Electronic Preprint for Journal of Information Processing Vol.20 No.1

Invited Paper

Approximate Homogeneous Graph Summarization

Zheng Liu[1] Jeffrey Xu Yu[1,a)] Hong Cheng[1]

Received: July 8, 2011, Accepted: October 18, 2011

Abstract: Graph patterns are able to represent the complex structural relations among objects in many applications in various domains. The objective of graph summarization is to obtain a concise representation of a single large graph, which is interpretable and suitable for analysis. A good summary can reveal the hidden relationships between nodes in a graph. The key issue is how to construct a high-quality and representative super-graph, G_S, in which a super-node summarizes a collection of nodes based on the similarity of attribute values and neighborhood relationships associated with nodes in G, and a super-edge summarizes the edges between nodes in G that are represented by two different super-nodes in G_S. We propose an entropy-based unified model for measuring the homogeneity of the super-graph. The best summary in terms of homogeneity could be too large to explore. By using the unified model, we relax three summarization criteria to obtain an approximate homogeneous summary of reasonable size. We propose both agglomerative and divisive algorithms for approximate summarization, as well as pruning techniques and heuristics for both algorithms to save computation cost. Experimental results confirm that our approaches can efficiently generate high-quality summaries.

Keywords: graph summary, graph clustering

1. Introduction

Researchers have made great efforts on mining graph data recently, because of its ability to represent complex relationships among entities in many applicable areas such as Web, social networks, biological networks, telecommunication, etc. It is not an easy task for users to manage and explore graph data, due to the complex structure, and the increasing size of graphs themselves. Graph summarization is a potential solution to this problem.

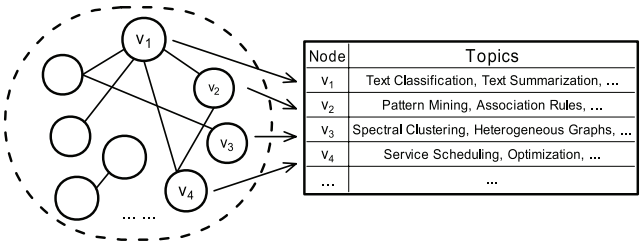
The goal of summarizing a large graph G is to obtain a concise graph representation G_S, which is smaller than G in size, for visualization or analysis. Although specific summarization representations can be various in different approaches, the main idea behind them is to construct a super-graph G_S with super-nodes and super-edges. The nodes in G are partitioned into several node sets and each node set is represented by a single super-node in G_S. Two super-nodes are connected by a super-edge in G_S if there exist edges in G between nodes from the two corresponding node sets. The basic assumption is that nodes in the same node set is similar to each other under certain criteria, which is called homogeneous in this paper, otherwise using a single node to represent them will not be reasonable.

In the literature, there are two major approaches for super-graph construction, where the main difference lies in how to create super-edges between two super-nodes. A strict approach [6] requires that a super-edge exists between two super-nodes in G_S only if every pair of nodes residing in the two corresponding super-nodes is connected by an edge in G. A relaxed approach [7], [10] allows two super-nodes to be connected with a super-edge in G_S if there is at least one connected node pair in G among all the node pairs summarized by the two super-nodes. Here, each super-edge is associated with a participation ratio to indicate the percentage of connected nodes in the two super-nodes among all the nodes in the two super-nodes.

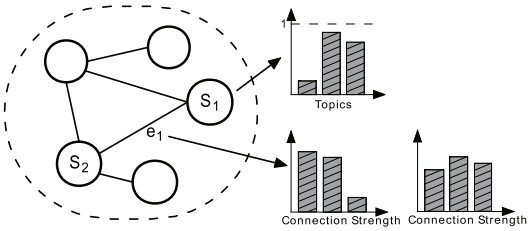
Unfortunately, both approaches have their disadvantages. In the strict approach, since only cliques or bipartite cliques can be represented by super-nodes according to the very rigorous requirement, the size of the summarized graph cannot be small in most cases, even when super-nodes are near-cliques, which makes the summarized super-graph still difficult to explore and access. In the relaxed approach, the issue lies in the quality of the summarization, which we will discuss soon. For example, if the participation ratio between two super-nodes is close to 1, it means almost all nodes in one super-node have neighbors in the other super-node. If the participation ratio is close to 0, it means almost no nodes have neighbors in the other super-node. So we can infer whether nodes in one corresponding node set may have edges connected to certain nodes in the other node set with high confidence. However, if the participation ratio is somewhat around 0.5, then the summarized super-nodes cannot provide much connection information of the neighborhood in the original graph. Because it implies that only partial nodes in one super-node have neighbors in the other super-node, and the chance of a node having neighbors almost equals the one of a random guess.

We focus on an information-preserving graph summarization for attribute graphs, which means the summarized representation must satisfy the quality criteria as much as possible. The summary for a graph in our solution consists of two parts: a super-graph and a list of probability distributions for each super-node and super-edge. **Figure 1** shows a conceptual example. A DBLP

[1] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong
[a)] yu@se.cuhk.edu.hk



(a) DBLP Co-Author Network



(b) Our Proposed Graph Summary

Fig. 1 An example of graph summarization.

co-author graph to be summarized is presented in Fig. 1 (a). Inside the dotted area is the structure information of the co-author graph, where nodes represent authors and edges represent collaborations between these authors. There is attribute information associated with authors possibly, for example, the table in Fig. 1 (a) associated with nodes shows the main research topics of each author. Figure 1 (b) shows our proposed summarized representation, where the summarized super-graph is within the dotted area. Each super-node represents a number of authors, and is affiliated with one topic distribution indicating the research topics of the authors in the super-node, as well as the homogeneity of these research topics. Each super-edge has two connection strength distributions indicating the homogeneity of the neighbor relationship between nodes in the two connected super-nodes in two different directions. We will have a careful analysis of the meaning of homogeneity in Section 2.

The major contributions of this research are summarized below.

- We focus on how to obtain an optimized approximate homogeneous partition on which a graph summarization can be constructed by relaxing both attribute requirement and structure requirement. Inspired by information theory, we propose a unified entropy model which unifies both attribute information and structural information.
- We propose a new lazy algorithm to compute the exact homogeneous partition by delaying the reconstruction of matrix, as well as two new approximate homogeneous algorithms aiming to find the optimized approximate partition.
- We conduct experiments on various real datasets and the results confirm that our proposed approaches can efficiently summarize a graph to achieve low average entropy.

The remainder of this paper is organized as follows. Section 2 starts a careful analysis of the graph summarization problem and Section 3 presents our concept of approximate homogeneous partition based on information theory. We introduce the proposed summarization framework in Section 4 and reported experimental results in Section 5. Related works are discussed in Section 6 and finally, Section 7 concludes this paper.

2. Problem Statement

An attribute graph G is a triple (V, E, Γ), where V and E are the node set and the edge set of the graph, respectively. Γ is a finite set of attributes, and each node $v \in V$ or edge $(u, v) \in E$ is mapped to one or more attributes in Γ, denoted as $\Gamma(v)$ or $\Gamma(u, v)$. Given $\Gamma(v) = (A_1, A_2, \cdots, A_d)$, let $\gamma(v) = (a_1, a_2, \cdots, a_d)$ denote the attribute value vector of v, where a_i is the value of attribute A_i. In this work, we concentrate on categorical attributes. For a categorical attribute A_i with l distinct values, we can represent an attribute value using a l-bitmap, where all bits are zero except for the bit which corresponds to the attribute value. To simplify the presentation in this paper, we assume that the edges of the graph to be summarized are of the same attribute value, but our framework can be extended to handle graphs which have multiple attributes associated with both nodes and edges.

