Finding Collections of k-Clique Percolated Components in Attributed Graphs

Pierre-Nicolas Mougel[1,2], Christophe Rigotti[1,2], and Olivier Gandrillon[1,3]

[1] Université de Lyon, CNRS, INRIA
[2] INSA-Lyon, LIRIS, UMR5205, F-69621, France
[3] Université Lyon 1, CGPhiMC, UMR5534, F-69622, France

Abstract. In this paper, we consider graphs where a set of Boolean attributes is associated to each vertex, and we are interested in k-clique percolated components (components made of overlapping cliques) in such graphs. We propose the task of finding the collections of homogeneous k-clique percolated components, where homogeneity means sharing a common set of attributes having value true. A sound and complete algorithm based on subgraph enumeration is proposed. We report experiments on two real databases (a social network of scientific collaborations and a network of gene interactions), showing that the extracted patterns capture meaningful structures.

Keywords: graph mining, network analysis, attributed graph, k-clique percolated component.

1 Introduction

During the last decade, graph mining has received an increasing interest in the data mining community. More recently, several works have considered the mining of *enriched* graphs where attributes are associated to the vertices. These works led to interesting results, for instance in clustering [4,8,15,16], dense graph mining [7,12] or graph matching [14].

In this paper, we focus on the special case where the domain of the attributes is Boolean and we propose to extract collections of components called *k-clique percolated components* [1]. More precisely, we define a pattern as a Collection of Homogeneous k-clique Percolated components (CoHoP), where homogeneity means that the vertices in all components share a common set of Boolean attributes having value *true*. A CoHoP pattern must also satisfy two additional constraints: it must contain more than a given number of k-clique percolated components and the vertices must have in common more than a given number of attributes set to *true*. A k-clique percolated component has been defined in [1] as a union of cliques of size k connected by overlaps of $k-1$ vertices (we recall the more formal definition in the next section), and since then, it has been widely accept as one structure that can be used to represent the notion of community. A CoHoP, as introduced here, can thus be interpreted as a set of communities, where elements in all communities share similar Boolean properties.

P.-N. Tan et al. (Eds.): PAKDD 2012, Part II, LNAI 7302, pp. 181–192, 2012.
© Springer-Verlag Berlin Heidelberg 2012

In this paper, we also present a sound and complete algorithm to extract the CoHoPs, and we show on two datasets (a coauthor graph and a gene interaction graph) that these patterns can be used to capture useful information, depicting underlying hidden structure of the graph.

The rest of the paper is organized as follows. Section 2 introduces the definition of the CoHoP patterns. The extraction algorithm is described in Section 3 and the experiments are reported in Section 4. The related works are discussed in Section 5, and Section 6 briefly concludes.

2 Pattern Definition

In this section, we first define the dataset structure and recall the notion of k-clique percolated component. Then, we define the targeted patterns, that are collections of k-clique percolated components.

Graphs where information are associated to vertices have been used in different research areas under various names, e.g. attributed graphs [12,14,15,16], itemset-associated graphs [2], informative graphs [4,8], graphs with feature vectors [7]. In this paper, we use the term attributed graphs, and restrict ourselves to Boolean attributed graphs.

Definition 1 (Boolean attributed graph). *A* Boolean attributed graph *is denoted* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{F})$ *where* $\mathcal{V}$ *is the set of vertices,* $\mathcal{E}$ *is the set of edges,* $\mathcal{A}$ *is the set of Boolean attributes, and* $\mathcal{F} : \mathcal{V} \to 2^{\mathcal{A}}$ *is the function returning for a vertex the set of attributes having value true.*

For notational convenience, let us define the following functions.

Definition 2 (Functions *vert* **and** Atb**).** *Let* x *be an attribute. The function* $vert(x) = \{v \in \mathcal{V} \mid x \in \mathcal{F}(v)\}$ *returns the set of vertices having value true for the attribute* x*. Let* M *be a collection of sets of vertices. Then,* $\textsc{Atb}(M) = \bigcap_{V \in M}(\cap_{v \in V}\mathcal{F}(v))$ *is the set of attributes shared by all vertices in* M*.*

Let $\mathcal{G}$ be an attributed graph. We denote $\mathcal{G}[V]$ the subgraph of $\mathcal{G}$ induced by the set of vertices V, i.e., $\mathcal{G}$ restricted to the vertices in V. The notation $\mathcal{G}[\![X]\!]$ denotes the subgraph induced by the set of vertices having value true for all attributes in X, i.e., $\mathcal{G}[\![X]\!] = \mathcal{G}[\cap_{x \in X} vert(x)]$.

