Knowledge-Based Systems 240 (2022) 108126

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Reduction of large-scale graphs: Effective edge shedding at a controllable ratio under resource constraints

Yiling Zeng [a], Chunyao Song [a,*], Tingjian Ge [b], Ying Zhang [a]

[a] College of Computer Science, Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin, China
[b] University of Massachusetts Lowell, Lowell, USA

ARTICLE INFO

Article history:
Received 24 July 2021
Received in revised form 29 December 2021
Accepted 1 January 2022
Available online 7 January 2022

Keywords:
Degree distribution
Edge shedding
Graph reduction
Limited resources

ABSTRACT

As technology advances, many complicated systems can be represented by networks/graphs. However, when using limited computing resources such as portable computers or personal desktop computers, users are not able to store and mine large-scale graphs due to the unparalleled growth of the amount of data we generate. In order to address this challenge, we present effective edge shedding. Effective edge shedding can reduce the amount of data to be processed and the corresponding storage space while speeding up graph algorithms and queries, thereby supporting interactive analysis, helping knowledge discovery, and eliminating noise. In this paper, to extract the underlying features of a graph, we present two effective edge shedding methods on the basis of preserving the expected vertex degree. Both methods allow users to control the edge shedding process, thus generating a reduced graph of the predefined size based on the computing resource constraint. Using four real-world datasets in different domains, we performed an extensive experimental evaluation of our methods and compared them with the state-of-the-art graph summarization method on seven graph analysis tasks. The results indicate that our methods can achieve up to 58.6% higher accuracy on graph analysis tasks compared with the state-of-the-art method. For very large datasets, our methods consumes only 0.3% of the running time of the competitive method when generating the reduced graph. The above results fully illustrate the advantages of our methods.

1. Introduction

With the fast progress of the information age, there are multiple ways to store and represent data. Among these methods, graphs develop expressive data representation patterns for describing entities and their relationships. Massive graphs arise in many applications, such as social networks, academic networks, transportation networks, and many knowledge-based systems. There are many types of information hidden in complex networks. Through the in-depth exploration of these graphs, the potential relationships within the data can be found and then used effectively.

Nevertheless, the volume of data is expanding rapidly over time. Daily activities such as social media interactions, smartphone usage, web browsing, online shopping, and wellness smart bracelets will produce a large amount of valuable information. Therefore, the size of network models is constantly growing. As the world's largest social network platform, Facebook had approximately 2.89 billion active users per month by the second quarter of 2021 [1]. As of the first quarter of 2019, Twitter, a social platform called an "Internet SMS", had nearly 330 million users and generated more than 10 million "tweets" every day [2].

Although we can use networks to represent and record the data produced in daily human life, the computing feasibility and efficiency of large-scale network analysis and mining is incomparable to the volume of data with explosive growth, especially when mining using the personal computers of everyday users. For example, graph mining tasks such as link prediction and node clustering cannot be executed on the desktop or laptop computer of an ordinary user, such as a scientist, due to the sheer sizes of the graphs. Hence, the use of graph reduction techniques, e.g., edge shedding, to decrease the network storage and analysis costs can achieve substantial practical benefits.

Applying graph reduction results in the following four main advantages. First, graph reduction greatly saves storage space by reducing the amount of data that needs to be processed for later analysis. Second, the use of reduced graphs accelerates the processing speed of graph queries. Third, for data visualization, graph reduction provides it with more feasibility. Finally, in real datasets, there are usually many hidden or incorrect links and labels. Through graph reduction, the noise in the network can be well cleaned, and the underlying pattern features can also be mined out.

* Corresponding author.
E-mail address: chunyao.song@nankai.edu.cn (C. Song).

https://doi.org/10.1016/j.knosys.2022.108126
0950-7051/

In 2020, COVID-19 swept across the world. Owing to the impact of COVID-19, many people began to work from home, and the demand for online platforms increased dramatically. Companies and institutions are struggling to meet the increased demand for storage space, which reveals that in the big data era, storage capacity and data processing capacity are of great importance. For example, regarding the current research results, most graph reduction techniques require high hardware resources, memory, and CPUs. 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 performing large-scale data analysis on a regular basis, cannot afford the cost of acquiring or maintaining high-end data servers. This leads to the failure of large-scale data being effectively utilized. Moreover, if scientists can obtain preliminary results on reduced graphs using their personal computer very quickly, they can then decide if it is necessary to send the data to data centers with powerful servers for more precise analysis. Our experiments demonstrate that for large datasets, we can consume only 0.3% of the original graph analysis time to obtain an accurate preliminary result. Moreover, the demand, and the market for edge computing [3] are rapidly growing. As preliminary data processing in edge computing is pushed to less powerful devices, graph reduction techniques will be much needed. To the best of our knowledge, efficient graph reduction methods under resource constraints have not been proposed before.

1.1. Related work

Graph reduction techniques can be divided into several popular categories, such as grouping- or aggregation-based techniques, bit compression-based techniques, 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 graph reduction method that greedily grouped nodes to minimize 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 graphs. Graph Dedensification [6] is an edge-grouping-based method that compresses neighborhoods around high-degree nodes, thereby 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 an input graph. Navlakha et al. [9] introduced a highly compact two-part representation, which allows for both lossless and lossy graph compression with bounds on the introduced error. Ahnert [10] proposed a framework for the discovery of dominant relationship patterns in transcription networks by compressing a network into a power graph with overlapping power nodes. Lee et al. [11] introduced SSumM, a scalable and effective graph summarization algorithm. The method not only merges nodes together but also sparsifies the summary graph, and the two strategies are carefully balanced based on the minimum description length principle.

Simplification-based reduction methods generate reduced graphs by removing less "important" nodes or edges from original graphs. Shen et al. [12] presented a visual analytics tool, OntoVis, which allows users to perform structural abstraction and importance filtering to make large networks manageable. Li et al. [13] designed several abstraction criteria to distill representative and important information to construct abstracted graphs for visualization. However, these methods mainly study heterogeneous social networks. By modeling the complex semantics of heterogeneous social networks, they generate a condensed feature graph representation for egocentric information abstraction. Therefore, these methods are not suitable for homogeneous graphs.

Papers [14,15] are surveys of graph reduction. Most of the above techniques are constrained by minimizing the reconstruction error or ensuring the utility of the results. However, these reduction methods often need to use extremely expensive hardware resources and are not suitable for small companies, some universities and laboratories with limited resources, as we mentioned above. Furthermore, the performance could be arbitrarily poor.

From another perspective, in different scenarios, users prefer different properties. Therefore, the key issue of graph reduction is what properties of the original graph should be preserved in the reduced graph. The properties that have been studied include aggregate functions [16], degree distributions [17,18], community [19,20] and PageRank values [21]. Among the previous studies, [17,18] adopted different graph reduction methods, such as random walks; and finally retained the degree distribution well. Different from their goals, we aim to generate a reduced graph that retains multiple properties by preserving the distribution of vertex degrees.

Inspired by [22], we propose two graph selective edge shedding techniques to efficiently solve the graph reduction problem. In a short paper published in ICDE 2021 [23], we briefly outlined some preliminary high-level ideas of our graph reduction work. In contrast, we present the complete work by including detailed graph reduction algorithms and running examples and error analysis in this work. We also provide some theoretical analysis of the error bound for each algorithm and show the comprehensive evaluation results. Experimental evaluation and comparisons with the SOTA algorithm illustrate the advantages of our methods for highly efficient graph reduction and their higher utility with much more accurate data analysis results.

In addition, TCM [24] and GSS [25] are novel graph stream summarization techniques concentrating on graphs with multiple edges that appear at different timestamps. Learning automata-based algorithms have also been utilized in network sampling [26]. Graph scaling techniques such as [27,28], as well as the abovementioned techniques, focus on different problems from ours.

1.2. Our contributions

In this work, 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.

The key contributions of this paper are as follows:

(1) We propose two effective methods for graph reduction using edge shedding.

(2) As a basic topological characteristic of the network, the vertex degree plays an important role in network analysis. In order to achieve the best graph preservation effect, we constrain the edge shedding process by maintaining the vertex degree distribution, which can preserve the basic network topology and is therefore beneficial to the subsequent network mining tasks. In addition to the graph reduction methods themselves, different estimation techniques are given for different graph analysis tasks to

Table 1
List of symbols used in the paper.

