Journal of Computational Mathematics
Vol.xx, No.x, 2025, 1–33.

http://www.global-sci.org/jcm
doi:10.4208/jcm.2506-m2024-0192

A RESTARTED AND RANDOMIZED ALGORITHM FOR EVALUATING ENERGY OF EXTREMELY LARGE-SCALE GRAPH*

Gang Wu[1)]
School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China
Email: gangwu@cumt.edu.cn
Ke Li
School of Medical Information and Engineering, Xuzhou Medical University, Xuzhou 221004, China
Email: likeer1002@163.com
Jianjian Wang
School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China
Email: 1377152300@qq.com

Abstract

Graph energy is a spectrum-based graph invariant that has been studied extensively in network sciences. However, as far as we are aware, most of the existing works try to establish theoretical bounds, and there are few efficient algorithms for computing energy of extremely large-scale graph. To fill-in this gap, we first propose a randomized algorithm for evaluating energy of large-scale graphs, under the assumption that the adjacency matrix is approximately low (numerical) rank. However, the number of sampling used in this algorithm is difficult to determine in advance, and the graph energy is often underestimated. In order to improve the quality of the evaluation, we then propose a non-restarted randomized algorithm that updates the columns of the search basis incrementally. The error analysis and the convergence of the algorithm are established. However, the non-restarted algorithm may suffer from heavy overhead as the iteration proceeds. So as to release the overhead of the non-restarted algorithm, we finally propose a restarted randomized algorithm for evaluating energy of extremely large-scale graphs. The rationality of the restarted algorithm is given. Extensive numerical experiments are performed on extremely large-scale real-world and synthetic graphs, to show the effectiveness of our strategies and efficiency of the proposed algorithms.

Mathematics subject classification: 65F10, 65F15.
Key words: Graph energy, Extremely large-scale graph, Low (numerical) rank, Complex network analysis, Randomized singular value decomposition, Restarting.

1. Introduction

Let $\mathcal{G}$ be a graph with n vertices and m edges, then the graph energy $E(\mathcal{G})$ is defined as the sum of the absolute values of all the eigenvalues of the adjacency matrix of $\mathcal{G}$ [25, 26]. For undirected graph, graph energy is defined as the energy norm of the corresponding adjacency matrix, which is one of the most important unitary norms [32]. The concept of graph energy arose in the context of the study of conjugated hydrocarbons using a tight-binding method known in chemistry as the Hückel molecular orbital (HMO) method [9, 25]. Now, it appears in

* Received September 3, 2024 / Revised version received December 29, 2024 / Accepted June 23, 2025 / Published online September 25, 2025 /
1) Corresponding author

many applications such as graph theory [6,7,13,16,21,43], quantum chemistry [24,25,28,52], complex network analysis [19], scientific computation [24,59], and so on.

The mathematical theory of graph energy is nowadays relatively elaborated [27,45]. A large number of upper bounds [4,11,47,53] or lower bounds [38,39,55,69] have been established. To name a few, McClelland [52] established an upper bound on the energy of a general graph as follows:

$$E(\mathcal{G}) \leq \sqrt{2mn}. \tag{1.1}$$

Caporossi *et al.* [8] provided a lower bound based on the number of edges

$$E(\mathcal{G}) \geq 2\sqrt{m}. \tag{1.2}$$

Koolen and Moulton [41] presented an upper bound based on the number of vertices

$$E(\mathcal{G}) \leq \frac{n}{2}(1+\sqrt{n}). \tag{1.3}$$

Some upper and lower bounds on energy of special graphs, such as bipartite graphs [13,57, 72], regular graphs [5,29,37], cyclic graphs [36,62], random graphs [16], singular graphs [65], weighted graphs [21], and unitary Cayley graphs [35], were also investigated intensively. To name a few, for a simple, undirected, connected graphs without multiple edges or self-loops, Estrada and Benzi [19] proved the following upper bound on graph energy:

$$E(\mathcal{G}) \leq \frac{\lambda_1}{2}n + \frac{1}{\lambda_1}m,$$

where λ_1 is the maximum eigenvalue of the adjacency matrix of the graph $\mathcal{G}$. For a regular graph of size n with degree q, Balakrishnan [7] provided an upper bound as follows:

$$E(\mathcal{G}) \leq q + \sqrt{q(n-1)(n-q)}.$$

For the energy of bipartite graphs, Koolen [42] proved the following upper bound:

$$E(\mathcal{G}) \leq \frac{1}{\sqrt{8}}n(\sqrt{n}+\sqrt{2}).$$

Moreover, the relationships between graph energy and specific parameters were also studied, such as graph energy and degree [1,48], graph energy and vertex coverage [22,68], graph energy and the smallest eigenvalue of the adjacency matrix [3], and so on. One refers to [27,45,60] and the references therein for review on these bounds.

Although there are thousands of theoretical results on graph energy, to the best of our knowledge, few of them are concerned with the problem of computing graph energy. As far as we know, most of these studies focus on seeking lower or upper bounds on graph energy, however, these bounds may be greatly overestimated or underestimated, and even may be far away from its real values. Gutman [27] mentioned whether it is possible to use some effective algorithms to solve graph energy. Recently, Safaei *et al.* [59] proposed to calculate graph energy based on the Estrada-Benzi method, however, the method is inappropriate to large-scale graphs. Indeed, for small graphs (with a dozen or so vertices) this is no problem at all, while for large-scale graphs (with over tens of thousands or even over millions of vertices), it seems that the problem was ignored until now. The key reason it that computing all the eigenvalues or singular values of a large-scale graph is prohibitive or even infeasible [23,58].

Graph energy is a very useful parameter in studying large-scale graphs or networks. Indeed, as a graph invariant, graph energy contains very important structural information about the graph, its subgraphs, and ingredient segments [18, p. 72], [59, p. 214]. Evaluating graph energy of large-scale graphs or networks efficiently is very attractive in practice. For example, ranking nodes in a large-scale network is one of the most pressing and challenging tasks in complex network [18]. Graph energy can be used to measure the node importance. More precisely, the importance of a node is equivalent to the change of graph energy caused by its removal [2, 14], [49, p. 13]. Moreover, the Laplacian energy, which comes from graph energy [11–13, 56], is a broad measure of graph complexity [12, p. 52], [56, p. 242]. Indeed, Laplacian graph energy can be used to measure the importance (centrality) of a vertex by the relative drop of Laplacian energy in the network caused by the deactivation of this vertex from the network [56, p. 242]. Song *et al.* [63] introduced component-wise Laplacian graph energy, as a complexity measure useful to filter image description hierarchies.

To the best of our knowledge, however, efficient algorithms for computing energy of extremely large-scale graphs are still lacking. To fill in this gap, we propose three randomized algorithms for evaluating energy of large-scale undirected and directed graphs. The idea is based on the observation that big data matrices are approximately low (numerical) rank [33,40,61,66,70], and our aim is to give estimations which are in the same magnitude as the "exact" graph energy. Indeed, for large-scale graphs with over tens of thousands or even over millions of vertices, the computation of "real" values of graph energy is prohibitive or even infeasible, and one has to evaluate it numerically. In this situation, the numerical and computational errors arising in the algorithms are unavoidable. Fortunately, in some real applications, it is enough to know the magnitude of a graph energy. If the "exact" values are needed or it is desirable to know the values in a very high accuracy, the proposed algorithms are not applicable. However, they could be used to provide nice initial guesses.

The contribution of this work is as follows. First, we propose a low-rank approximation approach based on randomized singular value decomposition (RSVD) [30, 46] for graph energy. However, the target rank required in this algorithm is difficult to determine in advance, and the estimation is often underestimated. In order to deal with this problem, the second contribution is to propose a non-restarted and randomized algorithm that expands the search subspace consecutively by block Gram-Schmidt orthogonalization. However, the storage requirements and the computational cost will increase gradually as the iteration proceeds. So as to release the overhead, the third contribution is to propose a restarted and randomized algorithm instead, in which the dimension of the search subspace is fixed and an initial block matrix is constructed at each restarting, such that the algorithm can be run from scratch. Theoretical analysis is given to show the rationality of our new strategies, and numerical experiments are performed on some real-world and synthetic extremely large-scale graphs to illustrate the effectiveness of the proposed algorithms.

In this paper, we denote by $\mathcal{G}$ a graph with n vertices and m edges, by $E(\mathcal{G})$ the graph energy, and by $A \in R^{n\times n}$ the adjacency matrix of $\mathcal{G}$. Denote by $\mathbb{E}(\cdot)$ the expectation with respect to random test matrix. Let $\|\cdot\|_*$, $\|\cdot\|_2$, and $\|\cdot\|_F$ be the energy norm, the 2-norm and the Frobenius norm of a matrix, respectively. We denote by $\otimes$ the Kronecker product, and by vec$(\cdot)$ the "vec operator" that stacks the columns of a matrix into a long vector. Let I and $\mathbf{0}$ be the identity matrix and zero matrix (or zero vector), whose order are clear from context.

2. A Low-rank Approximation Method for Evaluating Energy of Large-scale Graph

Graph energy is the sum of the absolute value of eigenvalues of the adjacency matrix. In practice, the adjacency matrix is so large that it is impractical or infeasible to compute all its eigenvalues. With the help of the random singular value decomposition [30,46], in this section, we propose an algorithm for evaluating energy of large-scale directed and undirected graphs. We first introduce some definitions, and then present the new algorithm.

Definition 2.1 ([45]). *Let $\mathcal{G}$ be a graph with m edges and n vertices, and let $A \in R^{n\times n}$ be the adjacency matrix. Then the energy of the graph G is defined as*

$$E(\mathcal{G}) = \sum_{i=1}^{n} |\lambda_i|, \tag{2.1}$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A. Specifically, if $\mathcal{G}$ is an undirected graph, then the energy of the undirected graph $\mathcal{G}$ can be rewritten as

$$E(\mathcal{G}) = \|A\|_* = \sum_{i=1}^{n} \sigma_i, \tag{2.2}$$

where $\sigma_1, \dots, \sigma_n$ are the singular values of A, and $\|A\|_*$ denotes the energy norm of A.

The aim of this paper is to provide fast algorithms for energy of extremely large-scale graphs such that the estimation and the “exact” value can be in the same magnitude. Here “in the same magnitude” means:

Definition 2.2. *Let $x, y > 0$ be two real numbers, then under the scientific notation, if they can be expressed as*

$$x = a_1 \times 10^{b_1}, \quad y = a_2 \times 10^{b_2},$$

where $1 \le a_1, a_2 < 10$ and $b_1 = b_2$, then we say that x and y are in the same magnitude.

Matrices of approximately low (numerical) rank are pervasive in data science [66]. We have the following definition for low (numerical) rank of a matrix.

Definition 2.3 ([31]). *Let $A \in \mathbb{R}^{n\times n}$ be nonzero. For $k \le n$, the rank-k accuracy of A is*

$$\varepsilon_k(A) = \min_{W_k \in \mathbb{R}^{n\times n}} \left\{ \frac{\|A - W_k\|_2}{\|A\|_2} : \operatorname{rank}(W_k) \le k \right\}. \tag{2.3}$$

We call W_k an optimal rank-k approximation to A if W_k achieves the minimum in (2.3). The numerical rank of A at accuracy ε, denoted by $k_\varepsilon(A)$, is

$$k_\varepsilon(A) = \min\{k : \varepsilon_k(A) \le \varepsilon\}.$$

The matrix A is of low (numerical) rank if $\varepsilon_k(A) \ll 1$ for some $k \ll n$.

