Discovery-Driven Graph Summarization

Ning Zhang #1, Yuanyuan Tian *2, Jignesh M. Patel #3

#*Computer Sciences Department, University of Wisconsin-Madison, USA*
1nzhang@cs.wisc.edu 3jignesh@cs.wisc.edu

**IBM Almaden Research Center, USA*
2ytian@us.ibm.com

Abstract— **Large graph datasets are ubiquitous in many domains, including social networking and biology. Graph summarization techniques are crucial in such domains as they can assist in uncovering useful insights about the patterns hidden in the underlying data. One important type of graph summarization is to produce small and informative summaries based on user-selected node attributes and relationships, and allowing users to interactively drill-down or roll-up to navigate through summaries with different resolutions. However, two key components are missing from the previous work in this area that limit the use of this method in practice. First, the previous work only deals with categorical node attributes. Consequently, users have to manually bucketize numerical attributes based on domain knowledge, which is not always possible. Moreover, users often have to manually iterate through many resolutions of summaries to identify the most interesting ones. This paper addresses both these key issues to make the interactive graph summarization approach more useful in practice. We first present a method to automatically categorize numerical attributes values by exploiting the domain knowledge hidden inside the node attributes values and graph link structures. Furthermore, we propose an interestingness measure for graph summaries to point users to the potentially most insightful summaries. Using two real datasets, we demonstrate the effectiveness and efficiency of our techniques.**

I. Introduction

Many real world datasets can be modeled as graphs, where nodes represent objects and edges indicate relationships between nodes. Today, large amounts of graph data have been generated by various applications. For example, the popular social networking web site Facebook (`www.facebook.com`) contains a large network of registered users and their friendships. In September 2009, it has been reported that Facebook has over 300 million users [8]. As another example, the DBLP data [2] can be used to construct a network of authors connected by their coauthorships. This dataset, currently has about 700 thousand distinct authors. With the overwhelming wealth of information encoded in these graph datasets, there is a crucial need to summarize large graph datasets into concise forms that can be easily understood.

Graph summarization has attracted a lot of interest in the database community. Various graph summarization techniques have been proposed to help understand the characteristics of large graphs [15], [20]. An interactive graph summarization approach, called k-SNAP, is proposed in [20]. This approach produces summaries which themselves are also graphs, called *summary graphs*. In addition, it is amenable to an interactive querying scheme by allowing users to customize the summaries based on user-selected node attributes and relationships, and to control the resolutions of the resulting summaries. By controlling resolutions, users can interactively drill-down or roll-up to navigate through summaries with different resolutions.

Examples of the k-SNAP summaries is shown in Figure 1. This summary is constructed over a coauthorship network (sketched in Figure 1b), where each node represents an author and has a numerical attribute indicating the number of papers published by the author. Figure 1a and Figure 1c show two k-SNAP summary graphs with 5 nodes and 3 nodes, respectively. We call the number of nodes in a summary graph as the resolution or the size of the summary. In these summaries, the numeric attribute (number of publications for each author) has been “manually” mapped by the user to a categorical attribute with three values: HP, P and LP, which stand for highly prolific, prolific and low prolific, respectively. Each node in a summary graph represents a set of authors in the original data graph, and the edges in the summary graph capture the relationships between the groups of authors.

Although the k-SNAP summarization method provides useful features that can help users extract and understand the information encoded in large graphs, two key components are missing that limit the practical application of this technique in many cases.

First, the k-SNAP approach only deals with categorical node attributes. But in the real world, many node attributes are numerical, such as the age of a Facebook user or the number of publications of an author in a coauthorship network. Simply running the graph summarization method on the numerical attributes will result in summaries with large sizes (at least as large as the number of distinct numerical values). Due to the prevalence of numerical attributes in many graph datasets, it is natural to consider transforming these numeric attributes to categorical domains with smaller cardinalities, so that the resulting summaries are easier for a human to understand. Often, users have to draw upon their domain knowledge to provide the cutoffs to bucketize numerical values into categorical values. However, even for domain experts, providing such cutoffs can be tricky, and is not always possible (especially if the data is changing!). On the other hand, the graph data already contains information that can be used to “bucketize” the numeric domain based on the patterns of the attributes values and link structures of nodes in the graph. By leveraging such hidden knowledge, automatic categorization is possible.

The second missing component that affects the usability of

978-1-4244-5444-0/10/$26.00 © 2010 IEEE

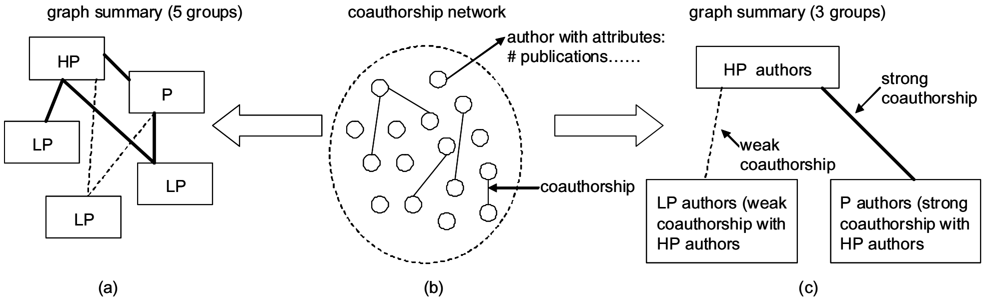


Fig. 1. Graph summary based on grouping nodes

k-SNAP, is the lack of an automatic method to direct the users to a small set of interesting summaries, from the set of all possible summaries. The k-SNAP approach provides users summaries with various resolutions. Exploring different resolutions of summaries is guided purely by user intuition. Users may have to go through a large number of summaries until some interesting summaries are found. Automatic discovery allows the system to focus the user's attention on a small subspace that contains the most promising/interesting summaries. (Within this subspace, the OLAP-style navigation of k-SNAP can be used to navigate through a far smaller set the summaries.) This approach can significantly improve the usability of the graph summarization method.

In this paper, we address both these key issues, to further improve the usability of the k-SNAP graph summarization method. The main contributions of this paper are:

- We introduce a method, called CANAL that automatically categorizes numerical attributes values based on both the attributes values and the link structures of nodes in the graph.
- To guide the automatic discovery of interesting summaries, we propose a measure to assess the interestingness of summaries.
- We apply our techniques to two well-known datasets, namely DBLP [2] and CiteSeer [1], and demonstrate the effectiveness and efficiency of our methods.

The remainder of this paper is organized as follows: In Section II, we present some background information. The CANAL method is introduced in Section III. Section IV discusses the interestingness measure for graph summaries, and Section V contains experimental results. Related work is described in Section VI, and Section VII contains our conclusions.

II. Preliminaries

In this paper, we consider a general graph model where nodes in the graph have arbitrary number of associated attributes, and are connected by multiple types of edges. More formally, a graph is denoted as $G = (V, E)$, where V is the set of nodes and E is the set of edges. The set of attributes associated with the nodes is denoted as $A = \{a_1, a_2, ..., a_m\}$. We require that each node $v \in V$ has a value for every attribute in A. The set of edge types present in the graph is denoted as $T = \{t_1, t_2, ..., t_n\}$. Each edge $(u, v) \in E$ can be marked by a non-trivial subset of edge types denoted as $T(u, v)$ ($\emptyset \subset T(u, v) \subseteq T$). For example, in a graph of scientific researchers connected by coauthorships and colleague relationships (i.e. working in the same organization), some pairs of researchers (nodes) are only coauthors or only colleagues, while others can be both coauthors and colleagues.

This paper focuses on a special kind of graph summary, which themselves are also graphs. These graphs are more compact in size and provide valuable insight into the characteristics of the original graphs. We call them *summary graphs*.

We can formally define summary graphs as follows: Given a graph $G = (V, E)$, and a partition of V, $\Phi = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_k\}$ ($\bigcup_{i=1}^{k} \mathcal{G}_i = V$ and $\forall i \neq j \ \mathcal{G}_i \cap \mathcal{G}_j = \emptyset$), the *summary graph* based on Φ is $S = (V_S, E_S)$, where $V_S = \Phi$, and $E_S = \{(\mathcal{G}_i, \mathcal{G}_j) | \exists u \in \mathcal{G}_i, v \in \mathcal{G}_j, (u, v) \in E\}$. The set of edge types for each $(\mathcal{G}_i, \mathcal{G}_j) \in E_S$ is defined as $T(\mathcal{G}_i, \mathcal{G}_j) = \bigcup_{(u,v) \in E, u \in \mathcal{G}_i, v \in \mathcal{G}_j} T(u, v)$.