Given an attribute graph, we aim to find a concise and interpretable summary which is friendly for users to explore and analyze. This can be done by partitioning all nodes V in a graph G into k homogeneous non-overlapping node sets $\{V_1, V_2, \cdots, V_k\}$, where the criterion of homogeneity is discussed later. Here, each V_i represents a non-empty subset of node set V. Let $\mathcal{P}$ denote the node partition $\{V_1, V_2, \cdots, V_k\}$, and let $\mathcal{P}(v)$ denote the unique node set V_i that a node v belongs. Furthermore, because a node v in a node set V_i has edges to link other nodes in another node set V_j, we use $N(v) = \{\mathcal{P}(v_k) | (v, v_k) \in E(G)\}$ to denote the set of V_j. In addition, for $V_j \in N(v)$, we use $|V_j|_v$ to denote the number of edges from v to any nodes in V_j.

Based on the homogeneous partition $\mathcal{P}$, a graph summarization G_S can be constructed as follows. A super-node S_i represents a node set V_i, for all node sets in $\mathcal{P}$, and all nodes of G summarized by a super-node in G_S have the same attribute values. The super-edges among super-nodes in G_S imply that every node of G summarized by a super-node has the same pattern of connecting nodes to other nodes summarized by other super-nodes. For example, suppose that S_i has super-edges to S_j, S_k, and S_l. It shows that every node of G summarized by S_i has edges to some nodes of G summarized by S_j, S_k, and S_l. In the following of this paper, we use V_i and S_i interchangeably.

Now the question is what is a homogeneous partition. In a homogeneous partition $\mathcal{P}$, every node set V_i in $\mathcal{P}$ is considered to be homogeneous, which consists of the following three criteria: First, nodes are homogenous according to the attribute information, i.e., nodes in the same node set must have the same attribute value vectors. Second, nodes are homogenous according to the neighbor information, i.e., if a node $v \in V_i$ connects to V_j, then all the nodes in V_i must connect to V_j. Third, nodes are homogenous according to the connection strength, which is measured in terms of edges. If V_i and V_j are connected, all nodes in V_i have the same number of edges to nodes in V_j. With these three criteria, we present the definition of exact homogeneous partition below.

Definition 1 (**Exact Homogeneous Partition**) An exact homogeneous partition $\mathcal{P} = \{V_1, V_2, \cdots, V_k\}$ of a graph $G = (V, E, \Gamma)$ satisfies the following three criteria for every node $v \in V$: (1) $\gamma(v) = \gamma(V_i)$; (2) $N(v) = N(V_i)$; and (3) $|V_j|_v = |V_j|_{V_i}$, for every $V_j \in N(v)$. Here, $\gamma(V_i)$ denotes the common attribute values of

nodes in V_i under the assumption that all nodes in V_i have the same attribute values. $N(V_i)$ denotes the common node sets for every node in the node set V_i, and $|V_j|_{V_i}$ denotes the common number of edges from every node in V_i linking to nodes in node set V_j.

The above definition of exact homogeneous partition extends the definition of exact grouping in Ref. [7]. The difference is that the exact grouping in Ref. [7] only considers the first two criteria but not the third one. Without the third one, nodes in a certain node set having more edges connecting to another node set, are considered to be the same as the one having less edges, which is obviously not reasonable. For example, in a DBLP co-author work, authors with more collaborations to a certain research group are more important than authors having few collaborations. It is not reasonable to place them together into the same node set apparently.

A summary G_S of a graph G constructed by an exact homogeneous partition can be considered as the best summarization with respect to the homogeneity criterion, since nodes in the same node sets are exactly the same in terms of attribute and structure information. Unfortunately, due to the high complexity of graph attributes and structures, as well as the increasing size of graph itself, such exact homogeneous partition cannot achieve a high compression ratio. The size of G_S based on the exact homogeneous partition is too large to serve as a graph summarization, which makes it beyond possible for users to handle. As we will see later in the experimental results, the size of G_S based on exact homogeneous partition can be almost as large as G. To solve this issue, we need to relax partial or all the criteria in Definition 1. The approach in Ref. [7] loosens only the second criterion, by allowing nodes in V_i connect to similar node sets of V_j but not necessarily to be the same. But it still requires that all attribute values of nodes in the same node set V_i must be exactly the same vector.

It is questionable if it is sufficient to relax only the second criterion in Definition 1 due to the following issues: (1) Keeping the same attribute vector in each node set makes it very difficult to handle a graph with multiple attributes, in particular, when the number of attributes is not small. Suppose a node has m attributes and each attribute has d possible values, there are total d^m possible combinations of these values. Though the real existing combinations may not be so many, it is still impossible to find a partition of a relative small size, say k, such that all nodes in the same node set V_i have the same attributes, when k is less than the number of existing combinations. (2) In the third criterion in Definition 1, it requests that all nodes in the same node set V_i should have the same number of edges connecting to nodes in any other node set. Due to the various possibility of neighborhood structures, this can also lead to a graph summary which is not much smaller than the original graph G.

To achieve compact summarization G_S, we propose to relax all the criteria in Definition 1. In order to relax these criteria, a quality function for each criterion is needed to control the quality of relaxation. Let us first give a high level definition for approximate homogeneous partition, and explain it later.

Definition 2 (**Approximate Homogeneous Partition**) Given a graph $G = (V, E, \Gamma)$, a number k, a graph node partition $\mathcal{P}$ is called approximate homogeneous partition, if it satisfies the following three criteria for every $V_i \in \mathcal{P}$. (1) $Q_\gamma(V_i) \leq \epsilon_1$; (2) $Q_{N(v_k)}(V_i) \leq \epsilon_2$; and (3) $Q_{|V_j|_{v_k}}(V_i) \leq \epsilon_3$, $\forall V_k \in (N(v_i) \cup N(v_j))$. Here, let $v_k \in V_i$, $Q_\gamma(\cdot)$, $Q_{N(v_k)}(\cdot)$, $Q_{|V_j|_{v_k}}(\cdot)$ are three quality measure functions, and ϵ_1, ϵ_2, and ϵ_3 are three thresholds to control the quality of the partition.

In an approximate homogeneous partition, nodes in the same node set are considered to be homogeneous as long as their attributes and neighborhood relationship patterns to other node sets are similar to each other. Besides, a overall ranking function is necessary to rank the partitions based on the overall summarization quality to obtain the best one. Suppose $R(\cdot)$ is the function that reflects the three criteria in Definition 2 to measure the quality of approximate homogeneous partition, we study how to compute an approximate homogeneous partition $\mathcal{P}$ of size k for a graph $G = (V, E, \Gamma)$ by minimizing the ranking function $R(\mathcal{P})$. The key issues are as follows. What quality measure and the function $R(\mathcal{P})$ should we use? Can we make it threshold free (without ϵ_1, ϵ_2, and ϵ_3)? We address these issues in the following sections.

3. An Approximate Homogeneous Partition Based on Information Theory

In this paper, we propose an information-preserving criterion, based on information theory. We first review some background knowledge, followed by detailed discussions about how to utilize a unified entropy model to measure the quality of the three relaxations in Definition 2.

Let x_i be a boolean random binary variable and $p(x_i)$ be its Bernoulli distribution function, $\mathbf{p(x)} = [\, p(x_1), \cdots, p(x_d)]$ is a Bernoulli distribution vector [8] over d independent Boolean random variables $x_1, \cdots, x_d$. Let $\mathbf{b_j}$ denote a binary d-element vector. Given a set of binary vectors $D = \{\mathbf{b_1}, \cdots, \mathbf{b_n}\}$, under the assumption of independence, the probability by which they are generated by a distribution vector θ is estimated as

$$P(D|\theta) = \prod_{\mathbf{b_j} \in D} \prod_{i=1}^{d} p(x_i = \mathbf{b_j^i}), \tag{1}$$

where $\mathbf{b_j^i}$ is the ith element of the binary vector $\mathbf{b_j}$. The best θ, which fits the model, is

$$\hat{\theta} = \arg\max_{\theta} \log(P(D|\theta)). \tag{2}$$

The well-known solution based on the maximum likelihood estimation is

$$p(x_i = 1) = \frac{\sum_{\mathbf{b_j} \in D}^{n} \mathbf{b_j^i}}{|D|}. \tag{3}$$

We use information theory to measure the quality of these distribution vectors. Recall that in information theory, entropy [2] is a measure of the uncertainty (randomness) associated with a random variable X, which is defined as

