Selective Edge Shedding in Large Graphs Under Resource Constraints

Yiling Zeng†, Chunyao Song†, Tingjian Ge‡
†Nankai University, Tianjin, China
‡University of Massachusetts Lowell, Lowell, USA
yilingzeng@mail.nankai.edu.cn, chunyao.song@nankai.edu.cn, ge@cs.uml.edu

***Abstract*—With the rapid development of the information age, many complex systems can be modeled as graphs. However, the unprecedented growth of data makes it extremely difficult for everyday users to process and mine very large graphs, given their limited computing resources such as personal computers and laptops. To address this challenge, we propose selective edge shedding, which can reduce the data volume, accelerate graph algorithms and queries, support interactive analysis, and eliminate noises. By estimating the original graph information from the reduced graph, it provides an efficient solution for network analysis at a low price. Previous graph reduction/summarization methods have a high demand on the hardware and are far less accurate in graph analysis results than ours; hence, there is a great need for efficient and effective graph reduction methods under resource constraints.**

In this paper, we propose two vertex-degree preserving edge shedding methods, the core of which are to maintain the expected vertex degree, so as to capture the basic characteristics of the network. Both methods allow users to control the process of edge shedding to generate a reduced graph of a chosen size based on the computing resource constraint. Using four real-world datasets in different domains, we have performed an extensive experimental evaluation of our methods, comparing against the state-of-the-art graph summarization method on as many as seven graph analysis tasks. The experimental results show that the methods proposed in this paper can achieve up to 65% higher accuracy on graph analysis tasks compared to the competitive method, while consuming only 26%-57% running time, which fully demonstrates the advantages of the methods proposed in this work.

***Index Terms*—Degree distribution, Edge shedding, Graph reduction, Limited resources**

I. Introduction

In modern society, there are various forms of data representation and storage methods. Many complex systems can be represented as graphs, such as social networks, academic networks and transportation networks. Complex graphs/information networks contain a lot of information. Through deep mining of the network, we can discover the potential connections between the data and make subsequent use of information effectively.

However, with the rapid development of the information age, the data volume is expanding rapidly. Daily activities like social media interaction, smart phone usage, web browsing, product and service purchases, and wellness sensors generate large amounts of data, and hence the scale of network models is gradually expanding. As of the fourth quarter of 2019, Facebook had nearly 2.5 billion monthly active users, making it the world's largest social network [1] . As of the first quarter of 2019, Twitter, a social platform called "Internet SMS", had nearly 330 million users and generated more than 10 million "tweets" every day [2].

Although the data generated by human activities can be stored and represented with networks, the speed and computing feasibility of processing, analysis, and mining of large-scale networks cannot keep up with the data-volume growth, using the computers of everyday users. For example, graph mining tasks such as link prediction and node clustering are impossible to be executed on the desktop or laptop computer of an ordinary user, such as a scientist, due to the sheer size of the graphs. Therefore, it is of great practical significance to reduce the cost of network analysis using graph reduction techniques such as edge shedding.

There are four main advantages of performing graph reduction. First, graph reduction can reduce the size of data, thus saving storage space. Second, using reduced graphs can speed up graph processing algorithms and queries. Third, it makes the visualization of data more feasible. Finally, real datasets often have many hidden or wrong links and labels. Graph reduction can filter noises and reveal the pattern characteristics of the data.

In 2020, the outbreak of COVID-19 has led to a surge in demand for online platforms since lots of people began to work from home. Various companies and institutions are having a hard time trying to keep up with the increasing demand on computing bandwidth and storage space. This shows the importance of storage capacity and data processing ability in the current big data era. At present, most graph reduction methods have high requirements on hardware resources such as memory and CPU. Even though hardware becomes cheaper, most laboratories in schools and small-scale enterprises with limited funds, as well as individual users such as scientists not doing large-scale data analysis on a regular basis, cannot afford the cost of acquiring or maintaining high-end data servers, resulting in large graph data not being used effectively. In addition, there has been increasing demand and market for edge computing [3], where preliminary data processing is pushed to less powerful devices—our techniques for graph reduction will be much needed. To the best of our knowledge, efficient graph reduction methods under resource constraints have not been proposed before.

A. Related Work

Graph reduction techniques can be divided into several popular categories, such as grouping or aggregation based, bit compression based and simplification or sparsification based techniques.

Grouping-based reduction methods involve node-grouping based methods and edge-grouping based methods. LeFevre et al. [4] proposed a method of generating graph reduction by greedily grouping nodes with the goal of minimizing the normalized reconstruction errors. Riondato et al. [5] focused on generating supernodes and superedges with guarantees, while completely ignoring the preservation of important parts and regions of the graph. Graph Dedensification [6] is an edge-grouping based method that compresses neighborhoods around high-degree nodes, accelerating query processing and enabling direct operations on the compressed graph. Fan et al. [7] proposed a "blueprint" for lossless queries on compressed attributed graphs. Kumar et al. [8] proposed a novel iterative utility-driven graph reduction approach. The utility is defined as the useful information of the summarized graph and is user-specified. This is also the state-of-the-art method, and we demonstrate the superiority of our methods over it in the experiments.

Bit Compression-based reduction methods aim to minimize the number of bits needed to describe the input graph. Navlakha et al. [9] propose a highly compact two-part representation, which allows for both lossless and lossy graph compression with bounds on the introduced error. Ahnert [10] introduces a framework for the discovery of dominant relationship patterns in transcription networks, by compressing the network into a power graph with overlapping power nodes.

Simplification-based reduction methods generate reduced graphs by removing less "important" nodes or edges from original graphs. Shen et al. [11] present a visual analytics tool, OntoVis, which allows users to do structural abstraction and importance filtering to make large networks manageable. Li et al. [12] design several abstraction criteria to distill representative and important information to construct the abstracted graphs for visualization. Both of our proposed methods using selective edge shedding belong to this category.

Papers [13] and [14] are surveys of graph reduction. Most of the techniques discussed above aim to minimize reconstruction error or guarantee output utility. However, these reduction methods have high requirements on hardware resources, which is not suitable for laboratories in some universities, small enterprises, or other environments with resource constraints, as discussed above.

In addition, TCM [15] and GSS [16] are novel graph stream summarization techniques concentrating on graphs with multiple edges which appear at different timestamps. Graph scaling techniques such as [17] and [18], as well as the above mentioned techniques, focus on different problems from ours.

B. Our Contributions

In this paper, we propose two novel edge shedding based graph reduction techniques: Centrality Ranking with Rewiring (CRR) and B-Matching with Bipartite Matching (BM2). Both of our methods aim to preserve the distribution of vertex degrees with edge shedding in different ways. Our contributions in this work are summarized as follows:

(1) We propose two efficient algorithms for judicious edge shedding of large graphs to achieve parameterized graph reduction.

(2) At present, research in the graph reduction field pays more and more attention to the controllability of the reduced graph size. Therefore, according to the actual needs of users in different scenarios, the edge preservation ratio p is used in our methods to generate graphs of different sizes, where $p \in (0, 1)$ represents the size ratio between the reduced and the original graph.

(3) We conduct a comprehensive experimental study using real datasets in four different domains performing seven different graph analysis tasks. The results show that our methods can generate reduced graphs with limited resources while assuring a certain level of accuracy, which has greatly improved the efficiency and accuracy compared to other competitive methods.

The rest of the paper is organized as follows. In Section II, we state the problem and introduce the relevant background knowledge. In Sections III and IV, we present the proposed CRR and BM2 algorithms, and illustrate them with detailed running examples. Section V contains comprehensive experiments which evaluate the proposed algorithms' efficiency and accuracy on real-world datasets in four different domains. At last, we conclude the paper in Section VI.

II. Preliminaries

A. Problem Definition

The primary idea of our graph reduction is to capture and preserve the structure of the original network. By using the reduced graph, we can estimate various properties of the original graph, so as to accelerate the downstream graph analysis tasks. How to capture and preserve the structure of the original network efficiently is the key requirement of graph reduction techniques. Vertex degree is one of the most basic features in a network topology and plays a crucial role in network analysis [19]. In current studies, the importance of vertex degree in capturing network properties has been well established, including communication network topology [20] and complex network modeling [21].

Based on these observations, we arrive at the following idea: with preserving vertex degrees as the main principle, we aim to capture the essence of the original graph by maintaining the expected vertex degrees of each node, so as to accurately preserve and approximate other characteristics.

Table I summarizes the symbols we will use later in the paper.

