Chapter 15
Interactive Graph Summarization

Yuanyuan Tian and Jignesh M. Patel

Abstract Graphs are widely used to model real-world objects and their relationships, and large graph data sets are common in many application domains. To understand the underlying characteristics of large graphs, graph summarization techniques are critical. Existing graph summarization methods are mostly statistical (studying statistics such as degree distributions, hop-plots, and clustering coefficients). These statistical methods are very useful, but the resolutions of the summaries are hard to control. In this chapter, we introduce database-style operations to summarize graphs. Like the OLAP-style aggregation methods that allow users to interactively drill-down or roll-up to control the resolution of summarization, the methods described in this chapter provide an analogous functionality for large graph data sets.

15.1 Introduction

Graphs provide a powerful primitive for modeling real-world objects and the relationships between objects. Various modern applications have generated large amount of graph data. Some of these application domains are listed below:

- Popular social networking web sites, such as Facebook (www.facebook.com), MySpace (www.myspace.com), and LinkedIn (www.linkedin.com), attract millions of users (nodes) connected by their friendships (edges). By April 2009, the number of active users on Facebook has grown to 200 million, and on average each user has 120 friends. Mining these social networks can provide valuable information on social relationships and user communities with common interests. Besides mining the friendship network, one can also mine the "implicit" interaction network formed by dynamic interactions (such as sending a message to a friend).

J.M. Patel (✉)
University of Wisconsin Madison, WI, USA
e-mail: jignesh@cs.wisc.edu

P.S. Yu, et al. (eds.), *Link Mining: Models, Algorithms, and Applications*,
DOI 10.1007/978-1-4419-6515-8_15,

- Coauthorship networks and citation networks constructed from DBLP (www.informatik.uni-trier.de/~ley/db/) and CiteSeer (citeseer.ist.psu.edu) can help understand publication patterns of researchers.
- Market basket data, such as those produced from Amazon (www.amazon.com) and Netflix (www.netflix.com), contain information about millions of products purchased by millions of customers, which forms a bipartite graph with edges connecting customers to products. Exploiting the graph structure of the market basket data can improve customer segmentation and targeted advertising.
- The link structure of the World Wide Web can be naturally represented as a graph with nodes representing web pages and directed edges representing the hyperlinks. According to the estimate at www.worldwidewebsize.com, by May 15, 2009, the World Wide Web contains at least 30.05 billion webpages. The graph structure of the World Wide Web has been extensively exploited to improve search quality [8], discover web communities [16], and detect link spam[23].

With the overwhelming wealth of information encoded in these graphs, there is a critical need for tools to summarize large graph data sets into concise forms that can be easily understood.

Graph summarization has attracted a lot of interest from a variety of research communities, including sociology, physics, and computer science. It is a very broad research area that covers many topics. Different ways of summarizing and understanding graphs have been invented across these different research communities. These different summarization approaches extract graph characteristics from different perspectives and are often complementary to each other. Sociologists and physicists mostly apply statistical methods to study graph characteristics. The summaries of graphs are statistical measures, such as degree distributions for investigating the scale-free property of graphs, hop-plots for studying the small world effect, and clustering coefficients for measuring the clumpiness of large graphs. Some examples of this approach were presented in Chapter 8. In the database research community, methods for mining frequent subgraph patterns are used to understand the characteristics of large graphs, which was the focus of Chapter 4. The summaries produced by these methods are sets of frequently occurring subgraphs (in the original graphs). Various graph clustering (or partitioning) algorithms are used to detect community structures (dense subgraphs) in large graphs. For these methods, the summaries that are produced are partitions of the original graphs. This topic is covered in Chapters 3 and 7. Graph compression and graph visualization are also related to the graph summarization problem. These two topics will be discussed in Section 15.7 of this chapter.

This chapter, however, focuses on a graph summarization method that produces small and informative summaries, which themselves are also graphs. We call them *summary graphs*. These summary graphs are much more compact in size and provide valuable insight into the characteristics of the original graphs. For example, in Fig. 15.1, a graph with 7445 nodes and 19,971 edges is shown on the left. Understanding this fairly small graph by mere visual inspection of the raw graph structure is very challenging. However, the summarization method introduced in this chapter

will generate much compact and informative graphs that summarize the high-level structure characteristics of the original graph and the dominant relationships among clusters of nodes. An example summary graph for the original graph is shown on the right of Fig. 15.1. In the summary graph, each node represents a set of nodes from the original graph, and each edge of the summary graph represents the connections between two corresponding sets of nodes. The formal definition of summary graphs will be introduced in Section 15.2.

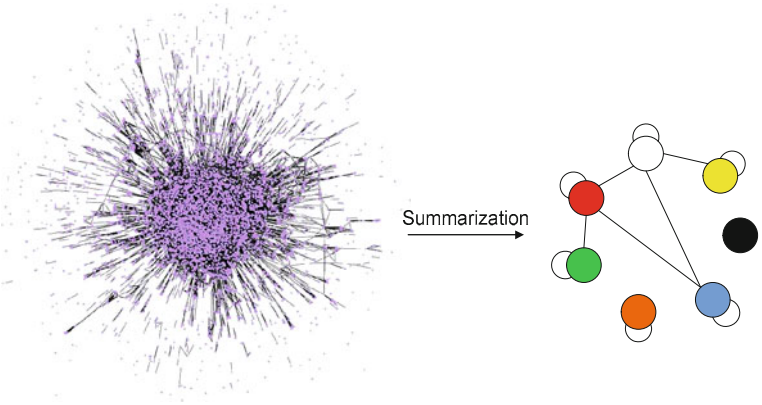


Fig. 15.1 A summary graph (*right*) is generated for the original graph (*left*)

The concept of summary graph is the foundation of the summarization method presented in this chapter. This method is very unique in that it is amenable to an interactive querying scheme by allowing users to customize the summaries based on user-selected node attributes and relationships. Furthermore, this method empowers users to control the resolutions of the resulting summaries, in conjunction with an intuitive "drill-down" or "roll-up" paradigm to navigate through summaries with different resolution. This last aspect of drill-down or roll-up capability is inspired by the OLAP-style aggregation methods [11] in the traditional database systems.

