Graph Summarization with Bounded Error

Yue Tan

tanxiangyueer@foxmail.com

2015.5.7

Data Mining Group, Web Sciences Center Institute of Computer Science and Technology, UESTC

Outlines

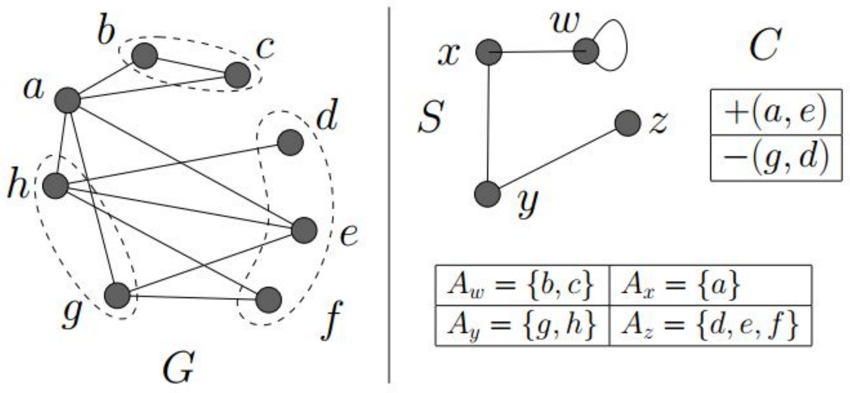
1. Introduction

- World Wide Web.
- Social Networking.
- IP Network Monitoring.
- Market Basket Data.

Large graphs with millions and even billions of nodes and edges.

1.1 A Generic Graph Representation

A graph G = (V_G, E_G), and a representation for it R = (S, C) consists of a graph summary S = (V_S, E_S) and a set of edge corrections C. The graph summary is an aggregated graph structure in which each node v $\in V_S$, called a supernode, corresponds to a set A_v of nodes in G, and each edge (u, v) $\in$ E_S , called a superedge, represents the edges between all pair of nodes in A_u and A_v .

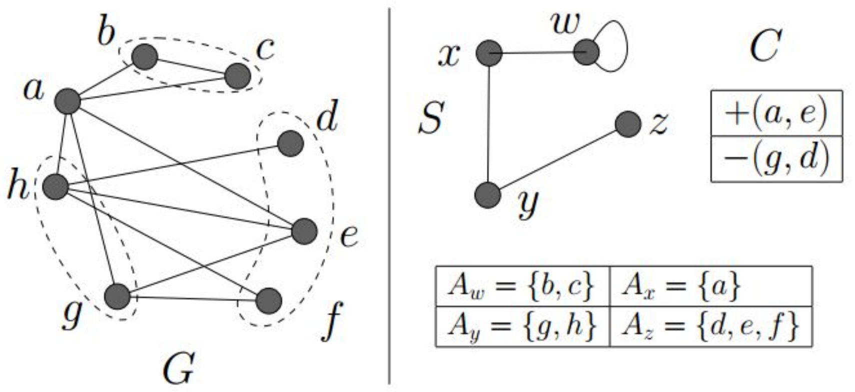


1.1 A Generic Graph Representation

a. If two nodes have edges to the same set (or very similar set) of other nodes, then we can collapse them into a supernode and replace the two edges going to each common neighbor with a single superedge.

b. Further, if there is a complete bi-partite subgraph, then we can collapse the two bi-partite cores into two supernodes and simply replace all the edges with a superedge between the supernodes.

c. Similarly, we can collapse a complete clique to a single supernode with a self-edge.