Given an undirected graph $G = (V, E)$ and an edge preservation ratio p, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $p \in (0, 1)$ is the specified edge preservation ratio. Let $G' = (V', E')$ be the reduced graph, where V' is the node set, $E' \subseteq V' \times V'$ is the edge set, and G' is a subgraph of G. For the original graph G, $\forall u \in V, deg_G(u)$

TABLE I
LIST OF SYMBOLS USED IN THE PAPER

Symbol	Definition
G	Initial graph
G'	Reduced graph, which is a subset of G
p	Edge preservation ratio
$deg_G(u)$	Degree of node u in G
$deg_{G'}(u)$	Degree of node u in G'
$E(deg_{G'}(u))$	Expected degree of node u in G'
$E(deg_{G'})$	Expected average node-degree of G'
$dis(u)$	Discrepancy between the actual and expected degree of u in G'
Δ	Sum of the absolute values of the discrepancy of each node in G'

denotes the degree of a single node u. For a reduced graph G', $\forall u \in V'$, $deg_{G'}(u)$ represents the degree of a single node u in G'; $E(deg_{G'}(u))$ represents this node's expected degree in G'; and $E(deg_{G'})$ denotes the average expected degree of the reduced graph G'.

The edge preservation ratio p is a controllable parameter in the edge shedding process. The smaller p is, the smaller the size of the final reduced graph. Therefore,

$$E(deg_{G'}(u)) = deg_G(u) \times p \tag{1}$$

The average expected degree of the reduced graph can be transformed into:

$$\begin{aligned} E(deg_{G'}) &= \frac{1}{|V|}\sum_{u\in V} E(deg_{G'}(u)) = \frac{1}{|V|}\sum_{u\in V}\{deg_G(u) * p\} \\ &= \frac{p}{|V|} * \sum_{u\in V} deg_G(u) \end{aligned} \tag{2}$$

Let $dis(u)$ represent the degree difference between the original and the reduced graphs of vertex u. It can be expressed as:

$$dis(u) = deg_{G'}(u) - E(deg_{G'}(u)) \tag{3}$$

The sum of the degree differences of all nodes Δ between the two graphs can be computed as:

$$\Delta = \sum_{u\in V} |dis(u)| \tag{4}$$

According to the above description, the measurement of edge shedding methods based on vertex degrees depends on the degree difference Δ between the expected and the reduced graphs. The smaller the degree difference Δ is, the closer the vertex degree distributions are between the reduced and the expected graphs.

So the problem can be defined as follows. Given an original graph G, obtain a representative reduced graph G', where $G' = \arg\min_{G^* \sqsubseteq \mathcal{G}} \Delta(G^*)$ and $\mathcal{G}$ is the set of all possible reduced graphs of G. According to [22], the problem is NP-hard; so we decide to provide efficient approximate methods to solve it. Two different methods in this paper are proposed to judiciously control the edge shedding process based on the ratio parameter p, which preserves not only the vertex-degree distribution but also the key topological connectivity of the graph.

B. Centrality

To preserve the key topological connectivity, we need to introduce the notion of centrality. Centrality is a commonly used concept in social network analysis which is used to express the importance of vertices or edges in graphs. Among different centrality computing methods, *betweenness centrality* is the one most frequently employed in network analysis [23].

The betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v. Betweenness centrality is computed as:

$$C_B(v) = \sum_{s,t\in V} \frac{\sigma(s,t \mid v)}{\sigma(s,t)}$$

where V is the set of nodes, $\sigma(s,t)$ is the number of shortest (s,t)-paths, and $\sigma(s,t \mid v)$ is the number of those paths passing through some node v other than s, t.

For the measurement of centrality of *edges*, betweenness centrality is still applicable. The formula is as follows:

$$C_B(e) = \sum_{s,t\in V} \frac{\sigma(s,t \mid e)}{\sigma(s,t)}$$

where symbols share similar meanings as $C_B(v)$.

Betweenness centrality focuses on the role of an edge serving as a "bridge". Brandes [24] has proposed a fast algorithm to compute betweenness centrality which requires $O(|V|+|E|)$ space and runs in $O(|V||E|)$ time on unweighted networks.

III. CENTRALITY RANKING WITH REWIRING (CRR)

Preserving the degree distribution of the original graph can start from preserving the expected average vertex degree. The proposed algorithm CRR has two phases. Firstly, according to the importance of all edges, it generates an initial reduced graph with the same expected average degree computed by fixing the edge-preservation ratio p following Equation 2. Since the importance of edges is taken into account, the initial result violates the goal of minimizing degree difference to some extent. Therefore, in the second phase, we replace some edges to reduce the node-degree discrepancy.

Algorithm 1 illustrates the process of CRR.

The first phase (lines 1-6) performs the initial edge-shedding. According to the discussion in Section II-A, the expected average degree of the reduced graph $E(deg_{G'})$ is equal to p times the average degree of the original graph. That is, when the reduced graph contains $p \times |E| = P$ edges, the required expected average degree is satisfied. Based on this, we let the reduced graph contain $[P]$ (the nearest integer of P) edges. Given that edges have different roles in preserving the graph topology, intuitively, edges that are more "important" in preserving the graph structure should be kept (and the rest are subject to edge shedding). As mentioned in Section II-B, betweenness centrality focuses on the role of an edge for serving as a "bridge". The higher the betweenness centrality of an edge is, the greater its contribution to the network connectivity. Kumar et al. [8] have confirmed that

Algorithm 1: Centrality Ranking with Rewiring (CRR)

Input: undirected graph $G = (V, E)$, edge preservation ratio p, $steps$
Output: reduced graph $G' = (V', E')$

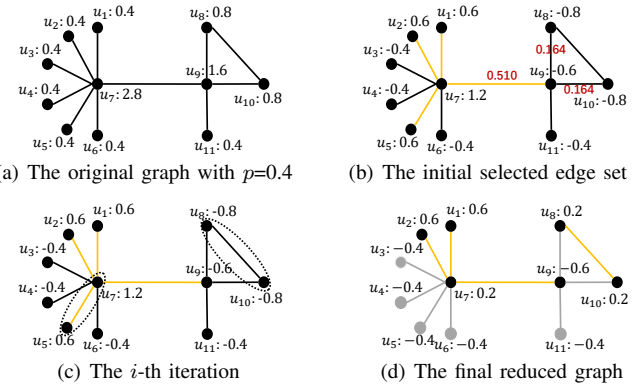
1 initialize $E' \leftarrow \emptyset, i \leftarrow 0$
2 $P \leftarrow p \times |E|$
3 calculate the betweenness centrality of all edges, and sort E in non-increasing order by their betweenness centrality
4 **while** $|E'| < [P]$ *(nearest integer of P)* **do**
5 $\quad e \leftarrow E.$ next ()
6 $\quad E' \leftarrow E' \cup e$
7 **for** $i \leftarrow 1 \ldots steps$ **do**
8 $\quad$ pick a random edge $e_1 = (u, v)$ from E'
9 $\quad$ pick a random edge $e_2 = (x, y)$ from $E \backslash E'$
10 $\quad d_1 \leftarrow |dis(u) - 1| + |dis(v) - 1| - (|dis(u)| + |dis(v)|)$
11 $\quad d_2 \leftarrow |dis(x) + 1| + |dis(y) + 1| - (|dis(x)| + |dis(y)|)$
12 $\quad$ **if** $d_1 + d_2 < 0$ **then**
13 $\quad\quad E' \leftarrow (E' - \{e_1\}) \cup \{e_2\}$
14 return G'

betweenness centrality has a unique advantage in measuring edge importance. Therefore, in Phase 1, CRR first calculates the betweenness centrality of all edges (line 3), and then selects $[P]$ edges with top betweenness centrality to form the initial reduced graph (lines 4-6).

The second phase (lines 7-13) takes care of edge rewiring on the initial reduced graph. In Phase 1, taking $[P]$ top centrality edges does not ensure the degrees of each node to be near their expected values. Also, taking edge importance as the basis for initial selection may compromise the goal of minimizing the total degree difference Δ to a certain extent. Therefore, in Phase 2, CRR performs several iterations of edge swapping to improve the overall degree difference. During each iterative step, edge replacement ensures that the number of edges in the reduced graph is always $[P]$, thereby maintaining the expected average degree.

The number of iterations $steps$ depends on the user's preference for efficiency vs. accuracy. Based on the results of extensive experiments, it is generally recommended that users choose $steps$ as $[10 \times P]$ (the nearest integer of $10 \times P$). At each iteration i, let the current edge set be E'. CRR randomly chooses two edges $e_1 = (u, v) \in E', e_2 = (x, y) \in E \backslash E'$ (lines 8-9). $d_1 = |dis(u) - 1| + |dis(v) - 1| - (|dis(u)| + |dis(v)|)$ represents the change in overall degree difference caused by removing e_1, and $d_2 = |dis(x) + 1| + |dis(y) + 1| - (|dis(x)| + |dis(y)|)$ represents the change in overall degree difference caused by adding e_2 (lines 10-11). If $d_1 + d_2 < 0$, it means the edge replacement will reduce the overall degree difference, which should be performed. Otherwise, we do nothing (lines 12-13). The final edge set is returned as the reduced graph (line 14).