Note that the method introduced in this chapter is applicable for both directed and undirected graphs. For ease of presentation, we only consider undirected graphs in this chapter.

The remainder of this chapter is organized as follows: We first introduce the formal definition of summary graph in Section 15.2, then discuss the aggregation-based graph summarization method in Section 15.3. Section 15.4 shows an interesting example of applying this graph summarization method to the DBLP [17] coauthorship graph. Section 15.5 demonstrates the scalability of the described method. Section 15.6 provides some discussion on the summarization method. Related topics, such as graph compression and graph visualization are discussed in Section 15.7. Finally, Section 15.8 concludes this chapter.

15.2 Summary Graphs

In this chapter, 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 $A = \{a_1, a_2, \ldots, a_m\}$. 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 $T = \{t_1, t_2, \ldots, t_n\}$. 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, Fig. 15.2a shows a sample social networking graph. In this graph, nodes represent students. Each student node has attributes such as gender and department. In addition, there are two types of relationships present in this graph: friends and classmates. While some students are only friends or classmates with each other, others are connected by both relationships. Note that in this figure, only a few edges are shown for compactness.

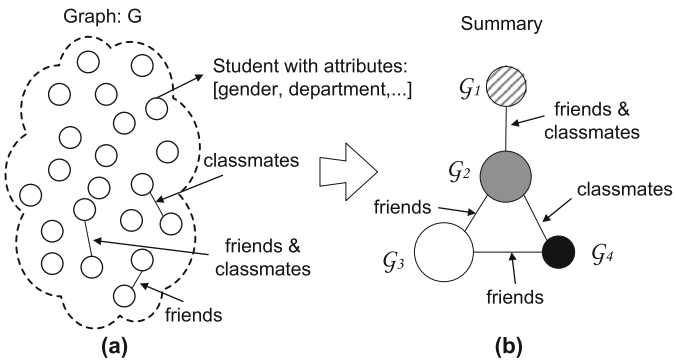


Fig. 15.2 Graph summarization by aggregation

We can formally define summary graphs as follows: Given a graph $G = (V, E)$, and a partition of V, $\Phi = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ ($\bigcup_{i=1}^{k} \mathcal{G}_i = V$ 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 $T(\mathcal{G}_i, \mathcal{G}_j) = \bigcup_{(u,v)\in E, u\in\mathcal{G}_i, v\in\mathcal{G}_j} T(u, v)$.

More intuitively, each node of the summary graph, called a *group* or a *supernode*, corresponds to one group in the partition of the original node set, and an edge, called *group relationships* or *superedges*, 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.

15.3 Aggregation-Based Graph Summarization

The graph summarization method we will discuss in this chapter is a database-style graph aggregation approach [28]. This aggregation-based graph summarization approach contains two operations.

The first operation, called *SNAP* (*S*ummarization by Grouping *N*odes on *A*ttributes and *P*airwise Relationships), produces a summary graph of the input graph by grouping nodes based on user-selected node attributes and relationships. The *SNAP* summary for the graph in Fig. 15.2a is shown in Fig. 15.2b. This summary contains four groups of students and the relationships between these groups. Students in each group have the same gender and are in the same department, and they relate to students belonging to the same set of groups with friends and classmates relationships. This compact summary reveals the underlying characteristics about the nodes and their relationships in the original graph.

The second operation, called *k-SNAP*, further allows users to control the resolutions of summaries. This operation is pictorially depicted in Fig. 15.3. Here using the slider, a user can "drill-down" to a larger summary with more details or "roll-up" to a smaller summary with less details.

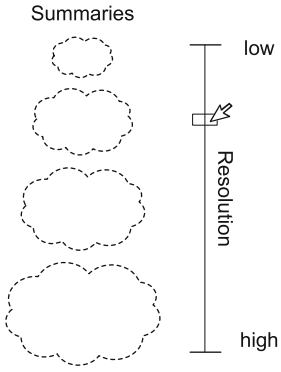


Fig. 15.3 Illustration of multi-resolution summaries

Next we describe the two operations in more detail and present algorithms to evaluate the *SNAP* and *k-SNAP* operations.

15.3.1 SNAP Operation

The *SNAP* operation produces a summary graph through a homogeneous grouping of the input graph's nodes, based on user-selected node attributes and relationships. Figure 15.2b shows an example summary graph for the original graph in Fig. 15.2a, generated by the *SNAP* operation based on gender and department attributes, and classmates and friends relationships.

The summary graph produced by *SNAP* operation has to satisfy the following three requirements:

Attributes Homogeneity: Nodes in each group have the same value for each user-selected attribute.

Relationships Homogeneity: Nodes in each group connect to nodes belonging to the same set of groups with respect to each type of user-selected relationships. For example, in Fig. 15.2b, every student (node) in group $\mathcal{G}_2$ is a friend of some student(s) in $\mathcal{G}_3$, a classmate of some student(s) in $\mathcal{G}_4$, and has at least a friend as well as a classmate in $\mathcal{G}_1$. By the definition of relationships homogeneity, for each pair of groups in the result of the *SNAP* operation, if there is a group relationship between the two, then every node in both groups has to participate in this group relationship (i.e. every node in one group connects to at least one node in the other group).

Minimality: The number of groups in the summary graph is the minimal among all possible groupings that satisfy attributes homogeneity and relationships homogeneity requirements.

There could be more than one grouping satisfying the attributes homogeneity and relationships homogeneity requirements. In fact, the grouping in which each node forms a group is always compatible with any given attributes and relationships (see [28] for more details). The minimality requirement guarantees that the summary graph is the most compact in size.

