计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(6):189−196 [doi: 10.15888/j.cnki.csa.009135]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 图的有损摘要问题的两阶段算法①

冯　康, 陈卫东

(华南师范大学 计算机学院, 广州 510631)
通信作者: 冯　康, E-mail: 1091889767@qq.com

**摘　要**: 图的有损摘要问题如下: 给定图 $G=(V, E)$ 和正整数 $k$, 要求将图 $G$ 中所有节点合并成为 $k$ 个超节点, 满足由这些超节点组成的摘要图能够在一定误差范围内表示原图 $G$. 这是一个基于图划分的组合优化问题, 一个主要求解思路是逐次地随机抽取节点对集并用启发式方法从中选取节点对进行合并. 本文提出一个有效的两阶段求解算法 TS_LGS. 算法根据图 $G$ 的平均点度特征设置阶段阈值: 当前超节点数大于阶段阈值为第 1 阶段, 期间算法在采样节点对中基于当前最佳合并分数批量选择节点对合并, 旨在有效减少迭代次数; 否则为第 2 阶段, 期间算法在加权采样的基础上优先挑选相邻的节点对, 旨在找到重构误差增量较小的节点对合并, 直至超节点的个数为 $k$. 在典型的真实网络实例图上与现有最好算法 SAA 进行了实验对比, 结果表明, 算法 TS_LGS 以较低时间复杂度提取到的图摘要具有更低的重构误差和查询误差.

**关键词**: 图摘要; 图有损摘要; 重构误差; 平均度

引用格式: 冯康,陈卫东.图的有损摘要问题的两阶段算法.计算机系统应用,2023,32(6):189–196. http://www.c-s-a.org.cn/1003-3254/9135.html

## Two-stage Algorithm for Lossy Graph Summarization

FENG Kang, CHEN Wei-Dong

(School of Computer Science, South China Normal University, Guangzhou 510631, China)

**Abstract**: The problem of lossy graph summarization is as follows: Given a graph $G=(V, E)$ and a positive integer $k$, it is required to merge all nodes in graph $G$ into $k$ super nodes so that the resulting summary graph composed of these super nodes can represent the original graph $G$ within a certain error range. As a combinatorial optimization problem based on graph partitioning, this problem is usually solved by randomly extracting node pairs successively and using heuristic methods to select node pairs for merging. This study proposes an effective two-stage algorithm, namely TS_LGS. The algorithm first sets the stage threshold according to the average degree of graph $G$. Specifically, in the first stage, the current number of super nodes is greater than the stage threshold, and the algorithm selects node pairs among the sampled node pairs in batch for merging based on the current best merging score, so as to effectively reduce the number of iterations; in the second stage, the algorithm preferentially selects adjacent node pairs based on weighted sampling, so as to merge the node pairs with small reconstruction error increment until the number of super nodes is $k$. The experimental results on several typical real network instances show that TS_LGS can extract graph summarization with lower reconstruction and query errors on the premise of lower time complexity compared with the existing best SAA algorithm.

**Key words**: graph summarization; lossy graph summarization (LGS); reconstruction error (RE); average degree

① 基金项目: 国家自然科学基金 (61370003)
收稿时间: 2022-12-05; 修改时间: 2023-01-06; 采用时间: 2023-01-19; csa 在线出版时间: 2023-04-07
CNKI 网络首发时间: 2023-04-10

复杂网络中的许多应用通常能够用图来表示, 如: 万维网、社交网络、分子生物学等[1–3]. 这些网络图的规模随着时间的推移而增大. 在大规模的图中进行查询、计算等操作时, 难以保证时效性. 图摘要 (graph summarization) 的目的就是将原图转换为一个更紧凑的摘要图, 同时尽可能地保留原图的结构模式、查询答案或特定属性分布[4]. 随着摘要图的规模变小, 在其上进行的模式识别、社团挖掘、度或特征向量中心性查询等图操作方法耗时更小[5].

图摘要问题分为图有损摘要问题和图无损摘要问题两大类. 对于图无损摘要问题来说, 该问题要求找出的摘要图能够在使用校正信息的条件下重建为原图. 最小描述长度 (minimum description length, MDL)[6] 和空间复杂度[7] 通常用于衡量所生成摘要图的质量. 同时, 图无损摘要问题通过将团、星、链等结构信息和校正信息一起存储在超点、超边中来辅助摘要图的重建[8–10].

图有损摘要则不保存校正信息, 因此空间复杂度更低. 其通过摘要图与原始图的邻接矩阵差异范数来衡量摘要图的质量. 常见的算法是采用节点分组和聚合的方法寻找差异范数最小的摘要图. LeFevre 等人[11] 提出的算法 GraSS 在每次迭代中贪心地选取导致当前 $l_1$-重构误差增量最小的节点对合并; 在此基础上, Beg 等人提出了算法 SAA[12], 其中新增了节点对的评分和计算策略, 用于优化算法的时间与效果; Riondato 等人[13] 则将每个顶点作为一个 $n$ 维向量, 采用欧氏距离聚类方法 (K-median 和 K-means) 寻找摘要图, 目标是最小化 $l_2$-重构误差. 此外, Lee 等人[14] 通过结合 MDL 方法和图的稀疏化来寻找尽可能小的摘要图; Zhou 等人[15] 提出了保留度的邻接矩阵作为目标函数来摘要图, 从而使得摘要图的度查询误差更小; 而 Fan 等人[16] 采用压缩的技术寻找摘要图.