It is often the case that the graphs are massive in size involving over hundreds of millions of vertices. An important tool for analysis and interpretation of the data is the low rank approximation of the adjacency matrices or graph Laplacians related to the graphs, and low rank approximations in these applications are extremely useful for many different reasons [61]. For example, one of the recent studies on generating adversarial attacks for graph data is Nettack [73]. The Nettack model has shown to be very successful in deceiving the graph convolutional network (GCN) model. Entezari *et al.* [17] showed that the GCN model can significantly resist the attacks when a low-rank approximation of the graph is used. Wu *et al.* [70]

considered the use of low rank approximation to reconstruct the graph topology from the randomized network.

It was pointed out that adjacency matrices may have low rank structure [33, 40, 61, 66, 70]. In particular, Hsieh *et al.* [33, p. 509] showed that the adjacency matrix of a complete k-weakly balanced network is low rank. In practice, there are a large number of dangling nodes [44] and unreferenced nodes [71] contained in the Web graphs, and there are many zero rows or zero columns in the adjacency matrices. Hence, the adjacency matrices can be low (numerical) rank in practice. To illustrate this more precisely, we plot in Figs. 2.1-2.2 the singular values of 8 (with 4 symmetric and 4 unsymmetric) adjacency matrices of directed or undirected graphs, respectively, in which 7 of them are from the SNAP (Stanford network analysis platform) network data sets[1)], and the sticky-2e4 matrix is from running the MATLAB toolbox CONTEST [64]. The details of the adjacency matrices are listed in Table 2.1. It is obvious to see from the figures that the singular values of the adjacency matrices decay quickly, and the adjacency matrices are low (numerical) rank.

Table 2.1: Details of some adjacency matrices.

Adjacency matrix	Size (n)	Symmetric	Title or background
as-22july06	22963	Yes	(Symmetrized) Structure of internet routers as of July 22, 2006
bfly	49152	Yes	Butterfly graph sequence
Oregon-1	11492	Yes	(9 graphs) AS peering info inferred from Oregon route-views, 3/31-5/26/01
sticky-2e4	20000	Yes	Synthetic graph created by the MATLAB toolbox CONTEST [64], with $n = 20000$
enron	69244	No	Enron email network
EPA	4772	No	Pajek network: Kleinberg, pages linking to *www.epa.gov*
p2p-Gnutella24	26518	No	Gnutella peer to peer network from August 24, 2002
wiki-Vote	8297	No	Wikipedia who-votes-on-whom network

On the other hand, for adjacency matrices that do not have a clear low rank structure, a very promising approach is clustered low-rank approximation: instead of computing a global low-rank approximation, the adjacency matrix is first clustered (which does not change the graph energy), and then a low-rank approximation of each cluster (i.e. diagonal block) is computed [61]. A framework called clustered low rank matrix approximation for massive graphs was proposed in [61]. A low-rank sparse decomposition of adjacency matrices for graphs having clusters was given in [40].

Motivated by these observations, the idea is that we first compute a low-rank approximation to the adjacency matrix, and then evaluate graph energy by using a projection method. To this aim, one can use some randomized algorithms, such as the randomized singular value decomposition [30, 46, 51], and some sparse-preserving algorithms such as the randomized column–row–column decomposition (CUR) and interpolative decompositions (ID) [67], and the randomized unitary–triangular–unitary decomposition (UTV) [50], and so on. As the energy norm is a unitary invariant norm and the graph energy is an eigenvalue problem in essence, we exploit the widely used randomized singular value decomposition in this work.

[1)] https://sparse.tamu.edu/SNAP

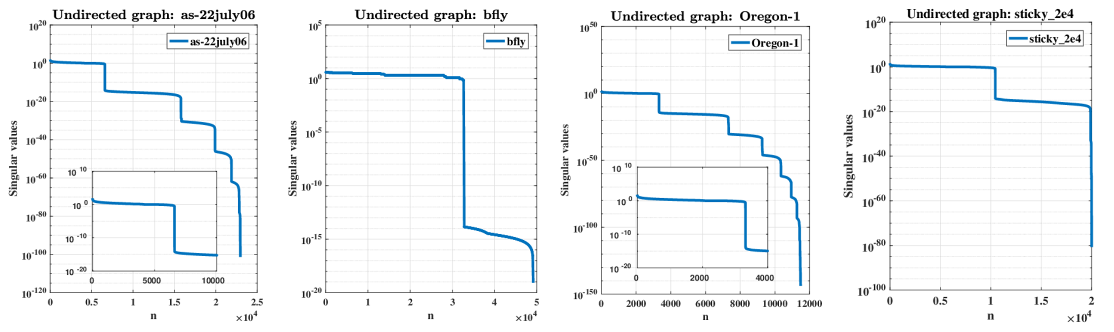

Fig. 2.1. Singular values of some adjacency matrices of undirected graphs.

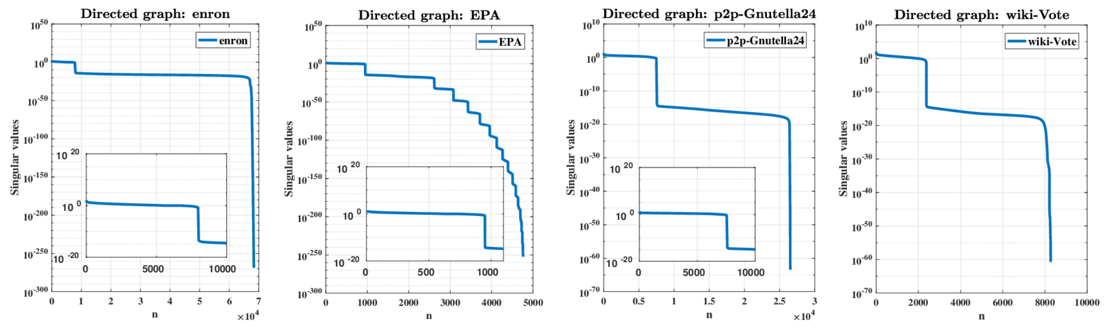

Fig. 2.2. Singular values of some adjacency matrices of directed graphs.

The randomized singular value decomposition [30, 46, 51] is a powerful tool for computing low-rank approximations of large-scale matrices. This method is simply divided into two phases [30]: First, constructing a low-dimensional subspace capturing most of the information of a given matrix. Second, projecting the original matrix into the subspace and then computing standard decomposition of the reduced matrix. The framework of this method is described in Algorithm 2.1, for more details, refer to [30].

The following theorem indicates that if the singular values of a matrix A decay rapidly, the randomized singular value decomposition will provide a sufficiently good approximation to the matrix.

Algorithm 2.1: A Randomized Singular Value Decomposition [30].

```
Input  : The data matrix $A \in \mathbb{R}^{m\times n}$, a target rank $\ell$, an over-sampling parameter $p$,
         and let $s = \ell + p$.
Output: The matrices $U, V_B$ and $S_B$.
1 Generate a random matrix $\Omega \in \mathbb{R}^{n\times s}$.
2 Compute the matrix $Y = A\Omega$, and perform the economized orthogonal triangular
    decomposition (QR) decomposition for an orthonormal basis $Q$ of $Y$.
3 Form the matrix $B = Q^\top A \in \mathbb{R}^{s\times n}$, and compute the economized singular value
    decomposition (SVD) of $B = U_B S_B V_B^\top$.
4 Return $U = QU_B, V_B$ and $S_B$ with $A \approx (QU_B)S_B V_B^\top = US_B V_B^\top = QQ^\top A$.
```

Theorem 2.1 ([30]). *Let $A \in R^{m\times n}$ be a real matrix with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. Choose a target rank $\ell \geq 2$ and an over-sampling parameter $p \geq 2$, where $\ell + p \leq \min\{m, n\}$. Draw an $n \times (\ell + p)$ standard Gaussian matrix Ω, and construct the sample matrix $Y = A\Omega$. Then*

$$\mathbb{E}(\|A - QQ^\top A\|_F) \leq \left(1 + \frac{\ell}{p-1}\right)^{\frac{1}{2}} \left(\sum_{j>\ell} \sigma_j^2\right)^{\frac{1}{2}},$$

where $\mathbb{E}(\cdot)$ denotes expectation with respect to the random test matrix.

We are ready to estimate energy of large-scale graphs. We first focus on the undirected graph. In this case, the adjacency matrix is symmetric, and the graph energy is the sum of all the singular values of A. Let $US_BV_B^\top$ be the RSVD of A obtained from Algorithm 2.1. Thus, the energy norm of A can be approximated by using that of $US_BV_B^\top$. More precisely, as Q, U and V_B are orthonormal matrices, we have

$$\|A\|_* \approx \|QQ^\top A\|_* = \left\|US_BV_B^\top\right\|_* = \|S_B\|_*. \tag{2.4}$$

Notice that there is no need to form or store the low-rank approximation $US_BV_B^\top$ explicitly.

The following theorem indicates that the quality of the estimation is closely related to the decaying of the singular values of the adjacency matrix. More precisely, if $\sigma_{\ell+1}, \ldots, \sigma_n$ are relatively small, then $\|S_B\|_*$ and $\|A\|_*$ can be in the same magnitude.

Theorem 2.2. *Let A be the adjacency matrix of an undirected graph, then under the above notations, we have*

$$\mathbb{E}\big(\big|\|A\|_* - \|S_B\|_*\big|\big) \leq \sqrt{r}\left(1 + \frac{\ell}{p-1}\right)^{\frac{1}{2}} \left(\sum_{j>\ell} \sigma_j^2\right)^{\frac{1}{2}}, \tag{2.5}$$

where r is the rank of the error matrix $A - US_BV_B^\top$.

Proof. Note that the energy norm is an unitary invariant norm, so we have

$$\begin{aligned}\big|\|S_B\|_* - \|A\|_*\big| &= \big|\left\|US_BV_B^\top\right\|_* - \|A\|_*\big| \leq \left\|A - US_BV_B^\top\right\|_* \\ &= \|A - QQ^\top A\|_* \leq \sqrt{r}\|QQ^\top A - A\|_F, \end{aligned} \tag{2.6}$$

where r is the rank of $A - US_BV_B^\top$, and (2.5) follows from a combination of Theorem 2.1 and (2.6). □

Now we consider the large-scale directed graph. In this case, the adjacency matrix is non-symmetric. By (2.1), the graph energy is the sum of the absolute values of all the eigenvalues of the adjacency matrix. Similar to the case of undirected graphs, the idea is to reduce the large matrix eigenvalue problem into a much smaller one, so that one can save the computational cost significantly. We need the following result.