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x). \tag{4}$$

Consider a random variable x_i whose value domain is $\{0, 1\}$, the

probability of x_i equals 0 or 1 is $p(x_i = 0)$ or $p(x_i = 1)$. The entropy of an unknown sample of the random variable x_i is maximized when $p(x_i = 0) = p(x_i = 1) = 1/2$, which is the most difficult situation to predict the value of an unknown sample. When $p(x_i = 0) \neq p(x_i = 1)$, we know that the value of the unknown sample is more likely to be either 0 or 1 accordingly, which is quantified in a lower entropy. The entropy is zero when $p(x_i = 0) = 1$ or $p(x_i = 1) = 1$. For a Bernoulli distribution vector $\mathbf{p}(\mathbf{x})$, assuming the contained random variables are independent of each other, the total entropy of a Bernoulli distribution vector is

$$H(\mathbf{p}(\mathbf{x})) = -\sum_{i=1}^{d}\sum_{x_i=0}^{1} p(x_i)\log_2 p(x_i). \quad (5)$$

If binary vectors within the set D are similar to each other, or homogeneous, then for each random variable x_i, most of its values should be similar, resulting in a low $H(\mathbf{p}(\mathbf{x}))$.

3.1 Entropy-Based Relaxations of Criteria

In the following part, we discuss the three relaxations in Definition 2. Based on these observations, we can measure the quality of the three relaxations in a unified model inspired by information theory.

Observation for Q_γ. For each node $v_i \in V$, $\gamma(v_i) = (a_1, \ldots, a_d)$ is the attribute vector of v_i, where a_i is the value of attribute A_i. As mentioned, we represent categorical attribute values as bitmaps, so we also use a_i to indicate the bitmap when there is no confusion. For a certain node set, V_j, in an approximate homogeneous partition, the attribute information of each node $v_i \in V_j$ is in form of a binary vector by concatenating these bitmaps together, denoted as $\mathbf{a} = (a_1, \ldots, a_d)$. The attribute information of a node set is homogeneous if the corresponding binary vectors are similar to each other. A binary Bernoulli distribution vector can be estimated from these vectors by Eq. (2). When the majority of nodes in a node sets share a same attribute value, the corresponding bit in the Bernoulli distribution vectors approaches to 1. When the majority of nodes do not have a certain attribute value, the corresponding bit approaches to 0. In this case, we can infer from the Bernoulli distribution where a node has or has not a certain attribute value by the expected value of the corresponding bit. Then it is better if each column in the Bernoulli distribution vector approaches to 1 or 0. When the value is 0.5, it is the worst case that we are uncertain to infer any useful attribute information, since the confidence of the expected value is like the one of a random guess. Entropy is an excellent quality measure in this case, and low entropy means high confidence based on Eq. (5).

As shown in **Fig. 2**, each row in the top table represents a node in the graph. For each node, there are four attributes: (a_1, a_2, a_3, a_4). The first three rows belong to the super-node S_1 (or node set V_1), while the remaining belong to the super-node S_2 (or node set V_2). It is easy to see that nodes in S_1 are more similar to each other than nodes in S_2. The corresponding Bernoulli distribution vectors for S_1 and S_2, are represented in the lower table, as well as their entropy values. As we can see, the entropy value of S_1 is much lower than that of S_2, which is consistent to

	Node	a_1	a_2	a_3	a_4
S_1	v_1	1	1	1	0
	v_2	1	1	1	1
	v_3	0	1	1	1
S_2	v_4	1	1	0	0
	v_5	1	0	0	1
	v_6	0	0	1	1

	Entropy	$p(a_1=1)$	$p(a_2=1)$	$p(a_3=1)$	$p(a_4=1)$
S_1	1.85	0.66	1	1	0.66
S_2	3.70	0.66	0.33	0.33	0.66

Fig. 2 Entropy-based attribute homogeneity.

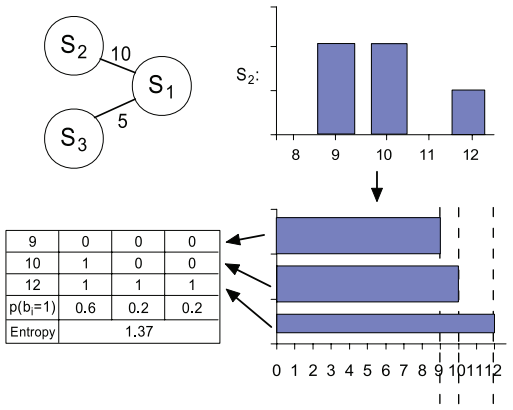


Fig. 3 Entropy-based homogeneity of connection strength.

that nodes in S_1 are more similar to each other.

Observation for $Q_{N(v_k)}$ Nodes in the same homogeneous node set V_i should have similar neighborhood relationship in the super-graph. Note that if a node set has good quality according to the third criterion, it must be also good by the second one, since the second criterion is in fact a special case of the third one. If there is only one neighbor for nodes in a node set, then the second criterion and the third one are the same. Obviously, $Q_{|V_j|_{v_k}}$ is a stronger criterion than $Q_{N(v_k)}$, because $Q_{|V_j|_{v_k}}$ measures the quality based on not only whether there are connections between nodes in V_i and V_j, but also the number of connections for $v_k \in V_i$. Therefore, we can ignore $Q_{N(v_k)}$, and concentrate on $Q_{|V_j|_{v_k}}$ which we will discuss next.

Observation for $Q_{|V_j|_{v_k}}$ Consider a super-graph G_S, and we use V_i (node set) and S_i (super-node) in G_S interchangeably. If there is a super-edge between super-nodes S_i and S_j, then nodes in S_i should have similar total number of edges to nodes in S_j. As discussed, it is not appropriate to put two nodes together, whose connection strengths to a certain node set differ a lot, because their importance to the node set is not in the same level. We can keep two histograms for each super-edge (S_i, S_j), namely, S_i-to-S_j and S_j-to-S_i, to record the distributions of neighbors in S_j (S_i) of nodes in S_i (S_j). We explain it by using an example as shown in **Fig. 3**. There are three node sets (super-nodes) in the partition: S_1, S_2, and S_3. At the upper left corner in Fig. 3, it shows how these super-nodes are connected by super-edges. For example, it indicates that every node in S_2 has 10 neighbors in S_1 on average. The histogram of S_2-to-S_1 is drawn on upper right corner, where the x-axis indicates the number of neighbors in S_1

for a node in S_2. The y-axis indicates the number of nodes in S_2 corresponding to each value on x-axis. The histogram shows that there are 2 nodes within S_2 which have 9 edges connected to nodes in S_1, 2 nodes which have 10 edges and 1 node which has 12 edges. Intuitively, a homogeneous node set should have a tight spread range on x-axis in the histogram. Again, entropy is a good measure to show how homogeneous inside each node set. To do so, we present the histogram in another way as shown in the bottom right corner. The x-axis still indicates the number of neighbors in S_1 for a node in S_2, while the thickness of each bar indicate the number of nodes in S_1 corresponding to each value on x-axis. Based on this intuition, we transform each bar in the bottom histogram to a binary vector of all 1's. For example, for bar indicating the number of neighbors is 9, a binary vector of length 9 is constructed. We first concatenate 0's at the end of each binary vector to make them of the same length. Then we remove the common 1's in the suffix of these vectors, because the entropy on these columns are all zero and we focus on only the difference in these binary vectors. The remaining binary vectors are shown in the bottom left table in Fig. 3. Similar to Q_γ, a Bernoulli distribution vector is learned from these binary vectors. The more similar these vectors are, the lower entropy of the distribution vector is, as shown in the table.