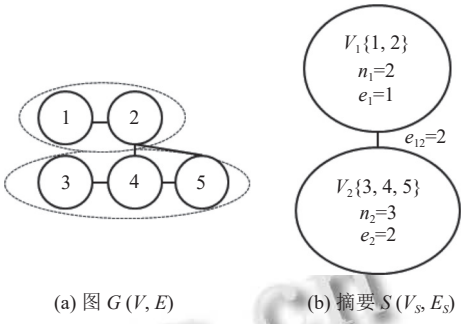
针对不同类型的网络图也有一些启发式的图摘要算法, 例如, 基于社会背景和特征来摘要社交网络[17–19]; 基于时序动态图[20–22] 和异构网络[23] 等进行图摘要的系列算法[24]. 并且, 图摘要技术还时常被用作数据匹配、模式挖掘等其他问题[5,25] 的预处理过程.

本文研究图的有损摘要问题的有效算法. 现有的基于节点聚合思想的图有损摘要算法每次迭代中仅贪心地合并一个节点对, 导致迭代轮次过多, 时间上较慢; 而在节点对选取上则没有考虑节点之间的结构关系. 针对以上问题, 本文提出了一个两阶段算法 TS_LGS, 在第一阶段批量合并对结果影响较小的节点对, 加快算法的迭代; 在第 2 阶段则结合节点的权重和相连性选择对结果影响更小的节点对合并, 从而更快地找到质量更好的摘要图.

本文组织和结构如下: 第 1 节是图有损摘要问题的数学表述, 第 2 节简要介绍了基于节点聚合的图有损摘要的主要算法, 第 3 节描述了我们的 TS_LGS 算法, 第 4 节给出了实验比较结果, 最后是本文的总结.

## 1 问题表述

图有损摘要 (lossy graph summarization, LGS) 问题定义如下: 给定一个包含 $n$ 个节点的原图 $G(V, E)$ 和正整数 $k$ ($k \leqslant n$), 找到具有 $k$ 个超节点的摘要图 $S(V_S, E_S)$ 来表示原图 $G$, 满足摘要图与原图的重构误差 (reconstruction error, $RE$) 要最小[11]. 图有损摘要下文简称为图摘要. 图 1 中给出了一个图摘要示例.




(a) 图 $G(V, E)$ (b) 摘要 $S(V_S, E_S)$

图 1 图摘要示例

摘要图 $S(V_S, E_S)$ 是一个 $k$ 点带权无向图, 其中 $V_S=\{V_1, \cdots, V_k\}$ 是 $V$ 的一个 $k$ 划分, 每个 $V_i$ 称作是摘要图的一个超节点, 有两个属性: 内部节点数目 $n_i$, 内部点之间的边数 $e_i$; 超节点对 $(V_i, V_j)$ 之间的边权重 $e_{ij}$ 为 $V_i$ 与 $V_j$ 点之间边数, 而 $E_S$ 为所有超节点之间带权边的集合.

给定一个摘要图 $S$, 由 $S$ 重建的图的期望邻接矩阵为 $n$ 阶方阵 $\overline{A}$ 给定节点对 $(u, v)$, 其元素的计算方式如下:

$$\overline{A}(u,v)=\begin{cases}\dfrac{e_i}{\binom{n_i}{2}}, & u,v\in V\\ 0, & u=v\\ \dfrac{e_{ij}}{n_i n_j}, & u\in V_i, v\in V_j\end{cases} \tag{1}$$

摘要图 $S$ 的质量可以用 $l_p$-重构误差来度量, 即 $\overline{A}$ 与原图 $G$ 的邻接矩阵为 $A$ 的差异情况. $l_p$-重构误差的定义如式 (2) 所示:

$$RE_p(G|S)=\left(\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}|\overline{A}(i,j)-A(i,j)|^p\right)^{1/p} \quad (2)$$

其中, $A(i,j)$ 表示图 $G$ 的邻接矩阵中 $i$ 行 $j$ 列的值, $\overline{A}(i,j)$ 表示摘要图 $S$ 的期望邻接矩阵中 $i$ 行 $j$ 列的值, 重构误差 $RE_P(G|S)$ 表示摘要图与原图的差异情况, 值越小说明摘要图越接近原图, 下文简写为 $RE$. 当 $P$=1 时为 $l_1$-$RE$[10], $P$=2 时为 $l_2$-$RE$[11].

## 2 相关工作

Lefevre 等提出的算法 k_GS[11] 的核心思想是以最小重构误差增量为目标, 在算法迭代过程中依次选取合并后重构误差增量最小的节点对合并. 其主要流程如下: 在每次迭代过程中, 首先需要随机选取 $P(t)$ 个节点对, 然后分别计算节点对合并后的重构误差变化量, 其计算时间为 $O(P(t)\times n)$; 再将重构误差增量最小的节点对删除并聚合为新的超节点, 其计算时间为 $O(n)$. 由于算法最多迭代 $n$–$k$+1 轮, 因此当 $P(t)$=$n_t$= $O(n)$ 时算法时间复杂度为 $O((n\times n+n)\times n)$= $O(n^3)$.