Theorem 2.3 ([20, 54]). *Let the two matrices $M \in R^{n_1\times m_1}$ and $N \in R^{m_1\times n_1}$, then the non-zero eigenvalue of MN are the same as those of NM.*

We have from Algorithm 2.1 that $QQ^\top A$ is a low-rank approximation to A, i.e.

$$A \approx QQ^\top A = QB = QU_BS_BV_B^\top \equiv \widehat{A} \in \mathbb{R}^{n\times n}.$$

Denote by

$$C = S_B V_B^\top Q U_B \in \mathbb{R}^{(\ell+p)\times(\ell+p)},$$

then we have from Theorem 2.3 that the nonzero eigenvalues of the "small" matrix C and the "large" matrix $\widehat{A}$ are the same. Therefore, the energy of the directed graph G can be approximated by the sum of the absolute values of the eigenvalues of C.

In summary, we have the following algorithm for evaluating energy of large-scale direct and undirect graphs.

The main overhead of Algorithm 2.2 is from the economized QR decomposition of Y in Step 2 and the economized SVD of B in Step 3, in $\mathcal{O}(n(\ell+p)^2)$ flops altogether, as well as $2(\ell+p)$ sparse matrix-vector products in Steps 1 and 3, in $\mathcal{O}((\ell+p)\cdot nnz(A))$ flops. Here $nnz(A)$ denotes the number of nonzero elements of A.

Algorithm 2.2: A RSVD-based Algorithm for Evaluating Energy of Large-scale Graph.

```
Input  : The adjacency matrix $A$ of a graph $\mathcal{G}$, a target rank $\ell$ and an oversampling
         parameter $p$, and let $s = \ell + p$.
Output: An estimation to the energy of graph $E(\mathcal{G})$.
1 Form a random Gaussian matrix $\Omega$ of $n \times s$, and compute $Y = A\Omega$.
2 Perform the economized QR decomposition of $Y$: $Y = QR$, where $Q \in \mathbb{R}^{n\times s}$ is
    orthonormal and $R \in \mathbb{R}^{s\times s}$ is upper triangular.
3 Form $B = Q^\top A$ and calculate the economized singular value decomposition of $B$:
    $B = U_B S_B V_B^\top$.
4 On one hand, if $\mathcal{G}$ is an undirected graph, let $E(\mathcal{G}) \approx \|S_B\|_*$. On the other hand, if $\mathcal{G}$ is
    a directed graph, let $E(\mathcal{G}) \approx \sum_{i=1}^{s} |\lambda_i|$, where $\{\lambda_i\}_{i=1}^{s}$ are the eigenvalues of the
    $s$-by-$s$ matrix $C = S_B V_B^\top Q U_B$.
```

3. Non-restarted and Restarted Algorithms for Evaluating Energy of Large-scale Graph

In Algorithm 2.2, one has to provide the target rank ℓ and the oversampling parameter p, which are difficult to determine in advance, if there is no information available a priori. Consequently, the results evaluated can be underestimated and even very poor; see the numerical experiments made in Section 4.

In order to deal with these problems, in this work, we propose two randomized algorithms for evaluating energy of large-scale undirected and directed graphs. The first one is a "non-restarted" algorithm based on expanding the search subspace by block Gram-Schmidt orthogonalization. However, the storage requirements and computational cost of this algorithm will increase gradually as the iteration proceeds, especially for extremely large-scale graphs. To overcome this difficulty, we then propose a "restarted" randomized algorithm in which an initial block matrix is constructed at each restarting, such that the storage requirements of algorithm are limited.

3.1. A non-restarted and randomized algorithm for energy of large-scale graph

In this subsection, we propose a non-restarted and randomized algorithm for energy of large-scale graphs. We first suppose that $\mathcal{G}$ is an undirected graph, and the adjacency matrix $A \in R^{n\times n}$ is real symmetric. We will introduce this algorithm by recursion. Let $\mathcal{Q}_k = [Q_1, Q_2, \ldots, Q_k]$ be the orthonormal matrix obtained from the first k-th step of the algorithm. Then,

$$H_k = \mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top$$

is an orthogonal projection of A in the subspace span$\{\mathcal{Q}_k\}$ [23], which can be utilized as an approximation to A. As $\mathcal{Q}_k$ is orthonormal, the energy norm of $C_k \equiv \mathcal{Q}_k^\top A\mathcal{Q}_k$ is equal to that of H_k, i.e.

$$\|C_k\|_* = \|H_k\|_*,$$

and we make use of $\|C_k\|_*$ as an approximation to $\|A\|_*$.

If the approximation is not good enough, we expand $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1} = [\mathcal{Q}_k, Q_{k+1}]$, such that $\mathcal{Q}_{k+1}$ is orthonormal, with $Q_{k+1} \in \mathbb{R}^{n\times s}$ being orthogonalized to $\mathcal{Q}_k$. Therefore,

$$C_{k+1} = \mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1} = \begin{pmatrix} C_k & \mathcal{Q}_k^\top AQ_{k+1} \\ Q_{k+1}^\top A\mathcal{Q}_k & Q_{k+1}^\top AQ_{k+1}\end{pmatrix}, \tag{3.1}$$

and we use $\|C_{k+1}\|_*$ as a new approximation to $\|A\|_*$. However, as the columns of $\mathcal{Q}_{k+1}$ enlarges, the storage requirements and the overhead for calculating C_{k+1} will increase gradually.

The idea is that, if $\|Q_{k+1}^\top A\mathcal{Q}_k\|_*$ is sufficiently small compared with $\|C_{k+1}\|_*$, we can use the energy norm of

$$D_{k+1} = \begin{pmatrix} C_k & \mathbf{0} \\ \mathbf{0} & Q_{k+1}^\top AQ_{k+1}\end{pmatrix} \tag{3.2}$$

to approximate $\|C_{k+1}\|_*$ instead. Notice that

$$\|D_{k+1}\|_* = \|C_k\|_* + \left\|Q_{k+1}^\top AQ_{k+1}\right\|_*, \tag{3.3}$$

so one only needs to compute $\|Q_{k+1}^\top AQ_{k+1}\|_*$ for updating the approximation, where $Q_{k+1}^\top A\ Q_{k+1} \in \mathbb{R}^{s\times s}$ is a small matrix.

To show the feasibility of this proposed strategy more precisely, we analyze the upper bound of the error between the exact value $\|A\|_*$ and the approximation $\|D_{k+1}\|_*$. Recall that both $\mathcal{Q}_k$ and $\mathcal{Q}_{k+1} = [\mathcal{Q}_k, Q_{k+1}]$ are orthonormal matrices, and thus,

$$\begin{aligned}\big|\|A\|_* - \|D_{k+1}\|_*\big| &= \big|\|A\|_* - \|C_{k+1}\|_* + \|C_{k+1}\|_* - \|D_{k+1}\|_*\big| \\ &= \big|\|A\|_* - \left\|\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top\right\|_* + \|C_{k+1}\|_* - \|D_{k+1}\|_*\big| \\ &\le \left\|A - \mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top\right\|_* + \|C_{k+1} - D_{k+1}\|_*.\end{aligned}$$

If we denote by

$$\varepsilon_{k+1} = \left\|A - \mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top\right\|_* \tag{3.4}$$

the error from the low-rank approximation to A in the $(k+1)$-th step, then

$$\big|\|A\|_* - \|D_{k+1}\|_*\big| \le \varepsilon_{k+1} + \|C_{k+1} - D_{k+1}\|_*. \tag{3.5}$$

Moreover,

$$\begin{aligned}
C_{k+1}-D_{k+1} &= \begin{pmatrix} \mathcal{Q}_k^\top A\mathcal{Q}_k & \mathcal{Q}_k^\top AQ_{k+1} \\ Q_{k+1}^\top A\mathcal{Q}_k & Q_{k+1}^\top AQ_{k+1}\end{pmatrix} - \begin{pmatrix} \mathcal{Q}_k^\top A\mathcal{Q}_k & \mathbf{0} \\ \mathbf{0} & Q_{k+1}^\top AQ_{k+1}\end{pmatrix} \\
&= \begin{pmatrix} \mathbf{0} & \mathcal{Q}_k^\top AQ_{k+1} \\ Q_{k+1}^\top A\mathcal{Q}_k & \mathbf{0}\end{pmatrix}, \\
\|C_{k+1}-D_{k+1}\|_* &= 2\big\|\mathcal{Q}_k^\top AQ_{k+1}\big\|_*.
\end{aligned}$$

Denote by $T_k = A-\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top$, then

$$\begin{aligned}
\big\|\mathcal{Q}_k^\top AQ_{k+1}\big\|_* &= \big\|\mathcal{Q}_k^\top(\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top + T_k)Q_{k+1}\big\|_* = \big\|\mathcal{Q}_k^\top T_kQ_{k+1}\big\|_* \\
&= \big\|\mathcal{Q}_k^\top(A-\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top)Q_{k+1}\big\|_* \\
&\le \big\|A-\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top\big\|_* = \varepsilon_k,
\end{aligned}$$

where we used $\mathcal{Q}_k^\top Q_{k+1}=\mathbf{0}$, and

$$\varepsilon_k = \big\|A-\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top\big\|_* \tag{3.6}$$

is the error arising from the low-rank approximation to A in the k-th step. As $\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top$ is an orthogonal projector, we have $\varepsilon_{k+1}\le\varepsilon_k$.

In conclusion, we get the following result.

Theorem 3.1. *Let the adjacency matrix A be symmetric, then under the above notations, we have*

$$\begin{aligned}
D_{k+1} &= \operatorname{diag}\big(\mathcal{Q}_k^\top A\mathcal{Q}_k, Q_{k+1}^\top AQ_{k+1}\big), \\
\varepsilon_{k+1} &= \big\|A-\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top\big\|_*, \\
\varepsilon_k &= \big\|A-\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top\big\|_*,
\end{aligned}$$

then

$$\big|\|A\|_* - \|D_{k+1}\|_*\big| \le \varepsilon_{k+1}+2\varepsilon_k. \tag{3.7}$$

Remark 3.1. Theorem 3.1 indicates that the quality of the approximation is closely related to that of the low-rank approximations $\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top$ and $\mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top$. If $\varepsilon_k,\varepsilon_{k+1}$ are small enough, then $\|D_{k+1}\|_*$ will be a good approximation to $\|A\|_*$, and they can be in the same magnitude.

Remark 3.2. We point out that the strategy discussed above also applies to directed graphs. In this case, one only needs to calculate $\sum_{i=1}^s|\widetilde{\lambda}_i|$ instead of the energy norm of $Q_{k+1}^\top AQ_{k+1}$, where $\{\widetilde{\lambda}_i\}$'s are the eigenvalues of the matrix.

Based on the above discussions, we present in Algorithm 3.1 a non-restarted algorithm for evaluating graph energy of large-scale undirected and directed graphs.

The following theorem shows the convergence of this algorithm when $\mathcal{G}$ is an undirected graph.