15.3.1.1 Evaluating *SNAP* Operation

The *SNAP* summary graph can be produced by the following top-down approach:

Top-down *SNAP* Approach

Step 1: Partition nodes based only on the user-selected attributes.
Step 2: Iteration Step
while a group breaks the relationships homogeneity requirement **do**
Split the group based on its relationships with other groups
end while

In the first step of the top-down *SNAP* approach, nodes in the original graph are partitioned based only on the user-selected attributes. This step guarantees that further splits of the grouping always satisfy the attributes homogeneity requirement. For example, if nodes in a graph only have one attribute with values A, B, and C, then the first step produces a partition of three groups with attribute values A, B, and C, respectively, such as the example grouping shown in Fig. 15.4a. In the iterative step, the algorithm checks whether the current grouping satisfies the relationships homogeneity requirement. If not, the algorithm picks a group that breaks the requirement and splits this group based on how the nodes in this group connect to nodes in other groups. In the example shown in Fig. 15.4a, group A (the group

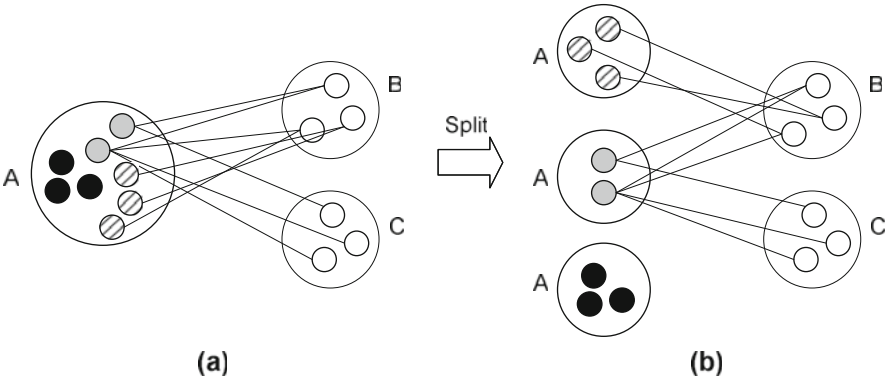


Fig. 15.4 An example of splitting a group based on its relationships with other groups in the top-down *SNAP* approach

with attribute value A) does not satisfy the relationships homogeneity requirement: the black nodes do not connect to any nodes in other groups, the shaded nodes only connect to nodes in group B, while the gray nodes connect to both group B and group C. This group is split based on how the nodes in this group connect to nodes in other groups, which ends up with three subgroups containing black nodes, gray nodes, and shaded nodes shown in Fig. 15.4b. In the next iteration, the algorithm continues to check whether the new grouping satisfies the homogeneity requirement. If not, it selects and splits a group that breaks the requirement. The iterative process continues until the current grouping satisfies the relationships homogeneity requirement. It is easy to prove that the grouping resulting from this algorithm contains the minimum number of groups satisfying both the attributes and relationships homogeneity requirements.

15.3.2 k-SNAP Operation

15.3.2.1 Limitations of the *SNAP* Operation

The *SNAP* operation produces a grouping in which nodes of each group are homogeneous with respect to user-selected attributes and relationships. Unfortunately, homogeneity is often too restrictive in practice, as most real life graph data are subject to noise and uncertainty; for example, some edges may be missing, and some edges may be spurious because of errors. Applying the *SNAP* operation on noisy data can result in a large number of small groups, and, in the worst case, each node may end up in an individual group. Such a large summary is not very useful in practice. A better alternative is to let users control the sizes of the results to get summaries with the resolutions that they can manage (as shown in Fig. 15.3).

The *k-SNAP* operation is introduced to relax the homogeneity requirement for the relationships and allow users to control the sizes of the summaries.

The relaxation of the homogeneity requirement for the relationships is based on the following observation. For each pair of groups in the result of the *SNAP* operation, if there is a group relationship between the two, then every node in both groups participates in this group relationship. In other words, every node in one group relates to some node(s) in the other group. On the other hand, if there is no group relationship between two groups, then absolutely no relationship connects any nodes across the two groups. However, in reality, if most (not all) nodes in the two groups participate in the group relationship, it is often a good indication of a "strong" relationship between the two groups. Likewise, it is intuitive to mark two groups as being "weakly" related if only a small fraction of nodes are connected between these groups.

Based on these observations, the *k-SNAP* operation relaxes the homogeneity requirement for the relationships by not requiring that every node participates in a group relationship. But it still maintains the homogeneity requirement for the attributes, i.e., all the groupings should be homogeneous with respect to the given attributes. Users control how many groups are present in the summary by specifying the required number of groups, denoted as k. There are many different groupings of size k, thus there is a need to measure the qualities of the different groupings. The Δ-measure is proposed to assess the quality of a *k-SNAP* summary by examining how different it is to a hypothetical "ideal summary."

15.3.2.2 Measuring the Quality of *k-SNAP* Summaries

We first define the set of nodes in group $\mathcal{G}_i$ that participate in a group relationship $(\mathcal{G}_i, \mathcal{G}_j)$ as $P_{\mathcal{G}_j}(\mathcal{G}_i) = \{u | u \in \mathcal{G}_i \text{ and } \exists v \in \mathcal{G}_j \text{ s.t. } (u, v) \in E\}$. 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%, we call it a strong group relationship, otherwise, we call it a weak group relationship. Note that in a *SNAP* summary, the participation ratios are either 100 or 0%.