为了降低算法的时间复杂度, 同时降低迭代过程中节点对随机选取对摘要图质量的影响, Beg 等人在算法 k_GS 的基础上提出了算法 SAA[12], 其主要优化策略包括: 简化了节点对合并后重构误差增量的计算, 并使用了加权采样选取增量更小的节点对的方法. 首先, SAA 将式 (2) 中的 $l_1$-$RE$ 计算过程拆分为式 (3):

$$RE(G|S)=\sum_{i=1}^{k}\left(4e_i-\frac{4e_i^2}{\binom{n_i}{2}}\right)+\sum_{i=1}^{k}\sum_{j=1,i\neq j}^{k}\left(2e_{ij}-\frac{2e_{ij}^2}{n_in_j}\right) \quad (3)$$

再将节点对合并后重构误差变化量定义为节点对的评估得分, 从而使得节点对合并后的影响可以形式化为评估得分的计算, 如式 (4) 所示:

$$\begin{aligned}score_t(a,b)=&RE(G|S_{t-1})-RE(G|S_t^{a,b})\\=&-\frac{4e_a^2}{\binom{n_a}{2}}-\sum_{i=1,i\neq a}^{n(t)}\frac{4e_{ai}^2}{n_an_i}+\frac{4e_{ab}^2}{n_an_b}-\frac{4e_b^2}{\binom{n_b}{2}}\\&-\sum_{i=1,i\neq a}^{n(t)}\frac{4e_{bi}^2}{n_bn_i}+\frac{4(e_a+e_b+e_{ab})^2}{\binom{n_a+n_b}{2}}\\&+\frac{4}{n_a+n_b}\sum_{i=1,i\neq a,b}^{n(t)}\left(\frac{e_{ai}^2}{n_i}+\frac{e_{bi}^2}{n_i}+\frac{2e_{ai}e_{bi}}{n_i}\right)\end{aligned} \quad (4)$$

其中, $RE(G|S_{t-1})$ 是第 $t$–1 轮迭代时形成的摘要图的重构误差, $RE(G|S_t^{a.b})$ 是第 $t$ 轮迭代时合并节点对 $(a, b)$ 后的摘要图的重构误差, $score_t(a, b)$ 则表示第 $t$ 轮迭代时合并节点对 $(a, b)$ 的评估得分. 从而将寻找合并后重构误差增量最小节点对转换为寻找得分最高的节点对. SAA 通过将 $n_a$, $e_a$ 等结构信息存储在超节点中并采用 count-min sketch[26] 的方法近似计算 $\sum_{\substack{i=1\\ i\neq a,b}}^{n(t)}\frac{e_{ai}e_{bi}}{n_i}$, 将节点对的得分计算时间复杂度降为了 $O(1)$. 同时, SAA 提取了式 (4) 中只与当前超节点有关的量 $f(a)$ 来构造该节点的权重 $w(a)$, 其中, $f(a)$ 与权重 $w(a)$ 的计算公式如下所示:

$$f(a)=-\frac{4e_a^2}{\binom{n_a}{2}}-\sum_{\substack{i=1\\ i\neq a}}^{n(t)}\frac{4e_{ai}^2}{n_an_i} \quad (5)$$

$$w(a)=\begin{cases}\dfrac{-1}{f(a)}, & f(a)\neq 0\\ 0, & f(a)=0\end{cases} \quad (6)$$

SAA 采用平衡二叉树的形式来辅助挑选节点对, 使得对摘要重构误差影响更小的节点对更容易被选中, 相较于 k_GS 中的随机采样更为合理. 当抽样节点数 $P(t)$=log$n$ 时, SAA 寻找的摘要图质量已经与抽取 $n$ 对节点的 k_GS 类似. 虽然使用维持的平衡二叉树抽取节点仍需要 log$n$ 的时间, 但整体时间复杂度已经降为了 $O(n\log^2 n)$ .

## 3 TS_LGS 算法

本文沿用了 k_GS 与 SAA 算法中迭代合并最小重构误差增量的节点对来寻找摘要图的基本模式. 并在 SAA 的加权抽样基础上进一步提出了 TS_LGS 算法, 通过批量合并节点对来加快算法的迭代, 并使用节点相连性来优化节点对的选取. 该算法将整个迭代过程分为了两个阶段 (使用阶段阈$th_{\overline{d}}$值$th_{\overline{d}}$. 来区分). 当前节点数大于$th_{\overline{d}}$为第 1 阶段, 此时根据当前最小合并分数快速合并节点对, 减少算法运行时间; 当前节点数小于等于第 2 阶段, 此时基于图的平均度和节点结构信息来选取节点对合并, 使抽样的节点对更合理, 从而整体上实现在更短的时间内寻找到质量较好的摘要图.

### 3.1 算法策略

本文使用原图中度数大于平均度的节点数作为阶

段阈值来划分算法的第 1 阶段和第 2 阶段, 在不同阶段使用不同的改进策略寻找摘要图. 阶段阈值的计算方法如式 (7) 所示:

$$th_{\overline{d}} = \left|\left\{v|\mathrm{v} \in V, d(v) > \overline{d}\right\}\right| \tag{7}$$

其中, $\overline{d}$是原图的平均度, 而 $d(v)$ 则是节点 $v$ 的度.