In summary, the homogeneity of a node set can be measured by the concept of entropy of these Bernoulli Vectors. Let $\mathbf{e} = (S_i, S_j)$ denote a super-edge between two super-nodes S_i and S_j, and let v_i denote a node in super-node S_i. The entropy of super-node S_i consists of two parts: the attribute part and the neighborhood connection strength part. We propose to convert the attribute homogeneity into neighbor relationship homogeneity to unify the two parts. **Figure 4** shows our conversion. For each attribute value, we add an additional node in the original graph G. Here we have four attribute values $\{a_1, a_2, a_3, a_4\}$, so we add four corresponding nodes in G. For each node, we add edges between it and those nodes corresponding to its attribute values. For example, in Fig. 4, node v_1 has attribute values $\{a_1, a_2, a_3\}$, so we add edges between v_1 and nodes representing a_1, a_2, and a_3. In this way, we convert the attribute homogeneity into neighbor relationship homogeneity. Then, we apply the same approach as we have discussed in Observation for $Q_{|V_j|_{v_k}}$ to calculate the attribute homogeneity. The entropy for S_i is

$$Entropy(S_i) = \sum_{j=1}^{k+l} H(\mathbf{p}(\mathbf{b_j^m} = \mathbf{1})), \tag{6}$$

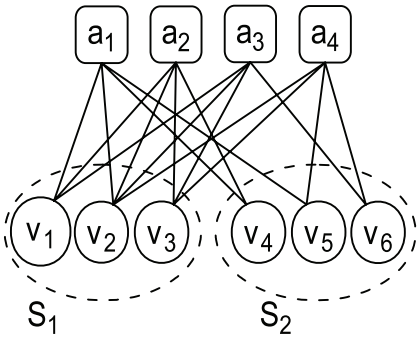


Fig. 4 The conversion from attributes to nodes.

where k is a user-given parameter for controlling the number of node subsets in the partition $\mathcal{P}$ and l is total number of distinct attribute values, $\mathbf{b^m}$ is the mth element in $\mathbf{b}$, and $\mathbf{p}(\mathbf{b_j^m} = \mathbf{1})$ is the Bernoulli distribution vector estimated by Eq. (3) for S_i to S_j or a_j, depending on whether the connections are to a super-node or attribute value node. As we can see from the above analysis, the total entropy for every super-node in exact homogeneous partition is zero.

Users might prefer attribute homogeneity over connection strength homogeneity or vice versa. To achieve this, we allow users to assign weights during the entropy calculation as follows in Eq. (7).

$$\begin{aligned} &WeightedEntropy(S_i) \\ &= \lambda \sum_{j=1}^{l} H(\mathbf{p}(\mathbf{a_j^m} = \mathbf{1})) + (1-\lambda) \sum_{j=1}^{k} H(\mathbf{p}(\mathbf{b_j^m} = \mathbf{1})) \end{aligned} \tag{7}$$

Now, $\mathbf{p}(\mathbf{a_j^m} = \mathbf{1})$ is the Bernoulli distribution vector estimated by Eq. (3) for S_i to node a_j, and $\mathbf{p}(\mathbf{b_j^m} = \mathbf{1})$ is the Bernoulli distribution vector estimated by Eq. (3) for S_i to super-node S_j. When λ equals 1/2, the entropy score is one half of the entropy computed by Eq. (6). And the weighted entropy for every super-node in the exact homogeneous partition is still zero.

The optimized approximate homogeneous partition is the partition that minimizes the ranking score of the super-graph, which is the total weighted entropy of all nodes:

$$R(\mathcal{P}) = \sum_{S_i \in \mathcal{P}} |S_i| \times WeightedEntropy(S_i), \tag{8}$$

where $|S_i|$ is the number of nodes contained in S_i. What we study next is how to find the optimized approximate homogeneous partition $\mathcal{P}$ for a given graph G. Based on $\mathcal{P}$, the graph summarization G_S can be constructed.

4. Homogeneous Graph Summarization

In this section, we present the algorithms for exact homogeneous partition and approximate homogeneous partition.

4.1 A Lazy Algorithm for Exact Homogeneous Partition

Exact homogeneous partition is the best summary in terms of homogeneity, and we extend the algorithm in Ref. [7] to compute exact homogeneous partition using a simple but effective approach.

Figure 5 outlines the procedures to compute the exact homogeneous partition based on Definition 1. Recall that $\mathbf{a}$ is the concatenated attribute vector for nodes. Suppose there are m distinct attribute vectors, the nodes in graph G are partitioned into m groups first according to the distinct vectors. Then the algorithm constructs an $n \times m$ node-to-group matrix M, where $M(i, j)$ is the number of v_i's neighbors in S_j. One thing worth noting is that nodes belonging to the same group are stored adjacently in M and the order of groups in rows is the same as the order of groups in columns. At line 5 in Fig. 5, the algorithm marks the split positions using a binary vector of length n. After inspecting all the groups, the algorithm reconstructs the node-to-group matrix M based on the marked split positions.

Lazy Exact Homogeneous Partition

Input: A graph $G = (V, E, \Gamma)$.
Output: The exact homogeneous partition $\mathcal{P}$.

1. Partition V into m node sets based on distinct attribute value vectors **a**;
2. Construct an $n \times m$ node-to-group matrix M;
3. **while** True
4. Sort rows within each group;
5. Let $splitflag$ be an all-zero binary vector of length n;
6. **for** each cell $M(i, j)$ in M
7. **if** $M(i, j) \neq M(i + 1, j)$
8. $splitflag(i)$ = true;
9. **if** $splitflag$ is all false
10. break;
11. Split each sets according to $splitflag$ to form m' new node subsets;
12. Reconstruct the $n \times m'$ node-to-group matrix M;
13. Output the exact homogeneous partition $\mathcal{P}$.

Fig. 5 Lazy exact homogeneous partition.

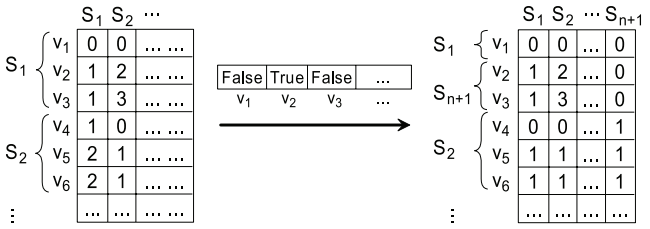


Fig. 6 Data structure for lazy exact homogeneous partition.

Because the matrix reconstruction is costly, we do not reconstruct M immediately after a split position is found. There are many unnecessary reconstructions during the split operations. An example is shown in **Fig. 6**. Suppose the left matrix is the initial node-to-group matrix after sorting, and we find S_1 should be split into two subsets. If we reconstruct the matrix in each loop, the matrix will be like the one on the right. As we can see that the next node set to be split is S_{n+1}, the last reconstruction of matrix is not necessary. Instead, we mark these split positions using a binary array and reconstruct only once after we check all the possible positions. We call it lazy exact homogeneous partition.

Next we will present two algorithms for approximate homogeneous partition: an agglomerative merging algorithm and a divisive k-means algorithm.

4.2 An Agglomerative Algorithm for Approximate Homogeneous Partition

As discussed, though exact homogeneous partition is of the highest quality based on homogeneity, its size is almost as large as the original graph. To further reduce the size of a summary of exact partition, we propose an agglomerative algorithm which is presented in **Fig. 7**, which takes the exact homogeneous partition $\mathcal{P}$ as the input. The main idea of the agglomerative algorithm is to maintain a matrix to record the change in total weighted entropy for each pair of node sets if they are merged, and merge the pair with the minimum value repeatedly. Each merging will decrease the total number of node sets by one.

In the loop from line 3 to line 6 in Fig. 7, the algorithm calculates the initial value of total weighted entropy of the exact partition after merging each possible node pair (V_i, V_j). Recall that the total weighted entropy of an exact partition is zero. At the

The Agglomerative Algorithm for Approximate Partition

Input: The exact homogeneous partition $\mathcal{P} = \{V_1, \cdots, V_m\}$; a number k.
Output: The approximate homogeneous partition $\mathcal{P}_A$.