Symbol	Definition
G	Initial graph
G'	Reduced graph, which is a subset of G
$\mathbb{G}$	Perfect 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'

estimate the results of the original graph. Through the estimation process, we can greatly reduce the costs of graph analysis.

(3) At present, graph reduction research field pays increasingly more attention to the controllability of the reduced graph size. Consequently, on the basis of the needs of users in different scenes, we use the edge preservation ratio p, where $p \in (0, 1)$ denotes the size ratio between the reduced and original graphs, to generate different sized graphs.

(4) We perform extensive experiments on four datasets in different areas, including seven graph analysis tasks. The results prove that the methods proposed in this work can run under resource constraints and achieve extremely high quality in terms of efficiency and accuracy, implying that our methods are highly competitive with other methods in terms of efficiency and accuracy.

The remainder of the paper is organized as follows. In Section 2, we state the problem and introduce the relevant background knowledge. In Sections 3 and 4, we present the proposed CRR and BM2 algorithms and illustrate them with detailed running examples. Section 5 contains comprehensive experiments that evaluate the efficiency and accuracy of the proposed algorithms on real-world datasets in four different domains. At last, we conclude the paper in Section 6.

2. Preliminaries

2.1. Problem definition

The core concept of our graph reduction methods is to capture and maintain the underlying structure of the original graph. Using the reduced graph, we can estimate various properties of the original graph and thus accelerate the downstream graph analysis tasks. How to efficiently capture and maintain the underlying structure of the original graph is the key requirement of graph reduction techniques. As one of the most fundamental characteristics in network topology, the vertex degree is of great significance in network analysis [29]. Current research has identified the importance of the vertex degree in capturing network features, including the communication network topology [30] and complex network modeling [31].

In light of the above conclusions, we come to the following idea: by ensuring that the expected vertex degree of each node fits the core idea of preserving the overall vertex degree, we can obtain the nature of the original graph and accurately approximate other features afterward.

In Table 1, we summarize the important notations that we use in our discussions and the remainder of the paper.

Assume 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 $G' = (V', E')$, where V' is the node set, $E' \subseteq V' \times V'$ is the edge set, and G' is a subgraph of G, be the reduced graph produced by our algorithms. Since the methods proposed in this paper are based on vertex degrees, it is necessary to define the symbolic representation of the vertex degrees of a graph. For the original graph G, $\forall u \in V$, $deg_G(u)$ denotes the degree of a single node u. For a reduced graph G', $\forall u \in V'$, $deg_{G'}(u)$ represents the actual degree of a single node u in G'. Ideally, the algorithm should generate a perfect graph $\mathbb{G}$ with the same degree distribution as G. However, in fact, the reduced graph G' can only be as close as possible to the standard of the perfect graph $\mathbb{G}$ but cannot meet the requirements exactly. Thus, we define the expected degree of a single node u in G' as $E(deg_{G'}(u))$, which corresponds to $deg_{\mathbb{G}}(u)$, the ideal vertex degree of u in the reduced graph. In the same way, the average expected degree of the reduced graph G' should be $E(deg_{G'})$, which corresponds to $deg_{\mathbb{G}}$, the average degree of $\mathbb{G}$.

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

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

According to Eq. (1), the average expected degree of the reduced graph can be expressed as:

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

Let $dis(u)$ denote the difference between the actual and expected degrees of vertex u in the reduced graph. It can be represented as:

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

The sum of the degree discrepancies of all nodes Δ in the reduced graph is:

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

As described above, the evaluation of edge shedding methods based on vertex degrees is in line with the degree difference Δ between the reduced graph and the original graph. The smaller the degree difference Δ is, the closer the vertex degree distributions between the reduced and perfect graphs.

Therefore, we give a formal definition of the problem. Given an original graph G, we need to obtain a representative reduced graph G', where $G' = \arg\min_{G^* \sqsubseteq \mathcal{G}} \Delta(G^*)$ and $\mathcal{G}$ is the collection of all possible reduced graphs of G. Based on the conclusion of [22], the problem is NP-hard; thus, we intend to propose effective approximate solutions. 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.

2.2. Centrality

To preserve the key topological connectivity, we need to introduce the notion of centrality. Centrality is a common concept used in social network analysis to express the importance of vertices or edges in graphs. Among the different centrality computing methods, the *betweenness centrality* is the one most frequently employed in network analysis [32].

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

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

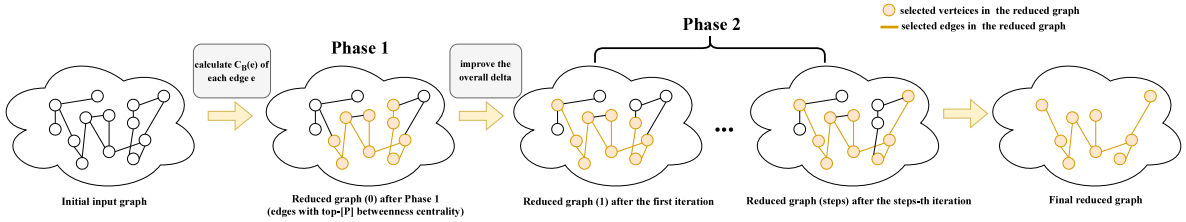


Fig. 1. The flow chart diagram of CRR.

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 paths passing through node v other than s,t.

To measure the centrality of *edges*, the 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 the symbols share similar meanings as $C_B(v)$.

The betweenness centrality focuses on the role of an edge serving as a "bridge". Brandes [33] proposed a fast algorithm to compute the betweenness centrality that requires $O(|V| + |E|)$ space and runs in $O(|V||E|)$ time on unweighted networks.

3. Centrality Ranking with Rewiring (CRR)

In order to preserve the degree distribution of the original graph, we can start by maintaining the expected average vertex degree. The proposed CRR algorithm has two phases, which are shown in Fig. 1. First, according to the importance of all edges, the CRR algorithm generates an initial reduced graph with the same expected average degree computed by fixing the edge-preservation ratio p following Eq. (2). Since we only consider the importance of edges at the beginning, the initial result may violate the goal of minimizing the degree differences to some extent. Therefore, in the second phase, we replace some edges to reduce the node-degree discrepancy.

Algorithm 1 presents the pseudocode of CRR.

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 $P \leftarrow p \cdot |E|$
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'

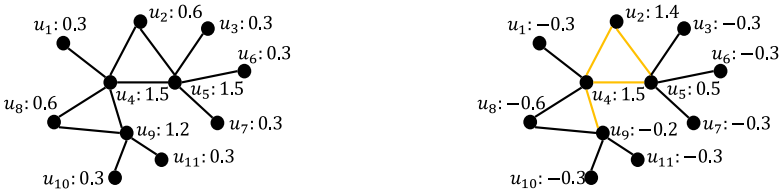
The first stage (Lines 1–6) implements the edge shedding strategy. According to the discussion in Section 2.1, the expected average degree of the reduced graph $E(deg_{G'})$ equals p times the average degree of the original graph. That is, if the reduced graph involves $p \cdot |E| = P$ edges, then the needed expected average degree is met. Therefore, the reduced graph should include $[P]$ (the nearest integer of P) edges. Since each edge makes different contributions in maintaining graph properties, intuitively, we should retain the edges whose contributions are higher, and the remaining edges are subject to edge shedding. In this way, the underlying structure of the graph can be preserved. As mentioned in Section 2.2, the betweenness centrality focuses on the role of an edge in 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] proved that the betweenness centrality is superior when estimating edge importance. Hence, in the first stage, CRR begins by computing the betweenness centrality of all edges (Line 3) and continuously picks $[P]$ edges with the highest betweenness centrality to form the initial reduced graph (Lines 4–6).

Fig. 2(a) shows an original graph, which includes 11 nodes and 12 edges, to be reduced. We set the edge preservation ratio p to 0.3, and the expected degrees are shown next to each node. First, CRR computes $[P] = [p \cdot |E|] = [0.3 \cdot 12] = 4$. Then, it selects the top-4 edges as the initial edge set based on the betweenness centrality. The result of Phase 1 is shown in Fig. 2(b), where yellow edges represent the initial reduced graph and the number next to each node indicates the degree difference $dis(u)$.