(1) 第 1 阶段: 批量合并策略

我们记录当前节点对合并得分的最佳值 $bestScore_t$, 在一定程度上它能反应出之前合并的节点对的得分情况. 假定 $t$ 轮迭代中节点对最高得分为 $maxScore_t$, 则当前的节点对最佳合并得分 $bestScore_t$ 可由式 (8) 计算:

$$bestScore_t = \min(bestScore_{t-1}, maxScore_t) \tag{8}$$

实验中我们观察到如下情况: 在第 1 阶段, 图中有许多度小于图的平均度的节点, 在合并这些节点时, 重构误差的变化量较小. 针对其他图摘要凝聚算法每轮迭代仅合并一个节点对的情况, 本文提出一个基于当前最佳合批量合并的方法. 具体来说, 在第 1 阶段迭代过程中, 若当前抽样的节点对得分高于 $bestScore_t$, 就直接将其合并. 反之, 若当前抽样节点对最高得分 $maxScore_t$ 依旧低于 $bestScore_t$ 时, 则将 $bestScore_t$ 更新为 $maxScore_t$. 这样有效地减少了算法迭代的次数, 从而能减少了整体的运行时间.

(2) 第 2 阶段: 相邻优先合并策略

节点权重只考虑了单个节点对得分的影响, 而忽略了节点对之间的关系. 在式 (3) 中, $e_{ab}$ 代表超节点对 $(a, b)$ 之间边的数量. 如果节点对不相连, 则 $e_{ab}$=0, 公式中$\frac{4e_{ab}^2}{n_a n_b}$与$\frac{4(e_a+e_b+e_{ab})^2}{\binom{n_a+n_b}{2}}$对应的值也就为 0. 为了在采样的时候选出 $e_{ab}\neq 0$ 的节点对, 本文考虑选取节点对的时候优先选取相连的节点对. 同时, 为了避免只考虑节点对相连性而忽略了式 (3) 中的其他参数. 本文使用平均度$\overline{d}$次数内来选取可能相连的节点对. 这样保证了挑选出的节点对要么是相连的, 要么是权重大的, 从而使得节点对合并后的重构误差增量更小.

### 3.2 算法描述

TS_LGS 算法的过程可以简单描述为: 首先计算图的平均度$\overline{d}$, 并计算度数大于平均度的节点数$th_{\overline{d}}$, 作为本算法的阶段阈值; 在第一阶段, 通过维持一个当前合并分数的最佳值 $bestScore_t$, 将该轮抽取的节点对中得分大于 $bestScore_t$ 的批量合并, 从而减少迭代的轮次; 在第 2 阶段, 算法在加权采样的基础上, 平均度次数内优先选取相连的节点对, 使得挑选的节点对合并后的重构误差增量更小. 直到图的超节点数降为 $k$.

TS_LGS 算法的具体描述如算法 1 所示.

算法 1. TS_LGS

输入: 图 $G(V, E)$, 摘要图节点数 $k$, 抽样节点数 $P(t)$
输出: 摘要图 $S$

```
1. d̄ = getAvgDregree(G) // 计算图 G 的平均度
2. th_d̄=getNumUpperAvgDregree(G) //计算阶段阈值
3. 构建抽样节点树 D
4. while n_t > k do
5.       if n_t > th_d̄ //第 1 阶段
6.          for i ∈ P(t) do
7.             (node1, node2) = getSample(D)
8.             score = getScore(node1, node2)
9.             保留最高得分 maxScore_t 节点对 maxPair
10.                if score ⩾ bestScore_{t-1}
11.                   合并节点对 (node1, node2)
12.         end for
13.         if maxScore_t < bestScore_{t-1}
14.            bestScore_t = maxScore_t
15.            合并得分最高节点对 maxPair
16.      else //第 2 阶段
17.         for i ∈ P(t) do
18.            for j ∈ d̄ do
19.               (node1, node2) = getSample(D)
20.               如果 node1, node2 相连则退出循环
21.            end for
22.            保留最高得分 maxScore_t 节点对 maxPair
23.         end for
24.         合并得分最高节点对 maxPair
25. end while
26. return S
```

### 3.3 复杂度分析

TS_LGS 算法分为了两个阶段, 第 1 阶段的时间复杂度为$\mathrm{O}\left(\alpha\left(n-th_{\overline{d}}\right)P(t)\log_n\right)\alpha\in(0,1)$, 第 2 阶段的时间复杂度为$\mathrm{O}\left(th_{\overline{d}}-k\right)\overline{d}P(t)\log_n)$. 其中$\overline{d}$为图的平均度, 一般为常数可忽略不计. 而为实际迭代次数与理论合并次数的比值, 取值在 0 到 1 之间. 所以, 当抽样节点数 $P(t)$ 取 $\log_n$ 时, 算法整体时间复杂度上限为 $\mathrm{O}(n\log^2{}_n)$.

## 4 实验结果与分析

本文实验所使用的对比算法包括 k-GS、SAA、SAA($w$=50)、SAA($w$=100) ($w$ 为近似计算参数取值)、S2L, 在几个典型的测试数据集上比较了重构误差和查询误差.