In a summary graph, each node, called a *group*, corresponds to one group in the partition of the original node set, and each edge, called a *group relationship*, represents the connections between two corresponding sets of nodes. A group relationship between two groups exists if and only if there exists at least one edge connecting some nodes in the two groups. The set of edge types for a group relationship is the union of all the types of the corresponding edges connecting nodes in the two groups.

Note that the methods introduced in this paper are applicable to both directed and undirected graphs. For ease of presentation, we only consider undirected graphs in this paper. (Our actual implementation supports directed graphs and in fact we present results for directed graphs in Section V.)

A. k-SNAP Summarization Approach

We now briefly describe the k-SNAP summarization method proposed in [20]. Our work in this paper builds on the k-SNAP summarization approach, providing two crucial components to significantly improve the usability k-SNAP – namely, attribute categorization and measuring the interestingness of summaries.

Given a user-specified resolution k, the k-SNAP summarization method produces a summary graph with size k (i.e. containing k groups) by grouping nodes based on user-selected node attributes and relationships. In the summary graph, nodes in each group are required to have the *same* value for each user-selected attribute(s), and they connect to nodes in a *similar* set of groups.

Given a resolution k, there could be many different summaries with the same size. A Δ-measure is proposed in [20] to assess the quality of a k-SNAP summary by examining how different it is to a hypothetical "ideal summary". First, we define the set of nodes in group $\mathcal{G}_i$ that participates in a group relationship $(\mathcal{G}_i, \mathcal{G}_j)$ as $P_{\mathcal{G}_j}(\mathcal{G}_i) = \{u | u \in \mathcal{G}_i$ and $\exists v \in \mathcal{G}_j$ s.t. $(u, v) \in E\}$. Then we define the participation ratio of the group relationship $(\mathcal{G}_i, \mathcal{G}_j)$ as $p_{i,j} = \frac{|P_{\mathcal{G}_j}(\mathcal{G}_i)| + |P_{\mathcal{G}_i}(\mathcal{G}_j)|}{|\mathcal{G}_i| + |\mathcal{G}_j|}$. For a group relationship, if its participation ratio is greater than 50%, then we call it a *strong* group relationship, otherwise, we call it a *weak* group relationship. Note that in an "ideal" summary, the participation ratios are either 100% or 0%.

Given a graph G, the Δ-measure of a grouping of nodes $\Phi = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_k\}$ is defined as follows:

$$\Delta(\Phi) = \sum_{\mathcal{G}_i, \mathcal{G}_j \in \Phi} (\delta_{\mathcal{G}_j}(\mathcal{G}_i) + \delta_{\mathcal{G}_i}(\mathcal{G}_j)) \tag{1}$$

where,

$$\delta_{\mathcal{G}_j}(\mathcal{G}_i) = \begin{cases} |P_{\mathcal{G}_j}(\mathcal{G}_i)| & \text{if } p_{i,j} \leq 0.5 \\ |\mathcal{G}_i| - |P_{\mathcal{G}_j}(\mathcal{G}_i)| & \text{otherwise} \end{cases} \tag{2}$$

Note that if the graph contains multiple types of relationships, then the Δ value for each edge type is aggregated as the final Δ value.

The Δ measure looks at each pairwise group relationship: If this group relationship is weak ($p_{i,k} \leq 0.5$), then it counts the participation differences between this weak relationship and a non-relationship ($p_{i,k} = 0$). On the other hand, if the group relationship is strong, then it counts the differences between this strong relationship and a 100% participation-ratio group relationship. The δ function, defined in Equation 2, evaluates the part of the Δ value contributed by a group $\mathcal{G}_i$ with one of its neighbors $\mathcal{G}_j$ in a group relationship.

Given the Δ-measure, and the user specified summary resolution k (i.e. number of groups in the summary is k), the goal of the *k-SNAP* operation is to find the summary of size k with the best quality. However, this is an NP-Complete problem [20], and a heuristic algorithm, called top-down k-SNAP, is proposed to produce suboptimal solutions. This top-down k-SNAP approach first partitions nodes based only on user-selected node attributes, and then iteratively splits existing groups until the number of groups reaches k. Note that the computation of the Δ measure can be broken down into components for each group with each of its neighbors (see the δ function in Equation 2). Consequently, at each iterative step, the heuristic chooses the group with the largest δ value with one of its neighboring groups, and then splits it based on whether nodes in this group connect to nodes in the neighboring group.

III. Categorization of Numerical Attributes

The k-SNAP graph summarization method only works when cardinalities of attribute domains are small. In a k-SNAP summary, nodes in each group are required to have the same value for each user-selected attributes. As a result, a summary has at least as many groups as the number of distinct attribute values. But many attributes of nodes in real graph datasets are numerical, such as the number of citations of an article in a citation network. The number of distinct values of a numerical attribute is often large. If we naively apply the k-SNAP summarization method on numerical values, the size of the summary can be very large, and hence of limited use.

For example, in a citation network with 196,015 articles connected by 823,432 citations generated from the CiteSeer dataset [1], the attribute "number of citations" has 423 distinct values. This means that a summary graph on this citation network has at least 423 groups. Figure 2 shows this summary, which has 423 groups and 28,307 group relationships. As is evident, this summary is overwhelming.

In contrast, by bucketizing the numerical values into 3 categories, LC (Low Cited: ≤ 3 citations), MC (Moderately Cited: ≥ 4 and ≤ 8 citations) and HC (Highly Cited: ≥ 9 citations), more compact summaries can be generated to help users understand the characteristics of the citation network. Figure 3 shows three summaries with resolutions 3, 4 and 5 based on this categorical attribute and the citation relationship. In this figure, groups with the LC value are colored in white, groups with the MC value are colored in light blue, and the groups with the HC value are in yellow. The bold edges between two groups indicate strong group relationships (with more than 50% participation ratio), while dashed edges are weak group relationships. (We analyze these summaries in more detail in Section V).

The categorization of numerical attribute values in the above example produces compact and informative summaries, but obtaining the right cutoffs for the categorization is very hard in practice. Experts can draw upon domain knowledge to decide such cutoffs. But this manual selection of cutoffs is not always possible, especially if the data is changing. To solve this problem, we propose a technique to automatically categorize numerical values by exploiting the similarities of attribute values and link structures of nodes in the graph. We call this technique CANAL (**C**ategorization of **A**ttributes with **N**umerical Values based on **A**ttribute Values and **L**ink Structures of Nodes). In fact, the categories in the above example are automatically generated by the CANAL algorithm.

Next, we discuss the CANAL algorithm in detail. For ease of presentation, the methods discussed in this paper are assumed to work on one attribute and one type of relationship; extending these methods to multiple attributes and relationships is straightforward, and omitted in the interest of space.

A. *Overview of the CANAL Algorithm*

Given a graph $G = (V, E)$, a numerical node attribute a and a desired number of categories C, the CANAL algorithm automatically finds $C - 1$ cutoffs to categorize the values of attribute a for all the nodes in G, such that the cutoffs result in high-quality summaries.