1. $\mathcal{P}_A = \mathcal{P}$;
2. Let Δ be an $m \times m$ empty matrix;
3. **for** each subset pair V_i and V_j in $\mathcal{P}$
4. $\mathcal{P}_{ij} = \mathcal{P}_A \cup \{V_i \cup V_j\} \setminus \{V_i, V_j\}$;
5. $\Delta_{ij} = R(\mathcal{P}_{ij})$; /* Eq. (8) */
6. **while** $|\mathcal{P}_A| > k$
7. Let (V_l, V_m) be the pair of node sets with the minimum Δ_{ij};
8. $\mathcal{P}_A = \mathcal{P}_A \cup \{V_l \cup V_m\} \setminus \{V_l, V_m\}$;
9. Update Δ based on V_l and V_m;
10. Output the approximate homogeneous partition $\mathcal{P}_A$.

Fig. 7 The agglomerative algorithm for approximate partition.

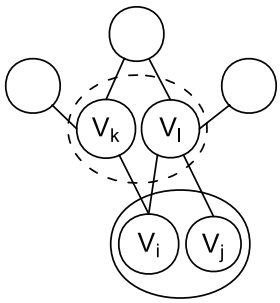


Fig. 8 An example of updating matrix Δ.

end of line 6, Matrix $\Delta(i, j)$ stores the change of total weighted entropy if we merge node set V_i and V_j. Note that we only use the upper half of $\Delta(i, j)$ since $\Delta(i, j) = \Delta(j, i)$. In each iteration from line 7 to line 11, the algorithm merges the pair of node sets with the minimum change in total weighted entropy to generate new partition, and update matrix Δ.

Now, the problem is how to update matrix Δ. A naive way is to recompute the whole Δ based on the current partition $\mathcal{P}_A$, which is slow and not necessary, since merging one pair of nodes only affects partial values in Δ. Suppose (V_i, V_j) is the pair to merge, and $i < j$. The merging is done by adding all nodes in V_j to V_i and deleting V_j. This operation only affects the values in two types of cells in Δ. The first type is the cells for pairs involving V_i, which is easy to understand, since V_i is now changed to $V_i \cup V_j$. Thus, we have to recompute the change in total weighted entropy for these pairs of node sets.

The other type is the pairs of node sets involving the neighbors of (V_i, V_j). An example is shown in **Fig. 8**. Suppose V_k and V_l are neighbors of (V_i, V_j). It does not matter whether V_k and V_l are both neighbors of (V_i, V_j), or just one of them is. Before the merging of (V_i, V_j), $\Delta(k, l)$ stores the change in total weighted entropy if we merge V_k and V_l, while V_i and V_j are still separated. Once V_i and V_j are merged, the change of $\Delta(k, l)$ consists of three parts:

(1) $|V_k \cup V_l| \times WeightedEntropy_{\{V_i, V_j\}}(V_k \cup V_l) - |V_k| \times WeightedEntropy_{\{V_i, V_j\}}(V_k) - |V_l| \times WeightedEntropy_{\{V_i, V_j\}}(V_l)$;

(2) $|V_i| \times WeightedEntropy_{\{V_k \cup V_l\}}(V_i) + |V_j| \times WeightedEntropy_{\{V_k \cup V_l\}}(V_j) - |V_i| \times WeightedEntropy_{\{V_k, V_l\}}(V_i) - |V_j| \times WeightedEntropy_{\{V_k, V_l\}}(V_j)$;

(3) The change of neighbors of (V_l, V_k) except V_i and V_j.

The subscript of $WeightedEntropy$ indicates the portion of the

total weighted entropy related to node sets in the subscript. As we can see, after the merging of (V_i, V_j), the third part does not change, so we only need to recompute the first part and the second part.

4.3 A Divisive k-Means Algorithm for Approximate Homogeneous Partition

In this section, we present a divisive k-means based approximate algorithm to find the optimized approximate homogeneous partition using the Kullback-Leibler (KL) divergence. The Kullback-Leibler divergence [8] is a measure of the difference between two distribution vectors $\mathbf{p}$ and $\mathbf{q}$, which is defined as follows,

$$KL(\mathbf{p} \parallel \mathbf{q}) = \sum_{i=1}^{d} \sum_{x_i=0}^{1} p(x_i) \log \frac{p(x_i)}{q(x_i)}. \quad (9)$$

In the view of information theory, KL divergence measures the expected number of extra bits required to encode samples from $\mathbf{p}$ when using a code based on $\mathbf{q}$, rather than using a code based on $\mathbf{p}$. We assume the Bernoulli distribution vector for a certain node group S_i is $\mathbf{p}$. For each node in group S_i, let $\mathbf{q}$ be the Bernoulli distribution vector for a node $v_i \in S_i$. Both $\mathbf{p}$ and $\mathbf{q}$ are the concatenated vector of $\mathbf{a_j}$ and $\mathbf{b_j}$ in Eq. (7). Then we have

$$\begin{aligned}
&-\sum_{v_i \in S_i} KL(\mathbf{p}(\mathbf{x}) \parallel \mathbf{q}(\mathbf{x})) \\
&= -\sum_{v_i \in S_i} \sum_{i=1}^{d} \left(p(x_i = 1) \log \frac{p(x_i = 1)}{q(x_i = 1)} - p(x_i = 0) \log \frac{p(x_i = 0)}{q(x_i = 0)} \right) \\
&= -\sum_{v_i \in S_i} \sum_{i=1}^{d} (p(x_i = 1)(\log p(x_i = 1) - \log q(x_i = 1)) \\
&\quad + p(x_i = 0)(\log p(x_i = 0) - \log q(x_i = 0))) \\
&= \sum_{v_i \in S_i} \sum_{i=1}^{d} (-p(x_i = 1) \log q(x_i = 1) - p(x_i = 0) \log q(x_i = 0) \\
&\quad - H(\mathbf{p})) \\
&= n(s_i) \sum_{i=1}^{d} \left(-\frac{\sum_{v_i \in S_i} p(x_i = 1)}{n(s_i)} \log q(x_i = 1) \right. \\
&\quad \left. - \frac{\sum_{v_i \in S_i} p(x_i = 0)}{n(s_i)} \log q(x_i = 0) \right) \\
&= n(s_i) \sum_{i=1}^{d} (-q(x_i = 1) \log q(x_i = 1) - q(x_i = 0) \log q(x_i = 0)) \\
&= n(s_i) * H(\mathbf{p}(\mathbf{x}))
\end{aligned}$$

Thus, the optimized approximate homogeneous partition that minimizes $R(\mathcal{P}_A)$ is the partition that minimizes the sum of $KL(\mathbf{p}(\mathbf{x}) \parallel \mathbf{q}(\mathbf{x}))$, which leads to the following divisive k-means approximate algorithm presented in **Fig. 9**.

The algorithm starts from one node set by putting all the nodes in G together. In each loop from line 2 to 20 in Fig. 9, the algorithm first splits the node set with the maximum total weighted entropy, and then applies k-means clustering method based on KL-divergence. The split procedure is from line 3 to line 14. First, a random perturbation of nodes in the node sets with the maximum weighted entropy is performed. Then we inspect these nodes one by one according to the order in the perturbation. If moving the node from the old node set to a new node set decreases the total weighted entropy, we move it, otherwise, it stays in the old node set. Once the split is finished, the algorithm performs k-means clustering from line 16 to line 20, to minimize the sum of KL-divergence. When the number of node sets equals k, the approximate homogeneous partition $\mathcal{P}_A$ is returned.