实验均用 Java 实现, 部分算法存在随机性, 实验结果取运行 100 次后的均值, 耗时较长的实验结果为运行 10 次后均值. 运行环境为 Windows 10 操作系统, 内存 8 GB, 处理器 Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz.

### 4.1 数据集

本文实验选用 6 个不同规模的真实网络数据集, 它们来自不同的应用场景[8,11,12,14], 网络的基本信息如表 1 所示.

表 1 数据集

| ID | 数据集 | 节点数 | 边数 |
|---|---|---|---|
| R1 | ego-Facebook | 4039 | 88234 |
| R2 | email-Enron | 36692 | 183831 |
| R3 | web-Stanford | 281903 | 1992636 |
| R4 | amazon0601 | 403394 | 2443408 |
| R5 | as-skitter | 1696415 | 11095298 |
| R6 | wiki-talk | 2394385 | 4659565 |

### 4.2 实验设置与评价指标

(1) 重构误差 $RE(G|S)$

本文使用重构误差来衡量摘要图与原始图的差异情况. 重构误差越小, 表示摘要图与原始图的差距越小. 重构误差的具体表达式见式 (2).

(2) 节点度平均差与标准差

给定节点 $v\in V$, 其在摘要图上的度计算方式如式 (9) 所示:

$$\overline{d}(v)=\sum_{j=1}^{|V|}\overline{A}(v,j) \tag{9}$$

在原始图与摘要图上对 $t$ 个节 $d(v_1)$ 点进行度查询, 其度分别为 $d(v_1), d(v_2), \cdots, d(v_t)$ 和 $\overline{d}(v_1), \overline{d}(v_2), \cdots, \overline{d}(v_t)$. 则原始图与摘要图的节点度平均差 $D_{\text{avgError}}$ 可由式 (10) 表示.

$$D_{\text{avgError}}=\frac{\sum_{i=1}^{t}\left|d(v_i)-\overline{d}(v_i)\right|}{t} \tag{10}$$

同理, 原始图与摘要图的节点度标准差 $D_{\text{stdError}}$ 可由式 (11) 表示.

$$D_{\text{stdError}}=\sqrt{\frac{\sum_{i=1}^{t}(\left|d(v_i)-\overline{d}(v_i)\right|-D_{\text{avgError}})^2}{t}} \tag{11}$$

(3) 特征向量中心性平均差与标准差

给定点 $v\in V$, 其在摘要图上的特征向量中心性计算方式如式 (12) 所示:

$$\overline{p}(v)=\frac{\overline{d}(v)}{2|E|} \tag{12}$$

在原始图与摘要图上对 $t$ 个节点进行特征向量中心性查询, 其特征向量中心性分别为 $p(v_1), p(v_2), \cdots, p(v_t)$ 和 $\overline{p}(v_1), \overline{p}(v_2), \cdots, \overline{p}(v_t)$. 则原始图与摘要图的特征向量中心性平均误差 $P_{\text{avgError}}$ 可由式 (13) 表示.

$$P_{\text{avgError}}=\frac{\sum_{i=1}^{t}|p(v_i)-\overline{p}(v_i)|}{t} \tag{13}$$

同理, 原始图与摘要图的特征向量中心性误差标准差 $P_{\text{stdError}}$ 可由式 (14) 表示.

$$P_{\text{stdError}}=\sqrt{\frac{\sum_{i=1}^{t}(|p(v_i)-\overline{p}(v_i)|-P_{\text{avgError}})^2}{t}} \tag{14}$$

(4) 三角形密度相对差

本文使用三角形密度相对差表示原始图与摘要图在三角形密度上的差异情况. 其可以用式 (15) 所表示.

$$TD=\frac{\overline{T}-T}{T} \tag{15}$$

其中, $T$ 表示原始图的三角形个数, $\overline{T}$ 为摘要图的三角形个数.

### 4.3 实验结果与分析

本文使用节约轮次 (理论迭代次数与实际迭代次数之差) 与节约率 (节约轮次与理论迭代次数之比) 来形式化表示第一阶段加速合并时 TS_LGS 算法减少的迭代轮次. 本文在不同规模真实数据集上 (见表 1), 通过控制摘要图剩余节点数 $k$ 和抽样节点数 $P(t)$ 来比较 TS_LGS 与其他算法寻找摘要图的运行时间以及摘要图的重构误差、查询误差等评价指标. 具体结果及分析如下.

(1) 不同 $k$ 与 $P(t)$ 下重构误差与运行时间

在小规模网络上 (如表 2 所示), TS_LGS 在 R1 数据集上能够在与 SAA 差不多的时间内找到 $l_1$-$RE$ 更低的摘要图. 在 R2 数据集上 TS_LGS 则能够更快地形成摘要图. 而在中规模网络 (如表 3 所示) 和大规模网络 (如表 4 所示) 上, 相较于现有的 k-GS、SAA 以及 SAA($w$=50)、SAA($w$=100) 算法, TS_LGS 算法提取的图摘要几乎都具有更小的 $l_1$-$RE$, 并且运行时间更少 (迭代轮次节约率在 60% 左右). 同时, 在不同数据集下, TS_LGS 找到的摘要图相比其他算法也具有较小的 $l_2$-$RE$ (如表 5 所示). 而随着图的规模变大或者抽样节点数 $P(t)$ 的增多, 可以看到, TS_LGS 算法不管是寻找的摘要图的重构误差, 还是运行时间, 都优于其余同类算法. 这说明 TS_LGS 算法能够很好地扩展到更大规模的网络上.