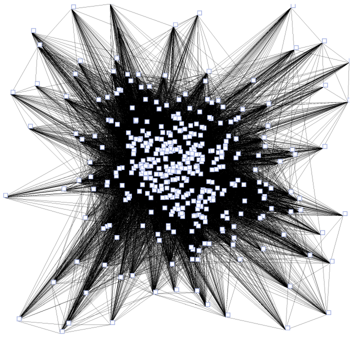
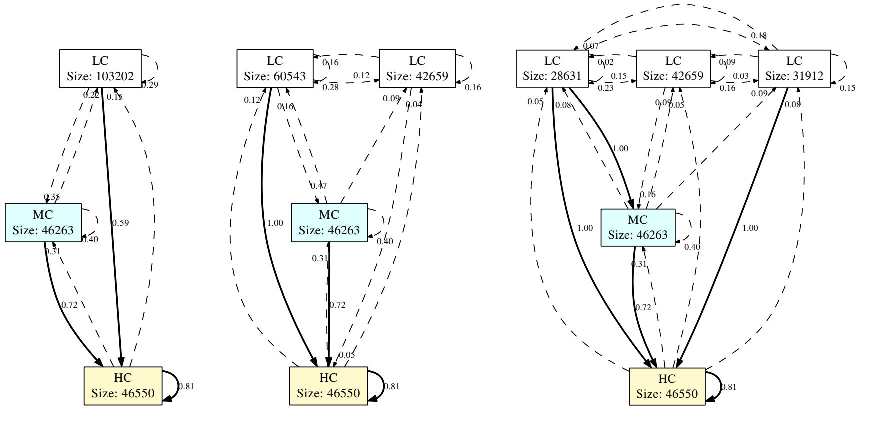


Fig. 2. A complex graph summary without categorization of numerical attributes



(a) 3 groups

(b) 4 groups

(c) 5 groups

Fig. 3. Graph summaries for the CiteSeer dataset

The CANAL algorithm operates in three phases, as outlined in Algorithm 1.

In the first phase, the CANAL algorithm groups nodes based on the attribute values of all the nodes in G. Nodes with the same value belong to the same group. Note that although the domain of a numerical attribute can be uncountable, such as real numbers, the number of distinct values in a particular graph is bounded by the number of nodes in the graph.

We can define the attribute range of a group $\mathcal{G}_i$ as $a(\mathcal{G}_i) = [s_i, t_i]$, where $s_i = \min_{v \in \mathcal{G}_i}\{a(v)\}$ and $t_i = \max_{v \in \mathcal{G}_i}\{a(v)\}$. In other words, the attribute range of a group is defined by the minimum attribute value and the maximum attribute value of nodes in the group. Since the initial groups are formed based on the values of attribute a, $s_i = t_i$ for each group initially. We impose an order on the groups. Given two groups $\mathcal{G}_i$ and $\mathcal{G}_j$, if $t_i < s_j$, then we order $\mathcal{G}_i$ before $\mathcal{G}_j$, denoted as $\mathcal{G}_i \prec \mathcal{G}_j$.

If $\mathcal{G}_i \prec \mathcal{G}_j$ and there exists no $\mathcal{G}_k$ such that $\mathcal{G}_i \prec \mathcal{G}_k \prec \mathcal{G}_j$, then we order $\mathcal{G}_i$ immediately before $\mathcal{G}_j$, denoted as $\mathcal{G}_i \rightsquigarrow \mathcal{G}_j$. Furthermore, we call $\mathcal{G}_i$ and $\mathcal{G}_j$ *value-adjacent* to each other if $\mathcal{G}_i \rightsquigarrow \mathcal{G}_j$ or $\mathcal{G}_j \rightsquigarrow \mathcal{G}_i$.

The second phase of the CANAL algorithm iteratively merges two value-adjacent groups based on the similarities of the link structures, until there is only one group left. When merging, the CANAL algorithm also monitors the changes of the quality of the summary corresponding to the grouping. If merging two value-adjacent groups significantly worsens the quality of the resulting summary, then the boundary between the two merged groups serves as good candidate for a cutoff of the attribute range.

In the final third phase, CANAL selects the top $C-1$ merges that cause the most deterioration in the summary quality and uses the corresponding boundaries as the cutoffs to categorize the numerical attribute.

Next we discuss in detail how CANAL decides which groups to merge at each iteration of the second phase (Section III-B), and how the algorithm measures the deterioration of summary quality and chooses the cutoffs in the third phase (Section III-C).

Algorithm 1 CANAL Algorithm

```
First Phase: Initialization
INPUT: A graph $G$, numerical attribute $a$ and the number of categories $C$.
Group nodes in $G$ based on numeric values of $a$
Sort the groups to obtain the value-adjacent groups
Maintain a heap for pairs of value-adjacent groups, sorted by $SimLink$
Pre-compute $\Delta_p$ for the current grouping, where $p$ is initialized to 0
Second Phase: Merge groups
while the $SimLink$ heap is not empty do
    Pop the pair of value-adjacent groups $\mathcal{G}_i$ and $\mathcal{G}_j$ with the smallest $SimLink$ values from the heap
    Merge $\mathcal{G}_i$ and $\mathcal{G}_j$, calculate $\mu_p$, and put $\mu_p$, $\mathcal{G}_i$ and $\mathcal{G}_j$ into $\mu_p$ heap
    Update the $SimLink$ heap
end while
Third Phase: Determine the $C-1$ cut-offs
for $C-1$ iterations do
    Pop the largest $\mu_p$ and its associated groups $\mathcal{G}_i$ and $\mathcal{G}_j$ from $\mu_p$ heap
    Pick the boundary of $\mathcal{G}_i$ and $\mathcal{G}_j$ as a cutoff , and save in an array $O$
end for
OUTPUT: The array $O$, which has the $C-1$ cutoffs for the attribute $a$.
```

B. Merging Groups

The goal of the CANAL algorithm is to exploit the domain knowledge hidden inside the attribute values and link structures of nodes in the graph, to automatically categorize numerical attributes. The merging of groups is guided by similarities of both attribute values and link structures of nodes in the graph. In other words, two groups of nodes with similar attribute values and link structures are likely to be put in a same category of the numerical attributes. Since similarities on attribute values have already been crudely captured in the definition of value-adjacent groups, CANAL can only merge groups that are value-adjacent to each other. In addition, CANAL requires that the groups to be merged should also have similar neighbor groups with similar participation ratios.

To characterize the link structural similarities between two value-adjacent groups, we define a measure called $SimLink$ as follows:

$$SimLink(\mathcal{G}_i, \mathcal{G}_j) = \sum_{i,j \neq k} |p_{i,k} - p_{j,k}|$$

Intuitively, $SimLink(\mathcal{G}_i, \mathcal{G}_j)$ accumulates the differences in participation ratio between the value-adjacent groups $\mathcal{G}_i$ and $\mathcal{G}_j$ with other groups. The smaller this value is, the more similar the two groups are. Our implementation of CANAL uses a heap to store pairs of groups based on the $SimLink(\mathcal{G}_i, \mathcal{G}_j)$ value. At each iteration, we pop the element from the heap corresponding to the two value-adjacent groups $\mathcal{G}_i$ and $\mathcal{G}_j$ with the smallest $SimLink(\mathcal{G}_i, \mathcal{G}_j)$ value, and merge the pair into one group. At the end of each iteration, we remove the heap elements (pairs of groups) involving either of the two merged groups, update elements involving the neighbors of the two merged groups, and insert new elements involving this new group. These algorithmic steps are shown in the second phase of Algorithm 1.

Note that the $SimLink$ measure is very similar to the $MergeDist$ measure introduced in [20]. However, the two measures are used for two different purposes. The $SimLink$ measure is used to automatically find the cutoffs to categorize numerical values, while the $MergeDist$ is used to produce summary graphs. Furthermore, $SimLink$ is only defined on value-adjacent groups, while $MergeDist$ is defined for any pair of groups.

C. Selecting Cutoffs

In the second phase of the algorithm, each generated group corresponds to a node in the summary graph. As the merge progresses, CANAL also monitors the changes in the qualities of these resulting summaries. The goal is to catch the merges of two groups that cause the most deterioration in the quality scores. These groups provide hints on how to divide the numerical attribute into categories.

First, we need to quantitatively measure the deterioration in the summary quality caused by a merge. We adopt the Δ quality measure of summaries in [20]. The brief description of this quality measure can be found in Section II.

We define μ_p as the relative Δ change between two grouping Φ and Φ', where Φ is the grouping before the pth merge and Φ' is the grouping after the pth merge. μ_p is defined as:

$$\mu_p = \frac{\Delta_p - \Delta_{p-1}}{\Delta_{p-1}}$$

Here, Δ_p is the shorthand of the Δ value of the grouping Φ after the pth merge, where p is bounded by $|\Phi| - 1$. Note that μ_p reflects the ratio of Δ changes from before and after the pth merge.