Theorem 3.2. *Let the adjacency matrix A be symmetric, the approximation sequence constructed by Algorithm* 3.1 *is non-decreasing and is upper-bounded from above, and it converges.*

Algorithm 3.1: A "Non-restarted" and Randomized Algorithm for Graph Energy of Large-scale Graph.

```
  Input  : The adjacency matrix $A$, a parameter $s\ (= \ell + p)$, and the convergence
           threshold $\delta$.
  Output: An estimation $E$ to the graph energy.
1 Generate a random matrix $\Omega \in R^{n\times s}$ with i.i.d. Gaussian entries.
2 Let $W = A\Omega$ and compute the economized QR decomposition of $W$ to get $\mathcal{Q}$.
3 Compute the energy norm of $\mathcal{Q}^\top A\mathcal{Q}$, and use it as the initial guess $E_0$, let $r = 1$.
  while $r > \delta$ do
4     Generate a random matrix $\Omega_s \in R^{n\times s}$ with i.i.d. Gaussian entries.
5     Let $W_s = A\Omega_s$ and compute the economized QR decomposition of $W_s$ to get $Q_s$.
6     Orthonormalize $Q_s$ with respect to $\mathcal{Q}$: $Q_s = Q_s - \mathcal{Q}(\mathcal{Q}^\top Q_s)$.
7     On one hand, if $\mathcal{G}$ is a undirected graph, we compute the energy norm of $Q_s^\top AQ_s$
        and set it to be $E_s$. On the other hand, if $\mathcal{G}$ is a directed graph, we calculate
        $\sum_{i=1}^{s}|\widetilde{\lambda}_i|$ for the energy norm of $Q_s^\top AQ_s$, where $\{\widetilde{\lambda}_i\}$'s are the eigenvalues of the
        matrix.
8     Let $E = E_0 + E_s$ and $r = (E - E_0)/E$.
      if $r < \delta$ then
          Output $E$ and stop.
      else
          Let $\mathcal{Q} = \begin{pmatrix}\mathcal{Q} & Q_s\end{pmatrix}$ and $E_0 = E$.
      end
  end
```

Proof. Let

$$\begin{aligned} E_k &= \left\|\operatorname{diag}\left(Q_1^\top AQ_1, Q_2^\top AQ_2, \ldots, Q_k^\top AQ_k\right)\right\|_*, \\ E_{k+1} &= \left\|\operatorname{diag}\left(Q_1^\top AQ_1, Q_2^\top AQ_2, \ldots, Q_{k+1}^\top AQ_{k+1}\right)\right\|_*, \end{aligned}$$

then it is seen that

$$E_1 \leq E_2 \leq \cdots \leq E_k \leq E_{k+1},$$

moreover,

$$E_{k+1} \leq \left\|\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\right\|_* \leq \|A\|_*.$$

That is, the sequence $\{E_k\}_{k=0}^{\infty}$ is non-decreasing and is upper-bounded from above, so it converges. □

3.2. A restarted and randomized algorithm for energy of extremely large-scale graph

In Algorithm 3.1, the columns of the matrix $\mathcal{Q}$ will enlarge consecutively as the iterations proceed. Consequently, the amount of calculation and storage requirements will increase gradually. Hence, the computational cost will be prohibitively high and the algorithm may be infeasible when the graph is extremely large. To settle this problem, we propose a "restarted" randomized algorithm for calculating the energy of extremely large graphs, whose order may be over, say, 10^6. Rather than increase the number of the columns of $\mathcal{Q}$ unlimited, if the size is larger than a given threshold, say, 10^3, we will choose a new initial block matrix and run the algorithm from scratch.

More precisely, suppose that $\mathcal{Q}_k = [Q_1, \ldots, Q_{k-1}, Q_k] = [\mathcal{Q}_{k-1}, Q_k]$ has reached the maximal number of columns, and we have to restart this algorithm. Thus, the block matrix Q_{k+1} generated in the next step will not be orthogonalized with respect to $\mathcal{Q}_k$ any more. Indeed, we only orthogonalize it against the "nearest" block matrix Q_k, and take advantage of the resulting matrix as the initial block matrix for the next outer iteration.

Let us discuss it in more detail. Denote by $\mathcal{Q}_{k+1} = [\mathcal{Q}_k,\ Q_{k+1}]$, and by

$$\begin{aligned}\mathcal{A}_{k+1} &= \mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\mathcal{Q}_{k+1}^\top \\ &= \mathcal{Q}_k\mathcal{Q}_k^\top A\mathcal{Q}_k\mathcal{Q}_k^\top + Q_{k+1}Q_{k+1}^\top AQ_{k+1}Q_{k+1}^\top \\ &= \begin{pmatrix}\mathcal{Q}_k & Q_{k+1}\end{pmatrix}\begin{pmatrix}\mathcal{Q}_k^\top A\mathcal{Q}_k & \mathbf{0} \\ \mathbf{0} & Q_{k+1}^\top AQ_{k+1}\end{pmatrix}\begin{pmatrix}\mathcal{Q}_k^\top \\ Q_{k+1}^\top\end{pmatrix} \\ &\equiv \begin{pmatrix}\mathcal{Q}_k & Q_{k+1}\end{pmatrix} D_{k+1}\begin{pmatrix}\mathcal{Q}_k^\top \\ Q_{k+1}^\top\end{pmatrix},\end{aligned} \tag{3.8}$$

where

$$D_{k+1} = \operatorname{diag}\big(\mathcal{Q}_k^\top A\mathcal{Q}_k,\ Q_{k+1}^\top AQ_{k+1}\big) \tag{3.9}$$

is a block diagonal matrix.

Unlike that in the non-restarted algorithm, here $\mathcal{Q}_{k+1}$ is not an orthonormal matrix any more. Without loss of generality, we assume that $\mathcal{Q}_{k+1} = [\mathcal{Q}_k,\ Q_{k+1}]$ is of full column rank. Let the economized QR decomposition be

$$\begin{pmatrix}\mathcal{Q}_k & Q_{k+1}\end{pmatrix} = \begin{pmatrix}\mathcal{Q}_k & \hat{Q}_{k+1}\end{pmatrix}\begin{pmatrix}I & R_{12} \\ \mathbf{0} & R_{22}\end{pmatrix}, \tag{3.10}$$

where $\hat{Q}_{k+1}$ is used as the initial block matrix for restarting, $\hat{Q}_{k+1}^\top\mathcal{Q}_k = \mathbf{0}$, and R_{22} is nonsingular. Moreover,

$$Q_{k+1} = \mathcal{Q}_k R_{12} + \hat{Q}_{k+1}R_{22}, \tag{3.11}$$

and

$$\|R_{12}\|_2 = \big\|\mathcal{Q}_k^\top Q_{k+1}\big\|_2 = \left\|\begin{pmatrix}\mathcal{Q}_{k-1}^\top \\ Q_k^\top\end{pmatrix}Q_{k+1}\right\|_2 = \left\|\begin{pmatrix}\mathcal{Q}_{k-1}^\top Q_{k+1} \\ \mathbf{0}\end{pmatrix}\right\|_2 = \big\|\mathcal{Q}_{k-1}^\top Q_{k+1}\big\|_2. \tag{3.12}$$

Furthermore, we have from (3.11) that

$$\begin{aligned}\hat{Q}_{k+1}R_{22} &= Q_{k+1} - \mathcal{Q}_k R_{12} = Q_{k+1} - \mathcal{Q}_k\mathcal{Q}_k^\top Q_{k+1} \\ &= Q_{k+1} - \begin{pmatrix}\mathcal{Q}_{k-1} & Q_k\end{pmatrix}\begin{pmatrix}\mathcal{Q}_{k-1}^\top \\ Q_k^\top\end{pmatrix}Q_{k+1} \\ &= Q_{k+1} - \mathcal{Q}_{k-1}\mathcal{Q}_{k-1}^\top Q_{k+1},\end{aligned} \tag{3.13}$$

where we used $Q_k^\top Q_{k+1} = \mathbf{0}$. Thus, it follows from (3.8) and (3.10) that

$$\begin{aligned}\mathcal{A}_{k+1} &= \begin{pmatrix}\mathcal{Q}_k & \hat{Q}_{k+1}\end{pmatrix}\begin{pmatrix}I & R_{12} \\ \mathbf{0} & R_{22}\end{pmatrix} D_{k+1}\begin{pmatrix}I & \mathbf{0} \\ R_{12}^\top & R_{22}^\top\end{pmatrix}\begin{pmatrix}\mathcal{Q}_k^\top \\ \hat{Q}_{k+1}^\top\end{pmatrix} \\ &\equiv \hat{\mathcal{Q}}_{k+1}\hat{D}_{k+1}\hat{\mathcal{Q}}_{k+1}^\top,\end{aligned} \tag{3.14}$$

where $\hat{\mathcal{Q}}_{k+1} = \begin{pmatrix}\mathcal{Q}_k & \hat{Q}_{k+1}\end{pmatrix}$ is orthonormal, and

$$\hat{D}_{k+1} = \begin{pmatrix} I & R_{12} \\ \mathbf{0} & R_{22} \end{pmatrix} D_{k+1} \begin{pmatrix} I & \mathbf{0} \\ R_{12}^\top & R_{22}^\top \end{pmatrix}. \tag{3.15}$$

As $\hat{\mathcal{Q}}_{k+1}$ is orthonormal, it follows from (3.14) that one can use $\|\hat{D}_{k+1}\|_*$ as an approximation to $\|A\|_*$. However, we see from (3.15) that $\hat{D}_{k+1}$ is a full matrix, whose size will become larger and larger as the restarting proceeds. To deal with this problem, we propose to exploit $\|D_{k+1}\|_*$ as an approximation to $\|A\|_*$ instead. The key is that D_{k+1} is a block diagonal matrix, and one only needs to compute the eigenvalues or singular values of a small matrix $Q_{k+1}^\top AQ_{k+1}$ at each restarting, refer to (3.9). The resulting algorithm is presented in Algorithm 3.2. Similarly, the restarting strategy also applies to evaluate energy of directed graphs. More precisely, one only needs to calculate $\sum_{i=1}^k |\widetilde{\lambda}_i|$ instead of energy norm of $Q_s^\top AQ_s$, where $\{\widetilde{\lambda}_i\}$'s are the eigenvalues of the s-by-s matrix.