表 2 小规模网络上算法对比

| 数据集 | $k/P(t)$ | $l_1$-$RE$ | | | 运行时间 (s) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | k-GS | SAA | TS_LGS | k-GS (s) | SAA (s) | TS_LGS (s) | 节约轮次 | 节约率 (%) |
| R1 | 1000/$\log_n$ | 43.84 | 38.81 | **36.31** | 0.43 | **0.2** | 0.33 | 1931 | 63.54 |
| | 1000/$n$ | 23.40 | 22.36 | **21.56** | **11.08** | 11.75 | 12.22 | 1659 | 54.59 |
| R2 | 10000/$\log_n$ | 6.45 | **5.81** | 6.02 | 4.17 | 0.82 | **0.56** | 28092 | 83.82 |
| | 10000/$n$ | 2.85 | **2.66** | 3.32 | 626.68 | 442.76 | **102.54** | 28969 | 86.43 |

表 3 中规模网络上算法对比

| 数据集 | $k/P(t)$ | $l_1$-$RE$ | | 运行时间 | | | |
|---|---|---|---|---|---|---|---|
| | | SAA | TS_LGS | SAA | TS_LGS | 节约轮次 | 节约率 (%) |
| R3 | 10000/$\log_n$ | **20.54** | 20.92 | 35.23 s | **30.36 s** | 159085 | 58.50 |
| | 10000/$n$ | 7.16 | **6.60** | 15.15 h | **5.42 h** | 162278 | 59.68 |
| R4 | 8000/$\log_n$ | 23.61 | **23.59** | 112.45 s | **98.02 s** | 253619 | 64.46 |
| | 8000/$n$ | 19.60 | **19.14** | 45.52 h | **15.43 h** | 236793 | 59.88 |

表 4 大规模网络上算法对比

| 数据集 | $k/P(t)$ | $l_1$-$RE$ | | | | 运行时间 (s) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | SAA | SAA($w$=50) | SAA($w$=100) | TS_LGS | SAA | SAA($w$=50) | SAA($w$=100) | TS_LGS |
| R5 | 10000/$\log_n$ | 23.80 | 25.41 | 24.78 | **23.76** | 495.58 | 470.46 | 482.73 | **468.49** |
| R6 | 10000/$\log_n$ | **6.72** | 7.28 | 7.08 | **6.72** | 392.81 | 233.79 | 293.71 | **195.89** |

表 5 不同算法下 $l_2$-$RE$ 对比

| 数据集 | $k/P(t)$ | SAA | SAA($w$=50) | SAA($w$=100) | S2L | TS_LGS |
|---|---|---|---|---|---|---|
| R1 | 1000/$\log_n$ | 19.51 | 35.00 | 28.66 | 581 | **18.16** |
| R2 | 10000/$\log_n$ | **2.91** | 3.01 | 2.92 | 72 | 2.95 |
| R3 | 10000/$\log_n$ | 10.57 | 11.83 | 11.81 | 38 | **10.46** |
| R4 | 8000/$\log_n$ | 11.87 | 11.89 | 11.87 | 51 | **11.85** |

(2) 剩余节点数下重构误差与运行时间

图 2 和图 3 是在 R3 网络上, 当 $k$=10000, $P(t)$=$n$ 时, 重构误差、运行时间与图的剩余节点数的关系. 可以看出, TS_LGS 算法在第 1 阶段时, 生成的摘要图的重构误差略高于算法 SAA 生成的摘要图, 低于其余算法生成的摘要图, TS_LGS 算法的运行时间则不超过其他算法的 1/2. 而在第 2 阶段时, TS_LGS 形成的摘要图重构误差增速开始变缓, 运行时间消耗开始增加. 当剩余节点数达到 $k$ 时, TS_LGS 的运行时间依旧比其余算法少, 同时找到的摘要图重构误差也是最低的. 这说明了 TS_LGS 算法分为两个阶段是合理且有效的.

(3) 查询结果对比

在 R1 与 R4 数据集上, 本文分别在不同 $k$ 下, 对原始图和摘要图进行了度、特征向量中心性、三角形密度等相关指标的查询结果对比. 从表 6 可以看出, 在小规模网络上, TS_LGS 算法在度和特征向量中心性的平均差与标准差上与 SAA 可以达到类似的效果, 在三角形密度相对差上则更接近 0. 而从表 7 可以看出, 随着图的规模增大, $k$ 的取值越来越小. TS_LGS 算法在各项查询指标上的结果都优于其余算法.