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

$$\Delta(\Phi) = \sum_{\mathcal{G}_i, \mathcal{G}_j \in \Phi} (\delta_{\mathcal{G}_j}(\mathcal{G}_i) + \delta_{\mathcal{G}_i}(\mathcal{G}_j)), \tag{15.1}$$

where

$$\delta_{\mathcal{G}_j}(\mathcal{G}_i) = \begin{cases} |P_{\mathcal{G}_j}(\mathcal{G}_i)| & \text{if } p_{i,j} \leq 0.5 \\ |\mathcal{G}_i| - |P_{\mathcal{G}_j}(\mathcal{G}_i)| & \text{otherwise.} \end{cases} \tag{15.2}$$

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.

Intuitively, the Δ-measure counts the minimum number of differences in participations of group relationships between the given *k-SNAP* grouping and a

hypothetical ideal grouping of the same size in which all the strong relationships have 100% participation ratio and all the weak relationships have 0% participation ratio. The 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, it counts the differences between this strong relationship and a 100% participation-ratio group relationship. The δ function, defined in (15.2), 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. Note that $\delta_{\mathcal{G}_i}(\mathcal{G}_i)$ measures the contribution to the Δ value by the connections within the group $\mathcal{G}_i$ itself.

Given this quality measure and the user-specified 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 problem has been proved to be NP-Complete [28]. Two heuristic-based algorithms are proposed to evaluate the *k-SNAP* operation approximately.

Top-Down *k-SNAP* Approach

Similar to the top-down *SNAP* algorithm, the top-down *k-SNAP* approach also starts from the grouping based only on attributes, and then iteratively splits existing groups until the number of groups reaches k.

However, in contrast to the *SNAP* evaluation algorithm, which randomly chooses a splittable group and splits it into subgroups based on its relationships with other groups, the top-down approach has to make the following decisions at each iterative step: (1) which group to split and (2) how to split it. Such decisions are critical as once a group is split, the next step will operate on the new grouping. At each step, the algorithm can only make the decision based on the current grouping. Each step should make the smallest move possible, to avoid going too far away from the right direction. Therefore, the algorithm splits one group into only two subgroups at each iterative step. There are different ways to split one group into two. One natural way is to divide the group based on whether nodes have relationships with nodes in a neighbor group. After the split, nodes in the two new groups either all or never participate in the group relationships with this neighbor group.

As discussed in Section 15.3.2.2, the *k-SNAP* operation tries to find the grouping with a minimum Δ measure (see (15.1)) for a given k. The computation of the Δ measure can be broken down into each group with each of its neighbors (see the δ function in (15.2)). Therefore, our heuristic chooses the group that makes the most contribution to the Δ value with one of its neighbor groups. More formally, for each group $\mathcal{G}_i$, we define $CT(\mathcal{G}_i)$ as follows:

$$CT(\mathcal{G}_i) = \max_{\mathcal{G}_j}\{\delta_{\mathcal{G}_j}(\mathcal{G}_i)\}. \tag{15.3}$$

Then, at each iterative step, we always choose the group with the maximum CT value to split and then split it based on whether nodes in this group $\mathcal{G}_i$ have relationships with nodes in its neighbor group $\mathcal{G}_t$, where

$$\mathcal{G}_t = arg \max_{\mathcal{G}_j}\{\delta_{\mathcal{G}_j}(\mathcal{G}_i)\}.$$

The top-down *k-SNAP* approach can be summarized as follows:

Top-down *k-SNAP* Approach

Step 1: Partition nodes based only on the user selected attributes.
Step 2: Iteration Step
while the grouping size is less than k **do**
Find $\mathcal{G}_i$ with the maximum $CT(\mathcal{G}_i)$ value
Split $\mathcal{G}_i$ based on its relationship with $\mathcal{G}_t = arg \max_{\mathcal{G}_j}\{\delta_{\mathcal{G}_j}(\mathcal{G}_i)\}$
end while

Bottom-Up *k-SNAP* Approach

The bottom-up approach first computes the *SNAP* summary using the top-down *SNAP* approach. The *SNAP* summary is the summary with the finest resolution, as the participation ratios of group relationships are either 100 or 0%. Starting from the finest summary, the bottom-up approach iteratively merges two groups until the number of groups reduces to k.

Choosing which two groups to merge in each iterative step is crucial for the bottom-up approach. First, the two groups are required to have the same attribute values. Second, the two groups must have similar group relationships with other groups. Now, this similarity between two groups can be formally defined as follows.

The two groups to be merged should have similar neighbor groups with similar participation ratios. We define a measure called $MergeDist$ to assess the similarity between two groups in the merging process.

$$MergeDist(\mathcal{G}_i, \mathcal{G}_j) = \sum_{k \neq i,j} |p_{i,k} - p_{j,k}|. \tag{15.4}$$

$MergeDist$ accumulates the differences in participation ratios between $\mathcal{G}_i$ and $\mathcal{G}_j$ with other groups. The smaller this value is, the more similar the two groups are.

The bottom-up *k-SNAP* approach can be summarized as follows:

Bottom-up *k-SNAP* Approach

Step 1: Compute the *SNAP* summary using the top-down *SNAP* approach
Step 2: Iteration Step
while the grouping size is greater than k **do**
Find $\mathcal{G}_i$ and $\mathcal{G}_j$ with the minimum $MergeDist(\mathcal{G}_i, \mathcal{G}_j)$ value
Merge $\mathcal{G}_i$ and $\mathcal{G}_j$
end while

15.3.3 Top-Down k-SNAP Approach vs. Bottom-Up k-SNAP Approach

In [28], extensive experiments show that the top-down *k-SNAP* approach significantly outperforms the bottom-up *k-SNAP* approach in both effectiveness and efficiency for small k values. In practice, users are more likely to choose small k values to generate summaries. Therefore, the top-down approach is preferred for most practical uses.