A clique is a set of vertices in which every pair of distinct vertices is connected by an edge and a k-clique is a clique of size k. A k-clique percolated component (termed also k-clique-community in [11]) is a relaxed version of the concept of cliques. The definition of k-clique percolated component given in [1] can be reformulated as follows using an equivalence relation over the cliques.

Definition 3 (Adjacency relation). *Let* $\mathcal{G}$ *be an attributed graph and* $\mathfrak{R}$ *be the adjacency relation over the* k*-cliques in* $\mathcal{G}$*. Two* k*-cliques are related by* $\mathfrak{R}$ *if and only if they have an intersection of at least* $k-1$ *vertices. Let* $\mathfrak{R}^t$ *be the transitive closure of* $\mathfrak{R}$*.*

The relation $\mathfrak{R}$ is symmetric and reflexive, so $\mathfrak{R}^t$ is symmetric, reflexive, and transitive. Consequently, $\mathfrak{R}^t$ is an equivalence relation.

Definition 4 (k-clique Percolated Component (k-PC)). *A k-PC is the union of all k-cliques in a class of equivalence over $\mathfrak{R}^t$.*

In other words, a k-PC is the union of all k-cliques that can be reached from each other through a series of adjacent k-cliques. We will denote $\mathcal{C}_{kpc}(\mathcal{G})$ the collection of all k-PCs in an attributed graph $\mathcal{G}$. Compared to other fault-tolerant clique definitions, the particularity of k-PC is to enforce the fact that each vertex can be reached from any other vertex through well connected subset of vertices [11]. In the context of social networks, it represents a community where each person, even if not directly connected to another member, can easily find a way to communicate with him/her. Note also that with this definition, a clique is contained in at most one k-PC. However, since a vertex can be in several cliques sharing less than $k-1$ vertices, it can be part of several k-PCs.

As mentioned in the introduction, our purpose is to explore the relation between strongly connected subgraphs. To perform this task we extract collections of set of vertices such that, with k, α, and γ three user defined positive integers, (1) all vertices are homogeneous, more precisely, they have at least α true-valued attributes in common, (2) the collection contains at least γ k-PCs and (3) all k-PCs showing the same true-valued attributes are in the collection. These patterns are defined more precisely as follows.

Definition 5 (Collection of Homogeneous k-PCs (CoHoP)). *Let k, α, and γ be three strictly positive integers, and $\mathcal{G}$ be an attributed graph. A collection M of sets of vertices is a CoHoP if and only if:*

- $|\textsc{Atb}(M)| \geq \alpha$ *(the vertices in M are homogeneous);*
- *M contains at least γ k-PCs, i.e., $|M| \geq \gamma$;*
- *M is the collection of all k-PCs in $\mathcal{G}[\![\textsc{Atb}(M)]\!]$, i.e., M contains* all *k-PCs sharing the attributes in* Atb*(M).*

Note that due to the constraint on homogeneity, a k-PC which is formed by vertices sharing less than α attributes cannot be part of a CoHoP.

3 Mining CoHoP Patterns

We first present a naive algorithm enumerating all subgraphs which might contain a pattern. Then we show how to safely reduce the subgraphs enumeration, and we describe the corresponding algorithm. Finally we describe implementation techniques.

Naive Algorithm

While Definition 5 is very declarative, we establish the following more constructive definition of the CoHoPs.

Lemma 1. *Let k, α, and γ be three strictly positive integers, and $\mathcal{G}$ be an attributed graph with $\mathcal{A}$ the set of Boolean attributes in $\mathcal{G}$. A collection M of sets of vertices is a CoHoP if and only if there exists $X \subseteq \mathcal{A}$ such that $M = \mathcal{C}_{kpc}(\mathcal{G}[\![X]\!])$, $|X| \geq \alpha$, and $|M| \geq \gamma$.*

Proof. First, consider a CoHoP M. By direct application of Definition 5, there exists $X = \text{ATB}(M) \subseteq \mathcal{A}$ such that $M = \mathcal{C}_{kpc}(\mathcal{G}[\![X]\!])$, $|X| \geq \alpha$, and $|M| \geq \gamma$. Now we prove the reciprocal. Consider X a set of attributes satisfying $|X| \geq \alpha$, and M a collection of sets of vertices such that $M = \mathcal{C}_{kpc}(\mathcal{G}[\![X]\!])$ and $|M| \geq \gamma$. Since $M = \mathcal{C}_{kpc}(\mathcal{G}[\![X]\!])$, then $X \subseteq \text{ATB}(M)$. So $\mathcal{G}[\![\text{ATB}(M)]\!]$ is a subgraph of $\mathcal{G}[\![X]\!]$ and since all vertices in M are also in $\mathcal{G}[\![\text{ATB}(M)]\!]$, we have $M = \mathcal{C}_{kpc}(\mathcal{G}[\![X]\!]) = \mathcal{C}_{kpc}(\mathcal{G}[\![\text{ATB}(M)]\!])$. Thus M is a CoHoP.