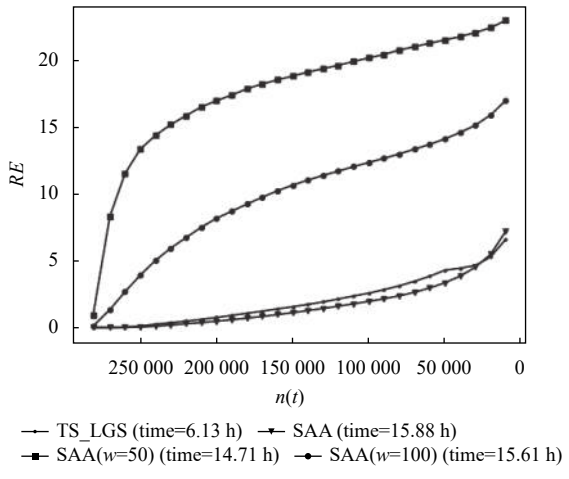



图 2 剩余节点数与重构误差

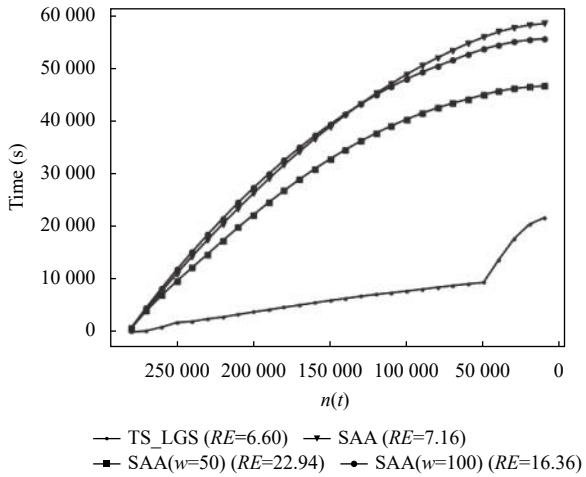



图 3 剩余节点数与运行时间

表 6 R1 上各算法查询结果对比

| $k$ | 算法 | $D_{\text{avgError}}$ | $D_{\text{stdError}}$ | $P_{\text{avgError}}$ | $P_{\text{stdError}}(\times 10^{-5})$ | $TD$ |
|---|---|---|---|---|---|---|
| 1 500 | SAA($w$=50) | 12.62 | 23.54 | 3.57 | 6.67 | −0.689 |
| | SAA($w$=100) | 6.67 | 12.42 | 1.89 | 3.52 | −0.419 |
| | SAA | **4.66** | **5.96** | **1.32** | **1.68** | −0.087 |
| | TS_LGS | 5.15 | 6.45 | 1.46 | 1.82 | **−0.084** |
| 1 000 | SAA($w$=50) | 16.59 | 22.87 | 4.70 | 6.48 | −0.800 |
| | SAA($w$=100) | 10.39 | 16.49 | 2.94 | 4.67 | −0.612 |
| | SAA | **7.63** | 8.66 | **2.16** | 2.45 | −0.150 |
| | TS_LGS | 7.86 | **8.62** | 2.22 | **2.44** | **−0.116** |
| 500 | SAA($w$=50) | 21.47 | 25.90 | 6.21 | 7.01 | −0.911 |
| | SAA($w$=100) | 15.35 | 23.94 | 4.43 | 5.48 | −0.832 |
| | SAA | 11.97 | 12.42 | 3.39 | 3.52 | −0.277 |
| | TS_LGS | **11.01** | **10.30** | **3.12** | **2.91** | **−0.161** |

表 7 R4 上各算法查询结果对比

| $k$ | 算法 | $D_{\text{avgError}}$ | $D_{\text{stdError}}$ | $P_{\text{avgError}}$ | $P_{\text{stdError}}(\times 10^{-7})$ | $TD$ |
|---|---|---|---|---|---|---|
| 10 000 | SAA($w$=50) | 4.67 | 8.87 | 4.77 | 9.08 | −0.965 |
| | SAA($w$=100) | 4.65 | 7.77 | 4.75 | 7.95 | −0.961 |
| | SAA | 4.68 | 7.84 | 4.79 | 8.02 | −0.960 |
| | TS_LGS | **4.57** | **7.37** | **4.67** | **7.54** | **−0.959** |
| 5 000 | SAA($w$=50) | 4.98 | 9.36 | 5.10 | 9.57 | −0.982 |
| | SAA($w$=100) | 4.96 | 8.69 | **5.04** | 8.89 | −0.980 |
| | SAA | 4.97 | 8.91 | 5.09 | 9.12 | −0.979 |
| | TS_LGS | **4.93** | **8.42** | 5.05 | **8.61** | **−0.977** |
| 1 000 | SAA($w$=50) | 5.41 | 11.48 | 5.53 | 11.17 | −0.996 |
| | SAA($w$=100) | 5.36 | 10.50 | 5.48 | 10.75 | −0.995 |
| | SAA | 5.33 | 10.16 | 5.46 | 10.03 | −0.995 |
| | TS_LGS | **4.93** | **8.42** | **5.05** | **8.51** | **−0.977** |

## 5 总结

对于图有损摘要问题, 本文提出了一种两阶段算法 TS_LGS. 首先, TS_LGS 计算了度大于平均度的节点数作为算法的阶段阈值. 在当前节点数大于阶段阈值的第 1 阶段, TS_LGS 提出了一种节点对合并的计算方法, 用于加速合并对摘要图质量影响较小的节点对; 当第 2 阶段, TS_LGS 基于图的平均度和结构信息提出了另一种节点对的挑选合并方法, 使得合并后的摘要图重构误差变化更小. 本文在 6 个不同规模的真实网络上进行了实验, 结果表明本算法与其他同类算法相比, 能够更快的寻找到质量更好的摘要图. 并且, TS_LGS 算法具有良好的扩展性, 能够适用于更大规模的网络.