The second stage (Lines 7–13) is responsible for edge rewiring based on the initial results. In the first stage, taking $[P]$ edges with the highest centrality does not guarantee that the degree of each node is close to its expected value. Additionally, taking the 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 the second stage, CRR conducts several iterations of edge substitution to better decrease 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 parameter *steps*, which is the number of iterations, is determined by users' preference for efficiency versus accuracy. Based on the results of extensive experiments, it is commonly recommended that users set *steps* as $[10 \cdot P]$ (the nearest integer of $10 \cdot P$). At each iteration, E' is used to denote the current edge set. CRR randomly selects two edges $e_1 = (u, v) \in E'$ and $e_2 = (x, y) \in E \backslash E'$ (Lines 8–9). Thus, $d_1 = |dis(u) - 1| + |dis(v) - 1| - (|dis(u)| + |dis(v)|)$ denotes the change in the overall degree difference caused by the removal of e_1, and $d_2 = |dis(x) + 1| + |dis(y) + 1| - (|dis(x)| + |dis(y)|)$ denotes the change in the overall degree difference caused by the addition of e_2 (Lines 10–11). If $d_1 + d_2 < 0$, the proceeding edge replacement will lead to a decrease in the overall degree difference, so we should adopt this change. Otherwise, we do nothing (Lines 12–13). The final edge set is returned as the reduced graph (Line 14).

Fig. 3 is an example iteration of CRR, where the number next to each node represents the current degree difference and the yellow edges are the current set of selected edges. According to Fig. 3, at the ith iteration, CRR randomly chooses $e_1 = (u_4, u_9) \in E'$, $e_2 = (u_3, u_5) \in E \backslash E'$ to attempt edge replacement. Since $d_1 = |1.5 - 1| + |-0.2 - 1| - (|1.5| + |-0.2|) = 0$, $d_2 = |-0.3 + 1| + |0.5 + 1| - (|-0.3| + |0.5|) = 1.4$, and $d_1 + d_2 >$



(a) The original graph with p=0.3 (b) The reduced graph after Phase 1

Fig. 2. CRR Phase 1 illustration.

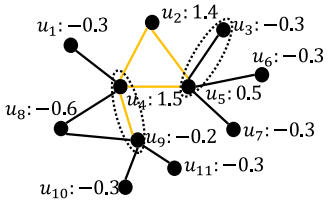


Fig. 3. An illustration of the ith iteration of CRR.

0, this edge replacement would not improve the overall degree difference, so we do not perform edge swapping at this time. After this iteration, E' remains $\{(u_2, u_4), (u_2, u_5), (u_4, u_5), (u_4, u_9)\}$.

Example 1 (*Running Example of CRR*)**.** Fig. 4(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 $[P] = [p \cdot |E|] = [0.4 \cdot 11] = 4$, 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, and the importance of unmarked edges is 0.182 in Fig. 4(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 Fig. 4(b), where the number next to each node indicates the degree difference dis_u, and the yellow edges form the initially selected edge set.

Next, CRR starts the second phase. First, we compute $steps = [10 \cdot P] = 44$, which is the number of iterations. At the ith iteration, CRR randomly chooses $e_1 = (u_5, u_7)$ and $e_2 = (u_8, u_{10})$ to attempt edge replacement, which is shown in Fig. 4(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 Fig. 4(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* $(0, 4p(1-p)\frac{|E|}{|V|})$.

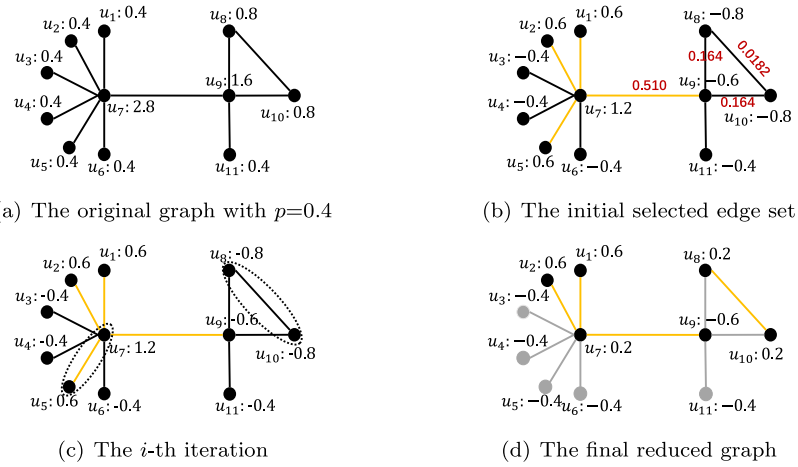
Proof. We first prove the claim that an upper bound of Δ (i.e., the total absolute difference between the node degrees in the reduced graph and their expectations) is achieved when a subset U of nodes all have degree 0 and the remaining nodes $V \setminus U$ all retain their original degrees (except 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 retain the node degree statistics but can arbitrarily rewire the edges while retaining 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 toward 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 occur: (1) All nodes in $V \setminus U$ have full degrees, and U may have some internal edges; or (2) All nodes in U have degree 0. In case (1), within the node set U, we can keep moving all remaining internal edges toward 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 is greater than the expectation. Therefore, in the end, we will reach the scenario described in the claim, i.e., a rewired graph G' with a subset of nodes having 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 $V \setminus U'$ have all the $p|E|$ edges at the end of the proof constructed above. Based on the result of the claim, we have $\Delta \leq \sum_{u \in U'} deg_G(u) \cdot p + \sum_{u \in V \setminus U'} deg_G(u) \cdot (1-p) = p \cdot (1-p) \cdot 2|E| + (1-p) \cdot 2p|E| = 4p(1-p)|E|$, and we obtain the result for the range of the average absolute difference as in the theorem. □

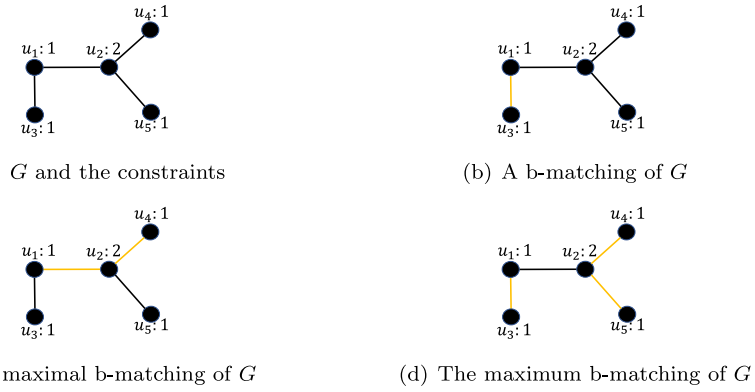
For CRR, we analyze the time complexity as follows. The first stage is the generation of the initial edge set, which mainly includes calculating and sorting the betweenness centralities of each edge. The time complexity of these two parts is $O(|V||E|)$ and $O(|E|log_2|E|)$, respectively. In addition, the initial edge selection has a cost of $O(|E|)$. The second phase is a linear process of edge replacement. Therefore, the overall time complexity of CRR is $O(|V||E| + |E|log_2|E| + |E| + steps)$, which is simplified to $O(|V||E| + |E|log_2|E| + steps)$. As for the space complexity, CRR requires $O(|V| + |E|)$ space when computing the betweenness centralities of all edges [33]. In addition, the spaces required for the graph construction and storage of node degree differences throughout the algorithm are $O(|V|+|E|)$ and $O(|V|)$, respectively. Therefore, the overall space complexity of CRR is $O(|V| + |E|)$.



(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. 4. An illustration of running CRR.



(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. 5. A B-matching example.

4. B-Matching with Bipartite Matching (BM2)

In this section, we first introduce the *maximum b-matching* problem [34], which is another type of edge shedding problem. Next, we introduce the BM2 algorithm based on the association between the maximum b-matching problem and the graph reduction problem.

4.1. B-matching

Let us consider an undirected graph $G = (V, E)$ and a set of capacity constraints $b(u) : V \to \mathbb{N}$. For subgraph $S = (V, E_S)$ of G, if the degree of any vertex $u \in V$ in S is at most $b(u)$, then S 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. Fig. 5 shows a b-matching instance, where the number next to each node indicates its capacity constraint, and the matched edges are shown in yellow.

In [22], Parchas et al. analyzed the relationship between the maximum b-matching problem and the problem of summarizing representative samples from uncertain graphs. It has been demonstrated that the solution of the maximum b-matching problem can be well applied in our situation only if we treat $E(deg_{G'}(u))$ as $b(u)$ in a b-matching problem.