We are ready to show the rationality of the restarted and randomized algorithm. For simplicity, we only consider the case of undirected graph. Suppose that the largest number h of columns of $\mathcal{Q}_k = [Q_1, \ldots, Q_{k-1}, Q_k] = [\mathcal{Q}_{k-1}, Q_k]$ has reached, where $k = h/s$, and we

Algorithm 3.2: A "Restarted" and Randomized Algorithm for Extremely Large-scale Graphs.

```
Input  : The adjacency matrix $A$, the number of $s$ $(= \ell + p)$ for sampling, the
         number $h$ for controlling the column number of $\mathcal{Q}$, and the convergence
         threshold $\delta$.
Output: An estimation $E$ to the graph energy.
1 Generate a random matrix $\Omega \in R^{n\times s}$ with i.i.d. Gaussian entries.
2 Let $W = A\Omega$ and compute the economized QR decomposition of $W$ to get $\mathcal{Q}$.
3 Compute the energy norm of $\mathcal{Q}^\top A\mathcal{Q}$, and use it as the initial guess $E_0$, let $r = 1$.
  while $r > \delta$ do
4     Generate a random matrix $G_s \in R^{n\times s}$ with i.i.d. Gaussian entries.
5     Let $W_s = AG_s$, and perform the economized QR factorization of $W_s$ to get $Q_s$.
6     Let $Q_s = Q_s - \mathcal{Q}(\mathcal{Q}^\top Q_s)$.
7     On one hand, if it is a undirected graph, computing the energy norm of $Q_s^\top AQ_s$,
       and set it to be $E_s$. On the other hand, if it is a directed graph, calculating
       $\sum_{i=1}^s |\widetilde{\lambda}_i|$ instead of the energy norm, where $\{\widetilde{\lambda}_i\}$'s are the eigenvalues of the
       matrix $Q_s^\top AQ_s$.
8     Let $E = E_0 + E_s$ and $r = (E - E_0)/E$.
      if $r < \delta$ then
          Output $E$.
      else
          $\mathcal{Q} = \begin{pmatrix}\mathcal{Q} & Q_s\end{pmatrix}$.
          $E_0 = E$.
          if the number of columns of $\mathcal{Q}$ is larger than $h$ then
              $\mathcal{Q} = Q_s$.   % for restarting
          end
      end
  end   % the outer iteration
```

have to restart the algorithm from scratch. For restarting, the next block matrix Q_{k+1} only orthogonalizes to Q_k, and it is not orthogonal to $\mathcal{Q}_{k-1}$ any more. Without loss of generality, we make the assumption that

$$\cos\angle(\mathcal{Q}_{k-1}, Q_{k+1}) = \big\|\mathcal{Q}_{k-1}\mathcal{Q}_{k-1}^\top Q_{k+1}\big\|_2 = \big\|\mathcal{Q}_{k-1}^\top Q_{k+1}\big\|_2 \ll 1, \tag{3.16}$$

which is the cosine of the angle between the two subspaces span$\{\mathcal{Q}_{k-1}\}$ and span$\{Q_{k+1}\}$. That is, the two subspaces are sufficiently far away from each other.

Consider the error

$$\begin{aligned}\big|\|A\|_* - \|D_{k+1}\|_*\big| &= \big|\|A\|_* - \|\mathcal{A}_{k+1}\|_* + \|\mathcal{A}_{k+1}\|_* - \|D_{k+1}\|_*\big|\\ &\le \big|\|A\|_* - \|\mathcal{A}_{k+1}\|_*\big| + \big|\|\mathcal{A}_{k+1}\|_* - \|D_{k+1}\|_*\big|\\ &\le \big|\|A\|_* - \|\mathcal{A}_{k+1}\|_*\big| + \big|\|\hat{D}_{k+1}\|_* - \|D_{k+1}\|_*\big|\\ &\le \|A - \mathcal{A}_{k+1}\|_* + \|\hat{D}_{k+1} - D_{k+1}\|_*, \end{aligned}\tag{3.17}$$

where we used the fact that $\hat{\mathcal{Q}}_{k+1}$ is orthonormal and thus $\|\mathcal{A}_{k+1}\|_* = \|\hat{D}_{k+1}\|_*$, see (3.14). Notice that $\|A - \mathcal{A}_{k+1}\|_*$ is the error from a low-rank approximation to A, and it is only necessary to consider $\|\hat{D}_{k+1} - D_{k+1}\|_*$. Denote by

$$F = \begin{pmatrix} \mathbf{0} & R_{12} \\ \mathbf{0} & R_{22} - I \end{pmatrix} \in R^{(k+1)s\times(k+1)s},$$

where $R_{22} \in R^{s\times s}$ and I is identity matrix. Notice that the rank of F is at most s. Thus, we have from (3.15) that

$$\begin{aligned}\hat{D}_{k+1} - D_{k+1} &= (I+F)D_{k+1}(I+F^\top) - D_{k+1}\\ &= (D_{k+1} + FD_{k+1})(I+F^\top) - D_{k+1}\\ &= D_{k+1} + D_{k+1}F^\top + FD_{k+1} + FD_{k+1}F^\top - D_{k+1}\\ &= D_{k+1}F^\top + FD_{k+1} + FD_{k+1}F^\top,\end{aligned}$$

and

$$\|\hat{D}_{k+1} - D_{k+1}\|_* \le 2\|D_{k+1}\|_*\|F\|_2 + \|D_{k+1}\|_*\|F\|_2^2. \tag{3.18}$$

From (3.16) and (3.12),

$$\|R_{12}\|_2 = \cos\angle(\mathcal{Q}_{k-1}, Q_{k+1}) \ll 1,$$

moreover, as both $R_{12}^\top R_{12}$ and $(R_{22}-I)^\top(R_{22}-I)$ are Hermitian positive semi-definite matrices, we obtain

$$\begin{aligned}\|F\|_2^2 = \lambda_{\max}(F^\top F) &= \lambda_{\max}\big(R_{12}^\top R_{12} + (R_{22}-I)^\top(R_{22}-I)\big)\\ &\le \lambda_{\max}\big(R_{12}^\top R_{12}\big) + \lambda_{\max}\big((R_{22}-I)^\top(R_{22}-I)\big)\\ &= \|R_{12}\|_2^2 + \|R_{22}-I\|_2^2\\ &= \cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \|R_{22}-I\|_2^2.\end{aligned}\tag{3.19}$$

We now consider $\|R_{22} - I\|_2$. It follows from (3.13) that

$$\begin{aligned}R_{22} &= \hat{Q}_{k+1}^\top\big(Q_{k+1} - \mathcal{Q}_{k-1}\mathcal{Q}_{k-1}^\top Q_{k+1}\big)\\ &= \hat{Q}_{k+1}^\top Q_{k+1} - \hat{Q}_{k+1}^\top \mathcal{Q}_{k-1}\mathcal{Q}_{k-1}^\top Q_{k+1}\\ &= \hat{Q}_{k+1}^\top Q_{k+1},\end{aligned}\tag{3.20}$$

where we used $\hat{Q}_{k+1}^\top \mathcal{Q}_{k-1} = \mathbf{0}$. Thus,

$$\begin{aligned} R_{22}^\top R_{22} &= Q_{k+1}^\top \big(\hat{Q}_{k+1}\hat{Q}_{k+1}^\top\big) Q_{k+1} \\ &= Q_{k+1}^\top \big(\hat{Q}_{k+1}\hat{Q}_{k+1}^\top - Q_{k+1}Q_{k+1}^\top + Q_{k+1}Q_{k+1}^\top\big) Q_{k+1} \\ &= Q_{k+1}^\top \big(\hat{Q}_{k+1}\hat{Q}_{k+1}^\top - Q_{k+1}Q_{k+1}^\top\big) Q_{k+1} + I. \end{aligned}$$

As a result,

$$\big\|R_{22}^\top R_{22} - I\big\|_2 \le \big\|\hat{Q}_{k+1}\hat{Q}_{k+1}^\top - Q_{k+1}Q_{k+1}^\top\big\|_2 = \sin\angle(\hat{Q}_{k+1}, Q_{k+1}). \tag{3.21}$$

If R_{22} is nonsingular, we have from (3.10) that the two subspaces span$\{[\mathcal{Q}_k, \hat{Q}_{k+1}]\}$ and span$\{[\mathcal{Q}_k, Q_{k+1}]\}$ are the same. As span$\{\hat{Q}_{k+1}\}$ is orthogonal to span$\{\mathcal{Q}_k\}$, moreover,

$$Q_{k+1}^\top Q_k = \mathbf{0}, \quad \cos\angle(\mathcal{Q}_{k-1}, Q_{k+1}) \ll 1,$$

so span$\{Q_{k+1}\}$ is "almost" orthogonal to span$\{\mathcal{Q}_k\}$. Thus, the two subspaces span$\{\hat{Q}_{k+1}\}$ and span$\{Q_{k+1}\}$ are close to each other, i.e.

$$\sin\angle(\hat{Q}_{k+1}, Q_{k+1}) \ll 1, \tag{3.22}$$

and $\|R_{22}^\top R_{22} - I\|_2 \ll 1$. Denote by

$$S = R_{22}^\top R_{22} - I, \tag{3.23}$$

we have

$$\begin{aligned} R_{22} - I &= \big(R_{22} - R_{22}^\top R_{22}\big) + \big(R_{22}^\top R_{22} - I\big) \\ &= \big(I - R_{22}^\top\big) R_{22} + S. \end{aligned} \tag{3.24}$$

Let $X = R_{22} - I$, then (3.24) can be rewritten as

$$I \cdot X + X^\top \cdot R_{22} = S, \tag{3.25}$$

which is a T-Sylvester equation [34], and $R_{22} - I$ is the solution to this problem.

We are in a position to consider the solution of (3.25). The following theorem gives a sufficient and necessary condition that 1 or -1 is an eigenvalue of the upper-triangular matrix R_{22}.

Theorem 3.3. *Let $\mathbf{q}_i$ and $\hat{\mathbf{q}}_i$ be the i-th column of Q_{k+1} and $\hat{Q}_{k+1}$, respectively. Then the i-th diagonal element of R_{22} is ± 1 if and only if $\mathbf{q}_i = \hat{\mathbf{q}}_i$ and $\mathcal{Q}_k^\top \mathbf{q}_i = 0, 1 \le i \le s$.*

Proof. We only prove the case of 1, and the proof of -1 is in a similar way. On one hand, we suppose that the i-th diagonal element of R_{22} is $1, 1 \le i \le s$. Let $\mathbf{e}_i$ be the i-th column of the identity matrix, then we obtain from (3.20) that

$$\mathbf{e}_i^\top R_{22} \mathbf{e}_i = \mathbf{e}_i^\top \hat{Q}_{k+1}^\top Q_{k+1} \mathbf{e}_i = 1.$$

That is, $\cos\angle(Q_{k+1}\mathbf{e}_i, \hat{Q}_{k+1}\mathbf{e}_i) = 1$, and the angle between $\mathbf{q}_i = Q_{k+1}\mathbf{e}_i$ and $\hat{\mathbf{q}}_i = \hat{Q}_{k+1}\mathbf{e}_i$ is zero. Let $\mathbf{q}_i = \alpha_i \hat{\mathbf{q}}_i$, where α_i is a real number, moreover, as both Q_{k+1} and $\hat{Q}_{k+1}$ are orthonormal and real, we have $|\alpha_i| = 1$, and $\alpha_i = \pm 1$.