Example 1. ***Running example of CRR:*** *Figure 1(a) shows an original graph before edge shedding, where the number next to each node represents the expected degree for the reduced graph with the edge preservation ratio $p = 0.4$. First, CRR computes $[P] = [p \times |E|] = [0.4 \times 11] = 4$, and then it calculates the importance of each edge using the betweenness centrality formula $C_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$. The importance of edges is shown in red, while the importance of unmarked edges is 0.182 in Figure 1(b). According to the above calculation, the four most important edges are selected as the initial chosen edge set, where edges of the same importance are selected randomly. The result is shown in Figure 1(b), where the number next to each node indicates the degree difference dis_u, and the yellow edges form the initial selected edge set.*



(a) The original graph with p=0.4 (b) The initial selected edge set

(c) The i-th iteration (d) The final reduced graph

Fig. 1. An illustration of running CRR

Next, CRR starts the second phase. First, we compute $steps = [10 \times P] = 44$, the number of iterations. At the i-th iteration, CRR randomly chooses $e_1 = (u_5, u_7)$ and $e_2 = (u_8, u_{10})$ for an edge replacement attempt, which is shown in Figure 1(c). Then we compute $d_1 = |0.6 - 1| + |1.2 - 1| - (|0.6| + |1.2|) = -1.2$ and $d_2 = |-0.8 + 1| + |-0.8 + 1| - (|-0.8| + |-0.8|) = -1.2$. Because $d_1 + d_2 < 0$, the edges are swapped. The overall degree difference is reduced by 2.4, and the current E' is composed of $\{(u_1, u_7), (u_2, u_7), (u_7, u_9), (u_8, u_{10})\}$. The subsequent iteration steps are similar, and the final selected edge set returned by CRR is $E' = \{(u_1, u_7), (u_2, u_7), (u_7, u_9), (u_8, u_{10})\}$, as shown in Figure 1(d). The black nodes correspond to the reduced graph's node set V', and the yellow edges form the final selected edge set E'.

Theorem 1. *The average absolute difference between the degree of a node in the graph produced by CRR and its expectation is in the range $(0, 4p(1-p)\frac{|E|}{|V|})$.*

Proof. We first prove the claim that an upper bound of Δ (i.e., the total absolute difference between node degrees in the reduced graph and their expectations) is achieved when a subset U of nodes all have degree 0 and the remaining nodes $V \backslash U$ all keep their original degrees (except possibly one node, which may have a smaller degree), subject to the constraint that the total number of edges is $p|E|$.

To prove this claim, suppose in the reduced graph G' with the maximum Δ, there is a subset U of nodes whose degrees are below their expectations (i.e., $p \cdot deg_G(u)$) and the remaining nodes $V \setminus U$ have degrees above or equal to their expectations. Suppose we can keep the node degree statistics, but can arbitrarily rewire the edges while keeping the same

number of edges. That is, each node has the same number of "half edges" or "spokes" as its degree and we are allowed to arbitrarily reconnect two half edges into one edge. Thus, in the above reduced graph G' with the maximum Δ, as long as there is a node in $V \setminus U$ with a degree in G' less than that in G, we can keep moving edges in G' to be connecting two nodes in $V \setminus U$ only—this could only increase Δ towards the upper bound. In particular, for every two "bridge" edges crossing U and $V \setminus U$, we can rewire them and have one edge connecting the same 2 nodes in U and one edge connecting the same 2 nodes in $V \setminus U$.

At the end of this moving process, one of the following two cases will happen: (1) All nodes in $V \setminus U$ have full degrees and U may have some internal edges; (2) All nodes in U have degree 0. In case (1), within the node set U, we can keep moving all remaining internal edges towards any particular node u or several nodes as self-edges until they reach their maximum degree in G. This does not change Δ initially when the degree of u in G' is below its expectation, and will increase Δ once the degree of u reaches above the expectation. So in the end, we will reach the scenario as described in the claim, i.e., a rewired graph G' with a subset of nodes have full degrees, possibly another node with a positive degree, and all remaining nodes with degree 0, will give an upper bound of Δ. In the same vein, for case (2), we can move the internal edges in $V \setminus U$ to saturate the degrees of a subset of nodes without decreasing Δ. Thus, the claim above is proven.

Let U' be the set of nodes with degree 0 and $V \setminus U'$ has all the $p|E|$ edges at the end of the proof construction above. Based on the result of the claim, we have $\Delta \leq \sum_{u \in U'} deg_G(u) \times p + \sum_{u \in V \setminus U'} deg_G(u) \times (1-p) = p \times (1-p) \times 2|E| + (1-p) \times 2p|E| = 4p(1-p)|E|$, and we obtain the result for the range of the average absolute difference as in the theorem.

□

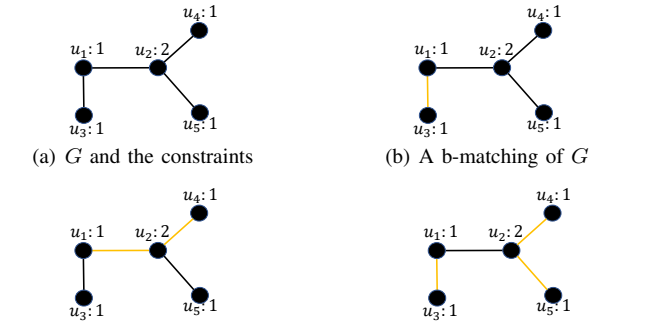
Let us now analyze the complexity of CRR. The first phase involves the calculation and ranking of betweenness centrality, whose time complexity is $O(|V||E|)$ and $O(|E|log_2|E|)$, respectively, and the initial edge selection has a cost of $O(|E|)$. The second phase is a linear process of edge replacement. So the overall time complexity of CRR is $O(|V||E| + |E|log_2|E| + |E| + steps)$, which is simplified to $O(|V||E| + |E|log_2|E| + steps)$.

IV. B-Matching with Bipartite Matching (BM2)

In this section, we first introduce the *maximum b-matching* problem [25], which is also an edge shedding problem, and then propose the BM2 algorithm, drawing the connection between the maximum b-matching problem and the graph reduction problem.

A. B-Matching

Let us consider an undirected graph $G = (V, E)$ and a set of capacity constraints $b(u) : V \to \mathbb{N}$. For the subgraph $H = (V, E_H)$ of G, if the degree of any vertex $u \in V$ in H is at most $b(u)$, then H is a b-matching of G. If the addition of any edge violates at least one capacity constraint, the current b-matching is maximal. A maximum b-matching is a maximal b-matching with the largest number of edges. Figure 2 shows a b-Matching instance, where the number next to each node indicates its capacity constraint, and the matched edges are shown in yellow.



(a) G and the constraints (b) A b-matching of G

(c) A maximal b-matching of G (d) The maximum b-matching of G

Fig. 2. A B-Matching example

Parchas et al. [22] have studied the connection between the maximum b-matching problem and the problem of extracting representative instances from uncertain graphs. It turns out that a solution to the maximum b-matching problem would also be a good initial constraint enforcer for our edge shedding problem. We may use $E(deg_{G'}(u))$ in the reduced graph as the $b(u)$ in a b-matching problem.

B. B-Matching with Bipartite Matching (BM2)

Since the edge preservation ratio p is in $(0, 1)$, $E(deg_{G'}(u))$ is possibly a fraction number. Thus, the b-matching model described in Section IV-A cannot be applied directly to the graph reduction problem, and we need to round the expected vertex degrees to integers. Therefore, BM2 involves two phases. First, it performs b-matching on a transformed graph to obtain an initial edge set. Next, it picks additional edges which can improve the total degree difference Δ by performing a bipartite matching. Algorithm 2 illustrates the process of BM2.

The first stage (lines 1-7) generates the initial selected edge set. First, we use Equation 1 to calculate the expected vertex degree for each node (line 1) and initialize the vertex degrees in the reduced graph (line 2). In order to solve our problem with real-value constraints, BM2 rounds the expected degrees to the closest integers and runs b-matching to get a maximum b-matching using the rounded values as capacity constraints (lines 3-7). Although there may be more than one maximum b-matching, our algorithm only finds one of them, and proceeds to the second phase using that one.