The Divisive k-Means Algorithm for Approximate Partition

Input: A graph $G = (V, E, \Gamma)$, a number k;
Output: The approximate homogeneous partition $\mathcal{P}_A$

1. $\mathcal{P}_A = \{V\}$;
2. **while** $|\mathcal{P}_A| < k$
3. Let V_m be the node set with the maximum value of $|V_m| \times WeightedEntropy(V_m)$;
4. Generate a random perturbation L of nodes in V_m,
5. $V_i = V_m$;
6. $V_j = \emptyset$;
7. **for** $v \in L$
8. $we = |V_i| \times WeightedEntropy(V_i) + |V_j| \times WeightedEntropy(V_j)$;
9. $we' = (|V_i| - 1) \times WeightedEntropy(V_i \setminus \{v\}) + (|V_j| + 1) \times WeightedEntropy(V_j \cup \{v\})$;
10. **if** $we' < we$
11. $V_i = V_i \setminus \{v\}$;
12. $V_j = V_j \cup \{v\}$;
13. $\mathcal{P}_A = \mathcal{P}_A \cup \{V_i, V_j\} \setminus V_m$;
14. **repeat**
15. Evaluate the Bernoulli distribution vectors $\mathbf{a_j}$'s and $\mathbf{b_j}$'s for $V_k \in \mathcal{P}$;
16. Concatenate $\mathbf{a_j}$'s and $\mathbf{b_j}$'s together for $V_k \in \mathcal{P}_A$;
17. Assign each node $v \in V(G)$ to a new cluster V_k according to the Kullback-Leibler divergence in Eq. (9);
18. **until** the change of $R(\mathcal{P}_A)$ is small or no more changes of the cluster assignment.

Fig. 9 The divisive k-means algorithm for approximate partition.

5. Experimental Results

In this section, we report the experimental results of our proposed summarization framework on various real datasets from DBLP Bibliography [*1]. The algorithms are implemented by using matlab and C++. All the experiments were run on a PC with Intel Core-2 Quad processor and 3 GB RAM, running Windows XP. One thing worthy noting is that we did not optimize our sources for multiple core environment.

5.1 Datasets

We construct a co-author graph with top authors and their co-author relationships, where the authors are from three research areas: database (DB), data mining (DM) and information retrieval (IR). Based on the publication titles of the selected authors, we use a topic modeling approach [3], [9] to extract 100 research topics. Each extracted topic consists of a probability distribution of keywords which are most representative for the topic.

By using authors from partial or all areas, we construct four real datasets in our experiments. The basic statistics of the four datasets are presented in **Table 1**, including the number of nodes, the number of edges and the average degree of nodes. There are total 100 topics in the original datasets and in the experiments, we remove the topics from authors, whose probabilities are extremely small. Each author is related to several topics whose

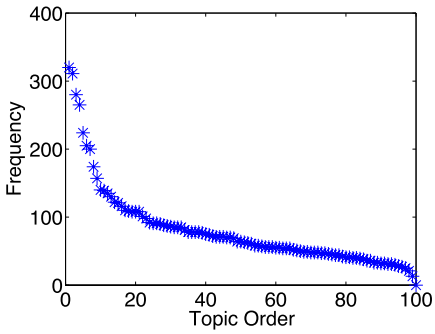
[*1] http://www.informatik.uni-trier.de/~ley/db/

Table 1 The DBLP bibliography datasets.

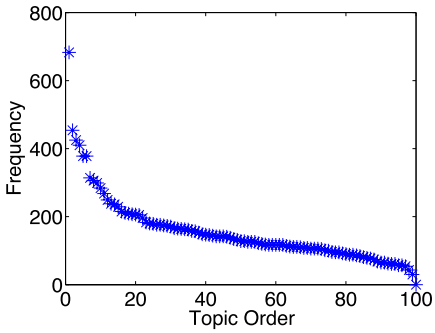
	Datasets	# of Nodes	# of Edges	Average Degree
D1	DM	1695	2282	1.35
D2	DB	3328	11379	3.42
D3	DB+DM	5023	15262	3.03
D4	DB+DM+IR	6184	18710	3.02

Table 2 The keywords of topics.

Topics #	Keywords
32	text, classification, vector, categorization
66	mining, patterns, frequent, sequential...
76	service, scheduling, extending, media
80	clustering, matrix, density, spectral



(a) D1 (DM)



(b) D2 (DB)

Fig. 10 Topic frequency in dataset D1 and dataset D2.

probabilities are larger than 5%. Example of the topics are shown in **Table 2**, as well as the top keywords in each topic.

All these topics are not of equal importance. We present the frequency distribution of topics in datasets D1 (DM) and D2 (DB) in **Fig. 10** in descending order. The x-axis is the topic order and the y-axis the frequency of a topic which is defined as the number authors doing research on the topic. For dataset D1 in Fig. 10 (a), the majority of topics appear less than 100 times, while only less than ten topics are very hot among authors. For dataset D2, the frequencies of most topics are below 200.

5.2 Exact Homogeneous Partition

Table 3 presents a comparison between the number of groups and the nodes in the original graphs. The number of distinct attribute vectors and the number of exact groups are quite close to the number of nodes. Therefore, the exact homogeneous partition cannot obtain a graph summary of a reasonable size. **Figure 11** shows the graph structure of the main connected component generated by exact partition algorithm on dataset DM, which is very large and not possible for users to explore. In **Fig. 12**, we compare the running time of our lazy exact homogeneous partition

Table 3 The DBLP bibliography datasets.

	D1	D2	D3	D4
# of Nodes	1695	3328	5023	6184
# of Distinct Attribute Vectors	1492	2931	4401	5409
# of Exact Groups	1604	3219	4829	5912

Fig. 11 Exact homogeneous summarization of dataset DM.

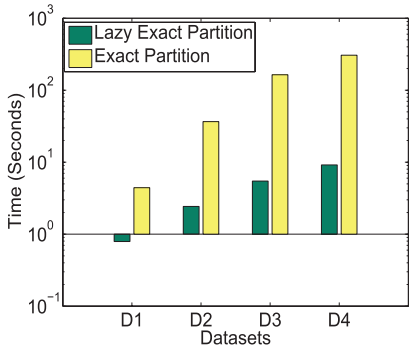


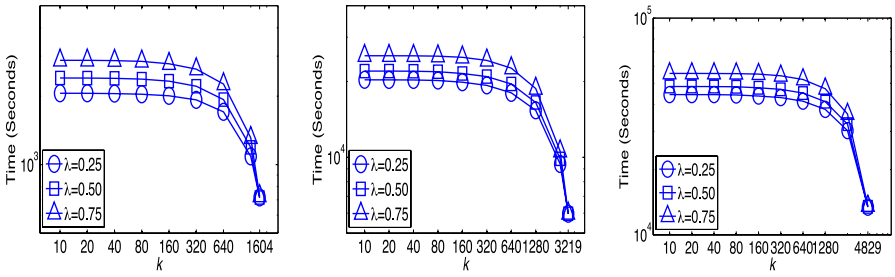
Fig. 12 The running time of exact Algorithms.

algorithm with the exact partition algorithm, denoted as exact partition. Unlike the lazy partition algorithm, the exact partition algorithm reconstructs the matrix M immediately after discovering a split position, as shown in Fig. 5. The lazy exact partition algorithm is more than 10 times faster than the exact partition algorithm due to the saved time of matrix construction.

5.3 Approximate Homogeneous Partition

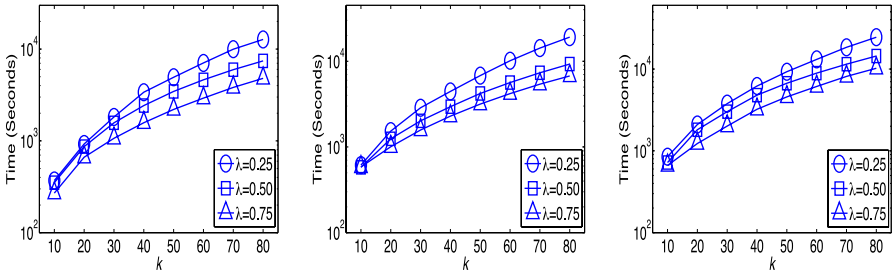
We performed our approximate homogeneous algorithms using three values of λ: 0.25, 0.5 and 0.75. Due to the high time complexity, we only apply our agglomerative algorithm on datasets D1, D2 and D3. **Figure 13** shows the running time of the agglomerative algorithm performed on these three datasets. Both the x-axis and the y-axis are in log scale. We performed our divisive k-means algorithm on all the four datasets and report the results for datasets D2, D3 and D4 in **Fig. 14**. k-means algorithm is almost 10x times faster than the agglomerative algorithm when k is small, which common in real applications. An interesting phenomenon is that the running time of small λ in the agglomerative algorithm is less, while the running time of large λ in the divisive k-means algorithm is less. The reason is that the agglomerative algorithm starts from the exact partition, and a large λ cannot boost the affect of attribute too much. In the divisive k-means algorithm, a large λ usually means less iterations in the k-means clustering algorithm, as we observed during the experiments.