4.2. B-Matching with Bipartite Matching (BM2)

The flow-chart of BM2 is shown in Fig. 6. Since the edge preservation ratio p is in $(0, 1)$, $E(deg_{G'}(u))$ is possibly a fraction.

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

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

calculate the expected degree $E(deg_{G'}(i))$ for all vertices in V
initialize $E_m \leftarrow 0$, $deg_{G'}(i) \leftarrow 0$
$b_i \leftarrow round(E(deg_{G'}(i)))$ for each vertex i
for *each* $(u, v) \in E$ **do**
  if $deg_{G'}(u) < b_u$ *and* $deg_{G'}(v) < b_v$ **then**
    $E_m \leftarrow E_m \cup \{e\}$
    $deg_{G'}(u) \leftarrow deg_{G'}(u) + 1$, $deg_{G'}(v) \leftarrow deg_{G'}(v) + 1$
$A \leftarrow \emptyset, B \leftarrow \emptyset, C \leftarrow \emptyset$
for *each* $u \in V$ **do**
  $dis(u) = deg_{G'}(u) - E(deg_{G'}(u))$
  if $dis(u) \leq -0.5$ **then**
    $A \leftarrow A \cup \{u\}$
  else if $-0.5 < dis(u) < 0$ **then**
    $B \leftarrow B \cup \{u\}$
  else
    $C \leftarrow C \cup \{u\}$
$E^* \leftarrow E - E_m$
for *each* $e = (u, v) \in E^*$ **do**
  gain $= |dis(u)| + 2\,|dis(v)| - |1 + dis(u)| - 1$
  if $u \in A$ *and* $v \in B$ *and* gain ≥ 0 **then**
    $w(e) \leftarrow$ gain
  else
    discard e from E^*
let G^* be $((A \cup B), E^*, W)$ where W is the set $\{w(e)\}$
E_{BP} = bipartite (G^*)
$E' = E_m \cup E_{BP}$

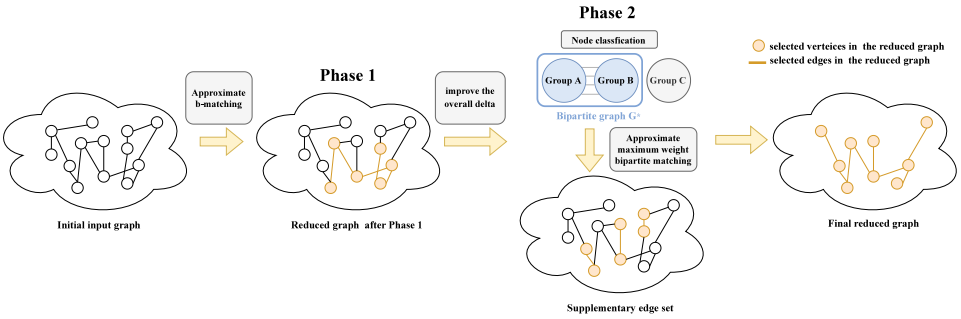
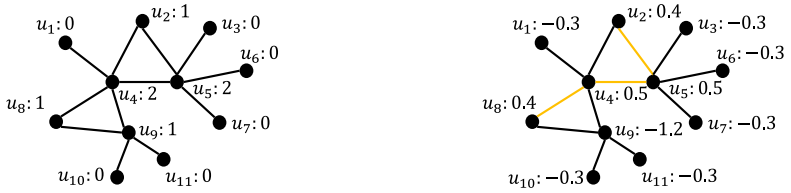


Fig. 6. The flow-chart diagram of BM2.



(a) The rounded graph (b) The reduced graph after Phase 1

Fig. 7. BM2 Phase 1 illustration.

Thus, the b-matching model described in Section 4.1 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, BM2 performs b-matching on a transformed graph to obtain an initial edge set. Next, BM2 picks additional edges that can improve the total degree difference Δ by performing bipartite matching.

Algorithm 2 presents the pseudocode of BM2.

The initially selected edge set is generated in the first stage (Lines 1–7) of BM2. First, we compute the expected vertex degree of each node for the reduced graph using Eq. (1) (Line 1). In order to solve our problem with real-value constraints, BM2 treats the rounded integer of the expected vertex degree as the capacity constraint of each node and runs approximate b-matching to obtain a maximum b-matching (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.

Fig. 7 shows the procedure, where the capacity constraints $b(u_i)$ are shown next to u_i. Fig. 7(a) is the transformed graph of Fig. 2(a) with the same compression ratio $p = 0.3$. The maximum b-matching of Fig. 7(a) is shown in Fig. 7(b), where the number next to each node indicates the current degree difference compared to the expected vertex degree (before rounding), and the yellow edges represent the initial selected edge set.

In Phase 2 (Lines 8–25), a supplementary edge set is selected to further reduce the overall degree difference. The rounding strategy in Phase 1 leads to some differences between the actual and expected degrees of the reduced graph. This requires correcting the deviation of the initial selected edge set to obtain the closest vertex degree distribution. BM2 first classifies nodes into three groups A, B, and C according to each node's degree such that $\forall u \in A, dis(u) \leq -0.5$, $\forall u \in B, -0.5 < dis(u) < 0$, and $\forall u \in C, dis(u) \geq 0$ (Lines 8–16). If an edge connecting vertex u is added to the reduced graph G', the real-time vertex degree of u in G' will increase by 1. According to Eq. (3), its degree difference $dis(u)$ will increase by 1 with a fixed expected degree. Due to the different initial values of the degree differences of the three groups, the change in the absolute degree differences for vertices of different groups varies after adding a connected edge. For a node in group A, adding a connected edge to the reduced graph will decrease its absolute degree difference. For a node in group B, adding a connected edge to the reduced graph will increase its absolute degree difference by less than 1. For a node u in group C, its absolute degree difference will increase by 1 since it has already reached or exceeded its expected degree. Specifically, $deg_{G'}(u)$ is directly increased by 1, and $dis(u)$ is also increased by 1. Since $dis(u)$ is positive before and after the increase, the absolute value of its change is 1.

Next, let us discuss different situations for all possible edges. Altogether, there are nine possible combinations where the two endpoints of an edge may belong to any of the three groups. Let us start by listing the first six combinations.

(1) $e = (u, v) \in E$, $u \in B$ and $v \in B$
(2) $e = (u, v) \in E$, $u \in A$ and $v \in C$
(3) $e = (u, v) \in E$, $u \in C$ and $v \in A$
(4) $e = (u, v) \in E$, $u \in B$ and $v \in C$
(5) $e = (u, v) \in E$, $u \in C$ and $v \in B$
(6) $e = (u, v) \in E$, $u \in C$ and $v \in C$

In the above 6 cases, the addition of an edge will definitely lead to an increase in the overall degree difference. Except for these, edges $e = (u, v) \in E$, where $u \in A$ and $v \in A$ have all been added to E_m, so we do not need to discuss them here. Therefore, we only focus on 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. □

In Phase 2, we select all edges connecting nodes between groups A and B with positive gains calculated by Lemma 1. We then generate a weighted bipartite graph G^*, whose edges are those selected items (Lines 17–23). Then, approximate maximum weight bipartite matching is performed on G^*, which can pick the edges minimizing the global degree difference (Lines 24–25). The ultimate selected edge set of BM2 consists of E_m and E_{BP} (Line 26).

Algorithm 3 shows the corresponding *bipartite* algorithm (Line 25) of Algorithm 2.

Algorithm 3: bipartite

```
   Input: bipartite graph G* = (A ∪ B, E*, W)
   Output: edge set E_BP ⊆ E*
1  initialize E_BP ← ∅
2  sort edges e ∈ E* in non-increasing order of their weights w(e)
     and add them into a priority queue Q
3  while Q ≠ ∅ do
4      e = (a, b) ← Q.next()
5      E_BP ← E_BP ∪ {e}
6      discard all edges in Q incident to b
7      dis(a) ← dis(a) + 1
8      if −1 < dis(a) ≤ −0.5 then
9          for each e' = (a, x) ∈ E* do
10             w(e') ← |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) ∈ E* do
17             discard edge e' from Q
```

First, edges are sorted in nonincreasing order of their weights and then added to a priority queue Q (Line 2). In 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 dynamically changes. The algorithm needs to maintain the correctness of vertex classification and the order of Q. The discussion is as follows.