$+(a,e)$	
$-(g,d)$	

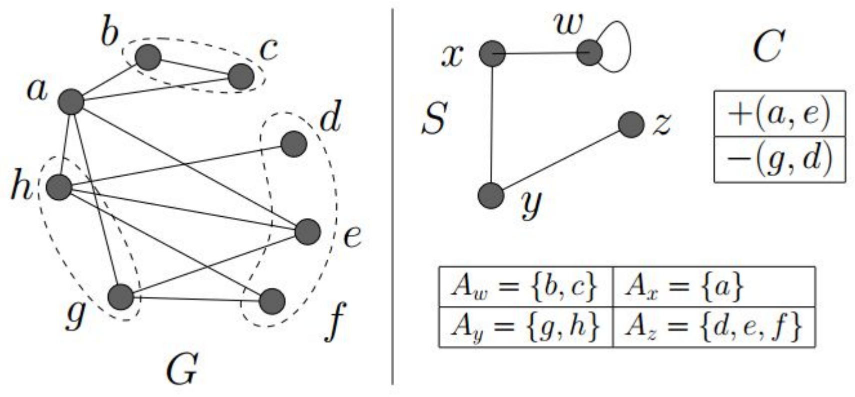
$A_w = \{b,c\}$	$A_x = \{a\}$
$A_y = \{g,h\}$	$A_z = \{d,e,f\}$

1.1 A Generic Graph Representation

Keeping the set of corrections C,which contains the list of edge-corrections that need to be applied to the graph constructed using S to recreate the original graph G.

a. For the superedge (u, v), C contains entries of the form ′-(x, y)′ for the edges that were not present in G.

b. If the same superedge was not added to S, C will contain entries of the form ′+ (x, y)′ for the edges that were actually present in G.

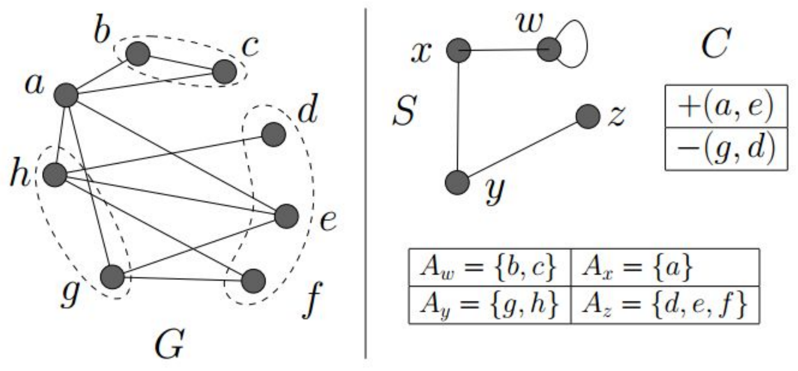


Defining the function g(R) that maps a representation R to the equivalent graph G. An edge (x, y) is present in G iff either (a) C contains an entry ′+ (x, y)′, or (b) S contains a superedge (u, v) s.t. x $\in A_u$ and y $\in A_v$ and C does not have an entry ′-(x, y)′.

1.1 A Generic Graph Representation

Example.

Note that the graph is compresse d from size (number of edges) 11 to 6 (4 edges in graph summary and 2 edge corrections). The neighborhood of a node (say g) in the graph is reconstructed as follows.



+(a, e)	
−(g, d)	

$A_w = \{b, c\}$	$A_x = \{a\}$
$A_y = \{g, h\}$	$A_z = \{d, e, f\}$

First, find the supernode (y) that contains g, then add edges from g to all the nodes in a supernode that is a neighbor of y. This gives the edges {(g, a), (g, d), (g, e), (g, f)}. Next, apply the corrections to the edge set, that is, delete all edges with a ′-′ entry (edge (g, d)), and add edges with a ′+′ entry (none in this example).This gives the set of edges in the neighborhood of g as {(g, a), (g, e), (g, f)}, which is the same as in the original graph. This can be repeated for all nodes in V_G to recover the original graph.

1.2 MDL Representation

Minimum Description Length (MDL) principle

It roughly states that the best theory to infer from a set of data is the one which minimizes the sum of (A) the size of the theory, and (B) the size of the data when encoded with the help of the theory.