To compute all patterns, a naive algorithm can enumerate the subgraphs $\mathcal{G}_e = \mathcal{G}[\![X]\!]$ for all non empty set of attributes X, and for each $\mathcal{G}_e$ computes all k-PCs in $\mathcal{G}_e$. Then, if $|X| \geq \alpha$ and if there is at least γ k-PCs in $\mathcal{G}_e$, this collection of k-PCs is a CoHoP. From Lemma 1, this algorithm is correct. However, with this enumeration technique, $2^{|\mathcal{A}|} - 1$ subgraphs will have to be enumerated ($2^{|\mathcal{A}|} - 1$ non empty subsets of $\mathcal{A}$). The following lemmas are used to reduce the collection of subgraphs that has to be enumerated.

Reducing the Collection of Graphs to Be Enumerated

First, we introduce the notion of k-max-clique which is a clique having at least k vertices and not being a subset of any other clique. The collection of all k-max-cliques in an attributed graph $\mathcal{G}$ is denoted $\mathcal{C}_{kmax}(\mathcal{G})$.

The next lemma states that we can discard the attributed graphs that do not contain at least γ k-max-cliques, and also their subgraphs.

Lemma 2. *Let $\mathcal{G}$ be an attributed graph. If $\mathcal{G}$ does not contain at least γ k-max-cliques, then neither $\mathcal{G}$ nor any subgraph of $\mathcal{G}$ can contain a CoHoP.*

Proof. Let $\mathcal{G}$ be an attributed graph having less than γ k-max-cliques. Since all k-cliques in a k-max-clique are in the same k-PC, then the number of k-max-cliques cannot be greater than the number of k-PCs. So, $\mathcal{G}$ cannot contain γ k-PCs and thus cannot contain a CoHoP. The same holds for any subgraph of $\mathcal{G}$, since a subgraph of $\mathcal{G}$ cannot contain more k-max-cliques than $\mathcal{G}$.

According to the following lemma, we can avoid the enumeration of graphs (and their subgraphs) if they are induced by sets of attributes shared by not enough vertices to contain a CoHoP.

Lemma 3. *Let $\mathcal{G}$ be an attributed graph and x an attribute shared by less than k vertices in $\mathcal{G}$. Then, the graph $\mathcal{G}[\![\{x\}]\!]$ and all its subgraphs cannot contain a CoHoP.*

Proof. Straightforward since $\mathcal{G}[\![\{x\}]\!]$ contains less than k vertices.

The following property allows to reduce the set of vertices under consideration.

Lemma 4. *Let $\mathcal{G}$ be an attributed graph. Only vertices in a k-max-clique of $\mathcal{G}$ can form a CoHoP in $\mathcal{G}$ or in any subgraph of $\mathcal{G}$.*

Proof. Direct, as a vertex which is not in a k-max-clique cannot be in any k-PC.

Algorithm Description

A recursive function `FindCoHoP`, that takes advantage of Lemmas 2, 3, and 4 to prune the search space, is presented as Algorithm 1. The input of the algorithm for the first call is the whole attributed graph, i.e., $\mathcal{G}_e = \mathcal{G}$, and $\mathcal{A}_c$, the set of candidate attributes remaining under consideration to find attributes shared by subgraph, is $\mathcal{A}$.

Line 1 checks that there is at least γ k-max-cliques in $\mathcal{G}_e$. If it is not the case, from Lemma 2 no subgraph of $\mathcal{G}_e$ including $\mathcal{G}_e$ itself can contain a k-PC. **Line 2** computes the set of vertices which might contain a k-PC (i.e., $\mathcal{V}_r$) as the union of all k-max-cliques in $\mathcal{G}_e$ according to Lemma 4. **Line 3** checks (1) if there is at least α attributes shared by all vertices in $\mathcal{V}_r$ ($|\cap_{v \in \mathcal{V}_r} \mathcal{F}(v)| \geq \alpha$) and (2) if there is at least γ k-PCs ($|\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])| \geq \gamma$). If so, the collection of k-PCs is a CoHoP, and is output on **line 4**. On **line 5**, attributes from $\mathcal{A}_c$ shared by all vertices in $\mathcal{V}_r$ are removed from $\mathcal{A}_c$. Removing these attributes does not change the collection of enumerated subgraphs, since if we pick such an attribute x we have $\mathcal{G}_e[\mathcal{V}_r \cap vert(x)]$ that is equal to $\mathcal{G}_e[\mathcal{V}_r]$ itself in the recursive call to `FindCoHoP` (**line 9**). On **line 6**, attributes shared by less than k vertices in $\mathcal{V}_r$ are removed from $\mathcal{A}_c$, according to Lemma 3. This avoids unnecessary calls to `FindCoHoP` with subgraphs having not enough vertices. **Lines 7 to 9** perform a standard recursive enumeration scheme to produce in a depth first way, and element by element (the x that is picked), all subsets of $\mathcal{A}_c$. While $\mathcal{A}_c$ is not empty, an attribute x is picked (**line 8**) and function `FindCoHoP` is called with the subgraph of $\mathcal{G}_e$ induced by the set of vertices in $\mathcal{V}_r$ sharing attribute x, i.e., $\mathcal{G}_e[\mathcal{V}_r \cap vert(x)]$.