In each iteration of the second phase, we keep track of the μ_p value as well as the merged $\mathcal{G}_i$ and $\mathcal{G}_j$ in another heap, which is sorted by μ_p. When the second phase finishes, we return and remove the current largest μ_p value and its associated groups $\mathcal{G}_i$ and $\mathcal{G}_j$. The boundary between the two value-adjacent groups $\mathcal{G}_i$ and $\mathcal{G}_j$ (the boundary can be either t_i or s_j assuming $\mathcal{G}_i \rightsquigarrow \mathcal{G}_j$) is considered as one cutoff. The process of returning the largest μ_p and deciding cutoff is repeated $C-1$ times. Then, the resulting $C-1$ cutoffs can be used to categorize the numerical values. The details of this step are shown in the third phase of Algorithm 1.

The time complexity of CANAL algorithm is the cost of initialization, merging groups, and determining the $C-1$ cutoffs. The algorithm takes at most $O(|V|\log|V|)$ time for sorting and grouping the attribute values, and $O(k_a^2)$ time for initializing the heaps, where k_a is the number of distinct values of the numerical attribute a for all the nodes in the graph. Note that k_a is bounded by $|V|$, but in practice is orders of magnitude smaller than $|V|$. (For example, in the CiteSeer citation network [1], $|V| = 823,432$ whereas $k_a = 423$.) In each iteration of the second phase, the algorithm needs at most $O(k_i^3 + k_i^2 \log k_i)$ steps to update the heaps, where k_i is the size of the current grouping at each iteration. The total number of iteration in the second phase is bounded by k_a. The determination of $C-1$ cutoffs only takes at most $O(C \log k_a)$ time. To sum up, the time complexity of the CANAL algorithm is bounded by $O(|V|\log|V| + k_a^4)$.

Note that CANAL is a pre-computation step. In practice, CANAL is only executed once to categorize the numerical values, then different summaries can be generated without re-running the CANAL algorithm again. (CANAL has to be rerun if the underlying data has changed significantly.) Furthermore, even choosing a different C value does not need a complete rerun of the CANAL algorithm – since all the potential cutoff candidates are maintained in a heap structure, CANAL just needs to return a different number of top cutoffs from the heap in the third phase of the algorithm. In addition, as we show in Section V-C, CANAL is very efficient, and hence is unlikely be a bottleneck for generating summaries.

IV. Automatic Discovery of Interesting Summaries

In this section, we propose a measure to assess the interestingness of a summary and discuss how to use this measure to automatically discover interesting summaries, allowing users to focus on a small number of summaries.

A. Measuring Interestingness of Summaries

The essence of graph summarization is to capture the high-level structural characteristics of the original graphs by extracting the dominant relationships amongst groups of nodes. A strong group relationship in a summary exhibits a frequent relationship pattern between two groups of nodes in the original graph. Leveraging the work from the data mining community [3], [9], [11], we now define the interestingness of a summary graph.

Note that in the following definition of the interestingness measure, we omit the weak group relationships, since they do not exhibit frequent patterns in the link structures and are less important.

Intuitively, the interestingness of a summary graph has the following three aspects:

- *Diversity*: Strong relationships between groups with different attribute values reveal more insights into the original graph, thus make a summary more informative.
- *Coverage*: If more nodes participate in strong group relationships, the resulting summary is more comprehensive.
- *Conciseness*: Summaries with fewer groups and strong group relationships are more concise, and hence are easier to understand and visualize.

Next, we will discuss each of these aspects in detail.

1) Diversity: A summary graph S is considered more diverse, if more strong group relationships in S are connecting groups with different categorical attribute values. A diverse summary is interesting because in the absence of background knowledge, a user often wants to see more relationships between groups with diverse features (e.g. the interactions between researchers with very different prolificacies). A formal definition of diversity is as follows:

$$Diversity(S) = \frac{|DPC(S)|}{C}$$

where C is the cardinality of the categorical attribute in S and $DPC(S)$ is the set of distinct pair of categorical values connected by one or more strong group relationships in S. Note that $|DPC(S)|$ is bounded by the number of all possible combinations of two different categorical values. When C increases, $|DPC(S)|$ usually increases as well. To reduce this bias, we divide $|DPC(S)|$ by C.

The rationale behind the above formula is: (1) We are largely interested in strong group relationships which reflect frequent structural patterns in the original graph. (2) We only pick the strong relationships across groups with different attribute values, since strong relationships between groups with the same attribute values are pervasive in many cases and can be expected by common knowledge. (3) Nodes with the same categorical values are often split into different groups in a summary. As a result, there may be more than one strong group relationships connecting different categorical values. In the $Diversity$ definition, we only count the distinct pair of categorical values that are connected by one or more strong group relationships.

2) Coverage: The coverage of a graph summary measures the comprehensiveness of strong group relationships, that is, the fraction of nodes in the original graph that are present in these strong group relationships. Now, we formally define the coverage of a summary graph S as follows:

$$Coverage(S) = \sum_{(\mathcal{G}_i,\mathcal{G}_j)\in SR(S)} \frac{|P_{\mathcal{G}_j}(\mathcal{G}_i)| + |P_{\mathcal{G}_i}(\mathcal{G}_j)|}{|V|}$$

where $SR(S)$ is the set of strong group relationships in S, and $P_{\mathcal{G}_j}(\mathcal{G}_i)$ is the set of nodes in group $\mathcal{G}_i$ that participate in a group relationship $(\mathcal{G}_i, \mathcal{G}_j)$.

3) Conciseness: A summary graph is concise if it contains relatively few groups and strong group relationships. A concise summary graph is easy to understand and visualize. Intuitively, the conciseness of a summary graph S is:

$$Conciseness(S) = |V_S| + |SR(S)|$$

where V_S is the set of groups in S and $SR(S)$ is the set of strong group relationships in S.

4) Putting it Together: To put the above three aspects together, we define the interestingness of a summary graph as:

$$Interestingness(S) = \frac{Diversity(S) \times Coverage(S)}{Conciseness(S)}$$

The product of diversity and coverage generally indicate the informativeness of a given summary. Large summaries usually tend to be more informative, but may not be usable, due to the lack of conciseness. The interestingness measure defined above aims to make a tradeoff between the two.

We acknowledge that the proposed interestingness measure is not perfect. However, quantitative assessment of graph summary interestingness is a hard problem. Although crude, the proposed measure intuitively captures the different aspects of the interestingness, and is a significant step towards the goal of automatic discovery of interesting summaries. We expect that future work in this area has the potential to improve this measure and/or produce better measures for specific domains.

B. Automatic Discovery of Interesting Summaries

The k-SNAP summarization technique allows generating numerous summaries with different resolutions efficiently [20]. However, users have to manually navigate through these numerous summaries to find out the interesting ones. Now, the proposed interestingness measure can help users automatically narrow down the search space to a very small subset of high scoring summaries. Within this space, they can examine a few selected summaries, and/or use the OLAP-style of drill-down or roll-up navigation provided by k-SNAP .

In Section V-E, we will demonstrate how to make use of the interestingness measure defined above to discover interesting summaries in two real applications.

V. Experimental Results

In this section, we present experimental results evaluating the effectiveness and efficiency of our methods on two real datasets, namely the DBLP and the CiteSeer datasets. All algorithms are implemented in C++. The graph data is stored in a node table and an edge table in a PostgreSQL database. Accesses to nodes and edges are implemented by issuing SQL queries to the database. At the start of summarization process, the graph data is fetched and cached in memory. This process of caching the graph data takes 6 seconds on the largest dataset that we use. In the results below, we do not include this

TABLE I

CITESEER DATASETS

	Description	# Nodes	# Edges	Avg. Degree
D50k	*CiteNum* ≥ 10	46,550	125,012	2.68
D100k	*CiteNum* ≥ 5	92,813	233,020	2.51
D150k	*CiteNum* ≥ 2	143,887	330,997	2.30
D200k	*CiteNum* ≥ 1	196,015	823,432	4.20

TABLE II

MANUAL 2-CUTOFFS AND CANAL CUTOFFS FOR THE DBLP DB DATASET

	Cutoffs on *PubNum*
Manual 2-cutoffs	[1,5] [6,20] [$\geq$ 21]
CANAL 2-cutoffs	[1,3] [4,10] [$\geq$ 11]
CANAL 3-cutoffs	[1] [2,3] [4,10] [$\geq$ 11]
CANAL 4-cutoffs	[1] [2,3] [4,10] [11,26] [$\geq$ 27]

TABLE III

CANAL CUTOFFS FOR THE CITESEER D200K DATASET

	Cutoffs on *CiteNum*
CANAL 2-cutoffs	[1,3] [4,8] [$\geq$ 9]
CANAL 3-cutoffs	[1,3] [4,8] [9,12] [$\geq$ 13]
CANAL 4-cutoffs	[1,3] [4,5] [6,8] [9,12] [$\geq$ 13]

caching time, since this cost is easily amortized over several summary generations. All experiments were run on a Linux server with 2.66GHz Intel Core-2 Quad processor and 8GB RAM.