Taking the head $e = (a, b)$ of Q and adding it to E_{BP}, 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; thus, 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 $dis(a) \le -2$. The change in $dis(a)$ does not influence the gains of the edges adjacent to a.*

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

According to Lemma 2, if $dis(a) \le -2$, the gains of the edges adjacent to a do not change, and we do nothing. If $-2 < dis(a) \le -1.5$, vertex a still belongs to group A, and we 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 $dis(a) > -1.5$ and $dis(a) + 1 > -0.5$, the new degree difference of vertex a does not meet the requirement of group A; thus, a and the edges adjacent to a are removed from the bipartite graph (Lines 15–17).

Let us use Fig. 8 as an example to illustrate how Algorithm 3 works. First, according to Fig. 7(b), we obtain the vertex classification: $A = \{u_9\}, B = \{u_1, u_3, u_6, u_7, u_{10}, u_{11}\}, C = \{u_2, u_4, u_5, u_8\}$. Then, Algorithm 3 selects the edges whose gains are positive to form a bipartite graph, as shown in Fig. 8(a). In the first iteration, $e = (u_9, u_{10})$ is added to E_{BP}, and then the degree differences are updated. u_9 no longer belongs to Group A; therefore, we delete (u_9, u_{11}). The result is shown in Fig. 8(b). The priority queue Q is empty, and Algorithm 3 ends.

Example 2 (*Running Example of BM2*)**.** Fig. 4(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 Fig. 9(a). Fig. 9(b) shows the maximum b-matching of the transformed graph, where the number next to each node indicates the degree difference between the currently 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 Fig. 9(b), the vertices are classified as $A = \{u_7, u_9\}, B = \{u_1, u_2, u_3, u_4, u_5, u_6, u_{11}\}, C = \{u_8, u_{10}\}$. Fig. 9(c) shows the weighted bipartite graph, including the degree differences and the gains of each edge.

In the first iteration, the *bipartite* algorithm 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 Fig. 9(d). In the second iteration, edge $e = (u_7, u_2)$ is selected to be added to E_{BP}. Additionally, the bipartite algorithm 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 Fig. 9(e), it can be selected or discarded according to users' preferences. 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 Fig. 9(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 Phase 1 of BM2, we first perform the rounding operation, which would cause a 0.5 degree difference for each node at most. Then, we give a linear time approximation algorithm for the Cardinality b-Matching Problem [34], and the Δ after Phase 1 should be at most $\sum_{u \in V} \frac{1}{2} + \frac{1}{2} \sum_{u \in V} |-deg_G(u) \cdot p| = \frac{1}{2}|V| + \frac{1}{2} \cdot p \sum_{u \in V} deg_G(u) = \frac{1}{2}|V| + \frac{1}{2}p \cdot 2|E| = \frac{1}{2}|V| + p|E|$.

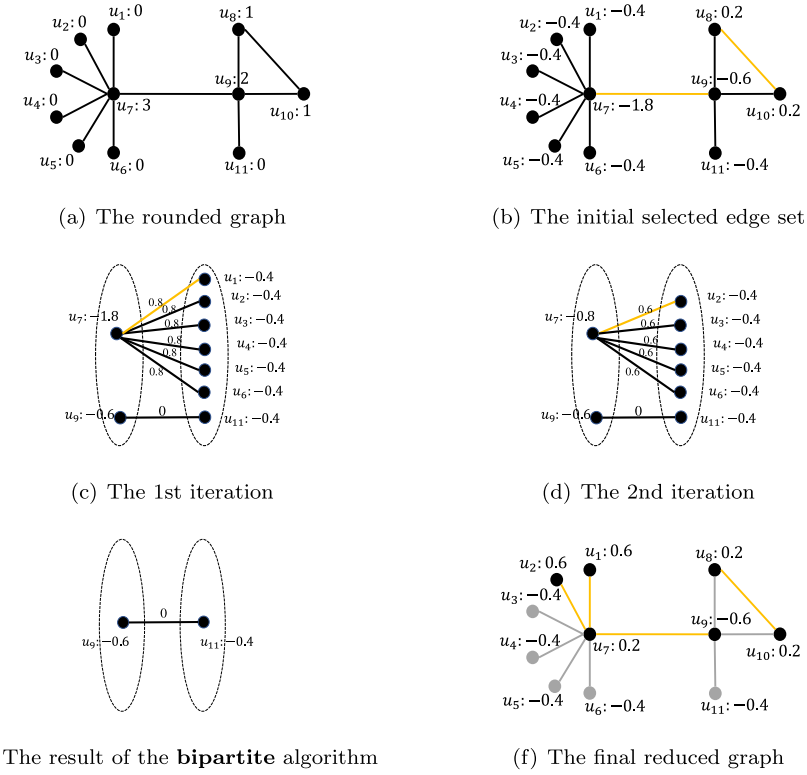
Considering the BM2 process, it starts to add edges from 0 and will not cause edge overload except for the 0.5 degree rounding. Therefore, the maximum degree difference before Phase 1 starts is $\sum_{u \in V} |-deg_G(u) \cdot p| = 2p|E|$ for BM2. We define $\frac{\frac{1}{2}|V| + p|E|}{2p|E|} = \frac{|V|}{4p|E|} + \frac{1}{2}$ to indicate the optimization ratio of the 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 $p \le 0.5$.

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

The first stage of BM2 includes the linear-time processing of nodes and edges. In the second stage, 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

(a) Input of the **bipartite** algorithm (b) The priority Q after the 1st iteration

Fig. 8. BM2 Phase 2 illustration.



(a) The rounded graph (b) The initial selected edge set

(c) The 1st iteration (d) The 2nd iteration

(e) The result of the **bipartite** algorithm (f) The final reduced graph

Fig. 9. An illustration of running BM2.

bipartite edge sorting and selection in which each edge of E^* can be processed at most $|B|$ times. Therefore, 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 $x{\cdot}P$ and $P = p{\cdot}|E|$. 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 5.2. Next, we analyze the space complexity of BM2. In the first stage, BM2 needs to construct the initial graph and maintain the degree differences and capacity constraints for each node, which requires $O(|V|+|E|+2|V|)$ space. In the second stage, BM2 uses a priority queue to store the candidate vertices and edges and thus requires at most $O(|A|+|B|+|E|)$ space. In summary, the upper bound on the space complexity of BM2 is $O(|V|+|E|)$.

5. Experimental evaluation

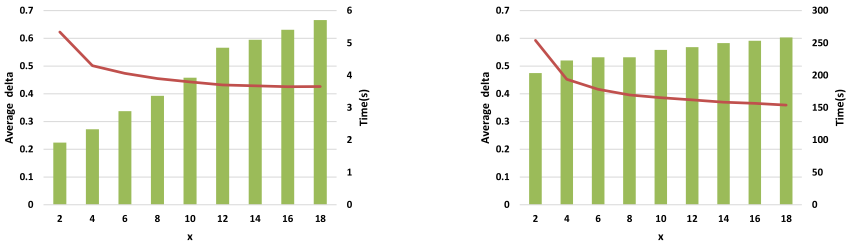
5.1. Experimental settings

Setup. We implement all our algorithms and the competitive method in Python. All the experiments were performed on a machine with an Intel Core i7 3.40 GHz processor and 16 GB memory. We create graphs using the snap library provided by the Stanford Network Analysis Project [35]. All our codes and experimental results are provided in GitHub [36].

Table 2
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

Datasets. We use four real-world datasets for the experiments. The detailed information is shown in Table 2. Among the datasets, ca-GrQc and ca-HepPh are author collaboration networks in different domains, email-Enron is an email communication network from Enron, and com-LiveJournal is an online blogging and gaming network. All of these datasets were downloaded from SNAP [37].



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

Fig. 10. Performances of *steps*.

Competitive method. We compare our proposed methods with the SOTA work, UDS (Utility-Driven Graph Summarization) proposed by Kumar et al. [8]. UDS provides a novel iterative solution for grouping-based greedy graph summarization and has achieved the best experimental results thus far. To overcome the scalability challenge, Kumar et al. [8] introduced a memorization technique as a scalable approach to UDS. However, the performance of UDS in terms of storage costs and time complexity is still poor, and there is great potential for improvement. On a server with 16 vCPUs, 64 GB memory, and 300 GB SSD storage, it takes approximately 10^6 seconds to process a dataset with 1 billion edges.