1.2 MDL Representation

•Defining the cost of a representation R = (S, C) to be the sum of the storage costs of its two components, that is, cost(R) =$|E_S|$ + $|C|$.

•Thus, if R^ =(S^,C^) denotes the minimum cost representation, then theMDL principle says thatˆS is the “best possible” summary of the graph.

•We define Πuv as the set of all the pairs (a, b), such that a $\in A_u$ and b $\in A_v$; this set represents all possible edges of G that may be present between the two supernodes. Furthermore, let $A_{uv} \subseteq \Pi_{uv}$ be the set of edges actually present in the original graph G ($A_{uv} = \Pi_{uv} \cap E_G$).

1.2 MDL Representation

•Now, we have two ways of encoding the edges in Auv using the summary and correction structures. The first way is to add the superedge (u, v) to S and the edges $\Pi_{uv} - A_{uv}$ as negative corrections to C, and the second is to simply add the edges in the set A_{uv} as positive corrections to C. The memory required for these two alternatives is $(1+|\Pi_{uv} - A_{uv}|)$ and $|A_{uv}|$, respectively. We will simply choose the one with the smaller memory requirements for encoding the edges A_{uv}.

•The cost of representing the edge set A_{uv} between supernodes u and v in the representation is $c_{uv} = \min\{|\Pi_{uv}| - |A_{uv}| + 1, |A_{uv}|\}$.

Problem Statement : Given a graph G, compute its MDL representation R^.

1.3 Approximate Representations

We now proceed to define an ε-approximate representation, denoted by R_ε, that can recreate the original graph within a user-specified bounded error ε ($0 \leq \varepsilon \leq 1$). The structure of R_ε is identical to representation R discussed earlier; thus, it too consists of a (summary, corrections) pair (S_ε, C_ε). But unlike R, it provides the following weaker guarantee for the reconstructed graph $G_\varepsilon = g(R_\varepsilon)$: For every node $v \in G$, if Nv and N'_v denote the set of v's neighbors in G and G_ε, respectively, then

$$error(v) = |N'_v - N_v| + |N_v - N'_v| \leq \varepsilon |N_v| \quad (1)$$

where, $N'_v - N_v$ is the set difference operation. Here, the first term in the equation represents the nodes included in the approximate neighbor set N'_v but were not present in the original neighbor set N_v, while the second term represents vice-versa.

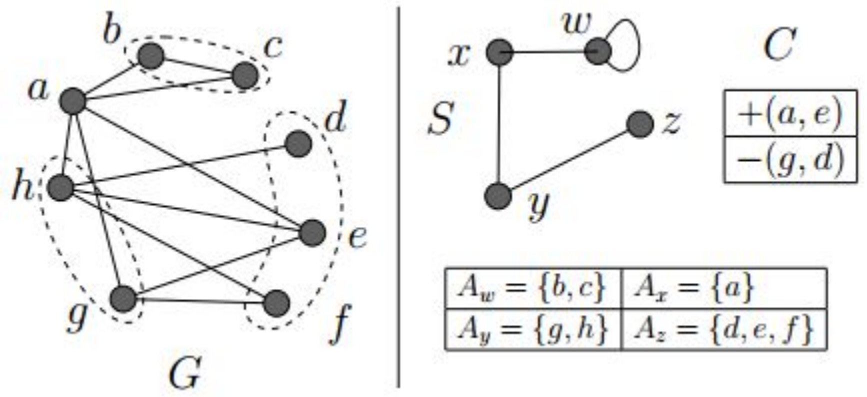
1.3 Approximate Representations

Problem Statement: Given a graph G and $0 \leq \varepsilon \leq 1$, compute the minimum cost ε-representation.