We first describe the datasets used in our empirical evaluation. Then, we present results evaluating the effectiveness and efficiency of the CANAL algorithm, and the effectiveness of the interestingness measure introduced in Section IV-A. After that, we use two example applications to demonstrate that summaries produced by the proposed methods can be used to conduct interesting analysis on real graph data.

A. Datasets

DBLP DB Dataset This dataset is constructed from the DBLP Bibliography data [2] by choosing publications from selected database conferences and journals[1]. An undirected coauthorship graph is formed from this dataset, with nodes corresponding to authors and edges indicating coauthorships. Each node has an attribute, called *PubNum*, which is the number of publications belonging to the corresponding author. This coauthorship graph contains 7,445 nodes and 19,971 edges.

CiteSeer Dataset We construct citation graphs from the CiteSeer dataset [1]. Citation graphs are *directed* graphs, where nodes denote articles and a directed edge $X \rightarrow Y$ indicates that article X cites article Y. Each node in a citation graph has an attribute, called *CiteNum*, representing the number of citations for the corresponding article. For scalability experiments, we partition the CiteSeer dataset and construct four datasets with increasing sizes. The characteristics of these datasets are shown in Table I. These four datasets are constructed as follows: dataset D50k contains articles with *CiteNum* ≥ 10. Dataset D100k includes articles with *CiteNum* ≥ 5. Articles with *CiteNum* ≥ 2 belong to the dataset D150k, and finally, dataset D200k contains articles with *CiteNum* ≥ 1.

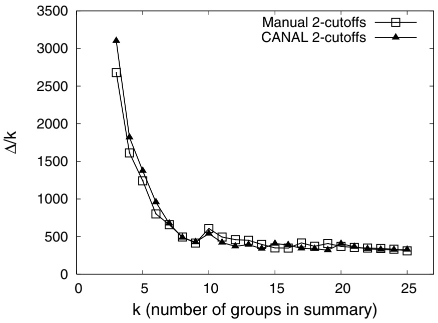


Fig. 4. Comparison between the Manual 2-cutoffs and the CANAL 2-cutoffs on the DBLP DB dataset

B. Effectiveness of The CANAL Algorithm

We first evaluate the effectiveness of the CANAL algorithm when categorizing numerical attributes, by comparing the cutoffs generated by CANAL to the manually selected cutoffs in [20] for analyzing the DBLP DB coauthorship graph dataset.

As described in Section V-A, each author (node) in the coauthorship graph has a numerical attribute, called *PubNum*, indicating the number of publications for the corresponding author. To summarize the graph, this numerical attribute was categorized into three categories in the previous work [20] based on domain knowledge and the authors' familiarity with this dataset. The manually selected cutoffs are shown in the first row of Table II.

In [20], the measure $\frac{\Delta}{k}$ was used to compare the qualities of summaries with different resolutions, where Δ is the value of the Δ-measure described in Section II-A, and k is the number of groups in the summary. With the $\frac{\Delta}{k}$ measure, for a fixed k, a smaller $\frac{\Delta}{k}$ value implies that the summary is of a higher quality, which in turn indirectly indicates better cutoffs. The cutoffs in [20] (we call them *Manual 2-cutoffs*) are manually tuned to achieve good $\frac{\Delta}{k}$ value. Note that as mentioned in [20] $\frac{\Delta}{k}$ is not claimed to be a perfect measure of summary quality. To facilitate direct comparison with this previous work, here we used the same $\frac{\Delta}{k}$ measure to compare the cutoffs generated by CANAL with the manually selected cutoffs. As our results (presented below) show, the cutoffs produced by CANAL (we call them *CANAL cutoffs*) results in $\frac{\Delta}{k}$ values that are very close to the manually selected ones.

For this experiment, we use CANAL to automatically generate 2 cutoffs for the DBLP DB dataset. The second row of Table II shows details about the resulting CANAL 2-cutoffs. Then, we run the same k-SNAP summarization algorithm proposed in [20] using the CANAL 2-cutoffs as well as the manually selected cutoffs to compare the qualities

[1] We selected papers longer than 4 papers from VLDB J., TODS, KDD, PODS, VLDB and SIGMOD. We included "long" industrial papers in our dataset, since we did not want to discriminate against these papers. We have also created a dataset in which we removed all industrial papers, and the results are similar as those presented in this paper. We omit these results in the interest of space.

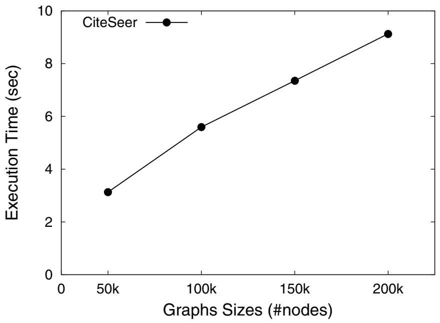


Fig. 5. Efficiency of the CANAL algorithm on the CiteSeer dataset

of the resulting summaries. Figure 4 plots the $\frac{\Delta}{k}$ values corresponding to the different cutoffs.

As shown in Figure 4, in the range of $8 \leq k \leq 14$, CANAL 2-cutoffs result in summaries with better quality than Manual 2-cutoffs. Even when $3 \leq k \leq 7$, the average difference for $\frac{\Delta}{k}$ between the Manual 2-cutoffs and the CANAL 2-cutoffs is within 10%. Overall, CANAL 2-cutoffs produce high-quality summaries that match the manually selected Manual 2-cutoffs.

C. *Efficiency of The CANAL Algorithm*

In this section, we present results evaluating the efficiency of the CANAL algorithm.

For this experiment, we use the four CiteSeer datasets shown in Table I, and set $C = 3$ (i.e. 2 cutoffs) for simplicity. The first row of Table III shows the cutoffs. Note that different C values do not significantly affect the run time of the CANAL algorithm, because all the cutoff candidates are maintained in a heap structure and running CANAL once can generate all different number of cutoffs (refer to Section III). We run each experiment 100 times, and report the average number.

Figure 5 shows the execution time of the CANAL algorithm on the CiteSeer datasets with increasing data sizes. As can be seen, CANAL scales nicely with increasing data sizes. After generating the cutoffs, the k-SNAP summarization method only takes a few seconds to produce each of the summaries of size less than 15 (e.g. $k < 15$) on these datasets.

D. *Effectiveness of the Interestingness Measure*

In Section IV, we proposed a measure to assess the interestingness of graph summaries. Now, we evaluate the effectiveness of this measure on the DBLP DB and the CiteSeer datasets for different combinations of k (number of groups in a summary) and C (the number of categories) values. All cutoffs in this experiment are produced by the CANAL algorithm and the cutoff details are shown in Table II and Table III. In this experiment, we select some interesting summaries, and visualize them to explain why they are interesting.

Table IV and Table V shows the interestingness values of the summaries with different combinations of k and C values for the DBLP and the CiteSeer D200k datasets respectively. As observed in Table IV and Table V, large k values often result in summaries with low interestingness values. This is because large summaries tend to have low conciseness. In addition, large C values also contribute to summaries with low interestingness values, because very detailed attribute categories may lead to summaries that are hard to interpret. Therefore, in the remainder of this section, we only consider small k and small C values.