15.4 An Example Application on Coauthorship Graphs

This section presents an example of applying the aggregation-based graph summarization approach to analyze the coauthorship graph of database researchers. The database researcher coauthorship graph is generated from the DBLP Bibliography data [17] by collecting the publications of a number of selected journals and conferences in the database area. The constructed coauthorship graph with 7445 authors and 19,971 coauthorships is shown in Fig. 15.5. Each node in this graph represents an author and has an attribute called *PubNum*, which is the number of publications belonging to the corresponding author. Another attribute called *Prolific* is assigned to each author in the graph indicating whether that author is prolific: authors with ≤ 5 papers are tagged as low prolific (LP), authors with > 5 but ≤ 20 papers are prolific (P), and the authors with > 20 papers are tagged as highly prolific (HP).

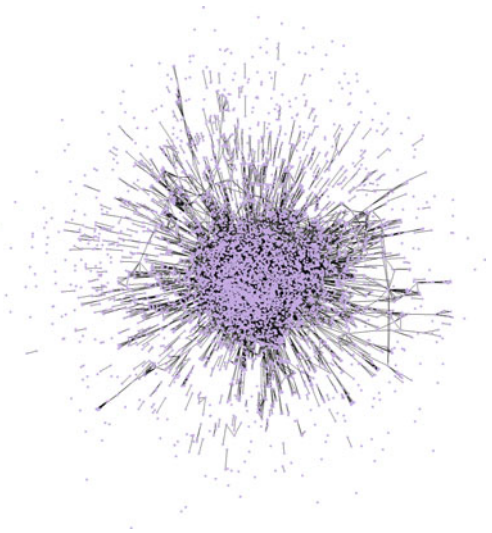


Fig. 15.5 DBLP DB coauthorship graph

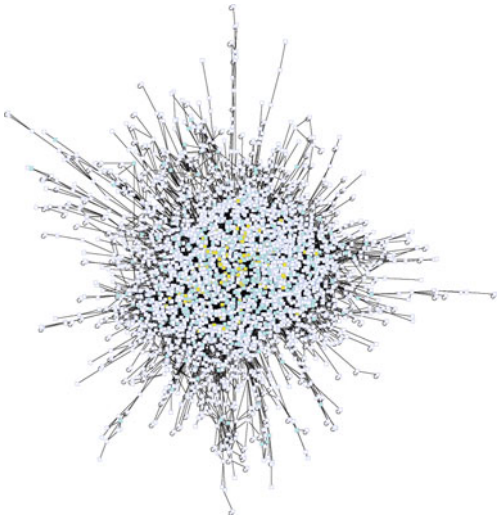


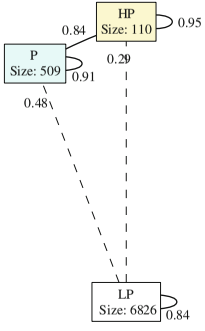
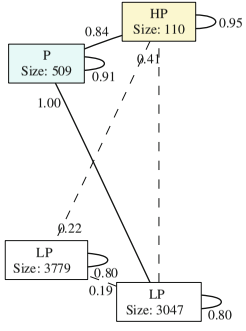
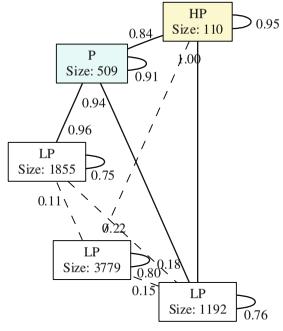
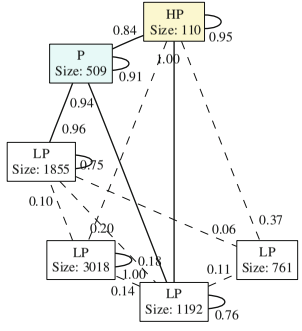
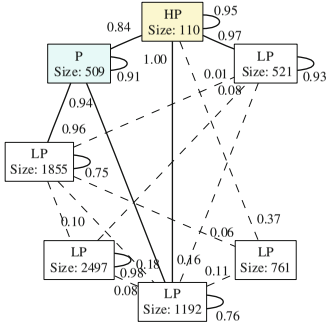
Fig. 15.6 The *SNAP* result for DBLP DB coauthorship graph

The *SNAP* operation on the *Prolific* attribute and the coauthorships produces a summary graph shown in Fig. 15.6. The *SNAP* operation results in a summary with 3569 groups and 11,293 group relationships. This summary is too big to analyze. On the other hand, by applying the *SNAP* operation on only the *Prolific* attribute (i.e., not considering any relationships in the *SNAP* operation), a summary with only three groups is produced shown in the leftmost figure of Table 15.1. The bold edges between two groups indicate strong group relationships (with more than 50% participation ratio), while dashed edges are weak group relationships. This summary shows that the HP researchers as a whole have very strong coauthorship with the P group of researchers. Researchers within both groups also tend to coauthor with people within their own groups. However, this summary does not provide a lot of information for the LP researchers: they tend to coauthor strongly within their group and they have some connection with the HP and P groups.

Now, making use of the *k-SNAP* operation, summaries with multiple resolutions are generated. The figures in Table 15.1 show the *k-SNAP* summaries for $k = 4, 5, 6,$ and 7. As k increases, more details are shown in the summaries.

When $k = 7$, the summary shows that there are five subgroups of LP researchers. One group of 1192 LP researchers strongly collaborates with both HP and P researchers. One group of 521 only strongly collaborates with HP researchers. One group of 1855 only strongly collaborates with P researchers. These three groups also strongly collaborate within their groups. There is another group of 2497 LP researchers that has very weak connections to other groups but strongly cooperates among themselves. The last group has 761 LP researchers, who neither coauthor with others within their own group nor collaborate strongly with researchers in other groups. They often write single author papers.

Table 15.1 Summaries with multiple resolutions for the DBLP DB coauthorship graph

Attribute Only	$k=4$	$k=5$	$k=6$	$k=7$
DB					

Now, in the *k-SNAP* summary for $k = 7$, we are curious if the average number of publications for each subgroup of the LP researchers is affected by the coauthorships with other groups. The above question can be easily answered by applying the *avg* operation on the *PubNum* attribute for each group in the result of the *k-SNAP* operation.