The second stage (lines 8-25) picks an edge set to further reduce the overall degree difference. The rounding strategy in Phase 1 leads to some differences between the actual and the expected degrees of the reduced graph. This requires deviation correction of the initial selected edge set to obtain the closest vertex degree distribution. BM2 first classifies nodes according to each node's degree difference. Specifically, nodes are divided into three groups A, B, and C, such that $\forall u \in A, dis(u) \leq -0.5$, $\forall u \in B, -0.5 < dis(u) < 0$, and

Algorithm 2: B-Matching with Bipartite Matching (BM2)

Input: undirected graph $G = (V, E)$, edge preservation ratio p
Output: reduced graph $G' = (V', E')$

1 calculate the expected degree $E(deg_{G'}(i))$ for all vertices in V
2 initialize $E_m \leftarrow 0, deg_{G'}(i) \leftarrow 0$
3 $b_i \leftarrow round(E(deg_{G'}(i)))$ for each vertex i
4 **for** *each* $(u, v) \in E$ **do**
5 **if** $deg_{G'}(u) < b_u$ *and* $deg_{G'}(v) < b_v$ **then**
6 $E_m \leftarrow E_m \cup \{e\}$
7 $deg_{G'}(u) \leftarrow deg_{G'}(u)+1, deg_{G'}(v) \leftarrow deg_{G'}(v)+1$
8 $A \leftarrow \emptyset, B \leftarrow \emptyset, C \leftarrow \emptyset$
9 **for** *each* $u \in V$ **do**
10 $dis(u) = deg_{G'}(u) - E(deg_{G'}(u))$
11 **if** $dis(u) \leq -0.5$ **then**
12 $A \leftarrow A \cup \{u\}$
13 **else if** $-0.5 < dis(u) < 0$ **then**
14 $B \leftarrow B \cup \{u\}$
15 **else**
16 $C \leftarrow C \cup \{u\}$
17 $E^* \leftarrow E - E_m$
18 **for** *each* $e = (u, v) \in E^*$ **do**
19 gain $= |dis(u)| + 2\,|dis(v)| - |1 + dis(u)| - 1$
20 **if** $u \in A$ *and* $v \in B$ *and* gain ≥ 0 **then**
21 $w(e) \leftarrow$ gain
22 **else**
23 discard e from E^*
24 let G^* be $((A \cup B), E^*, W)$ where W is the set $\{w(e)\}$
25 $E_{BP} =$ bipartite (G^*)
26 $E' = E_m \cup E_{BP}$

$\forall u \in C, dis(u) \geq 0$ (lines 8-16). For a node in group A, its absolute degree difference will decrease by adding a connected edge in the reduced graph. For a node in group B, its absolute degree difference will increase by less than 1 by adding a connected edge in the reduced graph. For a node in group C, its absolute degree difference will increase by 1 since it has already reached or exceeded its expected degree.

Next, let us discuss the different situations with respect to the membership of two endpoints of an edge in the sets A, B, and C.

For edges connecting nodes (1) within group B (2) between groups A and C (3) between groups B and C (4) within group C, the addition of an edge will definitely lead to an increase in the overall degree difference. Furthermore, those edges $e = (u, v) \in E$, where $u \in A$ and $v \in A$, must have all been added to E_m in Phase 1—so we do not need to consider them at this point. Therefore, we only focus on those edges $e = (u, v) \in E$, where $u \in A$ and $v \in B$ or vice versa.

Lemma 1. *Let $e = (u, v)$ where $u \in A$ and $v \in B$. The addition of e to the reduced graph changes the overall degree difference Δ by gain $= |dis(u)|+2|dis(v)|-|dis(u)+1|-1$.*

Proof. For $e = (u, v) \in E$, where $u \in A$ and $v \in B$, let us calculate the degree differences caused by its addition. Before adding this edge to the selected edge set, the total degree difference of vertices u and v is $d_1 = |dis(u)| + |dis(v)|$. After its addition, the total degree difference is $d_2 = |dis(u)+1| + |dis(v) + 1| = |dis(u) + 1| + (1 - |dis(v)|)$. Define $gain$ as the change by adding e, i.e., $gain = d_1 - d_2 = (|dis(u)| + |dis(v)|) - (|dis(u) + 1| + (1 - |dis(v)|)) = |dis(u)| + 2|dis(v)| - |dis(u) + 1| - 1$. If $gain > 0$, the degree difference between the reduced graph and the original is reduced. This concludes the proof. □

According to Lemma 1, in the second phase, BM2 adds edges connecting nodes between groups A and B into graph G^* if the $gain$ is > 0 to form a weighted bipartite graph, where the weight of an edge is the gain (lines 17-23). Next, it performs an approximate maximum weight bipartite matching on graph G^* to identify the edge set that minimizes the overall degree difference (lines 24-25). The final selected edge set of BM2 is composed of E_m and E_{BP} (line 26).

Algorithm 3 illustrates the *bipartite* algorithm called in line 25 of Algorithm 2.

Algorithm 3: bipartite

Input: bipartite graph $G^* = (A \cup B, E^*, W)$
Output: edge set $E_{BP} \subseteq E^*$

1 initialize $E_{BP} \leftarrow \varnothing$
2 sort edges $e \in E^*$ in non-increasing order of their weights $w(e)$ and add them into a priority queue Q
3 **while** $Q \neq \varnothing$ **do**
4 $e = (a, b) \leftarrow Q.next()$
5 $E_{BP} \leftarrow E_{BP} \cup \{e\}$
6 discard all edges in Q incident to b
7 $dis(a) \leftarrow dis(a) + 1$
8 **if** $-1 < dis(a) < -0.5$ **then**
9 **for** *each* $e' = (a, x) \in E^*$ **do**
10 $w(e') \leftarrow |dis(a)| + 2|dis(x)| - |1 + dis(a)| - 1$
11 **if** $w(e') > 0$ **then**
12 update order of e' in Q
13 **else**
14 discard edge e' from Q
15 **else if** $dis(a) > -0.5$ **then**
16 **for** *each* $e' = (a, x) \in E^*$ **do**
17 discard edge e' from Q

First, edges are sorted in non-increasing order of weights and then added to a priority queue Q (line 2). At each iteration, we add the head $e = (a, b)$ of Q to the bipartite set E_{BP} (lines 4-5). Due to the addition of edge e, we need to modify the degree differences of the relevant vertices, and update the bipartite graph and Q (lines 6-17). Algorithm 3 terminates when Q is empty.

Due to the addition to the selected edge set, the degree difference is dynamically changing. The algorithm needs to keep the correctness of vertex classification and the order of Q. The discussion is as follows.

Take the head $e = (a, b)$ of Q and add it to E_{BP}, then BM2 updates the classification of relevant vertices. The new degree difference of b is $dis_{new}(b) = dis(b) + 1 > 0$, which does not meet the requirement of group B; so b and the edges adjacent to b are removed from the bipartite graph (line 6). For a, there are different situations.

Lemma 2. *Let $e = (a, b) \in Q$ where $dis(a) \leq -2$. The change of $dis(a)$ does not influence the gains of edges adjacent to a.*

Proof. The gain of e is $|dis(a)|+2|dis(b)|-|dis(a)+1|-1$. Since $dis(a) \leq -2$, it can be computed as $|dis(a)| + 2|dis(b)| - (|dis(a)| - 1) - 1 = 2|dis(b)|$. So it depends only on $|dis(b)|$ and cannot be influenced by $dis(a)$. □

According to Lemma 2, if $dis(a) \leq -2$, the gains of edges adjacent to a do not change, and we do nothing. If $-2 < dis(a) < -1.5$, we just update the gains of edges adjacent to a using the formula in Lemma 1. If the new gain of an edge becomes negative, we remove it from Q (lines 8-14). If $dis(a) \geq -1.5$, $dis(a)+1 \geq -0.5$, the new degree difference of vertex a does not meet the requirement of group A; so a and the edges adjacent to a are removed from the bipartite graph (lines 15-17).

Example 2. ***Running example of BM2:*** *Figure 1(a) shows an original graph to be reduced. For the compression ratio $p = 0.4$, the number next to each node represents the expected degree for the reduced graph. First, the original graph is rounded and transformed to Figure 3(a). Figure 3(b) shows the maximum b-matching of the transformed graph, where the number next to each node indicates the degree difference between the current selected result and the expected reduced vertex degree. The initial selected edge set is $E_m = \{(u_7, u_9), (u_8, u_{10})\}$.*

According to Figure 3(b), the vertices are classified as $A = \{u_7, u_9\}, B = \{u_1, u_2, u_3, u_4, u_5, u_6, u_{11}\}, C = \{u_8, u_{10}\}$. Figure 3(c) shows the weighted bipartite graph, including the degree differences and the gains of each edge.

At the first iteration, Algorithm bipartite first selects the edge $e = (u_7, u_1)$ with the largest gain to join E_{BP}. Then it updates the degree difference of u_7 to -0.8, deletes u_1, and updates the gains of all edges connected to u_7. The result is shown in Figure 3(d). At the second iteration, edge $e = (u_7, u_2)$ is selected to be added to E_{BP}. Also, bipartite updates u_7's degree difference to 0.2 and deletes u_2. Because u_7 no longer belongs to group A, u_7 and all edges connected to it are removed. Since the gain of $e = (u_9, u_{11})$ is 0 as shown in Figure 3(e), it can be selected or discarded according to user's preference. If it is discarded, the bipartite algorithm ends, and $E_{BP} = \{(u_7, u_1), (u_7, u_2)\}$. The final edge set of BM2 is $E' = \{(u_7, u_9), (u_8, u_{10}), (u_7, u_1), (u_7, u_2)\}$. The final reduced graph is shown in Figure 3(f), where the yellow edges and the black nodes are selected.