Iteration steps. For CRR, the number of iterations will affect its reduction quality and operating efficiency. In order to determine the values of *steps* in subsequent experiments, a preliminary experiment is first conducted. According to the nature of the iterative process, we set *steps* to be $[x \cdot P]$, where x is a controllable variable. The graph reduction quality is measured by the average delta between the reduced graph G' and the initial graph G (average delta=$\frac{\Delta}{|V'|}$). A lower average delta indicates a higher graph reduction quality. In addition, 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 Fig. 10, 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 significantly improved 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 increase is relatively stable. Based on the above experiments, we observe that x should be set to greater than 4, and there is no need for x to be a very large number. Therefore, *steps* is set to $[10 \cdot P]$ in the subsequent experiments.

Parameter Settings. For UDS, the vertex importance *nodeIS* and edge importance *edgeIS* are set as the betweenness centrality. Additionally, the utility threshold $\tau_U = p$. For all methods, $p \in [0.1, 0.9]$, with a step size of 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, the utility is defined to show the usefulness with respect to the initial graph. A higher utility means a better graph reduction quality.

- *Vertex degree* refers to the percentage of nodes with a certain degree value. Our proposed methods, CRR and BM2, are closely related to this property since they aim to preserve vertex degrees.
- *Shortest-path distance* is the percentage of node pairs at a certain distance over all pairs of reachable vertices. For any graph analysis task involving shortest path computation, this property is decisive.
- *Betweenness centrality* represents the importance of the nodes in the network. It corresponds to the ratio of the shortest paths that pass through the node over all pairs of shortest paths.
- *Clustering coefficient* is used to measure how close neighbors of the average k-degree vertex form a clique. In particular, the clustering coefficient is a crucial property in social networks.
- *Hop-plot* represents the percentage of the number of reachable node pairs at a selected distance or less over all reachable pairs.
- *Top-k Query*: The Top-k or Top-t% Query is one of the most common applications. The Top-k query ranks the nodes using the PageRank algorithm and selects the top k nodes in descending order given the ranking. Given the value of t, we can compute $k = |V| \cdot t\%$ for the initial graph and $k = |V'| \cdot t\%$ for the reduced graph. When running PageRank on graphs G and G', we let $V_{t\%}$ be the set of the top k nodes in G and $V'_{t\%}$ be the set of the top k nodes in G' based on the PageRank values. Hence, the utility of the Top-k Query is expressed as follows:
  Utility of Top-k Query $= \frac{|V_{t\%} \cap V'_{t\%}|}{k}$
  For UDS, we adopt its own processing method of supernodes to obtain the Top-k utility.
- *Link prediction*: Another application is known as Link prediction within the community. Link prediction detects whether a given node pair is part of the same community. In our experiments, Link prediction is performed on all 2-hop vertex pairs in G and G'. Suppose L is the prediction result for G and L_s is the result for G'. The link prediction utility is expressed as follows:
  Utility of Link Prediction $= \frac{|L_s \cap L|}{L}$

5.2. Experimental results

The experiments are divided into two parts: the time efficiency of graph reduction and the graph reduction quality. We implement all the experiments on the first 3 datasets: ca-GrQc, ca-HepPh, and email-Enron. It is worth noting that UDS cannot complete graph reduction using 10 times more than our proposed methods' reduction time. Due to the huge costs of UDS, on the com-LiveJournal dataset, we only perform graph reduction using CRR and BM2 and perform the Top-k queries.

Running time. The total running time includes the graph reduction time on the initial graph and the graph analysis time on the reduced graph. The graph reduction times of all datasets are shown in Table 3, where the value of p varies from 0.9 to 0.1. We can observe a significant difference in the performances of

Table 3
Graph reduction time (s).

p	ca-GrQc			ca-HepPh			email-Enron			com-LiveJournal	
	UDS	CRR	BM2	UDS	CRR	BM2	UDS	CRR	BM2	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 4
Total processing time on ca-GrQc *I* (s).

	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

Table 5
Total processing time on ca-GrQc *II* (s).

	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 three methods as p decreases gradually. BM2 can complete graph reduction in almost constant time. The performance of CRR is slightly worse, but the increase in the time is nearly linear. In contrast, the time costs of UDS are too high such that UDS cannot complete the graph reduction task on the com-LiveJournal dataset running even ten times more than CRR's reduction time; therefore, we have to abandon running UDS on com-LiveJournal. There are two reasons for the rapidly growing graph reduction time of UDS. First, UDS needs to calculate the betweenness centrality of both nodes and edges. Second, as mentioned in [8], node merging and superedge decision-making are exhaustive in nature and perform redundant computations. Although a memorization technique is introduced, the high complexity of the algorithm still cannot be optimized much. In practical scenarios, when the large-scale email-Enron dataset is reduced significantly, the reduction time of CRR is only 0.3% (0.00307 = 1857.304/604679.461) of UDS while the reduction time of BM2 is even shorter and can be further reduced by 3 orders of magnitude. For the larger com-LiveJournal dataset, BM2 performs very well, taking only approximately 500 s. The above experimental results fully prove the advantage 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 evaluate the total 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. The experimental results of different p values on all datasets can be found in [36]. Since all the results show similar trends, they are omitted here.

The results are displayed in Tables 4–5, where the "T" lines show the processing times on the initial graphs. For Table 5, since the time complexities of these three graph analysis tasks, the Top-k query, Vertex degree, and Clustering coefficient, are low and the size of ca-GrQc is small, the entire processing time using edge shedding methods does not present significant advantages compared to performing the graph analysis tasks directly on the initial graph. However, the results still show that CRR and BM2 greatly surpass UDS, especially when we need a small compression ratio. Moreover, in practical scenarios, the reduced graph can be reused after being generated, and the time-savings are greater. In conclusion, CRR and BM2 can further reduce the processing time for graph analysis tasks with low time complexities.

Table 4 shows the results for the remaining four graph analysis tasks with relatively high time complexities. Again, the results show that CRR and BM2 perform much better than UDS. Moreover, CRR and BM2 exhibit great efficiency compared to performing the graph analysis tasks directly on the original graphs, especially when the compression ratio is small.

Next, we briefly show the graph analysis times of all graph analysis tasks on reduced graphs in Tables 6–7, using the email-Enron dataset as an illustration. The "T" lines represent the processing times 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 they mainly depend 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.

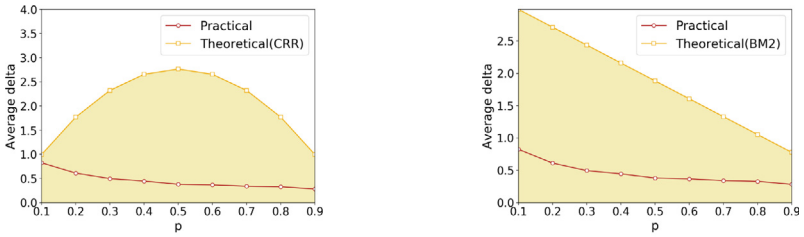
Graph reduction quality. In the following section, we concentrate on another evaluation part of the graph reduction methods, namely, the graph reduction quality on different graph analysis

Table 6
Graph analysis time on reduced graphs on email-Enron *I* (s).

	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

Table 7
Graph analysis time on reduced graphs on email-Enron *II* (s).

	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) CRR (b) BM2

Fig. 11. Error bounds on ca-GrOc.

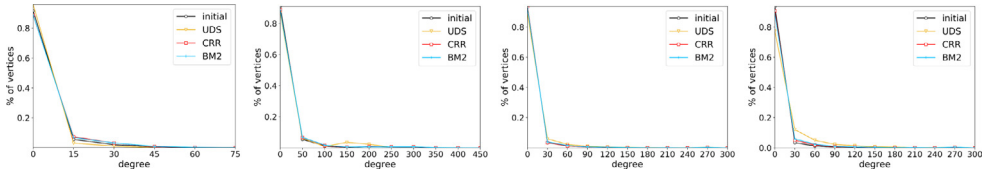


Fig. 12. Vertex degree.

tasks. Given that the performance of each of the three methods is great when p is large, we mainly show the experimental results when p is small for clarity.

First, we calculate the degree discrepancies of the proposed methods to verify the theoretical error bounds given in Sections 3 and 4. The results on caGrQc are displayed in Fig. 11. Although the error bounds defined in Sections 3 and 4 are not tight, it should be noted that both CRR and BM2 have completed the graph reduction well with small errors (no more than 1 for all values of p). In other words, our methods are feasible and effective in generating reduced graphs with low errors.