Observe that the MDL representation ˆR is essentially a representation R_0 with error parameter $\varepsilon = 0$ that has the minimum cost. Thus, one approach to compute a minimum cost R_ε is to first compute ˆR, and then delete edges from ˆC or ˆS that do not violate Equation (1) for any node $v \in G$. As ε increases, we can remove more edges from both the graph summary and corrections, and reduce the cost of the representation even further.

1.3 Approximate Representations

Example 2. Consider the graph and suppose ε = 1/3. From the corrections C, if we remove the entry +(a, e), then the approximate neighbor sets for a and e would be N'_a = {b, c, g, h} and N'_e = {g, h}. Since the neighbor sets for a and e in the original graph G are N_a = {b, c, e, g, h} and N_e = {a, g, h}, the approximate neighbor sets N'_a and N'_e satisfy Equation (1). So we can remove +(a, e) from C and reduce its size without violating the error bounds.



2. COMPUTING MDL REPRESENTATIONS

we present two algorithms for finding the MDL representation ˆ R. The first algorithm, called Greedy,iteratively combines node pairs that give the maximum cost reduction into supernodes. The second algorithm, called Randomized, is a light-weight randomized scheme that, instead of merging the globally best node pair, randomly picks a node and merges it with the best node in its vicinity.

2.1 The Greedy Algorithm

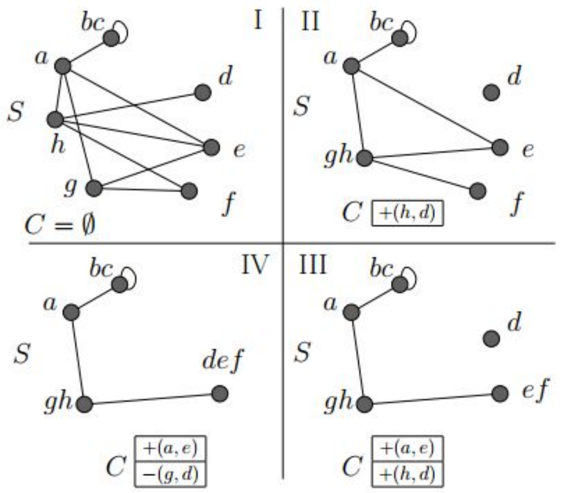
To understand the intuition behind this approach, recall that in a graph there may be many pairs of nodes that can be merged to give a reduction in cost. Typically, any two nodes that share common neighbors can give a cost reduction,with more number of common neighbors usually implying a higher cost reduction. Based on this observation, we define the cost reduction s(u, v) (see below) for any given pair of nodes (u, v). In Greedy, we iteratively merge the pair (u, v) in the graph with the maximum value of s(u, v) (the best pair).

2.1 The Greedy Algorithm

For any supernode $v \in V_S$, we define the neighbor set Nv to be the set of supernodes $u \in V_S$, s.t. there exists an edge (a, b) in graph G for some node $a \in A_v$ and $b \in A_u$. Recall that the cost of the superedge (v, x) from supernode v to a neighbor $x \in N_v$ is $c_{vx} = \min\{|\Pi_{vx}| - |A_{vx}| + 1, |A_{vx}|\}$.We will define the cost c_v of supernode v to be the sum of the costs of all the superedges (v, x) to its neighbors $x \in N_v$ .Now, given pair (u, v) of supernodes in V_S , the cost reduction s(u, v) is defined as the ratio of the reduction in cost as a result of merging u and v (into a new supernode w), and the combined cost of u and v before the merge.