Denote by $r_{i,j}$ the (i,j)-th element of R_{22}, and

$$R_{22}\mathbf{e}_i = (r_{1,i}, r_{2,i}, \ldots, r_{i-1,i}, 1, 0, \ldots, 0)^\top,$$

where $i, j = 1, 2, \ldots, s$. Then it follows from (3.13) that

$$\hat{Q}_{k+1} R_{22} \mathbf{e}_i = Q_{k+1} \mathbf{e}_i - \mathcal{Q}_{k-1} \mathcal{Q}_{k-1}^{\top} Q_{k+1} \mathbf{e}_i,$$

i.e.

$$\hat{\mathbf{q}}_1 r_{1,i} + \hat{\mathbf{q}}_2 r_{2,i} + \cdots + \hat{\mathbf{q}}_{i-1} r_{i-1,i} + r_{i,i} \hat{\mathbf{q}}_i = \mathbf{q}_i - \mathcal{Q}_{k-1} \big(\mathcal{Q}_{k-1}^{\top} \mathbf{q}_i \big). \tag{3.26}$$

Recall that $r_{ii} = 1$, so we have

$$\hat{\mathbf{q}}_1 r_{1,i} + \hat{\mathbf{q}}_2 r_{2,i} + \cdots + \hat{\mathbf{q}}_{i-1} r_{i-1,i} + (1 - \alpha_i) \hat{\mathbf{q}}_i = -\mathcal{Q}_{k-1} \big(\mathcal{Q}_{k-1}^{\top} \mathbf{q}_i \big) \equiv \mathbf{z}_i.$$

Hence, $\mathbf{z}_i \in \operatorname{span}\{\mathcal{Q}_{k-1}\} \bigcap \operatorname{span}\{\hat{Q}_{k+1}\}$. As these two subspaces are perpendicular to each other, we have $\mathbf{z}_i = \mathbf{0}$. Recall that $\hat{Q}_{k+1}$ is an orthonormal matrix, and $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_s$ are linearly independent, so we have

$$r_{1,i} = r_{2,i} = \cdots = r_{i-1,i} = 1 - \alpha_i = 0,$$

i.e. $\alpha_i = 1$, and thus $\mathbf{q}_i = \alpha_i \hat{\mathbf{q}}_i = \hat{\mathbf{q}}_i$. Furthermore, $\mathbf{z}_i = \mathbf{0}$ implies $\mathcal{Q}_{k-1}^{\top} \mathbf{q}_i = \mathbf{0}$. As $\mathbf{q}_i$ is the i-th column of Q_{k+1} and $Q_{k+1}^{\top} Q_k = \mathbf{0}$, we have $Q_k^{\top} \mathbf{q}_i = \mathbf{0}$ and $\mathcal{Q}_k^{\top} \mathbf{q}_i = \mathbf{0}$.

On the other hand, if $\mathbf{q}_i = \hat{\mathbf{q}}_i$ and $\mathcal{Q}_k^{\top} \mathbf{q}_i = \mathbf{0}$, it follows from (3.26) and the orthogonality of $\hat{Q}_{k+1}$ that $r_{ii} = 1$, $1 \leq i \leq s$. $\square$

Remark 3.3. In Algorithm 3.2, Q_{k+1} is not orthogonal to $\mathcal{Q}_k = [\mathcal{Q}_{k-1}, Q_k]$, and it is only orthogonalized with respect to Q_k. Thus, $\mathcal{Q}_k^{\top} \mathbf{q}_i \neq 0$, $i = 1, 2, \ldots, s$, and we have from Theorem 3.3 that 1 and -1 are not eigenvalues of R_{22}.

The following theorem gives the solution to a T-Sylvester equation.

Theorem 3.4 ([15]). *Let $A, B, C \in R^{n\times n}$, and A, B is non-singular, consider the following matrix equations:*

$$AX + X^{\top} B = C, \tag{3.27}$$

$$(B^{-\top} A) X - X (A^{-\top} B) = B^{-\top} C - B^{-\top} C^{\top} A^{-\top} B, \tag{3.28}$$

where $X \in R^{n\times n}$, then the following conclusions hold:

(1) If X_0 is the solution of (3.27)*, then X_0 is also the solution of* (3.28)*.*

(2) If (3.28) *has a unique solution X_0, then X_0 is also the unique solution of* (3.27)*.*

Let $A = I, B = R_{22}, C = S$ and $X = R_{22} - I$, then (3.25) can be written as (3.27). Notice that both A and B are nonsingular, by (3.28), we consider the following Sylvester equation:

$$R_{22}^{-\top} X - X R_{22} = R_{22}^{-\top} S - R_{22}^{-\top} S^{\top} R_{22}. \tag{3.29}$$

It is well known that the above Sylvester equation has a unique solution if and only if the eigenvalues of the matrices R_{22} and $R_{22}^{-\top}$ have no intersection [23, 34]. In view of the above discussions, we can present the following result.

Lemma 3.1. *Denote by*

$$\mathcal{L} = I_s \otimes R_{22}^{-\top} - R_{22}^{\top} \otimes I_s,$$

and by

$$W_1 = I_s \otimes R_{22}^{-\top}, \quad W_2 = R_{22}^{\top} \otimes R_{22}^{-\top},$$

then

$$\|R_{22} - I\|_2 \leq \omega\sqrt{s} \cdot \sin\angle(\hat{Q}_{k+1}, Q_{k+1}), \tag{3.30}$$

where

$$\omega = \|\mathcal{L}^{-1}\|_2(\|W_1\|_2 + \|W_2\|_2).$$

Proof. From (3.29), we have

$$\big(I_s \otimes R_{22}^{-\top} - R_{22}^{\top} \otimes I_s\big)\text{vec}(X) = \big(I_s \otimes R_{22}^{-\top}\big)\text{vec}(S) - \big(R_{22}^{\top} \otimes R_{22}^{-\top}\big)\text{vec}(S^{\top}),$$

where $\otimes$ is the Kronecker product, and vec$(\cdot)$ is the "vec operator" that stacks the columns of a matrix into a long vector. As R_{22} is upper-triangular, the diagonal elements (or the eigenvalues) of R_{22} are all real, and by Theorem 3.3, both 1 and -1 are not eigenvalues of R_{22}. As a result, the eigenvalues of the matrices R_{22} and $R_{22}^{-\top}$ have no intersection, and the Eq. (3.29) has a unique solution. Therefore, the matrix $\mathcal{L} = I_s \otimes R_{22}^{-\top} - R_{22}^{\top} \otimes I_s$ is nonsingular, and

$$\text{vec}(X) = \mathcal{L}^{-1}\big(W_1 \cdot \text{vec}(S) - W_2 \cdot \text{vec}(S^{\top})\big). \tag{3.31}$$

Note that $\|\text{vec}(S)\|_2 = \|\text{vec}(S^{\top})\|_2 = \|S\|_F$, we obtain from (3.31) that

$$\begin{aligned}\|R_{22} - I\|_2 = \|X\|_2 \leq \|X\|_F &= \|\text{vec}(X)\|_2 \\ &\leq \|\mathcal{L}^{-1}\|_2\|S\|_F(\|W_1\|_2 + \|W_2\|_2) \\ &\leq \sqrt{s}\|\mathcal{L}^{-1}\|_2(\|W_1\|_2 + \|W_2\|_2) \cdot \|S\|_2, \end{aligned} \tag{3.32}$$

where we used $\|S\|_F \leq \sqrt{s}\|S\|_2$, and the proof is completed by combining (3.21), (3.23) and (3.32). □

We have from (3.19) and (3.30) that

$$\begin{aligned}\|F\|_2^2 &\leq \|R_{12}\|_2^2 + \|R_{22} - I\|_2^2 \\ &\leq \cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1}).\end{aligned}$$

Hence,

$$\|F\|_2 \leq \sqrt{\cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1})}.$$

Applying the above equation to (3.18), we get

$$\begin{aligned}\|\hat{D}_{k+1} - D_{k+1}\|_* &\leq 2\|D_{k+1}\|_*\|F\|_2 + \|D_{k+1}\|_*\|F\|_2^2 \\ &\leq 2\|D_{k+1}\|_*\sqrt{\cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1})} \\ &\quad + \|D_{k+1}\|_*\big(\cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1})\big).\end{aligned}$$

Therefore, from the above equation and (3.17), we obtain the main result on the difference between the graph energy $\|A\|_*$ and our approximation $\|D_{k+1}\|_*$.

Theorem 3.5. *Let the adjacency matrix A be symmetric, then under the above notations and assumptions, we have that*

$$\begin{aligned}\big|\|A\|_* - \|D_{k+1}\|_*\big| \leq \|A - \mathcal{A}_{k+1}\|_* \\ + 2\|D_{k+1}\|_*\sqrt{\cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1})} \\ + \|D_{k+1}\|_*\big(\cos^2\angle(\mathcal{Q}_{k-1}, Q_{k+1}) + \omega^2 s \sin^2\angle(\hat{Q}_{k+1}, Q_{k+1})\big).\end{aligned}$$

Remark 3.4. Theorem 3.5 indicates that the quality of our approximation is closely related to the error from the low-rank approximation $\mathcal{A}_{k+1}$, as well as to the values $\cos\angle(\mathcal{Q}_{k-1}, Q_{k+1})$ and $\sin\angle(\hat{Q}_{k+1}, Q_{k+1})$. Moreover, the larger the angle between span$\{Q_{k+1}\}$ and span$\{\mathcal{Q}_{k-1}\}$, the better the approximation. By (3.9), this theorem shows the rationality of using

$$\|D_{k+1}\|_* = \left\|\mathcal{Q}_k^\top A\mathcal{Q}_k\right\|_* + \left\|Q_{k+1}^\top AQ_{k+1}\right\|_*$$

as an approximation to $\|A\|_*$ after restarting.

4. Numerical Experiments

In this section, we perform some numerical experiments on some real-world and synthetic data sets to illustrate the numerical behavior of the proposed algorithms. All the numerical experiments are run on a HP workstation with 16-core dual Intel (R) Xeon (R) E5-2637 v4 processor, and with CPU 3.50 GHz and RAM 256 GB. The operation system is 64-bit Windows 10. All the numerical results are obtained from using the MATLAB R2018b software.

In the first two examples, all the adjacency matrices are from `https://sparse.tamu.edu/`. Table 4.1 lists some details of the adjacency matrices, including the size n (the number of vertices), the number of non-zero elements m (Nnz, the number of edges), and whether the adjacency matrix is symmetric or not (undirected graph or directed graph). For an extremely large-scale graph, computation of the "real" graph energy is prohibitive or even infeasible. In order to show the effectiveness of our proposed strategies, we present in Table 4.2 the upper or lower bounds provided by using (1.1)-(1.3) for graph energy, which can be used as references to the computed values.

In all the experiments, we make use of the MATLAB build-in function `eig.m` (the implicitly restarted QR algorithm [23]) for all the eigenvalues of a matrix, which can be used as a baseline algorithm for computing graph energy. Indeed, the results obtained from `eig.m` will be used as the "exact values" of graph energy if available. In all the tables below, "CPU time" denotes CPU time in seconds. If an algorithm fails to compute the graph energy within 12 hours, we will stop it and denote the result by "–".

Example 4.1. In the non-restarted approach Algorithm 3.1, we assume that the norms of the off-diagonal blocks of the matrices $\|\mathcal{Q}_k^\top AQ_{k+1}\|_*$ are relatively small during iterations. In this example, we demonstrate that the assumption is reasonable. In Fig. 4.1, we depict the values of

$$\nu_k = \frac{\left\|\mathcal{Q}_k^\top AQ_{k+1}\right\|_F}{\left\|\mathcal{Q}_{k+1}^\top A\mathcal{Q}_{k+1}\right\|_F}, \quad k = 1, 2, \ldots \tag{4.1}$$

for the 4 undirected graphs oregon 1, as-22july06, cond-mat-2003, and cond-mat-2005. It is seen that the values of ν_k are much smaller than one during iterations. However, there is no guarantee that it is monotonically decreasing, see the figure for oregon 1.

Table 4.1: Details of some adjacency matrices.