With this analysis, we find out that the group of LP researchers who collaborate with both P and HP researchers has a high average number of publications: 2.24. The group only collaborating with HP researchers has 1.66 publications on average. The group collaborating with only the P researchers has on average 1.55 publications. The group that tends to only cooperate among themselves has a low average number of publications: 1.26. Finally, the group of mostly single authors has on average only 1.23 publications. Not surprisingly, these results suggest that collaborating with HP and P researchers is potentially helpful for the low prolific (often beginning) researchers.

15.5 Scalability of the Graph Summarization Method

In this section, we take the top-down *k-SNAP* approach as an example to demonstrate the scalability of the graph summarization method described in this chapter, as it is more effective and efficient for most practical uses (see Section 15.3.3). More comprehensive experimental results can be found in [28].

Most real-world graphs show power-law degree distributions and small-world effect [20]. Therefore, the R-MAT model [9] in the GTgraph suites [2] is used to generate synthetic graphs with power-law degree distributions and small-world characteristics. The generator uses the default parameters values, and the average node degree in each synthetic graph is set to 5. An attribute is also assigned to each node in a generated graph. The domain of this attribute has five values. Each node is assigned randomly one of the five values.

The top-down *k-SNAP* approach was implemented in C++ on top of PostgreSQL(http://www.postgresql.org) and is applied to different sized synthetic graphs with three resolutions (k values): 10, 100, and 1000. This experiment was run on a 2.8 GHz Pentium 4 machine running Fedora 2, and equipped with a 250 GB SATA disk. 512 MB of memory is allocated to the PostgreSQL database buffer pool, and another 256 MB of additional memory is assigned as the working space outside the database system.

The execution times with increasing graph sizes are shown in Fig. 15.7. When $k = 10$, even on the largest graph with 1 million nodes and 2.5 million edges, the top-down *k-SNAP* approach finishes in about 5 min. For a given k value, the algorithm scales nicely with increasing graph sizes.

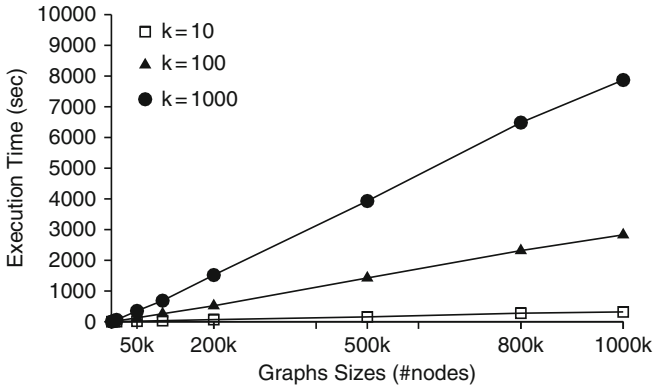


Fig. 15.7 Scalability experiment for synthetic data sets

15.6 Discussion

The example application and the experiments discussed above demonstrate the effectiveness and efficiency of the SNAP/k-SNAP summarization technique. However, the SNAP/k-SNAP approach described above has limitations, which we discuss below.

Attributes with Large Value Ranges: As can be seen from the algorithms in Section 15.3, the minimum summary size is bounded by the cardinality of the domain of the user-selected attributes. More precisely, the summary size is bounded by the actual number of distinct values in the graph data for the user-selected attributes. If one of the user-selected attributes has a large number of distinct values in the graph, then even the coarsest summary produced will be overwhelmingly large. One approach to solving this problem is to bucketize the attribute values into a small number of categories. One such approach is proposed in [30] to automatically categorize attributes with large distinct values by exploiting the domain knowledge hidden inside the node attributes values and graph link structures.

Large Number of Attributes: A related problem to the one discussed above is when a user selects a large number of node attributes for summarization. Now the attribute grouping in SNAP/K-SNAP has to be done over the cross-product of the distinct values used in each attribute domain. This cross-product space can be large. Bucketization methods can again be used in this case, though the problem is harder than the single attribute case. This is an interesting topic for future work.

15.7 Related Topics

15.7.1 Graph Compression

The graph summarization method introduced in this chapter produces small and informative summary graphs of the original graph. In some sense, these compact

summary graphs can be viewed as (lossy) compressed representations of the original graph, although the main goal of the summarization method we described is not to reduce the number of bits needed to encode the original graph, but enabling better understanding of the graph.

The related problem of graph compression has been extensively studied. Various compression techniques for unlabeled planar graphs have been proposed [10, 12, 15] and are generalized to graphs with constant genus [18]. In [5], a technique is proposed to compress unlabeled graphs based on the observation that most graphs in practice have small separators (subgraphs can be partitioned into two approximately equally sized parts by removing a relatively small number of vertices).

Due to the large scale of the web graph, a lot of attention has drawn to compress the web graph. Most of the studies have focused on lossless compression of the web graph so that the compact representation can be used to calculate measures such as PageRank [8]. Bharat et al. [4] proposed a compression technique making use of gaps between the nodes in the adjacency list. A reference encoding technique is introduced in [22], based on the observation that often a new web page adds links by copying links from an existing page. In this compression scheme, the adjacency list of one node is represented by referencing the adjacency list of another node. Alder and Mitzenmacher [1] proposed a minimum spanning tree-based algorithm to find the best reference list for the reference encoding scheme. The compression technique proposed in [27] takes advantage of the link structure of the web and achieves significant compression by distinguishing links based on whether they are inside or cross hosts, and by whether they are connecting popular pages or not. Boldi and Vigna [6, 7] developed a family of simple flat codes, called ζ codes, which are well suited for compressing power-law distributed data with small exponents. They achieve high edge compression and scale well to large graphs.