TABLE IV

EXPLORING INTERESTING SUMMARIES – DBLP DATASET

k	2-cutoffs	3-cutoffs	4-cutoffs
3	0.010		
4	**0.112**	0.054	
5	0.091	0.098	0.037
6	**0.093**	0.110	0.072
7	0.086	**0.122**	0.067
8	0.075	0.064	0.077
9	0.065	0.060	0.067
10	0.052	**0.080**	0.066
11	0.051	0.051	0.051
12	0.047	0.052	0.050
13	0.043	0.056	0.044
14	0.041	0.054	0.044

TABLE V

EXPLORING INTERESTING SUMMARIES – CITESEER D200K DATASET

k	2-cutoffs	3-cutoffs	4-cutoffs
3	0.105		
4	0.098	0.091	
5	**0.167**	0.087	0.086
6	0.138	**0.134**	0.084
7	0.137	0.115	0.110
8	**0.152**	0.114	0.128
9	0.149	0.104	0.125
10	0.119	0.095	0.012
11	0.105	0.094	0.121
12	0.099	**0.101**	0.118
13	0.081	0.100	0.116
14	0.080	0.080	0.093

Figure 6 and Figure 7 plot the interestingness values of summaries with 2 and 3 cutoffs for the DBLP and CiteSeer datasets. We observe that there is a common shape that is present in these two graphs. Figure 8 illustrates this common shape. The interestingness value is low when k is very small, then quickly increases as k increases. Then, it reaches a peak. After that, the interestingness value decreases a bit, and then rises again to a second peak. After this second peak, the interestingness value slowly declines.

The two peaks correspond to two types of interesting summaries. We call them *overall summary* and *informative summary*, respectively. The *overall summary*, with a small k value, concisely captures the general relationships amongst groups of nodes. On the other hand, the *informative summary* contains more details that lead to new discovery of diverse relationships.

According to the interestingness measure introduced in Section IV, the *overall summary* takes advantage of a smaller $Conciseness$ value, and also achieves good $Coverage$ and $Diversity$ values. On the other hand, although the *informative summary* is not as concise, it discovers new relationships and consequently has a higher $Diversity$ value.

In Table IV and Table V, we highlighted the interestingness

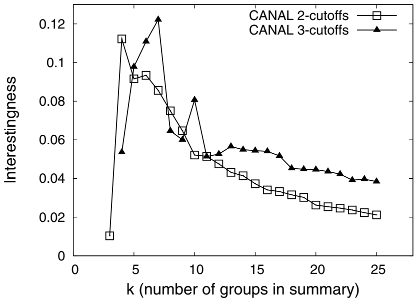


Fig. 6. Interestingness of summaries for the DBLP DB dataset

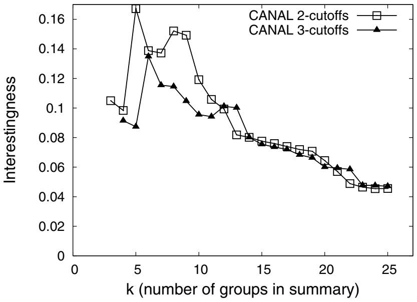


Fig. 7. Interestingness of summaries for the CiteSeer dataset

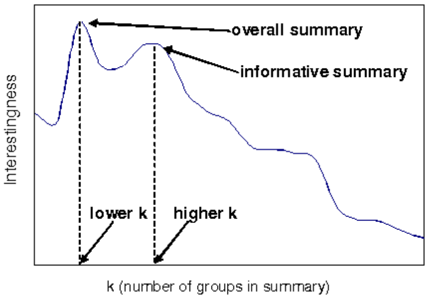
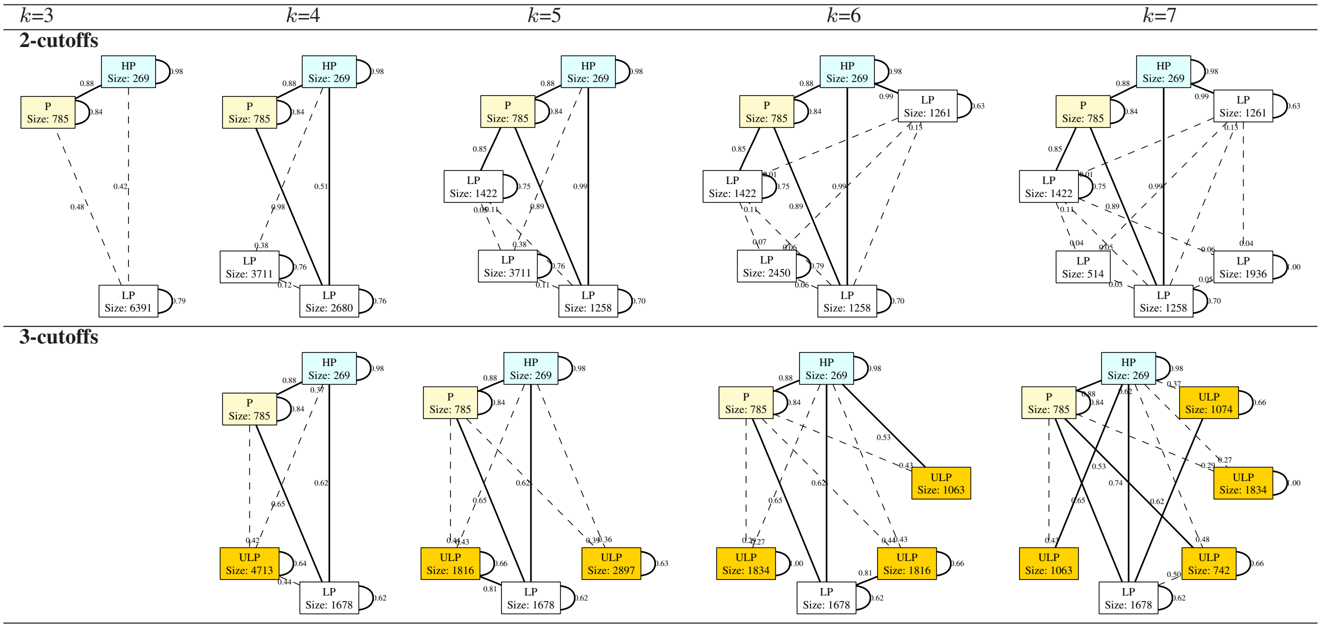


Fig. 8. Common shape of curves in Figure 7 and Figure 6

TABLE VI

SUMMARY GRAPHS WITH 2-CUTOFFS AND 3-CUTOFFS FOR THE DBLP DB DATASET



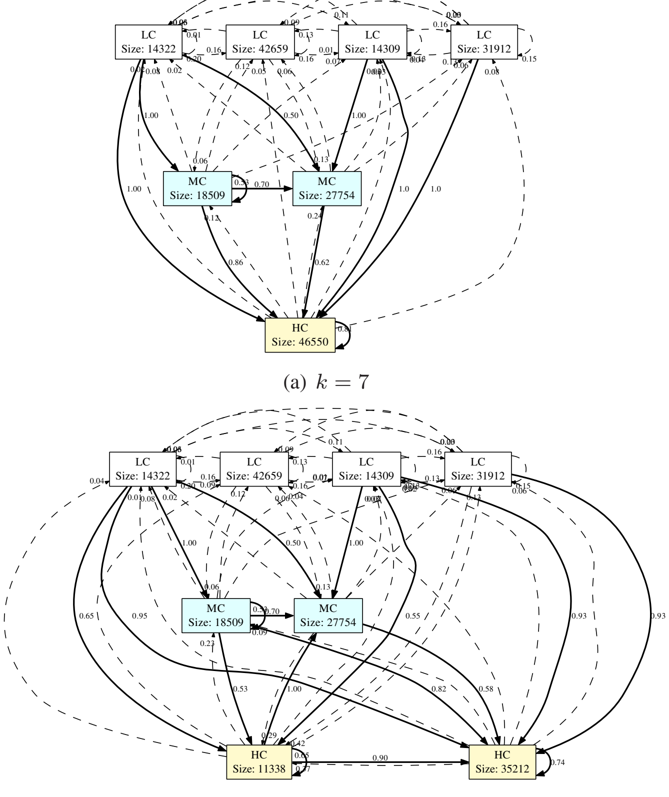
scores of the overall summaries and the informative summaries for the two datasets in bold for the 2-cutoffs and the 3-cutoffs. Now, let's take a look at some of the summaries and explain why there are interesting.

First, let's consider the case of the 2-cuttoffs for the DBLP dataset. The numerical attribute *PubNum* is cut into three categories. Using the same convention as in [20], we name the three categories as Low Prolific (LP) when $1 \leq PubNum \leq 3$, Prolific (P) when $4 \leq PubNum \leq 10$, and Highly Prolific (HP) when $PubNum \geq 11$. The first row of Table VI shows the summaries with sizes increasing from 3 to 7. For ease of presentation, we call them 3-summary, 4-summary and so on. With 2-cutoffs, the 4-summary corresponds to the *overall summary*. Compared to the 3-summary, it introduces one extra group and two new strong relationships between different categorical values, thus has better $Diversity$ and $Coverage$. On the other hand, it is more concise than the 5-summary. The 6-summary is considered the *informative summary*. It introduces a new LP group (1261 LP) that only strongly collaborates with the HP group, thus achieves better $Diversity$ and $Coverage$ than the 5-summary. Compared to the 7-summary, it has better $Conciseness$.