Adjacency matrix	Size (n)	Symmetric	Nnz (m)
oregon 1	11492	Yes	46818
as-22july06	22963	Yes	96872
cond-mat-2003	21163	Yes	240058
cond-mat-2005	40421	Yes	351382
mycielskian16	49151	Yes	33382480
caidaRouterLevel	192244	Yes	1218132
coAuthorsDBLP	299067	Yes	1955352
dblp-2010	326186	Yes	1615400
coPapersDBLP	540486	Yes	30491458
delaunayn-19	524288	Yes	3145646
roadNet-PA	1090920	Yes	3083796
wiki-Vote	8297	No	103689
p2p-Gnutella04	10879	No	39994
p2p-Gnutella24	26518	No	65369
cit-HepPh	34546	No	421578
cit-HepTh	27770	No	352807
soc-Slashdot0902	82168	No	948464
amazon0302	262111	No	1234877
wiki2006A	3148400	No	39383235
soc-LiveJournal1	4847571	No	68993773
wb-edu	9845725	No	57156537
333SP	3712815	Yes	22217266
delaunay-n22	4194304	Yes	25165738
channel-500	4802000	Yes	85362744

Moreover, for the four graphs, we depict in Fig. 4.2 the curves of the relative errors

$$r_k = \frac{|E_{k+1} - E_k|}{E_{k+1}}, \quad k = 1, 2, \ldots, \tag{4.2}$$

obtained from Algorithm 3.1 during iterations. It is seen that the approximation sequences constructed by Algorithm 3.1 are non-decreasing for the undirected graphs, which coincides with the theory established in Theorem 3.2. All these show the rationality of the schemes used in Algorithm 3.1, and this algorithm can be used to estimate energy of large-scale graphs.

Example 4.2. In this example, we show the numerical behavior of Algorithm 2.2 (RSVD-based algorithm), Algorithm 3.1 (non-restated and randomized algorithm) and Algorithm 3.2 (restated and randomized algorithm) for evaluating energy of both large-scale directed and undirected graphs, which are from `https://sparse.tamu.edu/`. Notice that the performance of Algorithm 2.2 relies on the sampling parameter s that is difficult to determine in advance. Thus, for an adjacency matrix of size n, we set the sampling parameter $s = 10\%n$ in Algorithm 2.2. In the non-restarted approach Algorithm 3.1, we choose both the number of columns in the initial matrix $\mathcal{Q}$ and the number of columns added in each time as $s = 100$. The convergence threshold is set to be $\delta = 0.01$. In Algorithm 3.2, we choose the largest number of columns

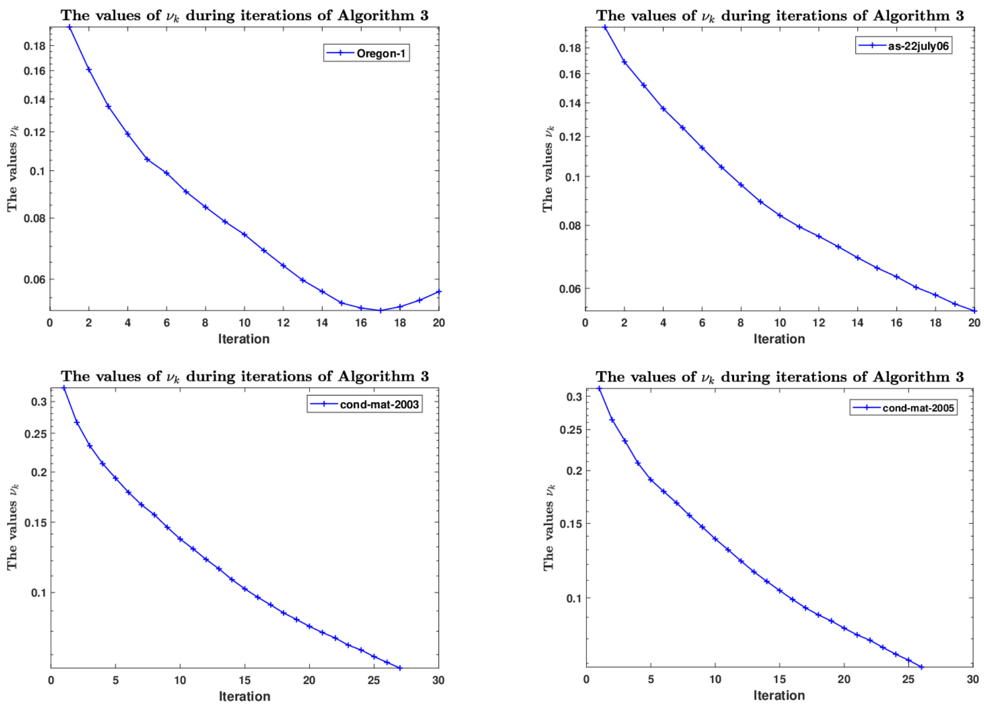

Fig. 4.1. Example 4.1: The values of v_k during iterations of Algorithm 3.1 for the four undirected graphs oregon 1, as-22july06, cond-mat-2003, and cond-mat-2005.

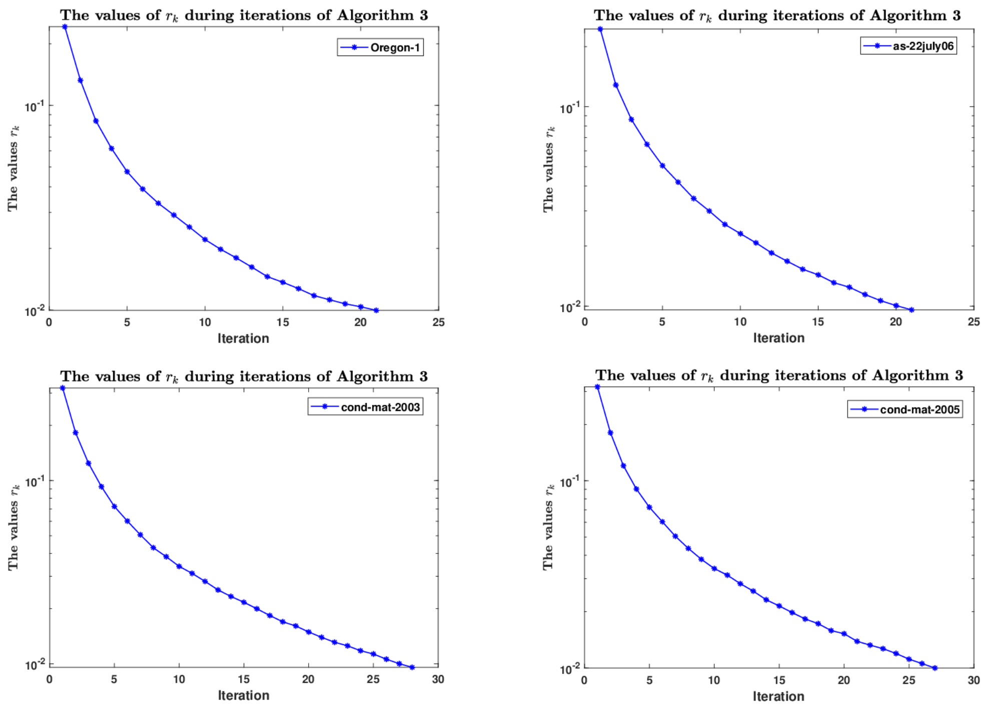

Fig. 4.2. Example 4.1: The values of r_k during iterations of Algorithm 3.1 for the four undirected graphs oregon 1, as-22july06, cond-mat-2003, and cond-mat-2005.

Table 4.2: Lower or upper bounds estimated by using (1.1)-(1.3) for graph energy.

Matrix	$\mathbf{E}(\mathcal{G}) \leq \sqrt{2mn}$	$\mathbf{E}(\mathcal{G}) \geq 2\sqrt{m}$	$\mathbf{E}(\mathcal{G}) \leq n(1+\sqrt{n})/2$
oregon 1	3.280×10^4	4.327×10^2	6.217×10^5
as-22july06	6.670×10^4	6.224×10^2	1.751×10^6
cond-mat-2003	1.223×10^5	9.799×10^2	2.116×10^6
cond-mat-2005	1.685×10^5	1.182×10^3	4.085×10^6
mycielskian16	1.811×10^6	1.155×10^4	5.473×10^6
caidaRouterLevel	6.843×10^5	2.207×10^3	4.224×10^7
dblp-2010	1.026×10^6	2.542×10^3	9.331×10^7
coAuthorsDBLP	1.081×10^6	2.796×10^3	8.192×10^7
delaunay-n19	1.816×10^6	3.547×10^3	1.901×10^8
roadNet-PA	2.593×10^6	3.512×10^3	5.702×10^8
wiki-Vote	4.148×10^4	6.441×10^2	3.821×10^5
p2p-Gnutella04	2.949×10^4	3.999×10^2	5.727×10^5
p2p-Gnutella24	1.888×10^4	5.114×10^2	2.172×10^4
cit-HepPh	1.706×10^5	1.298×10^3	3.227×10^6
cit-HepTh	1.399×10^5	1.118×10^3	2.372×10^6
soc-Slashdot0902	3.948×10^5	1.947×10^3	1.181×10^7
amazon0302	8.045×10^5	2.222×10^3	6.722×10^7
wiki2006A	1.574×10^7	1.255×10^4	2.794×10^9
soc-LiveJournal1	2.586×10^7	1.661×10^4	5.133×10^9
wb-edu	3.354×10^7	1.512×10^4	1.545×10^{10}
333SP	1.264×10^7	9.427×10^3	3.578×10^9
delaunay-n22	1.452×10^7	1.001×10^4	4.297×10^9
channel-500	2.863×10^7	1.847×10^4	5.263×10^9

as $h = 1000$, and set $\delta = 0.01$, $s = 100$. As a comparison, we also run the MATLAB build-in function `eig.m` on these problems. Tables 4.3 and 4.4 list the numerical results of the algorithms on the undirected graphs and the directed graphs, respectively.

Some remarks are in order. First, we observe from Tables 4.3 and 4.4 that for this problem, all the graph energy computed by using Algorithm 2.2 (if available) and Algorithm 3.2 are at the same magnitude as the "real value" obtained from `eig.m`. Moreover, all our proposed three algorithms run much faster than the MATLAB build-in function `eig.m`. Moreover, the evaluations from Algorithms 2.2 and 3.2 are much more effective than the upper and lower bounds given in Table 4.2. Taking the undirected graph cond-mat-2003 as an example, the "true" value obtained from `eig.m` are 4.242×10^4, and the estimated values from Algorithms 2.2 and 3.2 are 2.246×10^4 and 2.586×10^4, respectively, while the bound given in Table 4.2 is $[9.799 \times 10^2,\ 2.116 \times 10^6]$. Thus, our evaluations are preferable to the lower or upper bounds, and the proposed algorithms are promising for evaluating graph energy of large-scale graphs.

Second, for matrices whose size are in the order of 10^5, the CPU timings of Algorithms 3.1 and 3.2 are comparable, and the two algorithms run much faster than Algorithm 2.2. However, the estimated graph energy from Algorithm 3.1 can not be in about the same magnitude as the "real" value computed from `eig.m` in some cases. For example, for undirected graphs such as as-22july06, cond-mat-2003 and cond-mat-2005, the estimations from Algorithm 3.1

Table 4.3: Example 4.2: Estimated graph energy and CPU time in seconds (in brackets) of Algorithms 2.2, 3.1, 3.2 and `eig.m` for undirected graphs, where "–" denotes an algorithm fails to compute the graph energy within 12 hours. Here we also list the size n of the adjacency matrices.