Algorithm 1. `FindCoHoP`

Input: $\mathcal{G}_e$, $\mathcal{A}_c$
1 **if** $|\mathcal{C}_{kmax}(\mathcal{G}_e)| \geq \gamma$ **then**
2 $\mathcal{V}_r = \cup_{C \in \mathcal{C}_{kmax}(\mathcal{G}_e)}$
3 **if** $|\cap_{v \in \mathcal{V}_r} \mathcal{F}(v)| \geq \alpha$ **and** $|\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])| \geq \gamma$ **then**
4 **output** $\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])$
5 $\mathcal{A}_c \leftarrow \{x \in \mathcal{A}_c \mid \mathcal{V}_r \nsubseteq vert(x)\}$
6 $\mathcal{A}_c \leftarrow \{x \in \mathcal{A}_c \mid |vert(x) \cap \mathcal{V}_r| \geq k\}$
7 **while** $\mathcal{A}_c \neq \emptyset$ **do**
8 Pick and remove an attribute x from $\mathcal{A}_c$
9 `FindCoHoP`($\mathcal{G}_e[\mathcal{V}_r \cap vert(x)]$, $\mathcal{A}_c$)

Theorem 1. *Algorithm 1 returns all CoHoP patterns and only CoHoP patterns.*

Proof. Lemma 1 and Lemmas 2 to 4 (safety of the pruning) ensure the completeness of Algorithm 1. Line 3 ensures its soundness.

Note that a given CoHoP might be output several times by Algorithm 1. Such duplicates are removed in a simple post-processing step.

Implementation
We give here some details about the implementation of Algorithm 1 used in the experiments presented in the next section.

The algorithm used to compute the collection of k-PCs in a graph is the one described in [11] and also used for instance in [3]. It first builds a matrix representing the adjacency relation between the k-cliques, and then compute the connected components of k-cliques (the k-PCs) using this matrix. The algorithm used to compute the k-max-cliques is CLIQUES [13]. Both the collection of k-max-cliques (i.e., $\mathcal{C}_{kmax}(\mathcal{G}_e)$) and the collection of k-PCs (i.e., $\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])$) are computed only once for a given attributed graph on respectively lines 1 and 3, and are reused on lines 2 and 4. Moreover, the computation of the k-PCs is done on line 3 only if the vertices in $\mathcal{V}_r$ have at least α attributes in common (i.e., $|\cap_{v\in\mathcal{V}_r}\mathcal{F}(v)| \geq \alpha$).

Finally, since vertices in a pattern must share at least one attribute ($\alpha \geq 1$), in general it is not necessary to compute the k-max-cliques of the whole graph. So, the first level of the enumeration is computed using only lines 6 to 9 of Algorithm 1, with $\mathcal{V}_r$ the set of all vertices of the input attributed graph.

4 Experiments

In this section we report experiments on three datasets built using real data: two bibliographic datasets (DBLP_1 and DBLP_2), and a biological dataset (BioData). The size and density of these datasets are presented in Table 1. All experiments were performed on a PC running GNU/Linux with a 3 GHz Core 2 Duo CPU and 8 GB of main memory installed (no more than 2 GB where used). We first describe the datasets, then, we illustrate the interest of the CoHoPs by mean of three typical examples of pattern found. Next, we discuss the performances of the algorithm and parameter setting.

Collaboration Network: DBLP_1 and DBLP_2 datasets have been built using the public DBLP database[1]. This database contains rather exhaustive bibliographic information on most computer science conferences and journals. We built our datasets using all conferences and journal up to august 2011. A vertex corresponds to an author and the attributes associated to a vertex are the conferences and journals in which the author has published. An edge between two authors represents the fact that they have coauthored some papers.