Now, we switch to the summaries with 3 cutoffs. 3 cutoffs give us four categories as shown in the third row of Table II. We name them ULP (Ultra Low Prolific), LP, P and HP. Summaries with $k = 4$ to $k = 7$ are visualized in the second row of Table VI. The highest interestingness value is achieved at $k = 7$. The corresponding 7-summary is considered as the overall summary. In this summary, each group of ULP authors has unique coauthorship pattern with other groups.

Table V shows interestingness values of summaries for the CiteSeer D200k dataset. Here we focus on the interesting summaries in the 2-cutoffs column, because most of the high-scoring summaries are from this column. With 2-cutoffs, the $CiteNum$ attribute is divided into three categories as shown in the first row of Table III. We call the three categories Low Cited (LC), Moderately Cited (MC) and Highly Cited (HC).

The 5-summary shown in Figure 3c and the 8-summary shown in Figure 9b, correspond to the *overall summary* and



(a) $k = 7$

(b) $k = 8$

Fig. 9. Summary graphs with 2 cutoffs for the CiteSeer dataset ($k = 7, 8$)

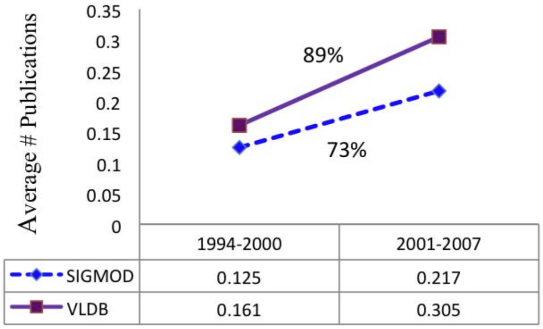
the *informative summary*, respectively. Compared to the 4-summary in Figure 3b, the 5-summary introduces an important group of LC (28631 LC) heavily citing both MC and HC groups, which results in a high interestingness value. The 8-summary introduces a new HC group (11338 HC) that heavily cites a MC group (27754 MC). This relationship, which is not shown in the 7-summary in Figure 9a, provides more insights to the original data, thus makes the 8-summary more interesting than the 7-summary.

From the above experiments, we have shown that using the proposed interestingness measure, the summaries with high scores often correspond to summaries with high interestingness. We have also shown that this measure provides two important summary points (the overall and the informative summaries). Therefore, our interestingness measure is effective in selecting a small set of interesting summaries that user can explore.

E. Example Applications

In this section, we demonstrate how summaries produced by the proposed methods can be used to answer interesting questions in two real examples applications:

1) DBLP Coauthorship Network: The first example application is to analyze the impact of the double-blind review process in SIGMOD (which started using double-blind review since 2001) using the DBLP DB coauthorship network.



	1994-2000	2001-2007
SIGMOD	0.125	0.217
VLDB	0.161	0.305

Fig. 10. Publication growth rates (SIGMOD vs VLDB)

There has been a lot of debate on whether double-blind review has any impact on SIGMOD publication. Madden *et al.* [14] and Tung [21] drew contradictory conclusions on this subject by using simple statistics. Here, we show how our method can be used to analyze the effect of double-blind reviewing.

As described in Section V-A, the coauthorship graph represents the coauthorship among the database researchers. Each node in this graph has a numerical attribute *PubNum*, indicating the number of publications for each author. Table II shows the different cutoffs produced by CANAL to categorize this numerical attribute. The interestingness scores of summaries with different C and k values are listed in Table IV. As shown in this table, 2-cutoffs and 3-cutoffs both lead to interesting summaries. For 2-cutoffs, summaries with size $k \leq 6$ have high interestingness scores, and for 3-cutoffs, summaries with size $k \leq 10$ are ranked among the top.

Let's first look at the 3-cutoffs summaries. 3-cutoffs single out authors with only one publication into a category called ULP (Ultra Low Prolific). The second row of Table VI visualizes the summaries with $k = 4$ to $k = 7$. The summaries split the ULP authors into more and more detailed groups, while the general LP group stays unchanged. However, in the analysis of the double-blind review effect, we do not want to single out authors with only one publications, instead we want to investigate a general group of low prolific authors. Therefore, we omit these 3-cutoffs summaries, although they score high with respect to the interestingness measure.

Now, we study the 2-cutoffs summaries. The first row of Table VI visualizes the summaries with size $k = 3$ to $k = 7$, among which the 4-summary is the *overall summary* and the 6-summary is the *informative summary* according to our summary interestingness measure (see Section V-D). In the 4-summary, the LP authors are divided into 2 groups: one group of 2680 strongly coauthor with both HP and P authors; and one group of 3711 only strongly coauthor among themselves. In the 6-summary, LP authors are divided into more detailed groups: one group of 1258 LP authors strongly coauthor with both HP and P authors, one group of 1261 only strongly coauthor with the HP group, one group of 1422 only strongly

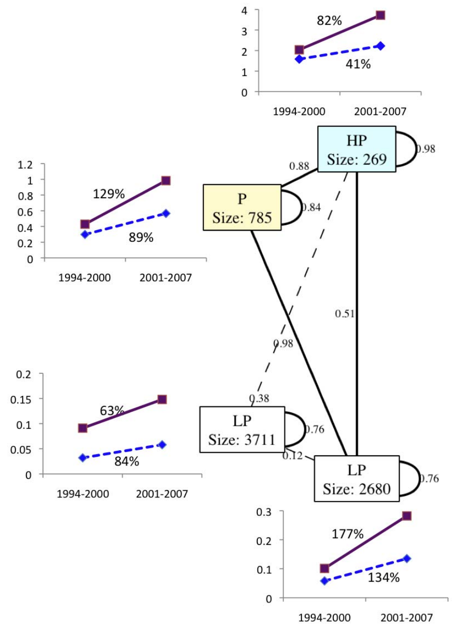


Fig. 11. Analysis of the double-blind review process with the 4-summary

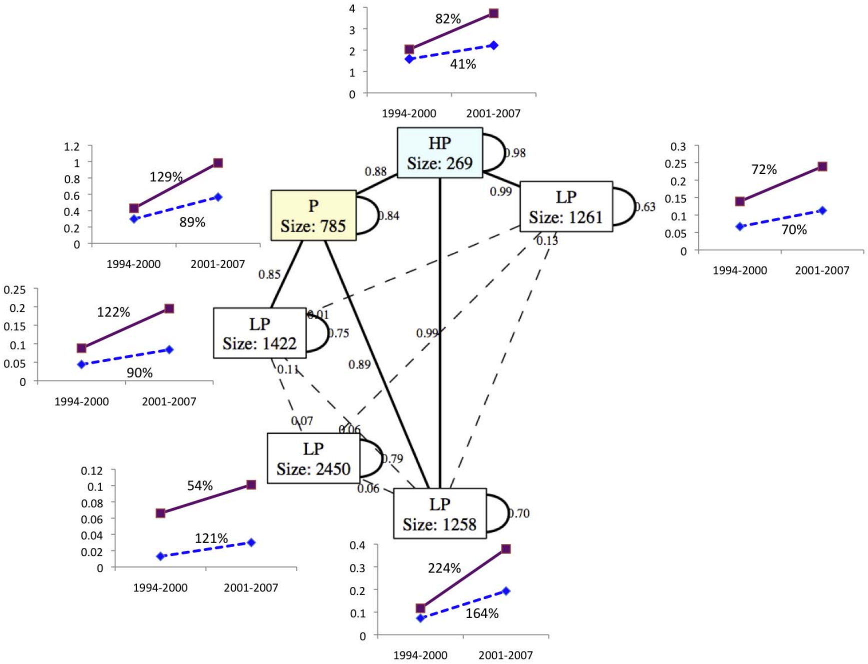


Fig. 12. Analysis of the double-blind review process with the 6-summary

coauthor with the P group, and the remaining 2450 LP authors are isolated from the P and HP authors (they only coauthor among themselves).