(1) Vertex degree. Fig. 12 illustrates the vertex degree distributions on different datasets. Since ca-GrOc has a wide degree range, vertex degrees larger than 300 are aggregated as 300. Fig. 12 shows that the vertex degree distribution curves of CRR and BM2 approach much more than the original graph, far better than UDS. To better illustrate the results, we zoom in on the most likely vertex degrees (1 to 18). The results are shown in Fig. 13. We can see that the CRR and BM2 curves fit the original graph precisely, which shows the high accuracy of our reduced graphs.

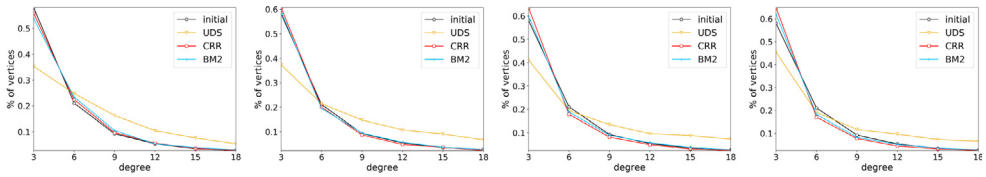
(2) Shortest-path distance. Fig. 14 shows the shortest-path distance distributions. For all datasets, the distributions of the three methods are almost the same as the initial graph when p is larger. When p is smaller, the results of CRR and BM2 still conform to the trend of the benchmark curve while UDS has an obvious deviation. As the results show, CRR and BM2 perform much better in the shortest path distribution while UDS loses most of the network connections.

(3) Betweenness centrality. Fig. 15 illustrates the betweenness centrality versus the vertex degree. It should be noted that on all datasets, CRR and BM2 are more accurate in measuring the betweenness centrality of nodes with lower degrees while the measure of nodes with higher degrees is relatively unstable. In general, the nature of the node aggregation operation in UDS makes it unable to accurately measure the attribute; thus, our algorithms have more advantages.

(4) Clustering coefficient. Fig. 16 shows the clustering coefficient versus the vertex degree. The outcomes are in line with those of the 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.

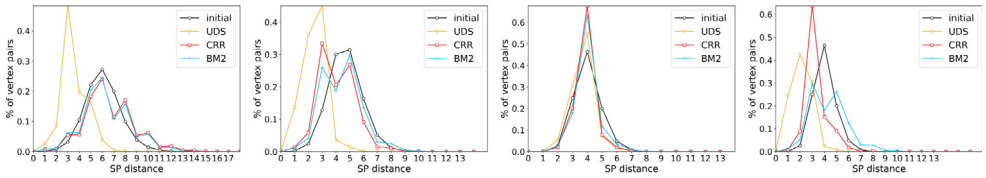
(5) Hop-plot. Fig. 17 shows the hop-plot distribution. The performances of all methods on the three datasets are similar, and they can restore the attributes of the original graph properly.

(6) Top-k Query. Tables 8–9 show the utility results of our experiments, in which we compare our methods and UDS with



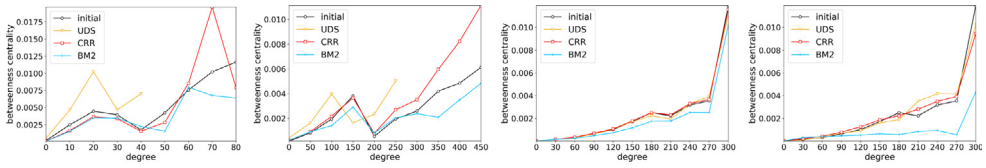
(a) email-Enron with p=0.7 (b) email-Enron with p=0.6 (c) email-Enron with p=0.5 (d) email-Enron with p=0.4

Fig. 13. Vertex degree (zoom-in).



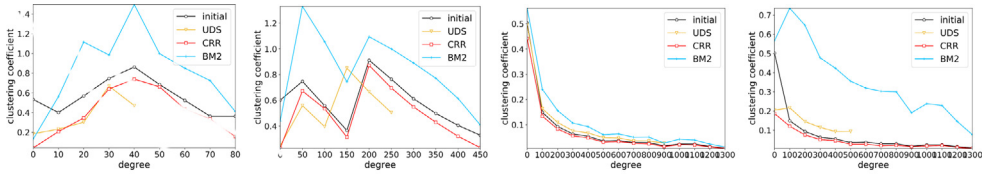
(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 14. Shortest-path distance.



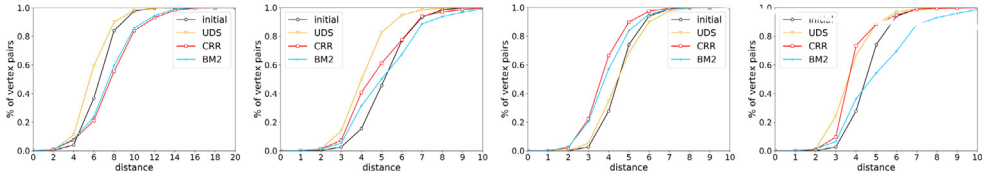
(a) ca-GrQc with p=0.3 (b) ca-HepPh with p=0.3 (c) email-Enron with p=0.8 (d) email-Enron with p=0.3

Fig. 15. 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. 16. 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. 17. Hop-plot.

respect to top-k queries, and set t to 10. The table shows that even if the graph has been greatly reduced, CRR can achieve at least 68% utility on all datasets when p is 0.3. The results fully demonstrate the advantages of CRR. BM2 performs great as well, ranking second only to CRR. When p is 0.1, the utility of UDS is below 0.2, which means that it is not able to retain effective information. It is worth mentioning that on large datasets such as com-LiveJournal, CRR and BM2 both perform very well with utilities greater than 75% even though p is only 0.1, again demonstrating the proposed methods' power again on large datasets. In contrast, due to the huge cost of UDS (UDS cannot complete the graph reduction task on com-LiveJournal dataset running even ten times more than CRR's reduction time), there are no accuracy results for UDS on the com-LiveJournal dataset.

(7) Link prediction. Table 10 presents the link prediction utilities on the first three datasets. In order to reduce the influence of link prediction methods, we select Node2vec [38] to generate models using graph embedding and then use K-means to classify the nodes on the models. The classification result is the basis of link prediction. Here, we set the parameter p to 1 and q to 1

Table 8
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 9
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

for Node2vec, and *n_clusters* to 5 for K-means. We can observe that for link prediction, each of the three methods has its own merits. On ca-GrQc, the utility differences of the three methods are not considerable, and they are all effective. However, for the remaining two datasets, UDS performs poorly. With a gradual decrease in p, the utility of UDS drops rapidly, and thus UDS cannot be compared with CRR and BM2.

Summary. First, we aim to propose effective graph reduction techniques under resource constraints. CRR and BM2 run on a machine with 16 GB of memory while UDS is poorly adapted to large-scale datasets in this case. In terms of time efficiency, CRR and BM2 take less than half of the time of UDS to finish the graph reduction. Moreover, when the size of datasets increases exponentially, CRR and BM2 can maintain a linear increase in time, demonstrating much stronger scalability than UDS. Regarding the graph reduction quality, our proposed methods are comparable with UDS or even better on different graph analysis tasks, again demonstrating their powerful graph reduction abilities. In general, BM2 is more efficient than CRR while CRR shows a better graph reduction quality in most cases. Therefore, users could choose different methods according to their needs. In summary, CRR and BM2, the methods proposed in this paper, have realized significantly better graph reduction performance than UDS, and satisfy the demand for processing large-scale graphs under resource constraints.

6. Conclusions

In this paper, we introduce two novel methods for graph reduction. Considering the key role of the vertex degree in network topology, we generate the reduced graph by retaining the expected vertex degree distribution, which can preserve the basic network topology and is therefore beneficial to the subsequent network mining tasks. Compared to the state-of-the-art UDS technique, our techniques have greatly reduced the graph reduction time, thus making it a reality to process large-scale graph datasets under resource constraints. For instance, CRR takes up to 25% and BM2 takes 1% at most of the graph reduction time of UDS on ca-GrQc . In terms of the reduction accuracy, a comprehensive experimental evaluation on real-world datasets confirms that CRR and BM2 indeed maintain plenty of vital graph features, which means that the generated reduction graphs can be effectively applied in downstream analysis tasks. In addition, with the continuous improvement of graph reduction techniques, users' various needs in different scenarios make autonomous control development trend. Another outstanding contribution of our methods is the controllability of the size of the generated reduced graph, which can meet the actual needs of users in different scenarios. Although this paper focuses on how to complete efficient graph reduction under resource constraints, we intend to subsequently implement and deploy this work in a parallelized environment to support a wider range of applications and achieve higher efficiency.