Figure 15 (a) shows the average entropy of the approximate



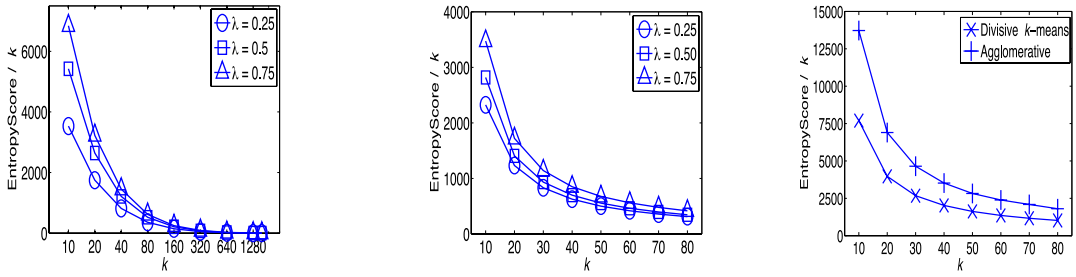
(a) D1 (DM) (b) D2 (DB) (c) D3 (DB+DM)

Fig. 13 The running time of the agglomerative algorithm.



(a) D2 (DB) (b) D3 (DB+DM) (c) D4 (DB+DM+IR)

Fig. 14 The running time of the divisive k-means algorithm.

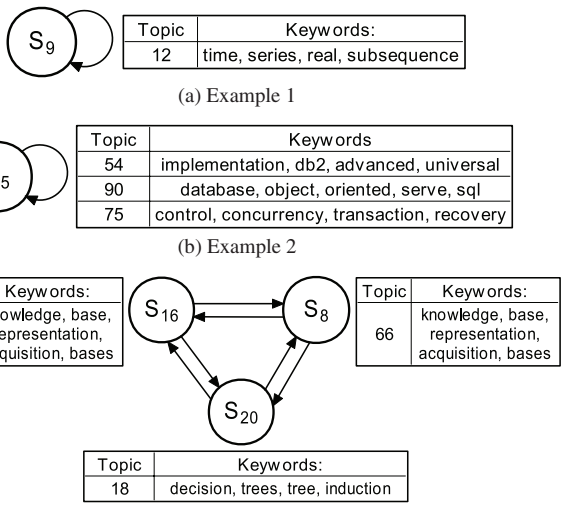


(a) The agglomerative algorithm on D1 (DM) (b) The divisive k-means algorithm on D1 (DM) (c) D2 (DB), $\lambda = 0.5$

Fig. 15 $R(\mathcal{P}_A)/k$.

homogeneous partition by the agglomerative algorithm on dataset D1, where we present the average entropy for different values of group number and λ. As the group number shrinks, the average entropy increases. Since the input of the bottom-up approximate algorithm is the exact homogeneous partition, the average entropy is 0 at the beginning. Figure 15 (b) shows the average entropy of the approximate homogeneous partition by the divisive k-means algorithm on dataset D1. As we can see, when k is in the range from 10 to 80, the summary generated by the divisive k-means algorithm is much better that one generated by the agglomerative algorithm, in terms of the average entropy. Figure 15 (c) reports the results on dataset D2 by these two algorithms when λ is 0.5, which once again shows that the divisive k-means algorithm performs better than the agglomerative algorithm, when k is small.

We present some interesting examples from summary of dataset D2 (DB), generated by the agglomerative algorithm when the group number is 60. For ease of presentation, we remove the distribution on edges, while the values of the entropy for these distributions are small. Each node in **Fig. 16** represents a group of researchers. The tables in Fig. 16 present the topic number and the main keywords of each topic. Figure 16 (a) shows that a group of researchers in time series domain tend to cooperate with themselves, where the size of node S_9 is 25. Figure 16 (b)



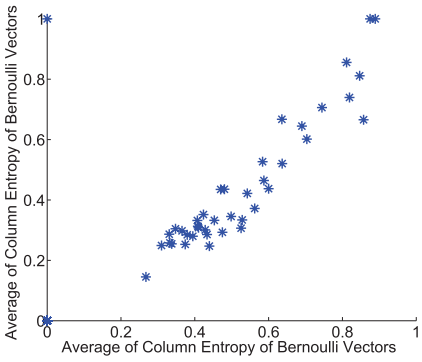
(a) Example 1

(b) Example 2

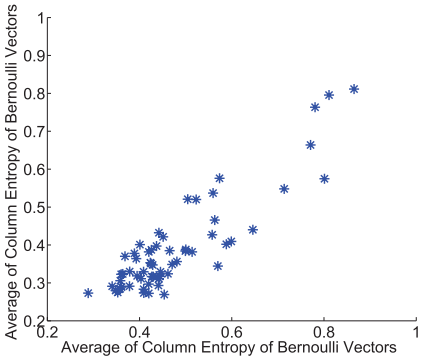
(c) Example 3

Fig. 16 Real examples from summaries.

shows that researchers working on three different topics cooperate a lot, where the size of node S_5 is 28. We can infer from these keywords that these researchers are working on the core database technology. Figure 16 (c) shows three groups of researchers cooperate a lot, where two of them mainly work on knowledge representation, while the third group mainly works on decision tree.



(a) D1 (DM), $k = 40$



(b) D2 (DB), $k = 60$

Fig. 17 Outliers found by the divisive k-means algorithm.

The size of node S_9, S_{16} and S_{20} are 26, 12 and 35, respectively.

Figure 17 plots the average entropy of all columns in all the Bernoulli distribution vectors from datasets D1 and D2. The x-axis is the average entropy of attributes and the y-axis is the average entropy of connection strength. Figure 17 (a) shows results from dataset D1 (DM) when $k = 40$. As we can see, most points are close to (0.3, 0.4) indicating a good confidence, while a few points are closed to (1, 1), which are considered as outliers in the summary. There are also outliers at (0, 1), which means these outliers have the same attribute information but not the neighborhood relationships.

Figure 17 (b) shows results from dataset D2 (DB) when $k = 60$. Most points are close to (0.35, 0.4), while a few outliers are far away from the main cluster.

6. Related Works

The graph summarization [1], [4], [5], [6], [7], [10] mainly contains researches on two aspects.

6.1 Graph Summarization

One of them focuses on obtaining an interpretable summary which is suitable for exploring and visualizing. Navlakha et al. [6] propose to substitute super-nodes for cliques in graphs without attribute information to generate summaries. Given a graph, each clique on the graph is represented by a super-node. The summary is a combination of super-nodes and the original nodes that cannot be represented. If there is an edge between two super-nodes, or a super-node and a original node, then all the possible pair of nodes must be connected by edges in the original graph. It is obvious that usually a graph cannot have many cliques, so they also use super-node to represent near-clique or dense areas, with an extra table to record the edges that do not exist or need to be removed. The quality of a summary is measured based on the size of the summary, which is measured by Minimum Description Length (MDL) principle. MDL can find the best hypothesis leading to the best compression of the data. Even with the help of the additional table, the compression ratio of a summary generated by the above method is still too large, which is almost one half of the original graph. To further reduce the summary size, an error bound ϵ is introduced for edges, that is, for an original node, if it or its super-node connects another super-node in a summary, then the number of missing edges is at most $(1 - \epsilon)$ times the number of nodes in the other super-node. They propose both greedy algorithm and randomized algorithm to calculate exact summary and error-bounded summary. The greedy algorithm iteratively merges two nodes which introduce small extra space cost. The randomized algorithm randomly selects a node u, and finds a node v of small extra space cost in u's 2-hop neighborhood to merge with u.