For DBLP_1 we wanted a large dataset to assess the performances of extraction algorithm. Consequently, in DBLP_1, we put an edge to represent each pair of coauthors, and for the attributes, an author is associated to all conferences and journals in which she/he has published. This led to a dataset containing $997,050$ authors and $5,963$ conferences/journals.

The dataset DBLP_2 was targeted to obtained more meaningful patterns. Thus, in DBLP_2, we focused on pairs of authors that have been collaborating more significantly, and we put an edge between two authors if they were coauthors of at least three articles. We also required a stronger relationship between authors

[1] `http://dblp.uni-trier.de/`

and conferences/journals. Indeed, we associated a conference or a journal to an author, only if this author has published at least three times in this conference/journal. Finally, in $DBLP_2$, authors that remain associated to no conference/journal (i.e., authors who have never published three times in the same conference/journal) were removed.

Protein Interaction Network: BioData has been built using two databases STRING[2] [5] and SQUAT[3] [6]. STRING integrates data on protein-protein interactions from different sources (e.g., genomic data, co-expression, experiments, literature). Among these interactions we only retained interactions with *confidence*[4] higher or equal to 400 (default STRING selection threshold). SQUAT is a public database of Boolean gene expression data resulting from SAGE experiments. SQUAT contains for thousands of genes, the sets of biological situations (termed *libraries*) where these genes are overexpressed. In our experiments, only *Human species* genes were used. We built the BioData dataset as follows. A vertex is a gene, and we put an edge between two genes if there was an interaction reported in STRING (confidence of at least 400) between the proteins corresponding to these two genes. The attributes associated to a gene were simply the biological situations in which the gene was overexpressed according to the SQUAT database. In our experiments, only Human species genes were used, this led to $15,571$ genes common to the two databases. For these genes we have expression data in SQUAT for 486 different biological situations.

Table 1. Size and density of datasets $DBLP_1$, $DBLP_2$, and BioData

	$DBLP_1$	$DBLP_2$	BioData
# Vertices	997,050	127,386	15,571
# Attributes	5,963	3,980	486
Avg. degree	6.88	3.69	20.01
Avg. attributes/vertex	3.06	2.15	11.46

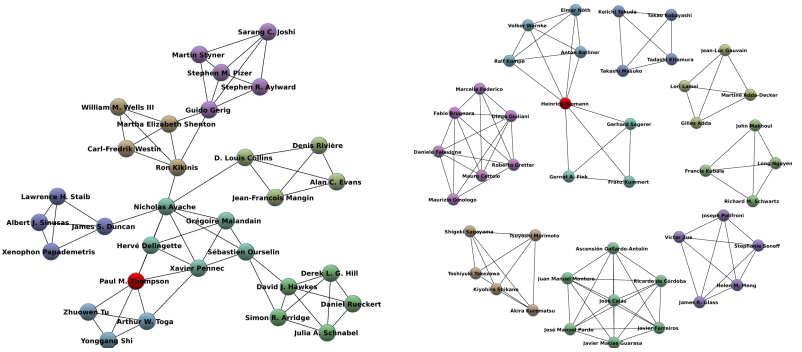
4.1 Illustration of the Interest of the Patterns

Collaboration Network: Let us first define some vocabulary in the context of a network of researchers. In [11] the authors consider that a k-PC is a community in the sense that "it consists of several complete subgraphs that tend to share many of their nodes". Consequently, we will use the term community for a k-PC. We will also say that two communities are connected if there is an edge between both communities. In $DBLP_2$, we searched for CoHoPs with at least seven 4-PCs were all authors have published in the same three conferences or journals

[2] `http://string-db.org/`

[3] `http://bsmc.insa-lyon.fr/squat/`

[4] This confidence is a measure provided by STRING. Low confidence means that there are not so many evidences that the interaction exists.



(a) Seven 4-PCs concerning conferences IPMI, ISBI, MICCAI and journal IEEE Trans. Med. Imaging.

(b) Nine 4-PCs concerning conferences INTERSPEECH, ICSLP, and EUROSPEECH.

Fig. 1. Two patterns extracted from DBLP_2 with $k = 4$, $\gamma = 7$, and $\alpha = 3$. Each color corresponds to a k-PC. Vertices in red are in several k-PCs

(i.e., $k = 4$, $\alpha = 3$ and $\gamma = 7$). With this parameter setting, 57 CoHoPs where extracted. To illustrate the kind of patterns that were retrieved, we focus on two patterns presented in Figures 1(a) and 1(b).