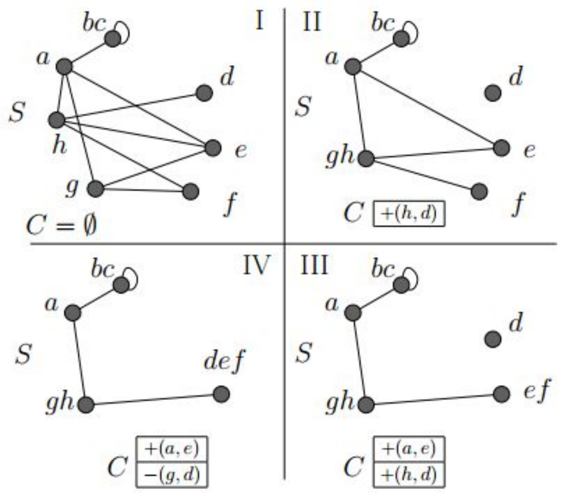
$$s(u, v) = (c_u + c_v - c_w)/(c_u + c_v) \quad (2)$$

2.1 The Greedy Algorithm



Example 3. Figure 2 shows the steps of the Greedy algorithm on the graph shown in Figure 1. For the sake of simplicity, we refer to the supernode formed due to merging nodes x and y as the concatenated string "xy". In the first step, we merge the pair (b, c), which has the highest cost reduction of .5 (since both b and c have two edges incident on them, each has a cost of 2, and supernode bc also has a cost of 2 because of a self-edge and a superedge to a; hence the cost reduction for (b, c) is .5). The other top contending pairs are (g, h) with an s(·) value of 3/7 and (e, f) with s(e, f) = 2/5. To see why s(g, h) = 3/7, lets derive the costs of nodes g and h before and after they are merged. Nodes g and h have 3 and 4 incident edges, respectively, and so c_g = 3 and c_h = 4. After g and h are merged to form supernode gh, we will have 3 superedges between supernode gh and nodes a, e and f , and one correction +(h, d). Thus,c_{gh} = 4 and s(g, h) = (c_g + c_h − cgh)/(cg + ch) = 3/7.