Theorem 2. *For $p \in (0, 1)$, the average absolute difference between the degree of a node in the graph produced by BM2 and the expectation is in $(0, \frac{1}{2} + (1-p)\frac{|E|}{|V|})$.*

Proof. During the Phase 1 of BM2, we first perform the rounding operation, which would cause 0.5 degree difference for each node at most. Then we give a linear time approximation algorithm for the Cardinality b-Matching Problem [25], and the Δ after Phase 1 should be at most $\sum_{u \in V} \frac{1}{2} + \frac{1}{2} \times \sum_{u \in V} | - deg_G(u) \times p| = \frac{1}{2} \times |V| + \frac{1}{2} \times p \sum_{u \in V} deg_G(u) = \frac{1}{2}|V| + \frac{1}{2} \times p \times 2|E| = \frac{1}{2}|V| + p|E|$.

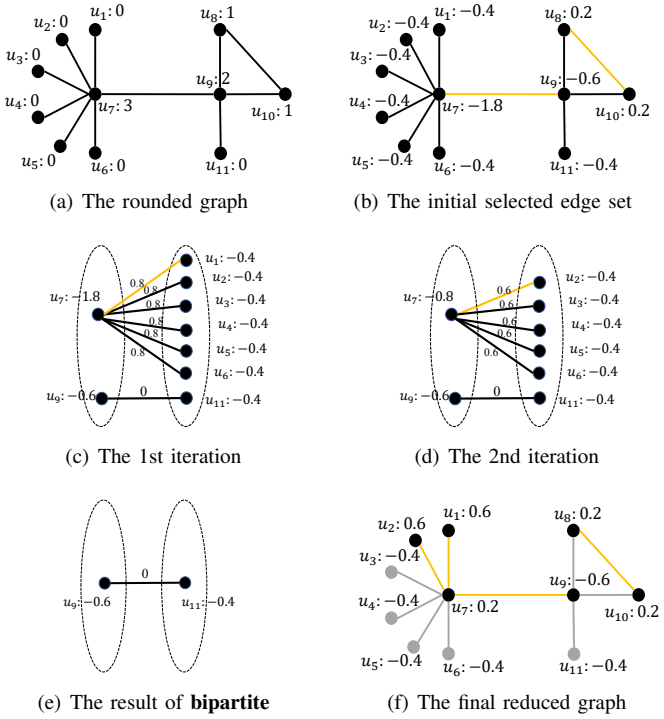


Fig. 3. An illustration of running BM2

Consider the process of BM2, it starts to add edges from 0 and will not cause edge overload except for the 0.5 degree rounding. So the maximum degree difference before Phase 1 starts is $\sum_{u \in V} | - deg_G(u) \times p| = 2p|E|$ for BM2. We define $\frac{\frac{1}{2}|V|+p|E|}{2p|E|} = \frac{|V|}{4p|E|} + \frac{1}{2}$ to indicate the optimization ratio of degree difference from BM2 under different values of p. It shows that a larger p results in a smaller degree difference. Thus, the Δ for $p > 0.5$ should be less than that of $p \leq 0.5$.

When $p \leq 0.5$, we have $p|E| \leq (1-p)|E|$, so $\Delta \leq \frac{1}{2}|V| + (1-p)|E|$. Thus, overall, we have $\Delta \leq \frac{1}{2}|V| + (1-p)|E|$, which gives the range of the average absolute difference as stated in the theorem.

□

In Phase 1 of BM2, it includes the linear-time processing of nodes and edges. In Phase 2, the vertex classification and edge selection are also in linear-time; so the time complexity of these two parts is $O(|V| + |E|)$. The rest of BM2 involves bipartite edge sorting and selection in which each edge of E^* can be processed at most $|B|$ times. So the overall time complexity of BM2 is $O(|V|+|E|+|E^*|log_2|E^*|+|B||E^*|)$.

The time complexity of CRR is $O(|V||E| + |E|log_2|E| + |E|+steps)$, where $steps$ is set to $x \times P$ and $P = p \times |E|$. Since x and p are constants, $steps$ depends only on $|E|$. In addition, $|B|$ is smaller than $|V|$ and $|E^*|$ is smaller than $|E|$, thus $|B||E^*|$ is smaller than $|V||E|$ and $|E^*|log_2|E^*|$ is smaller than $|E|log_2|E|$. This analysis shows that the time complexity of CRR is greater than that of BM2, which is consistent with our experimental results in Section V-B.

V. Experimental evaluation

A. Experimental Settings

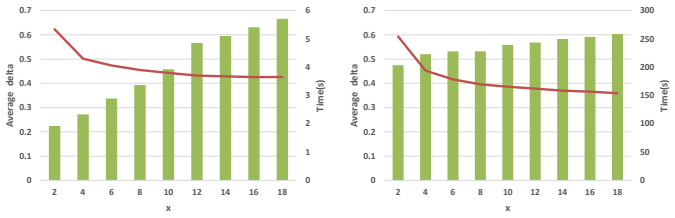
Setup. We perform the experiments on a desktop with 16GB memory, Intel Core i7 processor and 3.40GHz frequency. We use Python and create graphs using the snap library provided by Stanford Network Analysis Project [26]. The complete experimental codes and results can be found at [27].

Datasets. Table II summarizes the undirected graphs used in our experiments. Among them, ca-GrQc and ca-HepPh are author collaboration networks in different domains; email-Enron is an email communication network from Enron; com-LiveJournal is an online blogging and gaming network. All of these datasets are downloaded from SNAP [28].

TABLE II
Real-world network datasets

Dataset	# of Nodes	# of Edges	Description
ca-GrQc	5242	14,496	Collaboration network
ca-HepPh	12,008	118,521	Collaboration network
email-Enron	36,692	183,831	Email communication network
com-LiveJournal	3,997,962	34,681,189	Online social network

Competitive method. We compare with a state-of-the-art graph reduction algorithm, which is a grouping-based method UDS (Utility-Driven Graph Summarization) proposed by Kumar et al. [8]. It has achieved the best experimental results so far. Although Kumar et al. [8] have introduced a memorization technique as a scalable approach to UDS, the performance issues such as huge storage cost and high time complexity still cannot be improved much. It takes about 10^6 seconds to process a dataset containing 1 billion edges on a server with 16 vCPU, 64GB memory and 300GB SSD storage.



(a) ca-GrQc (b) ca-HepPh

Fig. 4. Performances of $steps$

Iteration steps. For CRR, the number of iterations affects its reduction quality and operating efficiency. In order to determine the values of $steps$ in the subsequent experiments, a preliminary experiment was first conducted. According to the nature of the iterative process, we set $steps$ to be $[x \times P]$, where x is a variable that can be controlled. The graph reduction quality is measured by the average delta between the reduced graph G' and the initial graph G (Average delta=$\frac{\Delta}{|V'|}$), with a lower average delta indicating a higher graph reduction quality, and the operating efficiency is measured by the running time. For the two smaller datasets ca-GrQc and ca-HepPh, the experimental results are shown in Figure 4, where the red curve represents the graph reduction quality and the green histograms represent the running time. According to the charts, on these two datasets, the graph reduction quality of CRR has improved significantly when $x > 4$, and tends to be flat when $x > 10$. In addition, for ca-GrQc, the rising trend of time slowed down when $x > 10$. For ca-HepPh, the time increasing is relatively stable. Based on the above experiments, we observe that x should be set to greater than 4, and there is no need to be a very large number. Therefore, $steps$ is set to $[10 \times P]$ in the subsequent experiments.

Parameter Settings. For UDS, the vertex importance $nodeIS$ and edge importance $edgeIS$ are set as the betweenness centrality. Also, the utility threshold $\tau_U = p$. For all methods, $p \in [0.1, 0.9]$, with a step size 0.1.

Evaluation Tasks. We evaluate our techniques using five common characteristics of graphs and two popular graph analysis applications, compared against the baseline method. For each application, utility is defined to show the usefulness with respect to the initial graph. A higher utility means a better graph reduction quality.