The pattern on Figure 1(a) contains seven 4-PCs, all authors having published in conferences or journals related to medical imaging. The authors N. Avache, H. Delingette, G. Malandain, S. Ourselin, X. Pennec, and P. M. Thompson are forming a community connected to all other communities except one and is the core of a star-based topology. Knowing such a structure is useful to make some decisions. For instance having researchers of the core community as partners in a project, or choosing this community as a destination for a post-doc position could be a great opportunity to benefit from contacts with all the other groups. We also investigated the role of the authors connecting two communities (i.e., the endpoints of edges connecting two communities) in this pattern using ArnetMiner[5]. We found that four of these *bridging nodes* [10] were advisor of at least half of the authors of their respective communities. So they are likely to be senior researchers and this is coherent with the fact that they appear as bridges between communities.

In the second CoHoP, presented on Figure 1(b), all authors have published at least three times in three conferences related to speech communication / spoken language. It contains nine communities, seven of them not being connected to any other. Moreover, from the personal page of the authors, we found out that in most cases a community is formed by people working in the same research institute. So, here most communities are formed by researchers working in the field of speech processing and not strongly publishing with researchers from other institutes. Such structure with disconnected groups of people sharing

[5] ArnetMiner (`http://arnetminer.org/`) is an application providing the relationship (e.g., coauthor, advisor, advisee) between researchers.

similar interests might be interesting for several tasks. For instance, it can give hints to funding organisms to set up long term development strategies of collaboration networks. Or it can also be helpful, in a normal day-to-day activity, like finding reviewers for a paper, by suggesting experts in the same domain as the authors, but having no closed collaborations (strong coauthor relationship) with these authors (and also eventually having no closed collaborations with the other experts).

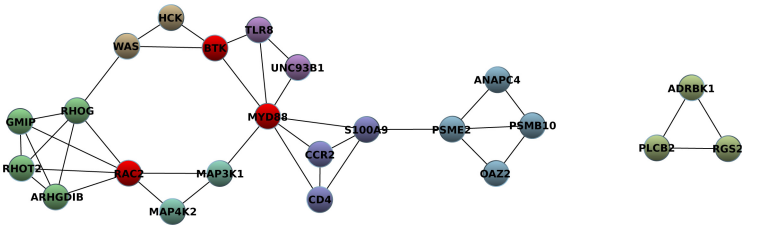


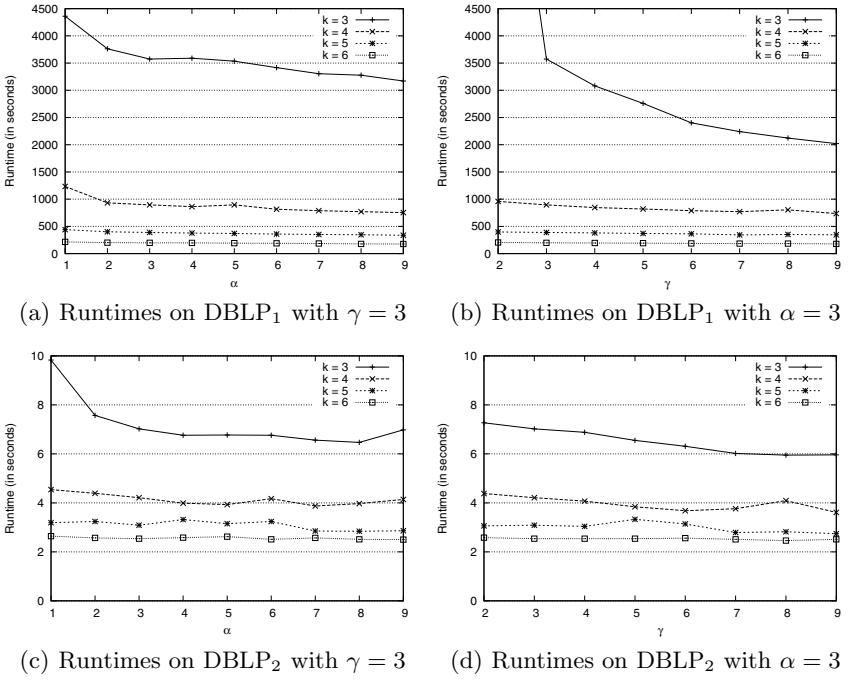
Fig. 2. A CoHoP extracted from BioData with $k = 3$, $\alpha = 4$, and $\gamma = 3$. Each color corresponds to a k-PC. Vertices in red are in several k-PCs.