While the previous studies [14], [21] treat all the low prolific authors as a whole, our methods can divide them into subgroups based on their coauthorship with the prolific or highly prolific authors, and analyze the different subgroups to get a more comprehensive understanding.

As in [14], [21], we compare the publication rates of SIGMOD before and after the double-blind review against VLDB (which does not have double-blind reviewing). Figure 10 shows the average number of publications for all authors in SIGMOD and VLDB from 1994 to 2007. The period 1994-2000 is before the start of double blind reviewing in SIGMOD, and the period 2001-2007 is after double blind reviewing was introduced in SIGMOD. This chart shows the average growth rates are 73% for SIGMOD and 89% for VLDB. The detailed growth rates of each author group for the 4-summary and the 6-summary are shown in Figure 11 and Figure 12, respectively.

In the overall 4-summary, we observe that the HP authors have much lower growth rate in SIGMOD (41%) than the average growth rate (73%), whereas the growth rate in VLDB (82%) is close the average (89%). So double-blind reviewing appears to result in fewer papers from the HP group of authors. Furthermore, the LP group (2680 LP) that coauthors with both HP and P authors has significantly more publications and much higher growth rates in both SIGMOD and VLDB than the LP group (3711 LP) that has little connection with the HP and P authors.

The more informative 6-summary with detailed growth rates (shown in Figure 12) provides us further insights into the double-blind review analysis. In this finer summary, the isolating group of LP authors (2450 LP) has significantly higher growth rate in SIGMOD (121%) than average (73%), but has a much lower growth rate in VLDB (54%) than average (89%). This observation seems to suggest that double-blind review in SIGMOD welcomes more papers from low prolific authors even without any collaboration with HP or P authors.

2) CiteSeer Citation Networks: In the second example application, we analyze what potentially helps a low cited paper get more citations.

In this analysis, we use the CiteSeer D200k dataset as described in Section V-A. Each node in the citation network has a numerical attribute *CiteNum*, which is the number of citations the article receives. We choose 2 cutoffs as shown in the first row of Table III, and name the resulting three categories as LC, MC and HC as in Section V-D.

The 5-summary shown in Figure 3c is an *overall summary* as discussed in Section V-D and sheds lights on answering this question. In this summary, the LC articles are divided into three subgroups: One group of 28631 LC articles both strongly cites MC and HC articles, one group of 31912 LC largely only strongly cites HC articles, and the last group of 42659 LC rarely cites either HC or MC articles. Now, we calculate the average of the *CiteNum* values for each of the three LC groups. We find that the LC group (28631 LC), citing both HC and MC articles, has a high average number of citations (1.75). In addition, the LC group (31912 LC), largely only citing HC articles also has a similar high average number of citations (1.74). However, the last group of LC articles (42659 LC), which does not frequently cite HC or MC articles, has on average only 1.61 citations. This seems to suggest that citing more influential articles potentially helps low cited articles get more citations. A potential explanation for this behavior is that for each article CiteSeer can also show which other articles have cited it in its history. Therefore, when a LC article L cites an influential article H, L can also be seen when a reader visits the web page of H. This citation relationship potentially helps L become frequently exposed to the public search, thus increases its chances of being cited as well.

VI. Related Work

Graph summarization has been an active research area, and various ways of summarizing graphs have been proposed. Statistical methods (e.g. [6], [7], [16]), are able to discover properties of large social networks, such as the scale-free property and the small world effect. However, these methods do not produce summaries that are graphs (which is the focus of this paper). Graph partitioning methods, such as [17], [19], [23], are useful in detecting communities in large networks, but node attributes are largely ignored in the analysis.

The related problem of graph compression has been extensively studied [5], [10], [12], [13], [18]. The recent work by Navlakha et al. [15] uses the MDL principle to compress graphs with bounded error. Although this MDL based approach produces summary-like structures, its main goal is to reduce the storage space needed to represent the original graph. Furthermore, unlike our methods, the MDL based approach does not consider node attributes or multiple edge types.

Work in graph visualization aims to design layout methods to improve the visualization of large graphs. A good survey of such methods is provided in [4]. In [22], Wang et al. proposed a method called CSV to mine and visualize cohesive subgraphs by mapping edges and nodes to a multidimensional space wherein dense areas correspond to cohesive subgraphs. Tools, such as [24], can visualize dynamic interaction graphs by incorporating common visualization paradigms such as zooming, coarsening and filtering. Our methods are complementary to these methods and can be combined with these visualization methods to provide an interactive way of summarizing graphs.

This paper is an improvement over our previous work [20]. We solve two main limitations of this previous work by proposing automatic categorization of numeric attributes and interestingness measurement of summaries. These two proposals significantly improve the usability of the previous work.

VII. Conclusions

This paper presents a graph summarization technique that builds on the previous work [20] which summarizes graphs based on node attributes and different edge types. However, there were two key limitations in that previous work, which we address in this paper. First, we allow automatic categorization of numeric node attributes (which is a common scenario), and second we automatically score the graph summaries to produce a small subset of interesting summaries (which results in a better paradigm as the user does not have to manually inspect a large number of summaries to find the interesting ones). Our solutions are crucial for the usability of graph summarization. Extensive evaluations using the DBLP and CiteSeer datasets show that our methods are effective in producing interesting graph summaries. Furthermore, the proposed methods are also very efficient and can be used to summarize large graph datasets.

Acknowledgment

This research was supported in part by the National Science Foundation under grant DBI-0926269 and by the National Institutes of Health under grant 1-U54-DA021519-01A1. We would like to thank the reviewers of this paper for their constructive comments on a previous version of this manuscript.

References

[1] Citeseer. http://citeseer.ist.psu.edu.

[2] DBLP Bibliography. http://www.informatik.uni-trier.de/~ley/db.

[3] R. Agrawal and R. Srikant. Fast algorithm for mining association rules. In *Proceedings of VLDB*, pages 487–499, 1994.

[4] G. Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[5] D. K. Blandford, G. E. Blelloch, and I. A. Kash. Compact representations of separable graphs. In *SODA*, pages 679–688, 2003.

[6] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.

[7] D. Chakrabarti, C. Faloutsos, and Y. Zhan. Visualization of large networks with min-cut plots, a-plots and r-mat. *Int. J. Hum.-Comput. Stud.*, 65(5):434–445, 2007.

[8] CNN. Facebook nearly as large as U.S. population. http://edition.cnn.com/2009/TECH/09/16/facebook.profit/, 2009.

[9] L. Geng and H. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), 2006.

[10] X. He, M.-Y. Kao, and H.-I. Lu. A fast general methodology for information - theoretically optimal encodings of graphs. In *Proceedings of ESA*, pages 540–549, 1999.

[11] R. J. Hilderman and H. J. Hamilton. *Knowledge Discovery and Measures of Interests*. Kluwer Academic, Boston, MA, 2001.

[12] K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58(3):239–252, 1995.

[13] H.-I. Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In *Proceedings of SODA*, pages 223–224, 2002.

[14] S. Madden and D. DeWitt. Impact of double-blind reviewing on SIGMOD publication rates. *SIGMOD Record*, 35(2):29–32, 2006.

[15] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of SIGMOD'08*, pages 567–580, 2008.

[16] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[17] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[18] S. Raghavan and H. Garcia-Molina. Representing Web graphs. In *Proceedings of ICDE*, pages 405–416, 2003.

[19] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Stat. Anal. Data Min.*, 1(1):6–22, 2008.

[20] Y. Tian, R. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of SIGMOD'08*, pages 419–432, 2008.

[21] A. Tung. Impact of double blind reviewing on SIGMOD publication: a more detail analysis. *SIGMOD Record*, 35(3):6–7, 2006.

[22] N. Wang, S. Parthasarathy, K. Tan, and A. Tung. Csv: Visualizing and mining cohesive subgraphs. In *Proceedings of SIGMOD'08*, pages 445–458, 2008.

[23] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: A structural clustering algorithm for networks. In *Proceedings of KDD'07*, pages 824–833, 2007.

[24] X. Yang, S. Asur, S. Parthasarathy, and S. Mehta. A visual-analytic toolkit for dynamic interaction graphs. In *Proceedings of KDD'08*, pages 1016–1024, 2008.