(1) Vertex degree corresponds to the percentage of vertices with a certain degree value. CRR and BM2 are edge shedding methods based on vertex degrees, which are closely related to this property. *(2) Shortest-path distance* refers to the percentage of reachable vertex pairs with different shortest path distances among all reachable vertex pairs. This property is crucial for any graph analysis tasks involving shortest path calculation. *(3) Betweenness centrality* represents node's importance in the network. It is related to the number of shortest paths through it in the graph. *(4) Clustering coefficient* is used to measure how close neighbors of the average k-degree vertex to form a clique. In particular, clustering coefficient is an important property in social networks. *(5) Hop-plot* is the percentage of all reachable vertex pairs in the network under the restriction of a certain distance k. *(6) Top-k Query*: Top-k or Top-$t\%$ Query is one of the most common applications. By running PageRank algorithm to rank the vertices, the top k vertices are selected. Given the value of t, we can compute $k = |V| \times t\%$ for the initial graph and $k = |V'| \times t\%$ for the reduced graph. If we run PageRank on both graphs G and G', we let $V_{t\%}$ be the set of top k nodes in G, and $V'_{t\%}$ be the set of top k nodes in G' based on PageRank values. Therefore, the utility of Top-k query is defined as:

$$\text{Utility of Top-}k\text{ Query}=\frac{|V_{t\%} \cap V'_{t\%}|}{k}$$

For UDS, we adopt its own processing method of supernodes to get Top-k utility. *(7) Link prediction within community* (abbreviated as link prediction) is another tested application. It predicts whether a given pair of vertices belongs to the same community. In our experiments, it is performed on all 2-hop vertex pairs in G and G' respectively. Suppose L is the prediction result for G and L_s is the result for G', the utility of link prediction is defined as follows:

$$\text{Utility of Link Prediction}=\frac{|L_s \cap L|}{L}$$

B. Experimental Results

The experiments include the graph reduction and graph analysis tasks on the first 3 datasets, ca-GrQc, ca-HepPh and email-Enron. Due to the huge cost of UDS, on the com-LiveJournal dataset, we only perform graph reduction using CRR and BM2, and perform the Top-k queries.

TABLE III
GRAPH REDUCTION TIME (SEC)

p	ca-GrQc			ca-HepPh			email-Enron			com-LiveJournal		
	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	15.212	14.861	0.257	268.972	275.619	1.084	6879.807	1885.879	2.645		3626.847	357.569
0.8	15.207	14.925	0.250	271.939	294.612	1.250	55965.467	1829.306	2.537		3312.434	397.061
0.7	15.426	14.965	0.258	302.764	315.623	1.423	160353.568	1956.268	2.889		3223.822	365.409
0.6	16.599	14.789	0.279	890.657	318.554	1.543	231575.210	1956.607	3.101		2631.308	459.907
0.5	19.217	14.934	0.257	3054.891	292.064	1.642	296826.905	1822.863	3.674		2426.456	293.872
0.4	27.516	14.510	0.297	5269.170	307.329	1.758	422570.755	1873.002	3.639		2043.653	323.950
0.3	66.699	14.510	0.283	8057.549	300.614	1.880	497718.257	1832.783	3.837		1776.021	314.252
0.2	179.200	13.895	0.359	11284.950	274.653	2.039	562725.773	1819.197	4.058		1419.634	343.632
0.1	365.766	13.246	0.349	15773.001	241.629	2.399	604679.461	1857.304	4.161		1096.614	330.989

TABLE IV
TOTAL PROCESSING TIME ON CA-GRQC I (SEC)

	Link prediction			SP distance			Betweenness centrality			Hop-plot		
T	321.68			74.182			110.466			141.429		
p	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	326.706	316.573	201.951	71.094	80.760	61.623	76.372	92.765	76.167	124.388	117.438	135.114
0.5	130.072	106.264	71.391	38.245	54.798	24.530	40.779	65.520	32.388	47.340	81.070	50.492
0.1	385.975	23.507	5.552	366.462	13.575	0.522	366.588	14.851	1.014	367.006	14.326	0.836

Running time. The total running time includes the graph reduction time on the initial graph and the graph analysis time on the reduced graph. Table III shows the graph reduction time of three methods with different values of p. It can be seen that when the size of the datasets grows exponentially, the graph reduction time of BM2 is almost unchanged, and CRR can achieve nearly linear growth. The time cost of UDS is so large that it cannot finish graph reduction on com-LiveJournal dataset within 10 times of CRR's reduction time; so we have to give up on com-LiveJournal with UDS. By contrast, BM2 can complete the corresponding graph reduction tasks within 500s. The above results fully demonstrate the superiority of CRR and BM2 in their edge shedding abilities, indicating that these two techniques can achieve fast graph reduction under resource constraints.

Next, let us examine the whole processing time (graph reduction time plus graph analysis time on reduced graphs) for different graph analysis tasks. We only show the results on ca-GrQc with p = 0.9, 0.5, and 0.1 for clarity. Results on other datasets with other p values show similar trends, and can be found in [27].

TABLE V
TOTAL PROCESSING TIME ON CA-GRQC II (SEC)

	Top-k			Vertex degree			Clustering coefficient		
T	1.016			0.075			0.202		
p	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	16.165	15.822	1.127	15.312	14.907	0.290	15.415	15.049	0.435
0.5	19.693	15.763	0.956	19.293	14.955	0.276	19.550	15.046	0.345
0.1	365.873	13.483	0.526	365.801	13.252	0.353	365.805	13.271	0.365

The results are displayed in Tables IV-V, where the "T" lines show the processing time on initial graphs. For Table V, since the time complexity of these three graph analysis tasks, Top-k query, Vertex degree and Clustering coefficient, is low, and the size of ca-GrQc is small, the whole processing time using edge shedding methods does not present significant advantages compared to performing the graph analysis tasks directly on the initial graph. However, we can still find CRR and BM2 surpass UDS a lot especially when we need a small compression ratio. Moreover, in practical scenarios, the reduced graph can be reused after being generated and the time-saving is more. As a conclusion, CRR and BM2 can further reduce the processing time for graph analysis tasks with low time complexities.

Table IV shows the results for the remaining four graph analysis tasks with relatively high time complexities. Again, we can see that CRR and BM2 perform much better than UDS. Moreover, CRR and BM2 exhibit great efficiency compared to performing graph analysis tasks directly on the original graphs, especially when the compression ratio is small.

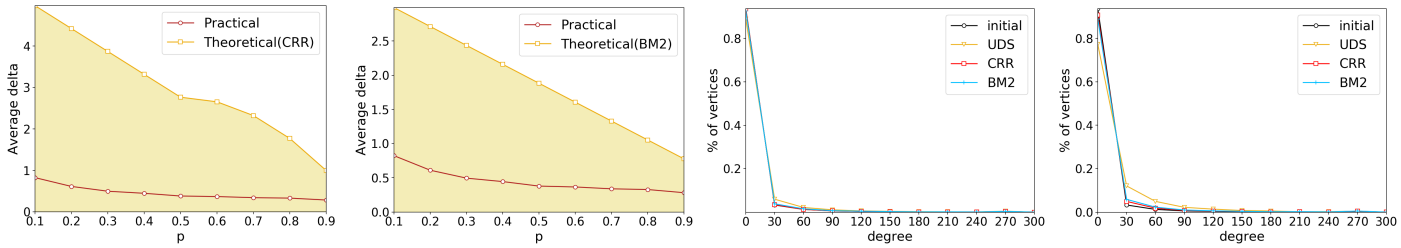
Next, we briefly show the graph analysis time of all graph analysis tasks on reduced graphs in Tables VI-VII, using the email-Enron dataset as an illustration. The "T" lines represent processing time on the initial graph. The three graph reduction methods can directly reduce the evaluation time of graph analysis tasks in most cases, but the performances of the three methods are not consistent on different graph analysis tasks since it mainly depends on the size of the reduced graph.

In general, both CRR and BM2 can greatly improve the time performance compared to UDS, especially when the dataset is large, and under resource constraints.

Graph reduction quality. Next we focus on the graph reduc-

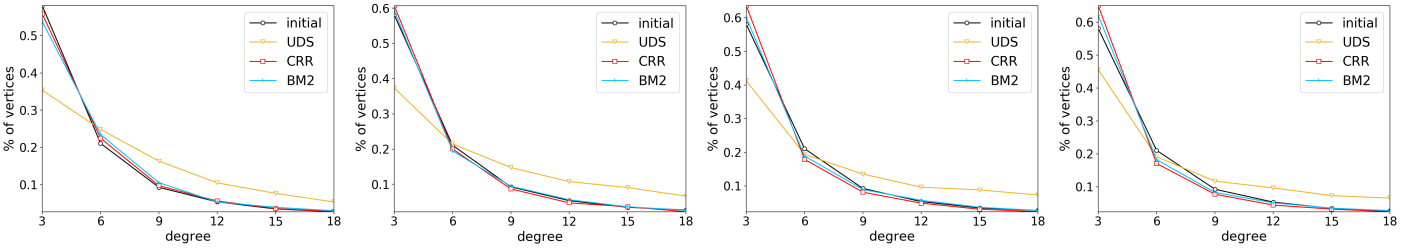
TABLE VI
GRAPH ANALYSIS TIME ON REDUCED GRAPHS ON EMAIL-ENRON I (SEC)