Protein Interaction Network: In the BioData dataset, we searched for CoHoPs with at least three 3-PCs were all genes are overexpressed in at least four biological situations (i.e., $k = 3$, $\alpha = 4$, and $\gamma = 3$) and obtained 25 patterns. The CoHoP containing the greatest number of k-PCs is presented Figure 2. This CoHoP is composed of 7 k-PCs, and all vertices are genes overexpressed in 4 situations corresponding to normal white blood cells activities. The pattern contains (from left to right) a ring made of 4 groups of genes (4 k-PCs), with two other groups forming a tail link to the ring, and an extra isolated group. Such a structure suggest, among others, the following biological questions. Is there any order in the activation of the groups along the ring ? Do the groups forming the tail act as a trigger for the whole ring activity ? Are there some interactions between the isolated group and the others (while no such interaction is reported in STRING with a confidence of 400 or greater) ? All these questions can lead to interesting deeper investigations through wet biology experiments.

4.2 Performance Study

Figure 3 shows that the extraction can be made in less than 25 minutes when $k \geq 4$ on DBLP_1 and DBLP_2 (on BioData, all extractions made for similar settings were run in less than 10 seconds). We can also notice that in all settings, the runtime increases significantly when k decreases. The runtime increase when γ decreases, as shown on Figure 3(b), is mainly due to the computation of k-PCs from a large number of k-max-cliques.

The number of output patterns is given on Figure 4. As expected this number decreases when the values of k, α, and γ increase. Such curves can be used to help setting the extraction parameters. For instance, for communities, the literature [11,3] recommends to use a value of k between 3 and 6. So, for DBLP_2,



(a) Runtimes on $DBLP_1$ with $\gamma = 3$ (b) Runtimes on $DBLP_1$ with $\alpha = 3$

(c) Runtimes on $DBLP_2$ with $\gamma = 3$ (d) Runtimes on $DBLP_2$ with $\alpha = 3$

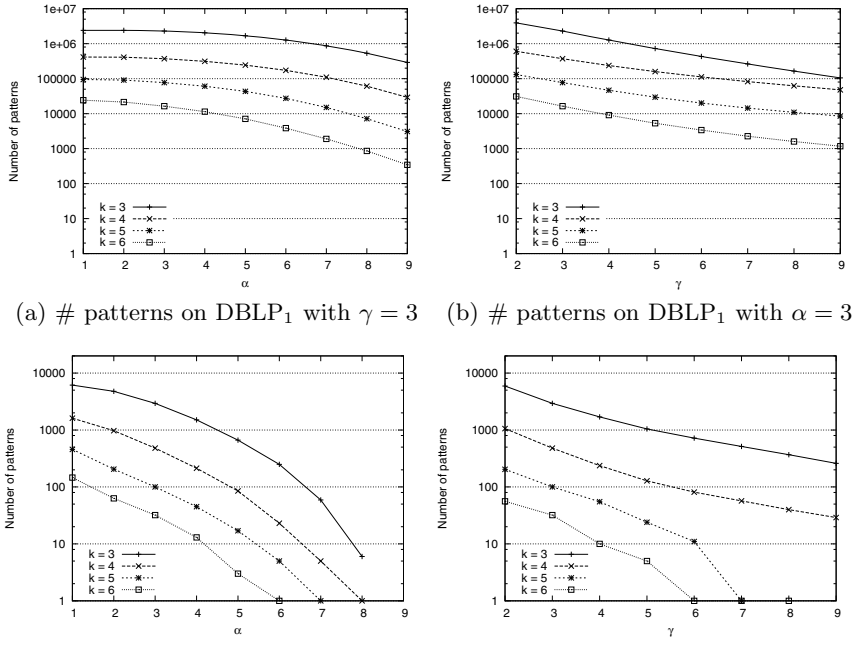
Fig. 3. Runtime for different sets of parameters on $DBLP_1$ and $DBLP_2$

since the running time is rather low, we could count the number of patterns for these values of k and a whole range of values of α and γ. Then we chose among these settings the ones that were meaningful and that lead to collections of patterns of reasonable size (for human browsing).

5 Related Work

Local pattern mining in attributed graphs to find homogeneous set of vertices is rather recent, and two main families of approaches have been developed.

In the first family [7,12], a pattern is a *single* densely connected subgraph (e.g., a quasi-clique) such that the vertices have homogeneous feature values. Such a pattern can reveal a module, a group or a community sharing similar properties or interests. A pattern in our approach is *set* of groups sharing similar attribute values, and thus exhibits a different kind of structures made of several groups (not a single one). Moreover the notion of group is also different, and corresponds for CoHoPs to an another well known form of communities, the k-clique percolated components. It should also be pointed out, that if the user is interested in extracting single groups, this can also be done, in the case of CoHoPs, by setting parameter γ to 1 and by outputting all k-clique percolated components as separated patterns.