Two recently proposed graph compression techniques that share similarities with the graph summarization technique described in this chapter will be discussed in detail below.

15.7.1.1 S-Node Representation of the Web Graph

The compression technique proposed in [21] compresses the web graph into a *S-Node representation*. As exemplified in Fig. 15.8, the *S-Node representation* of a web graph contains the following components:

SUPERNODE GRAPH: The supernode graph is essentially a summary graph of the web graph, in which groups are called *supernodes* and group relationships are called *superedges*.

INTRANODE GRAPHS: Each intranode graph (abbreviated as IN in Fig. 15.8) characterizes the connections between the nodes inside a supernode.

POSITIVE SUPEREDGE GRAPHS: Each positive superedge graph (abbreviated as PSE in Fig. 15.8) is a directed bipartite graph that represents the links between two corresponding supernodes.

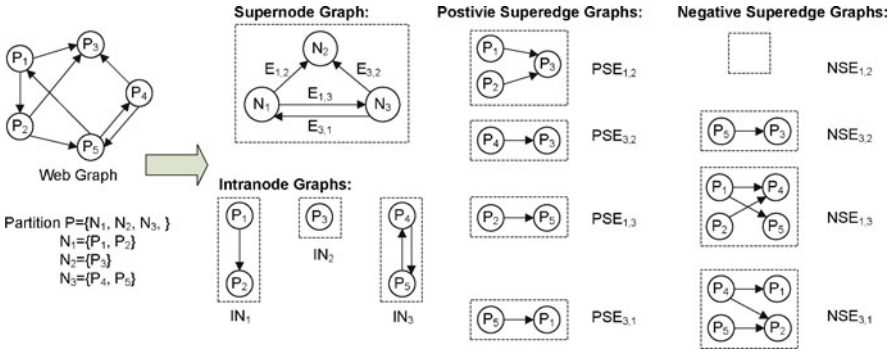


Fig. 15.8 An example of *S-Node representation* (from [21])

NEGATIVE SUPEREDGE GRAPHS: Each negative superedge graph (abbreviated as NSE in Fig. 15.8) captures, among all possible links between two supernodes, those that are absent from the actual web graph.

The compression technique in [21] employs a top-down approach to compute the S-Node representation. This algorithm starts from a set of supernodes that are generated based on the URL domain names, then iteratively splits an existing supernode by exploiting the URL patterns of the nodes inside this supernode and their links to other supernodes. However, different from the graph summarization method introduced in this chapter, this approach is specific to the web graph, thus are not directly applicable to other problem domains. Furthermore, since this approach aims at compressing the web graph, only one compressed S-Node representation is produced. Users have no control over the resolution of the summary graph.

15.7.1.2 MDL Representation of Graphs

Similar to the S-Node method described above, the technique proposed in [19] also compresses a graph into a *summary graph*. To reconstruct the original graph, a set of *edge corrections* are also produced. Figure 15.9 shows a sample graph G and its summary graph S with the set of edge corrections C.

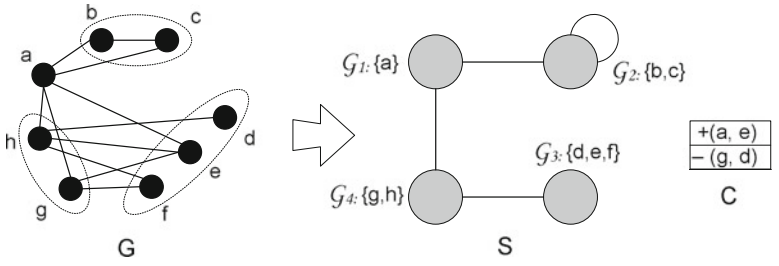


Fig. 15.9 An example of MDL-based summary: G is the original graph, S is the summary graph, and C is the set of edge corrections (from [25])

The original graph can be reconstructed from the summary graph by first adding an edge between each pair of nodes whose corresponding supernodes are connected by an superedge, then applying the edge corrections to remove non-existent edges from or add missing edges to the reconstructed graph. For example, to reconstruct the original graph in Fig. 15.9, the summary graph S is first expanded (now $V = \{a, b, c, d, e, f, g, h\}$ and $E = \{(a, b), (a, c), (a, h), (a, g), (b, c), (h, d), (h, e), (h, f), (g, d), (g, e), (g, f)\}$), then the set of corrections in C are applied: adding the edge (a, e) to E and removing the edge (g, d) from E.

Essentially, this proposed representation is equivalent to the S-Node representation described above. The intranode graphs, positive superedge graphs, and negative superedge graphs in the S-Node representation, collectively, can produce the edge corrections needed to reconstruct the original graph from the summary graph.

Based on Rissanen's minimum description length (MDL) principle [25], the authors in [19] formulated the graph compression problem into an optimization problem, which minimizes the sum of the size of the summary graph (the theory) and the size of the edge correction set (encoding of the original graph based on the theory). The representation with the minimum cost is called the *MDL representation*.

Two heuristic-based algorithms are proposed in [19] to compute the MDL representation of a graph. Both algorithms apply a bottom-up scheme: starting from the original graph and iteratively merging node pairs into supernodes until no further cost reduction can be achieved. The two algorithms differ in the policy of choosing which pair of nodes should merge in each iteration. The GREEDY algorithm always chooses the node pairs that give the maximum cost reduction, while the RANDOMIZED algorithm randomly picks a node and merges it with the best node in its vicinity.

The MDL representation can exactly reconstruct the original graph. However, for many applications, recreating the exact graph is not necessary. It is often adequate enough to construct a graph that is reasonably close to the original graph. As a result, the *ϵ-approximate MDL representation* is proposed to reconstruct the original graph within the user-specified bounded error ϵ ($0 \leq \epsilon \leq 1$).