	Link prediction			SP distance			Betweenness centrality			Hop-plot		
T	3302.558			13716.537			20290.111			16568.052		
p	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	2643.240	3533.481	2804.757	11828.763	14585.398	11652.593	13299.943	22405.572	14688.822	12015.097	16560.067	12049.356
0.5	682.331	2657.078	1688.226	1212.559	7811.894	3158.780	1030.596	11397.758	3816.374	1300.628	10465.631	3729.803
0.1	122.399	801.788	499.911	34.253	637.284	196.764	36.333	890.703	274.340	33.591	1003.209	344.473



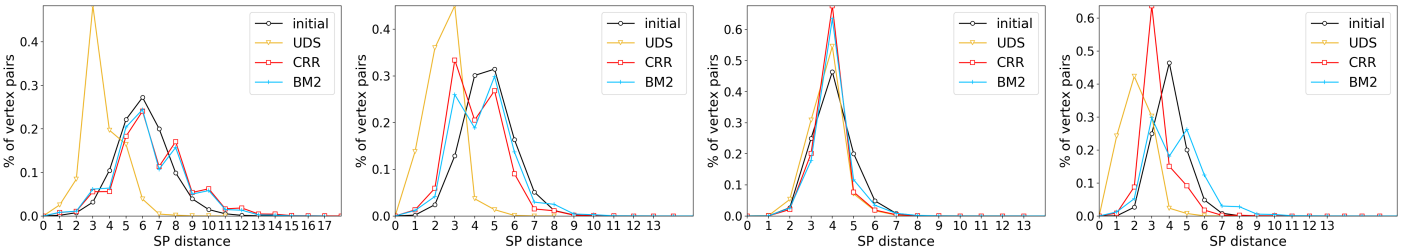
(a) Error bounds of CRR on caGrQc (b) Error bounds of BM2 on CaGrQc (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 5. Vertex degree



(a) email-Enron with p=0.7 (b) email-Enron with p=0.6 (c) email-Enron with p=0.5 (d) email-Enron with p=0.4

Fig. 6. Vertex degree (zoom-in)



(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 7. Shortest-path distance

tion quality of CRR and BM2 with respect to UDS on different graph analysis tasks. Since the three methods all perform well when p is large, we display the results mainly with small p for clarity.

We first examine the error bounds of our methods. We show the result on caGrQc in Figures 5(a)-5(b). It can be seen that although the error bounds we give are not tight, CRR and BM2 both perform very well and has very small errors (no more than 1 for all values of p). That is, our methods are effective and can generate the reduced graph with little error.

(1) Vertex degree. Figures 5(c)- 5(d) shows the vertex degree distributions on email-Enron. Since this dataset has a wide degree range, the vertex degrees larger than 300 are aggregated as 300. From Figures 5(c) and 5(d), we can see that the results from CRR and BM2 are very close to the initial vertex degree distributions, while UDS deviates more than CRR and BM2. To better illustrate the result, we zoom-in the most probable vertex degrees (1 to 18) on it. The result is shown in Figure 6. We can see that curves of CRR and BM2 fit the original graph precisely, which shows the high accuracy of our reduced graphs.

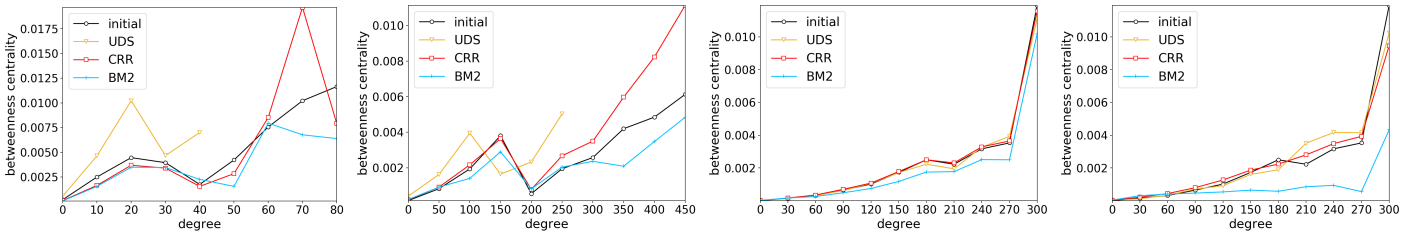
(2) Shortest-path distance. Figure 7 illustrates the shortest-path distance distributions. For all datasets, when p is larger, the distributions of the three techniques are nearly the same as the initial graph. When $p = 0.3$, the results of CRR and BM2 still conform to the trend of the curve, but UDS deviates significantly from the original curve. It can be seen that CRR and BM2 maintain the shortest-path distance much better, while UDS basically has lost this feature when p is small.

(3) Betweenness centrality. Figure 8 shows the betweenness centrality versus the vertex degree. We can see that CRR and BM2 measure the betweenness centrality of vertices with lower degrees very accurately in all cases, but the measure of vertices with higher degrees is relatively unstable. But on the whole, the performance of both is significantly better than UDS, due to the nature of supernode aggregation in UDS.

(4) Clustering coefficient. Figure 9 illustrates the clustering

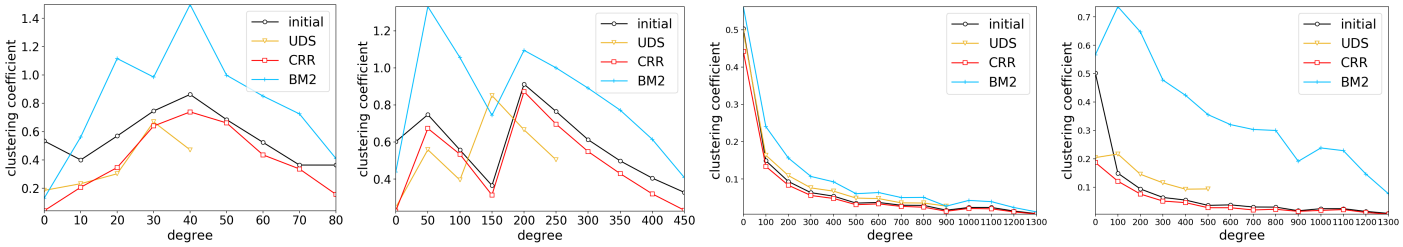
TABLE VII
GRAPH ANALYSIS TIME ON REDUCED GRAPHS ON EMAIL-ENRON *II* (SEC)

	Top-k			Vertex degree			Clustering coefficient		
T	10.407			1.152			10.489		
p	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	10.216	8.299	7.146	1.171	0.655	0.547	15.012	14.749	8.206
0.5	4.402	5.406	4.818	0.857	0.373	0.279	7.507	4.635	2.999
0.1	3.211	1.501	1.318	0.865	0.088	0.063	2.350	0.574	0.416



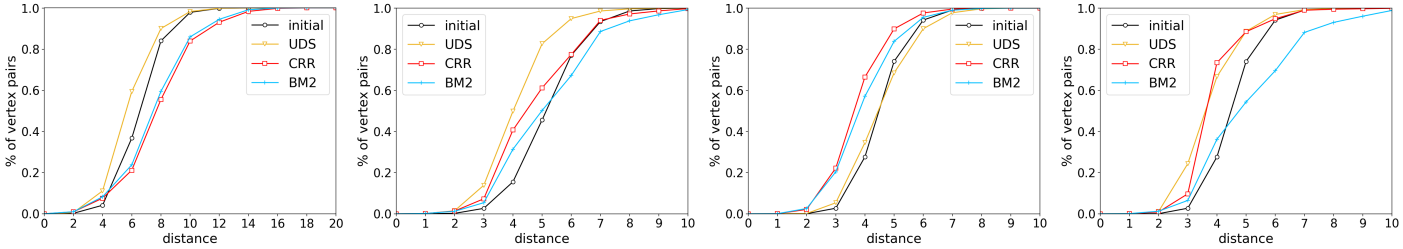
(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 8. Betweenness centrality



(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 9. Clustering coefficient



(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 10. Hop-plot

coefficient versus the vertex degree. The results are consistent with the results of shortest-path distance. When p is larger, CRR and BM2 are accurate in estimating the original graph. When p is smaller, CRR performs the best on the ca-GrQc and email-Enron datasets while BM2 performs the best on the ca-HepPh dataset.

(5) Hop-plot. Figure 10 illustrates the hop-plot distribution. On all datasets, the estimations of the three techniques for the initial graph are slightly different in different regions, but they all perform well on the whole.

(6) Top-k Query. Tables VIII-IX show the utility results of our experiments where we compare CRR, BM2, and UDS with respect to top-k queries, and t is set to 10. From the table we can see that the performance of CRR is excellent as it can reach almost 60% utility when p is reduced to 0.3 on all datasets. BM2's performance is also good, ranking only second to CRR. When p is 0.1, the utility of UDS is below 0.2, which means it has lost a lot of information. It is worth mentioning that, on large dataset such as com-LiveJournal, CRR and BM2 both perform very well with utilities greater than 75% even though p is only 0.1, demonstrating the proposed methods' power again for large datasets.