## 参考文献

1 Tian YY, Hankins RA, Patel JM. Efficient aggregation for graph summarization. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver: ACM, 2008. 567–580. [doi: 10.1145/1376616.1376675]

2 Maserrat H, Pei J. Community preserving lossy compression of social networks. Proceedings of the 12th IEEE International Conference on Data Mining. Brussels: IEEE, 2012. 509–518.

3 Shin K, Ghoting A, Kim M, *et al*. SWeG: Lossless and lossy summarization of Web-scale graphs. Proceedings of the World Wide Web Conference. San Francisco: ACM, 2019. 1679–1690.

4 Bonifati A, Dumbrava S, Kondylakis H. Graph summarization. arXiv. 2004.14794, 2020.

5 王鹤. 基于图摘要的图模式挖掘研究与实现 [ 硕士学位论文 ]. 南京: 东南大学, 2018.

6 Rissanen J. Modeling by shortest data description. Automatica, 1978, 14(5): 465–471. [doi: 10.1016/0005-1098(78)90005-5]

7 Lelewer DA, Hirschberg DS. Data compression. ACM Computing Surveys, 1987, 19(3): 261–296. [doi: 10.1145/45072.45074]

8 Navlakha S, Rastogi R, Shrivastava N. Graph summarization with bounded error. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver: ACM, 2008. 419–432. [doi: 10.1145/1376616.1376661]

9 Ko J, Kook Y, Shin K. Incremental lossless graph summarization. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2020. 317–327. [doi: 10.1145/3394486.3403074]

10 Koutra D, Kang U, Vreeken J, *et al*. VOG: Summarizing and understanding large graphs. Proceedings of the 2014 SIAM

International Conference on Data Mining. Philadelphia: Society for Industrial and Applied Mathematics, 2014. 91–99.

11 LeFevre K, Terzi E. GraSS: Graph structure summarization. Proceedings of the 2010 SIAM International Conference on Data Mining. Columbus: Society for Industrial and Applied Mathematics. 2010. 454–465.

12 Beg MA, Ahmad M, Zaman A, *et al*. Scalable approximation algorithm for graph summarization. Proceedings of the 22nd Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Melbourne: Springer, 2018. 502–514.

13 Riondato M, García-Soriano D, Bonchi F. Graph summarization with quality guarantees. Data Mining and Knowledge Discovery, 2017, 31(2): 314–349. [doi: 10.1007/s10618-016-0468-8]

14 Lee K, Jo H, Ko J, *et al*. SSumM: Sparse summarization of massive graphs. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2020. 144–154. [doi: 10.1145/3394486.3403057]

15 Zhou HQ, Liu SH, Lee K, *et al*. DPGS: Degree-preserving graph summarization. Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics. Waltham, 2021. 280–288.

16 Fan WF, Li JZ, Wang X, *et al*. Query preserving graph compression. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale: ACM, 2012. 157–168. [doi: 10.1145/2213836.2213855]

17 Zhuang H, Rahman R, Hu X, *et al*. Data summarization with social contexts. Proceedings of the 25th ACM International Conference on Information and Knowledge Management. Indianapolis: ACM, 2016. 397–406. [doi: 10.1145/2983323.2983736]

18 Chierichetti F, Kumar R, Lattanzi S, *et al*. On compressing social networks. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Paris: ACM, 2009. 219–228. [doi: 10.1145/1557019.1557049]

19 Safavi T, Belth C, Faber L, *et al*. Personalized knowledge graph summarization: From the cloud to your pocket. Proceedings of the 2019 IEEE International Conference on Data Mining. Beijing: IEEE, 2019. 528–537.

20 王健. 基于张量分解的高效图摘要算法研究 [ 硕士学位论文 ]. 武汉: 华中科技大学, 2021. [doi: 10.27157/d.cnki.ghzku.2021.001960]

21 McNeil M, Ma BY, Bogdanov P. SAGA: Signal-aware graph aggregation. Proceedings of the 2022 SIAM International Conference on Data Mining. Alexandria: Society for Industrial and Applied Mathematics. 2022. 136–144.

22 Shin K. Mining large dynamic graphs and tensors [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2019.

23 徐正祥, 王英, 汪洪吉, 等. 基于特征加强的异构网络潜在摘要模型. 计算机科学与探索, 2022, 16(11): 2537–2546. [doi: 10.3778/j.issn.1673-9418.2104081]

24 Liu YK, Safavi T, Dighe A, *et al*. Graph summarization methods and applications: A survey. ACM Computing Surveys, 2019, 51(3): 62.

25 Ahmadi N, Sand H, Papotti P. Unsupervised matching of data and text. Proceedings of the 38th IEEE International Conference on Data Engineering. Kuala Lumpur: IEEE, 2022. 1058–1070.

26 Cormode G, Muthukrishnan S. An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms, 2005, 55(1): 58–75. [doi: 10.1016/j.jalgor.2003.12.001]

(校对责编: 牛欣悦)