that $t > 1$, and $i = \min\{\max\{i_1, j_1\}, \max\{i_2, j_2\}\}$. Then, it is not difficult to check that the matrices $X(G,P)$ and $X(G,P')$ have the same $(i-1)$-main minor, and the first difference between them takes place at the i-th row.

Taking into account the definition of $\prec_u$, five cases where $a_t \prec_u b_t$ are given. Each one of these cases will be individually analyzed.

The first subcase of the first case, $i_1 < j_1$, $i_2 < j_2$ and $j_1 < j_2$, never takes place in connected permutations of the same graph. In fact, the tuple of the first edge of v_{j_1} in P is always located between b_{t-1} and b_t.

Let us suppose the first case, but in the second subcase $i_1 < j_1$, $i_2 < j_2$, $j_1 = j_2$ and $i_1 < i_2$. In this case, $i = j_1 = j_2$ and the i-th row of $X(G,P')$ has a non-zero element in a position lesser than the one the i-th row of $X(G,P)$. Therefore, P can not be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by reductio ad absurdum in this case.

The first subcase of the second case, $i_1 \geq j_1$, $i_2 \geq j_2$, and $i_1 < i_2$, never takes place in connected permutations of the same graph. In fact, the tuple of the first edge of v_{i_2} in P is always located between b_{t-1} and b_t.

Let us suppose the second case, but in the second subcase $i_1 \geq j_1$, $i_2 \geq j_2$, $i_1 = i_2$, and $j_1 < j_2$, takes place. In this case, $i = i_1 = i_2$ and the i-th row of $X(G,P')$ has a non-zero element in a position lesser than the one the i-th row of $X(G,P)$. Therefore, P cannot be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by reductio ad absurdum in this case.

In the third case $i_1 \geq j_1$ and $i_2 < j_2$, and $i_1 < j_2$, it is verified that $i = i_1$. Furthermore, the i-th row of $X(G,P')$ has a non-zero element in a position lesser than the one the i-th row of $X(G,P)$. Therefore, P cannot be the canonical permutation of G. This fact contradicts the initial assumption. Therefore, the theorem becomes true by reductio ad absurdum in this case.

The fourth case, $i_1 < j_1$, $i_2 \geq j_2$, and $j_1 \leq i_2$, never takes place in connected permutations of the same graph. In fact, the tuple of the first edge of v_{i_2} in P is always located between b_{t-1} and b_t.

In the last case, $i_1 = i_2$, $j_1 = j_2$, and $t_1 \prec_l t_2$, the matrices $X(G,P)$ and $X(G,P')$ have the first difference between them at the same cell position. In addition, it is verified that $a_1 = a_2$, since both matrices have the same $(i-1)$-main minor. Therefore, $c_1 \leq c_2$. Now, if $c_1 < c_2$, then P cannot be the canonical permutation of G. Otherwise, $c_1 = c_2$, it is verified

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 57-71

that $b_1 < b_2$, and then P cannot be the canonical permutation of G. Thus, the theorem becomes true by reductio ad absurdum.

Theorem 2 enacts an interesting linkage between adjacency matrices and spanning trees. Additionally, a new kind of code of tuples is defined keeping the semantics of adjacency matrices. This fact opens new skylines for mixing graph mining results coming from algorithms based on string of labels, for example: FSG (KURAMOCHI, 2001), FFSM (HUAN, 2001), grCAM (GAGO-ALONSO, 2010b), VEAM (ACOSTA-MENDOZA, 2012) and REAFUM (LI, 2015), and other ones based on codes of tuples, for example: gSpan (YAN, 2002) Gaston (NIJSSEN, 2004), MoFa (BORGELT, 2006), and gdFil (GAGO-ALONSO, 2010a).

Conclusions

The main conclusion of this paper is that only connected permutations need to be checked for calculating a kind the canonical adjacency matrix. A theorem supporting such affirmation was stated and mathematically proved. In addition, a characterization of the cardinality of the connected permutation subset was given for specific kind of graphs; including: path, cycles, and complete graphs without only one edge. Thus, the reduction of the cardinality achieved by this subset regarding the whole set of permutations was emphasized. Additionally, the proof of this characterization for paths shows a relationship among graph mining, graph theory and k-tope numbers coming from discrete mathematics. These properties could be used, in future work, for speeding up the redundancy checking in graph mining, since a reduction of the number of iterations could be attained.