To compute the ϵ-approximate MDL representation with the minimum cost, two heuristic-based algorithms are proposed in [19]. The first algorithm, called APXMDL, modifies the exact MDL representation by deleting corrections and summary edges while still satisfying the approximation constraint. The second algorithm, called APXGREEDY, incorporates the approximation constraint into the GREEDY algorithm, and constructs the ϵ-approximate representation directly from the original graph.

The key difference between this MDL-based approach and the *SNAP/k-SNAP* summarization approach is that the MDL method does not consider node attributes or multiple relationships in the summarization process and it does not allow users to control the resolutions of summaries.

15.7.2 Graph Visualization

Graph visualization methods are primarily designed to better layout a graph on a computer screen so that it is easier for users to understand the graph by visual inspection. Various graph drawing techniques are surveyed in [3]. However, as graphs become large, displaying an entire graph on the limited computer screen is challenging, both from the usability and the visual performance perspectives. To overcome the problems raised by the large graph sizes, navigation, interaction and, summarization techniques are often incorporated into graph visualization tools [13]. One common summarization technique used in graph visualization is structure-based clustering. Clustering provides abstraction of the original graph, and reduces the visual complexity. Graph visualization systems, such as [14, 24, 29], have applied clustering techniques to improve visualization clarity and at the same time increase performance of layout and rendering. The SuperGraph approach introduced in [26] employs a hierarchical graph partitioning technique to visualize large graphs in different resolution. In fact, the graph summarization technique introduced in this chapter can be coupled with visualization techniques to provide better understanding of large graphs.

15.8 Summary

This chapter studies an aggregation-based summarization method, which produces compact and informative graphs as the summaries of the original graphs. The summary graphs characterize the high-level structure embedded in the original graphs by aggregating nodes and edges from the original graph into node groups (supernodes) and group relationships (superedges), respectively. This summarization method utilizes the graph structure as well as user-specified node attributes and relationships to generate multi-resolution summaries. The users can interactively "drill-down" or "roll-up" to navigate through summaries with different resolution. Graph summarization is related to graph compression and can be coupled with graph visualization methods to enable better understanding of large graphs.

References

1. M. Adler and M. Mitzenmacher. Towards compressing web graphs. In *Proceedings of the Data Compression Conference (DCC'01)*, page 203, IEEE Computer Society, Washington, DC, USA, 2001.
2. D. A. Bader and K. Madduri. GTgraph: A suite of synthetic graph generators. http://www.cc.gatech.edu/~kamesh/GTgraph.
3. G. Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Englewood Cliffs, NJ 1999.
4. K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the Web. *Computer Networks and ISDN Systems*, 30(1–7):469–477, 1998.

5. D. K. Blandford, G. E. Blelloch, and I. A. Kash. Compact representations of separable graphs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pages 679–688, Baltimore, Maryland, USA, 2003.
6. P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In *Proceedings of the International World Wide Web Conference (WWW'04)*, pages 595–602, New York, NY, USA, 2004.
7. P. Boldi and S. Vigna. The WebGraph framework II: Codes for the World-Wide Web. In *Proceedings of the Data Compression Conference (DCC'04)*, page 528, Snowbird, Utah, USA, 2004.
8. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the International World Wide Web Conference (WWW'98)*, pages 107–117, Amsterdam, The Netherlands, 1998.
9. D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *Proceedings of the SIAM International Conference on Data Mining (SDM'04)*, Lake Buena Vista, Florida, USA, 2004.
10. H. Galperin and A. Wigderson. Succinct representations of graphs. *Information and Control*, 56(3):183–198, 1983.
11. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'96)*, pages 152–159, New Orleans, Louisiana, USA, 1996.
12. X. He, M.-Y. Kao, and H.-I. Lu. A fast general methodology for information - theoretically optimal encodings of graphs. In *Proceedings of the European Symposium on Algorithms (ESA'99)*, pages 540–549, London, UK, 1999.
13. I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.
14. M. L. Huang and P. Eades. A fully animated interactive system for clustering and navigating huge graphs. In *Proceedings of the International Symposium on Graph Drawing (GD'98)*, pages 374–383, London, UK, 1998.
15. K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58(3):239–252, 1995.
16. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. In *Proceedings of the International World Wide Web Conference (WWW'99)*, pages 1481–1493, Toronto, Canada, 1999.
17. M. Ley. DBLP Bibliography. http://www.informatik.uni-trier.de/~ley/db/.
18. H.-I. Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 223–224, San Francisco, California, USA, 2002.
19. S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pages 419–432, Vancouver, Canada, 2008.
20. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.
21. S. Raghavan and H. Garcia-Molina. Representing Web graphs. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'03)*, pages 405–416, Bangalore, India, 2003.
22. K. H. Randall, R. Stata, R. G. Wickremesinghe, and J. L. Wiener. The link database: Fast access to graphs of the Web. In *Proceedings of the Data Compression Conference (DCC'02)*, pages 122–131, Washington, DC, USA, 2002.
23. D. G. Ravi, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'05)*, pages 721–732, Trondheim, Norway, 2005.

24. J. S. Risch, D. B. Rex, S. T. Dowson, T. B. Walters, R. A. May, and B. D. Moon. The STARLIGHT information visualization system. In *Proceedings of the IEEE Conference on Information Visualisation (IV'97)*, San Francisco, CA, USA, page 42, 1997.
25. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
26. J. F. Rodrigues, A. J. M. Traina, C. Faloutsos, and C. Traina. SuperGraph visualization. In *Proceedings of the IEEE International Symposium on Multimedia (ISM'06)*, Washington, DC, USA, 2006.
27. T. Suel and J. Yuan. Compressing the graph structure of the Web. In *Proceedings of the Data Compression Conference (DCC'01)*, pages 213–222, Washington, DC, USA, 2001.
28. Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pages 567–580, Vancouver, Canada, 2008.
29. G. J. Wills. NicheWorks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.
30. N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'10)*, Long Beach, California, USA, 2010.