(a) # patterns on DBLP_1 with $\gamma = 3$ (b) # patterns on DBLP_1 with $\alpha = 3$

(c) # patterns on DBLP_2 with $\gamma = 3$ (d) # patterns on DBLP_2 with $\alpha = 3$

Fig. 4. Number of patterns for different sets of parameters on DBLP_1 and DBLP_2

Our proposal is closer to the second family of approaches [2,9], where a pattern is a *collection* of set of vertices in a subgraph made of vertices sharing similar attribute values. These previous works adopt opposite views on the kind of structures they consider. In [9] the constraint on the structure of a group is very strong, since the sets of vertices must be cliques. On the contrary, in [2], the choice was made to be very tolerant, since a set of vertices is simply required to form a connected subgraph. We introduce in this paper a complementary approach, that exhibits another kind of group structures, namely the k-clique percolated components, that are typical group structures used in the literature to capture the notion of community.

6 Conclusion

In this paper, we considered graphs having a set of Boolean attributes associated to each vertex. We proposed to find Collection of Homogeneous k-clique Percolated components (CoHoP) and gave a sound and complete algorithm for this task. We shown by means of experiments on real datasets that the extractions can be made in practice and lead to meaningful patterns.

Acknowledgments. This work is partly funded by the Rhône-Alpes Complex Systems Institute (IXXI) through the REHMI project, and by the French National Research Agency (ANR) through the projects FOSTER (ANR-2010-COSI-012-02).

References

1. Derényi, I., Palla, G., Vicsek, T.: Clique percolation in random networks. Phys. Rev. Lett. 94, 160–202 (2005)
2. Fukuzaki, M., Seki, M., Kashima, H., Sese, J.: Finding Itemset-Sharing Patterns in a Large Itemset-Associated Graph. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part II. LNCS, vol. 6119, pp. 147–159. Springer, Heidelberg (2010)
3. Gao, W., Wong, K.-F., Xia, Y., Xu, R.: Clique Percolation Method for Finding Naturally Cohesive and Overlapping Document Clusters. In: Matsumoto, Y., Sproat, R.W., Wong, K.-F., Zhang, M. (eds.) ICCPOL 2006. LNCS (LNAI), vol. 4285, pp. 97–108. Springer, Heidelberg (2006)
4. Ge, R., Ester, M., Gao, B.J., Hu, Z., Bhattacharya, B., Ben-Moshe, B.: Joint cluster analysis of attribute data and relationship data: The connected k-center problem. ACM Trans. Knowl. Discov. Data (TKDD) 2(2), 1–35 (2008)
5. Jensen, L.J., Kuhn, M., Stark, M., Chaffron, S., Creevey, C., Muller, J., Doerks, T., Julien, P., Roth, A., Simonovic, M., Bork, P., von Mering, C.: STRING 8-a global view on proteins and their functional interactions in 630 organisms. Nucleic Acids Research 37, 412–416 (2009)
6. Leyritz, J., Schicklin, S., Blachon, S., Keime, C., Robardet, C., Boulicaut, J.F., Besson, J., Pensa, R.G., Gandrillon, O.: Squat: A web tool to mine human, murine and avian sage data. BMC Bioinformatics 9(1), 378 (2008)
7. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SIAM Data Mining Conf (SDM), pp. 593–604 (2009)
8. Moser, F., Ge, R., Ester, M.: Joint cluster analysis of attribute and relationship data without a-priori specification of the number of clusters. In: Int. Conf. on Knowledge Discovery and Data Mining (KDD), p. 510 (2007)
9. Mougel, P.N., Plantevit, M., Rigotti, C., Gandrillon, O., Boulicaut, J.F.: Constraint-Based Mining of Sets of Cliques Sharing Vertex Properties. In: Workshop on Analysis of Complex NEtworks (ACNE 2010) co-located with ECML/PKDD 2010 (2010)
10. Musiał, K., Juszczyszyn, K.: Properties of Bridge Nodes in Social Networks. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS, vol. 5796, pp. 357–364. Springer, Heidelberg (2009)
11. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043), 814–818 (2005)
12. Silva, A., Meira Jr., W., Zaki, M.J.: Structural correlation pattern mining for large graphs. In: Workshop on Mining and Learning with Graphs (MLG), pp. 119–126 (2010)
13. Tomita, E., Tanaka, A., Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. Theor. Comput. Sci. (TCS) 363, 28–42 (2006)
14. Tong, H., Gallagher, B., Faloutsos, C., Eliassi-rad, T.: Fast best-effort pattern matching in large attributed graphs. In: Int. Conf. on Knowledge Discovery and Data Mining, KDD (2007)
15. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. 2, 718–729 (2009)
16. Zhou, Y., Cheng, H., Yu, J.X.: Clustering large attributed graphs: An efficient incremental approach. In: Int. Conf. on Data Mining (ICDM), pp. 689–698 (2010)