2.1 The Greedy Algorithm

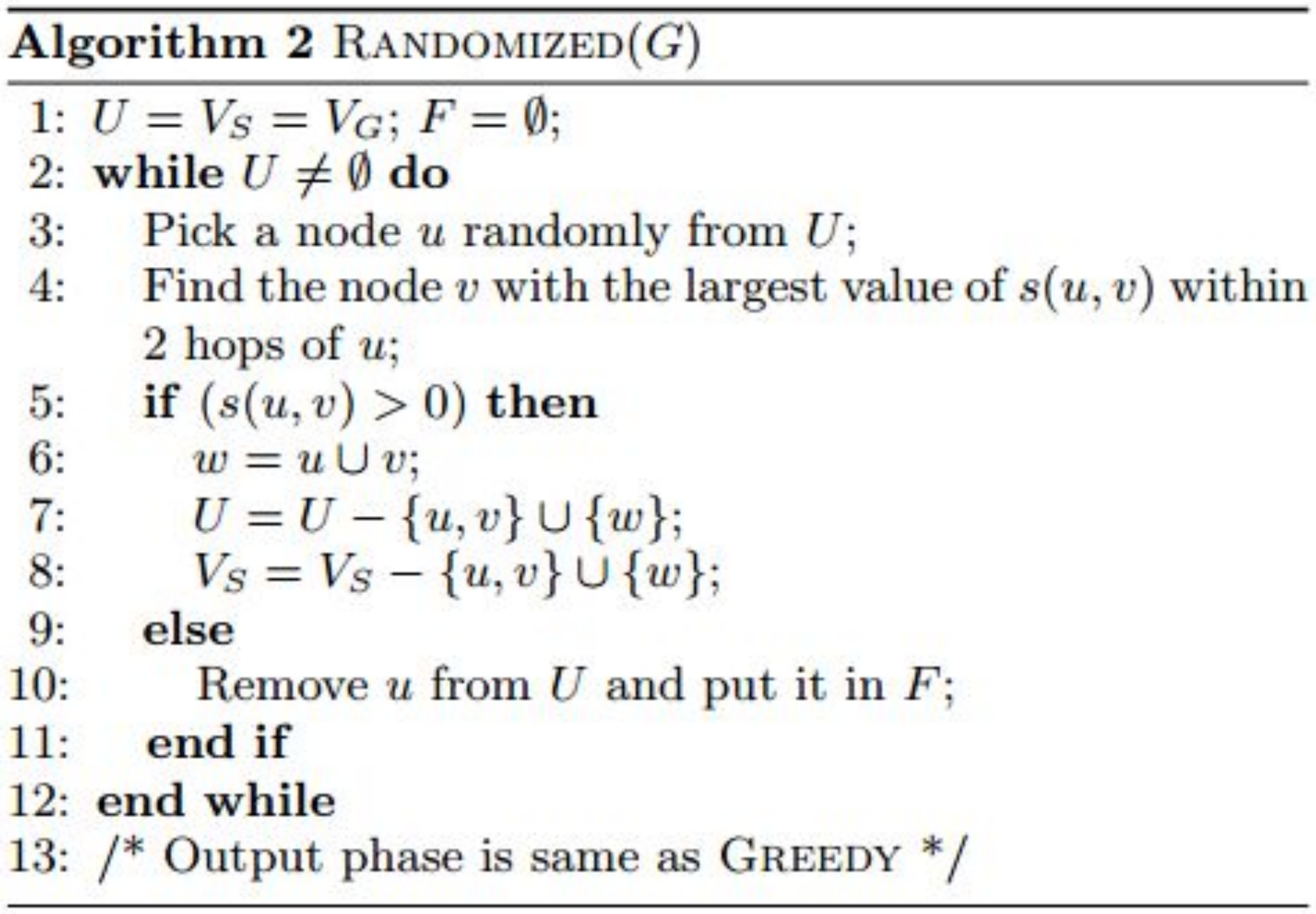


In the next 3 steps, we merge the pairs (g, h) with cost reduction 3/7, (e, f) with cost reduction 1/3 (since $c_e = 2, c_f = 1$, and $c_{ef} = 2$ because of a superedge between ef and gh, and a correction +(a, e)), and (d, ef) with cost reduction 0. Note that the last merge does not decrease the cost, but only reduces the number of supernodes in the summary S resulting in a more compact visualization. After these merges,the cost reduction is negative for all pairs, and so Greedy terminates.

2.2 The Randomized Algorithm

The Randomized algorithm (Algorithm 2) iteratively merges nodes to form a set of supernodes V_S ; these supernodes are divided into two categories, U (unfinished) and F (finished). The finished category tracks the nodes which do not give any cost reduction with any other node (that is, s(·) value is negative for all pairs containing them), while the unfinished category contains the remaining nodes that are considered for merging by the Randomized algorithm.Initially, all the nodes are in U . In each step, we choose a node u uniformly at random from U , and find the node v such that s(u, v) is the largest among all pairs containing u. If merging these nodes gives a positive cost reduction, we merge them into a supernode w. We then remove u and v from V_S (and U), and add w to V_S (and U). However, if s(u, v) is negative, we know that merging u with any othernode will only increase the cost; hence, we should not consider it for merging anymore and so we move u to F . We repeat these steps until all the nodes are in F . Finally, the graph summary and corrections are constructed from V_S ,similar to the Greedy algorithm.

2.2 The Randomized Algorithm



Algorithm 2 RANDOMIZED(G)

```
1:  $U = V_S = V_G$; $F = \emptyset$;
2:  while $U \neq \emptyset$ do
3:      Pick a node $u$ randomly from $U$;
4:      Find the node $v$ with the largest value of $s(u, v)$ within
        2 hops of $u$;
5:      if $(s(u, v) > 0)$ then
6:          $w = u \cup v$;
7:          $U = U - \{u, v\} \cup \{w\}$;
8:          $V_S = V_S - \{u, v\} \cup \{w\}$;
9:      else
10:         Remove $u$ from $U$ and put it in $F$;
11:     end if
12: end while
13: /* Output phase is same as GREEDY */
```

Thank You!