TABLE VIII
UTILITY OF TOP-10% I

p	ca-GrQc			ca-HepPh		
	UDS	CRR	BM2	UDS	CRR	BM2
0.9	0.876	0.966	0.935	0.947	0.978	0.943
0.8	0.735	0.937	0.908	0.927	0.963	0.917
0.7	0.611	0.916	0.870	0.867	0.941	0.887
0.6	0.571	0.863	0.828	0.609	0.917	0.851
0.5	0.498	0.809	0.702	0.419	0.865	0.756
0.4	0.443	0.731	0.693	0.320	0.838	0.733
0.3	0.370	0.681	0.586	0.230	0.772	0.684
0.2	0.269	0.500	0.460	0.151	0.685	0.604
0.1	0.174	0.313	0.254	0.092	0.514	0.439

TABLE IX
UTILITY OF TOP-10% II

p	email-Enron			com-LiveJournal		
	UDS	CRR	BM2	UDS	CRR	BM2
0.9	0.775	0.966	0.885		0.963	0.984
0.8	0.537	0.939	0.798		0.900	0.986
0.7	0.357	0.898	0.750		0.856	0.976
0.6	0.283	0.859	0.696		0.823	0.957
0.5	0.226	0.812	0.595		0.797	0.938
0.4	0.180	0.761	0.572		0.776	0.913
0.3	0.141	0.698	0.543		0.725	0.870
0.2	0.105	0.586	0.454		0.642	0.850
0.1	0.075	0.394	0.292		0.787	0.893

(7) Link prediction. Table X shows the utility results of link prediction. In order to reduce the influence of link prediction methods, we select Node2vec [29] to generate models using

graph embedding, and then use K-means to classify nodes on the models. The classification result is the basis of link prediction. Among them, we set the parameter p to 1, q to 1 of Node2vec, $n\_clusters$ to 5 of K-means. The results show that the performances of link prediction are different on various datasets. For ca-GrQc, all three techniques perform similarly. But for ca-HepPh and email-Enron, the utility of UDS drops very fast, which is much worse than CRR and BM2.

TABLE X
UTILITY OF LINK PREDICTION

p	ca-GrQc			ca-HepPh			email-Enron		
	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2
0.9	0.772	0.748	0.797	0.865	0.865	0.897	0.748	0.888	0.888
0.8	0.701	0.732	0.75	0.898	0.853	0.845	0.566	0.872	0.778
0.7	0.7	0.664	0.682	0.805	0.824	0.828	0.556	0.838	0.664
0.6	0.631	0.626	0.659	0.665	0.807	0.772	0.494	0.816	0.6
0.5	0.617	0.634	0.597	0.516	0.755	0.717	0.46	0.784	0.602
0.4	0.559	0.57	0.541	0.447	0.694	0.647	0.472	0.742	0.538
0.3	0.529	0.485	0.463	0.423	0.648	0.602	0.448	0.69	0.506
0.2	0.452	0.483	0.426	0.401	0.57	0.545	0.444	0.634	0.486
0.1	0.445	0.419	0.434	0.329	0.531	0.495	0.442	0.56	0.484

Summary. First, we aim to propose efficient graph reduction techniques under resource constraints. CRR and BM2 are running on a desktop with 16GB memory, while UDS is poorly adapted to large-scale datasets in this case. In terms of operating efficiency, it takes less than half of the time of UDS for CRR and BM2 to generate the reduced graph. Moreover, when the size of datasets increases exponentially, CRR and BM2 can maintain a linear increase in time, which is more feasible compared to UDS. In terms of the graph reduction quality, CRR and BM2 are much better than UDS in various graph analysis tasks. In general, BM2 is more efficient than CRR, while CRR shows a better graph reduction quality in most cases. So users could choose different methods according to their needs. In summary, the two methods proposed in this paper, CRR and BM2, have achieved significantly better performance than UDS in graph reduction, and meet the needs of processing large graphs under resource constraints.

VI. CONCLUSIONS

In this paper, we introduce two novel methods for graph reduction. Given the importance of vertex degree in capturing network characteristics, we aim at generating the reduced graph by preserving the expected degree distribution. The proposed methods have greatly reduced the graph reduction time compared to the state-of-the-art method UDS, and make processing large graph datasets under resource constraints become reality. For instance, CRR takes up to 25% of the graph reduction time of UDS on ca-GrQc and BM2 takes 1% at most. Meanwhile, a comprehensive experimental evaluations on real-world datasets confirm that CRR and BM2 indeed preserve well a number of important graph characteristics. In addition, with the continuous improvement of graph reduction techniques, users' various needs in different scenarios make autonomous control becoming the development trend. The controllability of CRR and BM2 for size reduction is one of the highlights of the proposed techniques.

REFERENCES

[1] J. Clement, "Number of facebook users worldwide 2008-2019," https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide, 2020.

[2] J. Clement, "Twitter: number of monthly active users 2010-2019," https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users, 2019.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[4] K. LeFevre and E. Terzi, "Grass: Graph structure summarization," in *SIAM*, 2010, pp. 454–465.

[5] M. Riondato, D. García-Soriano, and F. Bonchi, "Graph summarization with quality guarantees," *Data Min. Knowl. Discov.*, vol. 31, no. 2, pp. 314–349, 2017.

[6] A. Maccioni and D. J. Abadi, "Scalable pattern matching over compressed graphs via dedensification," in *SIGKDD*, 2016, p. 1755–1764.

[7] W. Fan, J. Li, X. Wang, and Y. Wu, "Query preserving graph compression," in *SIGMOD*, 2012, pp. 157–168.

[8] K. A. Kumar and P. Efstathopoulos, "Utility-driven graph summarization," *VLDB*, vol. 12, no. 4, pp. 335–347, 2019.

[9] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *SIGMOD*, 2008, p. 419–432.

[10] S. E. Ahnert, "Power graph compression reveals dominant relationships in genetic transcription networks," *Mol. BioSyst.*, vol. 9, pp. 2681–2685, 2013.

[11] Zeqian Shen, Kwan-Liu Ma, and T. Eliassi-Rad, "Visual analysis of large heterogeneous social networks by semantic and structural abstraction," *IEEE Trans Visual Comput Graphics*, vol. 12, no. 6, pp. 1427–1439, 2006.

[12] C. Li and S. Lin, "Egocentric information abstraction for heterogeneous social networks," in *ASONAM*, 2009, pp. 255–260.

[13] P. Hu and W. C. Lau, "A survey and taxonomy of graph sampling," *ArXiv*, vol. abs/1308.5865, 2013.

[14] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications," *CSUR*, vol. 51, pp. 1 – 34, 2018.

[15] N. Tang, Q. Chen, and P. Mitra, "Graph stream summarization: From big bang to big crunch," in *SIGMOD*, 2016, pp. 1481–1496.

[16] X. Gou, L. Zou, C. Zhao, and T. Yang, "Fast and accurate graph stream summarization," in *ICDE*, 2019, pp. 1118–1129.

[17] J. Zhang and Y. Tay, "Gscaler: Synthetically scaling a given graph." in *EDBT*, vol. 16, 2016, pp. 53–64.

[18] A. Musaafir, A. Uta, H. Dreuning, and A.-L. Varbanescu, "A sampling-based tool for scaling graph datasets," in *ICPE*, 2020, pp. 289–300.

[19] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," in *Proc IEEE INFOCOM*, vol. 2, 2002, pp. 638–647.

[20] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," *SIGCOMM*, vol. 36, no. 4, pp. 135–146, 2006.

[21] M. Mihail and N. K. Vishnoi, "On generating graphs with prescribed vertex degrees for complex network modeling," *Position Paper, Approx. and Randomized Algorithms for Communication Networks (ARACNE)*, vol. 142, 2002.

[22] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, "The pursuit of a good possible world: extracting representative instances of uncertain graphs," in *SIGMOD*, 2014, pp. 967–978.

[23] M. Barthelemy, "Betweenness centrality in large complex networks," *The European physical journal B*, vol. 38, no. 2, pp. 163–168, 2004.

[24] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[25] S. Hougardy, "Linear time approximation algorithms for degree constrained subgraph problems," in *Research Trends in Combinatorial Optimization*. Springer, 2009, pp. 185–200.

[26] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *TIST*, vol. 8, no. 1, p. 1, 2016.

[27] Y. Zeng, C. Song, and T. Ge, "Selective edge shedding in large graphs under resourceconstraints," https://github.com/ZYLpro/Selective-Edge-Shedding-in-Large-Graphs-Under-Resource-Constraints/.

[28] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, 2014.

[29] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *SIGKDD*, 2016.