Table 10
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	0.750	0.898	0.853	0.845	0.566	0.872	0.778
0.7	0.700	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	0.600
0.5	0.617	0.634	0.597	0.516	0.755	0.717	0.460	0.784	0.602
0.4	0.559	0.570	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	0.690	0.506
0.2	0.452	0.483	0.426	0.401	0.570	0.545	0.444	0.634	0.486
0.1	0.445	0.419	0.434	0.329	0.531	0.495	0.442	0.560	0.484

CRediT authorship contribution statement

Yiling Zeng: Methodology, Software, Validation, Data curation, Formal analysis, Writing – original draft. **Chunyao Song:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Funding acquisition. **Tingjian Ge:** Writing – review & editing, Supervision. **Ying Zhang:** Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the NSFC, China under grants 62172237, 61772289, U1903128, U1936206, U1936105 and 62077031; the NSF, USA under grant IIS-1633271, IIS-2124704, OAC-2106740; and the New England Transportation Consortium project 20–2.

References

[1] S.R. Department, Facebook: number of monthly active users worldwide 2008–2021, 2021, https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide.
[2] S.R. Department, Twitter: number of monthly active users 2010–2019, 2019, https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[4] K. LeFevre, E. Terzi, Grass: Graph structure summarization, in: Proceedings Of The 10th SIAM International Conference On Data Mining, SDM 2010, 2010, pp. 454–465.
[5] M. Riondato, D. García-Soriano, F. Bonchi, Graph summarization with quality guarantees, Data Mining Knowl. Discov. 31 (2) (2017) 314–349.
[6] A. Maccioni, D.J. Abadi, Scalable pattern matching over compressed graphs via dedensification, in: Proceedings Of The ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, 2016, pp. 1755–1764, 13-17-August-2016.
[7] W. Fan, J. Li, X. Wang, Y. Wu, Query preserving graph compression, in: Proceedings Of The ACM SIGMOD International Conference On Management Of Data, 2012, pp. 157–168.
[8] K. Ashwin Kumar, P. Efstathopoulos, Utility-driven graph summarization, Proc. VLDB Endow. 12 (4) (2018) 335–347.
[9] S. Navlakha, R. Rastogi, N. Shrivastava, Graph summarization with bounded error, in: Proceedings Of The ACM SIGMOD International Conference On Management Of Data, 2008, pp. 419–431.
[10] S.E. Ahnert, Power graph compression reveals dominant relationships in genetic transcription networks, Mol. Biosyst. 9 (11) (2013) 2681–2685.
[11] K. Lee, H. Jo, J. Ko, S. Lim, K. Shin, Ssumm: Sparse summarization of massive graphs, in: Proceedings Of The ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, 2020, pp. 144–154.
[12] Z. Shen, K.L. Ma, T. Eliassi-Rad, Visual analysis of large heterogeneous social networks by semantic and structural abstraction, IEEE Trans. Vis. Comput. Graphics 12 (6) (2006) 1427–1439.
[13] C.T. Li, S.D. Lin, Egocentric information abstraction for heterogeneous social networks, 2009, pp. 255–260.
[14] P. Hu, W.C. Lau, A survey and taxonomy of graph sampling, 2013, pp. 1–34.
[15] Y. Liu, T. Safavi, A. Dighe, D. Koutra, Graph summarization methods and applications: A survey, ACM Comput. Surv. 51 (3) (2018) http://dx.doi.org/10.1145/3186727.
[16] J. Lu, H. Wang, Variance reduction in large graph sampling, Inf. Process. Manage. 50 (3) (2014) 476–491, http://dx.doi.org/10.1016/j.ipm.2014.02.003.
[17] M.P. Stumpf, C. Wiuf, R.M. May, Subnets of scale-free networks are not scale-free: Sampling properties of networks, Proc. Natl. Acad. Sci. 102 (12) (2005) 4221–4224, http://dx.doi.org/10.1073/pnas.0501179102.
[18] L. Zhang, H. Jiang, F. Wang, D. Feng, Draws: A dual random-walk based sampling method to efficiently estimate distributions of degree and clique size over social networks, Knowl.-Based Syst. 198 (2020) 105891, http://dx.doi.org/10.1016/j.knosys.2020.105891.
[19] S.H. Yoon, K.N. Kim, J. Hong, S.W. Kim, S. Park, A community-based sampling method using DPL for online social networks, Inform. Sci. 306 (2015) 53–69, http://dx.doi.org/10.1016/j.ins.2015.02.014.
[20] C. Tong, Y. Lian, J. Niu, Z. Xie, Y. Zhang, A novel green algorithm for sampling complex networks, J. Netw. Comput. Appl. 59 (2016) 55–62.
[21] A. Vattani, D. Chakrabarti, M. Gurevich, Preserving personalized pagerank in subgraphs, in: Proceedings Of The 28th International Conference On Machine Learning, ICML 2011, 2011, pp. 793–800.
[22] P. Parchas, F. Gullo, D. Papadias, F. Bonchi, The pursuit of a good possible world: Extracting representative instances of uncertain graphs, in: Proceedings Of The ACM SIGMOD International Conference On Management Of Data, 2014, pp. 967–978, http://dx.doi.org/10.1145/2588555.2593668.
[23] Y. Zeng, C. Song, T. Ge, Selective edge shedding in large graphs under resource constraints, Proc. Int. Conf. Data Eng. (2021) 2057–2062, 2021-April.
[24] N. Tang, Q. Chen, P. Mitra, Graph stream summarization: From big bang to big crunch, 2016, pp. 1481–1496, 26-June-20.
[25] X. Gou, L. Zou, C. Zhao, T. Yang, Fast and accurate graph stream summarization, Proc. Int. Conf. Data Eng. (2019) 1118–1129, 2019-April.
[26] A. Rezvanian, M.R. Meybodi, Sampling algorithms for stochastic graphs: A learning automata approach, Knowl.-Based Syst. 127 (2017) 126–144.
[27] J.W. Zhang, Y.C. Tay, GSCALER: SYnthetically scaling a given graph, Adv. Database Technol. - EDBT (i) (2016) 53–64, 2016-March.
[28] A. Musaafir, A. Uta, H. Dreuning, A.L. Varbanescu, A sampling-based tool for scaling graph datasets, in: ICPE 2020 - Proceedings Of The ACM/SPEC International Conference On Performance Engineering, 2020, pp. 289–300.
[29] T. Bu, D. Towsley, On distinguishing between internet power law topology generators, Proc. - IEEE INFOCOM 2 (c) (2002) 638–647.
[30] P. Mahadevan, D. Krioukov, K. Fall, A. Vahdat, Systematic Topology Analysis and Generation Using Degree Correlations, 2006, p. 135.
[31] M. Mihail, N.K. Vishnoi, On generating graphs with prescribed vertex degrees for complex network modeling, in: Position Paper, Approx. And Randomized Algorithms For Communication Networks (ARACNE), Vol. 142, 2002, p. 2865.
[32] M. Barthélemy, Betweenness centrality in large complex networks, Eur. Phys. J. B 38 (2) (2004) 163–168.
[33] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Soc. 25 (2) (2001) 163–177.
[34] S. Hougardy, Linear time approximation algorithms for degree constrained subgraph problems, in: Research Trends In Combinatorial Optimization: Bonn 2008, 2009, pp. 185–200.
[35] J. Leskovec, R. Sosič, SNAP: A General-purpose network analysis and graph-mining library, ACM Trans. Intell. Syst. Technol. 8 (1) (2016).
[36] Y. Zeng, C. Song, T. Ge, Selective edge shedding in large graphs under ResourceConstraints, 2020, https://github.com/ZYLpro/Selective-Edge-Shedding-in-Large-Graphs-Under-Resource-Constraints/.
[37] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, 2014, http://snap.stanford.edu/data.
[38] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings Of The ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, 2016, pp. 855–864, 13-17-Augu.