Tian et al. [7] propose to summarize large attribute graphs by aggregating nodes into groups and use super-nodes to represent groups of nodes. The attributes are categorical. Two super-nodes are connected by a super-edge if there is a pair of nodes, one from each group, connected in the original graph. They require nodes in each group having the same attribution information, so the total number of possible attribute values cannot be too many. Otherwise, the size of summaries will be too large for users to explore. On the super-graph, there is a participation ratio associated with each edge, which is the percentage of pairs of nodes that are connected among all potential possible pairs. They prove NP-completeness of this problem and present two heuristic aggregating algorithms in a bottom-up fashion and a top-down fashion. They design a merging distance mainly based on the similarity between participation ratio vectors. Two super-nodes have a small merging distance if their participation ratio vectors are similar. Given a graph, the bottom-up algorithm iteratively merges two super-nodes with the minimum merging distance until the number of super-nodes left is k. In the top-down algorithm, nodes in the graph are initially grouped into clusters and nodes in each cluster have the same attribute information. A super-node S_i is first selected to be split based on the number of the connection errors to its neighbors. Suppose S_j is a neighbor of S_i, and the number of the connection errors between S_i and S_j is the largest among all the neighbors of S_i. Then S_i is split into two super-nodes whose participation ratio to S_j is 0 and 1. This procedure is repeatedly performed till there are k super-nodes. Their approach does not work when the number of attribute values is not small and their criteria are not so strict for summaries of high-quality.

Zhang et al. [10] extended Tian's approach [7] to summarize graph with two contributions. First, they propose to deal with numerical attribute values not just categorical. Second, they recommend possible values of k, which is the number of super-nodes in summaries. Their algorithm for categorizing numerical values is agglomerative, which iteratively merges two value-adjacent super-nodes until no super-nodes are value-adjacent. Then super-nodes of continuous values are cut into c groups of categories, where c is given by users. Next, they apply algorithms in Ref. [7]

to generate summaries. During splitting or merging process, their algorithm keeps tracking the interestingness measure of the current summary, and recommends the value of k. The interestingness measure is based on three characteristics: diversity, coverage and conciseness.

6.2 Graph Generation Models

Graph generating models can be considered as a summarization since they are able to partially reveal the hidden relationships between nodes in graphs. Chakrabarti et al. [1] study the problem from various points of views in physics, mathematics, sociology, and computer sciences. Based on the analysis of real social networks, the main characteristics they found for social graphs are power laws, small diameters and community effects. The characteristic of power laws indicates that most nodes in social graphs have few neighbors, while only a very small portion of nodes are of high degree. The characteristic of small diameters indicates that the distance between reachable pair of nodes is small, the effective diameter of the studied social graph is only 6. The characteristic of community effects indicates that nodes on graph can be grouped into clusters, whose clustering coefficients measure their clumpiness. Based on above characteristic of social graphs, they survey a lot of graph generators and suggest the possible solutions for each unique requirements.

Leskovec et al. [4] focus on the problem of generating a synthetic graph that has the same properties to a given one. The difficulty lies in that the parameters of generating model must be consistent to the given graph. The authors utilize Kronecker product of matrix to achieve fast synthetic graph generation. They estimate the parameters of Kronecker model using maximum likelihood estimation. The estimation process is speeded up by permutation distribution of the parameters. The same authors study the problem of evolving graph generator in Ref. [5]. Similar to Ref. [1], they first find the evolving rules from the sample graph data, including densification laws and shrinking diameters. Densification laws show that the average degree of nodes increases as time goes by, resulting in the diameters of graphs to be smaller. With these two observations, the forest fire model is introduced which simulates a burning fire of nodes and each node has a certain probability to link a new node which is found during the spread of the fire.

7. Conclusions

In this paper, we study graph summarization using a new information-preserving approach based on information theory. A graph is summarized by partitioning node set into subsets and constructing a super-graph based on the partition. We analyzed the exact and approximate homogeneous partition criterion and proposed a unified entropy framework to relax all three criteria in the exact partition. Our proposed summarization framework can obtain graph summary of small size and high quality, which is measured by the average entropy of each node subset in the partition. We proposed a lazy exact partition algorithm, as well as two other approximate partition algorithms to compute the exact homogeneous partition and the optimized approximate homogeneous partition, respectively. We conducted experiments on various datasets and the results demonstrate that our methods can summarize attribute graphs efficiently and homogeneously.

Acknowledgments The work was supported by grants of the Research Grants Council of the Hong Kong SAR, China No. 419008, 419109, and 411310.

Reference

[1] Chakrabarti, D. and Faloutsos, C.: Graph mining: Laws, generators, and algorithms, *ACM Comput. Surv.*, Vol.38, No.1, p.2 (2006).

[2] Cover, T.M. and Thomas, J.A.: *Elements of information theory*, Wiley-Interscience, New York, NY, USA (1991).

[3] Hofmann, T.: Probabilistic latent semantic indexing, *SIGIR '99: Proc. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.50–57, ACM, New York, NY, USA (1999).

[4] Leskovec, J. and Faloutsos, C.: Scalable modeling of real graphs using Kronecker multiplication, *ICML '07: Proc. 24th International Conference on Machine Learning*, pp.497–504, ACM, New York, NY, USA (2007).

[5] Leskovec, J., Kleinberg, J. and Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations, *KDD '05: Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp.177–187, ACM, New York, NY, USA (2005).

[6] Navlakha, S., Rastogi, R. and Shrivastava, N.: Graph summarization with bounded error, *SIGMOD '08: Proc. 2008 ACM SIGMOD International Conference on Management of Data*, pp.419–432, ACM, New York, NY, USA (2008).

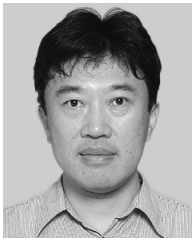
[7] Tian, Y., Hankins, R.A. and Patel, J.M.: Efficient aggregation for graph summarization, *SIGMOD '08: Proc. 2008 ACM SIGMOD International Conference on Management of Data*, pp.567–580, ACM, New York, NY, USA (2008).

[8] Yan, X., Cheng, H., Han, J. and Xin, D.: Summarizing itemset patterns: A profile-based approach, *KDD '05: Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp.314–323, ACM, New York, NY, USA (2005).

[9] Zhai, C., Velivelli, A. and Yu, B.: A cross-collection mixture model for comparative text mining, *KDD '04: Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.743–748, ACM, New York, NY, USA (2004).

[10] Zhang, N., Tian, Y. and Patel, J.M.P.: Discovery-driven graph summarization, *ICDE '10: Proc. 36th International Conference on Data Engineering*, pp.880–891, IEEE, Long Beach, CA, USA (2010).

Zheng Liu was born in 1980. He is current a Ph.D. candidate in the department of systems engineering and engineering management in the Chinese University of Hong Kong. He focuses on summarizing and querying large graph data. He has published research papers in several major conferences, including ICDE, ICDM, DASFAA, etc. He is a student member of IEEE.



Jeffrey Xu Yu is a Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong. His current main research interests include graph mining, graph query processing, graph pattern matching, and keywords search in relational databases. Dr. Yu served as an Information Director and a member in ACM SIGMOD Executive Committee (2007–2011), and an associate editor of IEEE Transactions on Knowledge and Data Engineering (2004–2008). He serves on the VLDB Journal Editorial Board and the International Journal of Cooperative Information Systems. He is an ACM member and an IEEE senior member.

Hong Cheng is an Assistant Professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong. She received her Ph.D. degree from University of Illinois at Urbana-Champaign in 2008. Her research interests include data mining, machine learning and database systems. She has published over 40 research papers in international conferences and journals, including SIGMOD, VLDB, SIGKDD, ICDE, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Knowledge Discovery from Data, and Data Mining and Knowledge Discovery, and received research paper awards at ICDE'07, SIGKDD'06 and SIGKDD'05. She is a finalist for the 2009 SIGKDD Doctoral Dissertation Award.