Additionally, the main idea of a previously published work (BORGELT, 2006), establishing connections between two existing kinds of canonical form based on tuple code, is expanded by including the link with canonical adjacency matrix. This fact was supported by a new theorem stated and proved in this paper. Moreover, the above linkage is achieved by means of the underlying code, a novel codification strategy for labeled graphs.

Future work will be devoted to implement computational algorithms for canonical form detection, taking advantage of the novel mathematical framework. In this sense, we are trying to enrich the already reported pruning strategies, for example the proposed one in the Nauty algorithm (MCKAY, 1981), by considering connected permutation prefixes. Besides, some hybrid approaches between string of labels and tuple codes will be designed and tested.

References

ACOSTA-MENDOZA, N.; GAGO-ALONSO, A.; MEDINA-PAGOLA, J.E. Frequent approximate subgraphs as features for graph-based image classification. Knowledge-Based Systems, 2012, 27, 381-392.

BORGELT, C. Canonical forms for frequent graph mining. In: 30th Annual Conference of the German Classification Society, Universitat Berlin, Springer-Verlag, 2006, 337-349.

CONWAY, J.; GUY, R. The Book of Numbers. New York, Copernicus, Springer-Verlag, 1996. 310 pages.
BAPAT, R. Graphs and Matrices. New Delhi, Hindustan Book Agency, India, 2010. 171 pages

DIESTEL, R. Graph Theory. Electronic Edition, Springer-Verlag, New York, 2000.

GAGO-ALONSO, A.; MEDINA-PAGOLA, J.E.; CARRASCO-OCHOA, J.A.; MARTÍNEZ-TRINIDAD, J.F. Full duplicate candidate pruning for frequent connected subgraph mining. Integrated Computer-Aided Engineering, 2010a, 17(3): 211-225.

GAGO-ALONSO, A.; PUENTES-LUBERTA, A.; CARRASCO-OCHOA, J.A.; MEDINA-PAGOLA, J.E.; MARTÍNEZ-TRINIDAD, J.F. A new algorithm for mining frequent connected subgraphs based on adjacency matrices. Intelligence Data Analysis, 2010b, 14 (3), 385-403.

HARARY, F.: Graph Theory. Addison-Wesley, Reading, MA, 1969, 178-180.

HUAN, J.; WANG, W.; PRINS, J. Efficient mining of frequent subgraphs in the presence of isomorphism. In: 3rd IEEE International Conference on Data Mining, Melbourne, FL, IEEE Computer Society, 2003, 549-552.

INOKUCHI, A.; WASHIO, T.; MOTODA, H. An apriori-based algorithm for mining frequent substructures from graph data. In: 4th European Conference on Principles of Data Mining and Knowledge Discovery, Lyon, France, Springer-Verlag, 2000, 13-23.

JIANG, C.; COENEN, F.; ZITO, M. A survey of frequent subgraph mining algorithms, The Knowledge Engineering Review, 2013, 28: 75-105.

KURAMOCHI, M.; KARYPIS, G. Frequent Subgraph Discovery. In: 1st IEEE International Conference on Data Mining, San Jose, CA, IEEE Computer Society, 2001, 313-320.

LI, R.; WANG, W.: REAFUM: Representative Approximate Frequent Subgraph Mining. In: SIAM International Conference on Data Mining, Vancouver, BC, Canada, 2015. ISSN 2167-0099.

MANSO, M.; PELLINO, S.; PETROSINO, A.; ROZZA, A. A Novel Graph Embedding Framework for Object Recognition. In: Computer Vision - ECCV 2014 Workshops, 2014, 341-352.

MCKAY, B. D. Practical graph isomorphism. Congressus Numerantium, 1981, 30: 45-87.

NIJSSEN, S.; KOK, J.N. A quickstart in frequent structure mining can make a difference. In: 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, ACM, 2004, 647-652.

VO, B.; NGUYEN, D.; NGUYEN, T.L. A Parallel Algorithm for Frequent Subgraph Mining. In: International Conference on Computer Science, Applied Mathematics and Applications, Metz, France, 2015, 163-173.

YAN, X.; HAN, J. gSpan: Graph-based substructure pattern mining. In: International Conference on Data Mining, Maebashi, Japan, IEEE, 2002, 721-724.