Matrix: n	`eig.m`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
oregon-1: 11492	7.493×10^3 (20.38)	5.009×10^3 (3.44)	3.532×10^3 (6.63)	8.289×10^3 (5.63)
as-22july06: 22963	1.525×10^4 (133.85)	1.026×10^4 (20.55)	$\mathbf{4.692 \times 10^3}$ (9.95)	1.154×10^4 (10.73)
cond-mat-2003: 21163	4.242×10^4 (286.80)	2.246×10^4 (46.71)	$\mathbf{7.710 \times 10^3}$ (20.62)	2.586×10^4 (24.47)
cond-mat-2005: 40421	5.606×10^4 (549.66)	3.071×10^4 (99.53)	$\mathbf{9.005 \times 10^3}$ (26.95)	3.067×10^4 (34.88)
mycielskian16: 49151	3.375×10^5 (1031.37)	2.307×10^5 (931.19)	2.977×10^5 (1064.51)	4.638×10^5 (603.33)
caidaRouterLevel: 192244	– –	1.144×10^5 (7685.52)	1.018×10^4 (179.57)	3.123×10^4 (179.12)
coAuthorsDBLP: 299067	– –	– –	4.086×10^4 (534.21)	1.104×10^5 (549.95)
dblp-2010: 326186	– –	– –	4.014×10^4 (518.52)	1.006×10^5 (441.59)
delaunay-n19: 524288	– –	– –	5.510×10^4 (2522.20)	5.807×10^4 (1412.09)
roadNet-PA: 1090920	– –	– –	9.299×10^3 (5087.10)	1.127×10^4 (2014.51)
333SP: 3712815	– –	– –	– –	5.628×10^4 (10401.93)
delaunay-n22: 4194304	– –	– –	– –	5.818×10^4 (11853.23)
channel-500: 4802000	– –	– –	– –	1.594×10^5 (12131.50)

are about one magnitude lower than the "real" value from `eig.m`. Similar situations occur in some directed graphs p2p-Gnutella04 and p2p-Gnutella24. One reason is that the number of columns s added in each iteration of Algorithm 3.1 is a little small. Specifically, for directed graphs, we see from Table 4.4 that Algorithm 2.2 is about 50 times faster than `eig.m`, which is a great improvement. Although Algorithm 2.2 is efficient to calculate graph energy of undirect and direct graphs of medium size, it does not work when the size of the coefficient matrix is larger than 5×10^4. This is because the size s of the initial block for sampling is very large in this situation, and this algorithm will suffer from heavy storage requirement and computational overhead.

Third, for large adjacency matrices whose size are over 3×10^6, all the algorithms do not work except for the restarted approach Algorithm 3.2. This is because Algorithm 3.2 will be

Table 4.4: Example 4.2: Estimated energy and CPU time in seconds (in brackets) of Algorithms 2.2, 3.1, 3.2 and `eig.m` for directed graphs, where "–" denotes an algorithm fails to compute the graph energy within 12 hours. Here we also list the size n of the adjacency matrices.

Matrix: n	`eig.m`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
wiki-Vote: 8297	2.905×10^3 (148.78)	2.139×10^3 (2.09)	1.882×10^3 (11.69)	4.215×10^3 (6.31)
p2p-Gnutella04: 10879	4.420×10^3 (174.82)	1.067×10^3 (4.30)	$\mathbf{7.824 \times 10^2}$ (10.49)	2.886×10^3 (17.49)
p2p-Gnutella24: 26518	6.451×10^3 (2027.49)	2.538×10^3 (37.60)	$\mathbf{8.838 \times 10^2}$ (38.23)	2.223×10^3 (40.58)
cit-HepTh: 27770	4.308×10^3 (11182.59)	6.378×10^3 (45.27)	3.072×10^3 (13.79)	9.936×10^3 (21.26)
cit-HepPh: 34546	– –	6.968×10^3 (83.12)	3.334×10^3 (19.47)	1.035×10^4 (28.15)
soc-Slashdot0902: 82168	– –	5.861×10^4 (927.44)	1.827×10^4 (182.98)	3.630×10^4 (105.43)
amazon0302: 262111	– –	– –	3.302×10^4 (825.63)	3.173×10^4 (473.10)
wiki2006A: 3148400	– –	– –	– –	4.116×10^4 (9731.82)
soc-LiveJournal1: 4847571	– –	– –	– –	3.566×10^5 (17513.09)
wb-edu: 9845725	– –	– –	– –	1.843×10^5 (18751.74)

restarted as soon as the columns of $\mathcal{Q}$ reaches 1000, and the storage requirement is limited. Furthermore, all the values evaluated by using Algorithm 3.2 meet the upper and lower bounds given in Table 4.2. Thus, Algorithm 3.2 performs the best both in terms of CPU time and the quality of the estimated value, especially when the adjacency matrix is extremely large.

Example 4.3. In this experiment, we run the algorithms on synthetic graphs created by the MATLAB toolbox CONTEST[1)] [64], and show the numerical behavior of them. The MATLAB toolbox CONTEST implements nine popular random network models. Here we make use of three of them [64]:

- The call $A = geo(n, r, m, per, pnorm)$ returns an instance of a geometric random graph, where n is the size of the adjacency matrix A, r specifies the radius, defaulting to $\sqrt{1.44/n}$, m specifies the dimension, defaulting to 2, per is a logical variable specifying whether periodic distance is to be used, defaulting to $per = 0$ (not periodic), and $pnorm$ specifies the Lp-norm to be used, defaulting to 2.

- The call $A = renga(n, lambda, alpha)$ returns an instance of the RENGA model arising from the protein-protein interaction (PPI) networks, with n being the size of the adjacency matrix, $lambda$ defaulting to 0.9 and $alpha$ defaulting to 1.

1) `http://www.maths.strath.ac.uk/research/groups/numerical analysis/contest`

- The call $A = sticky(n, gamma)$ returns an instance of a stickiness graph of dimension n with a scale-free expected degree distribution, with n being the size of the adjacency matrix and *gamma* defaulting to 2.5.

In each model, the adjacency matrix A is a sparse, symmetric, zero-diagonal matrix of dimension n, with n being the number of vertices. Moreover, these models have been designed to ensure that A corresponds to a connected (irreducible) graph with high probability, except for *sticky*, which may produce many small disconnected subgraphs [64]. In Fig. 4.3, we show a spy plot of the three models *geo*, *renga* and *sticky* with $n = 100$.

As is pointed out in [64], one can also create nonsymmetric adjacency matrices by using the MATLAB toolbox CONTEST. More precisely, corresponding to directed networks, we generate nonsymmetric adjacency matrices by combining the upper and lower triangles from two independent samples from the same model. For example, calling $A1 = sticky(n, gamma)$ and $A2 = sticky(n, gamma)$, one could set

$$A = \text{triu}(A1) + \text{tril}(A2) \tag{4.3}$$

as nonsymmetric adjacency matrices corresponding to directed networks.

We run all these functions with default values. In Algorithm 2.2, we use $s = 10\%n$ for randomized singular value decomposition. In the non-restarted approach Algorithm 3.1, we choose the parameters as $\delta = 0.01$ and $s = 100$, while in Algorithm 3.2, we choose the largest number of columns $h = 1000, \delta = 0.01$, and $s = 100$. As a base-line algorithm, we also run the MATLAB build-in function `eig.m` on these problems. Tables 4.5 and 4.6 present the numerical results of the algorithms on the undirected graphs and directed graphs, respectively.

We observe from Tables 4.5 and 4.6 that all our proposed algorithms run much faster than the MATLAB build-in function `eig.m`. Indeed, `eig.m` does not converge within 12 hours when the size of the adjacency matrices is over 2×10^5. Specifically, for the directed graphs, we see from Table 4.6 that Algorithm 2.2 is about 55 times faster than `eig.m`, which is very impressive. Indeed, when the adjacency matrix is relatively small, Algorithm 2.2 performs the best in terms of CPU time, and both Algorithms 2.2 and 3.1 may run faster than Algorithm 3.2. However, for large adjacency matrices whose size is larger than 9×10^4, both Algorithms 3.1 and 3.2 run much faster than Algorithm 2.2, and as the data size increases, this superiority becomes more obvious.

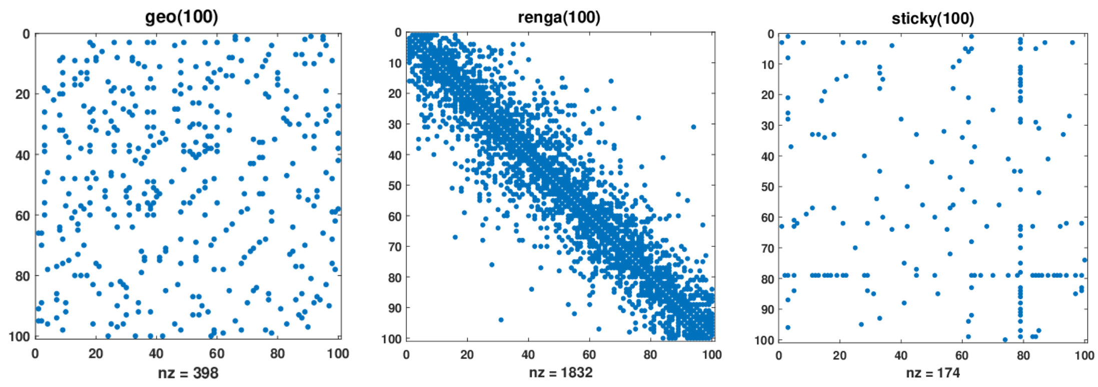

Fig. 4.3. Example 4.3: Spy plots showing nonzero patterns for a 100×100 sample from three models *geo*, *renga* and *sticky* (from left to right).

Table 4.5: Example 4.3: Estimated energy and CPU time in seconds (in brackets) of `eig.m`, Algorithms 2.2, 3.1 and 3.2 for symmetric adjacency matrices created by the MATLAB toolbox CONTEST, where "–" denotes an algorithm fails to compute the graph energy within 12 hours. Here we also list the size n of the adjacency matrices.

Model: n	`eig.m`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
geo: 1×10^4	1.647×10^4 (15.03)	$\mathbf{4.624 \times 10^3}$ (2.37)	$\mathbf{7.034 \times 10^3}$ (10.32)	2.362×10^4 (8.98)
renga: 1×10^4	3.239×10^4 (15.01)	$\mathbf{9.685 \times 10^3}$ (2.71)	1.084×10^4 (7.13)	2.783×10^4 (6.71)
renga: 9×10^4	2.916×10^5 (5248.63)	$\mathbf{8.692 \times 10^4}$ (966.04)	$\mathbf{5.202 \times 10^4}$ (83.76)	1.869×10^5 (174.20)
sticky: 9×10^4	7.020×10^4 (4330.08)	2.807×10^4 (890.78)	$\mathbf{3.914 \times 10^3}$ (56.06)	1.068×10^4 (48.62)
renga: 1×10^5	3.240×10^5 (6552.27)	$\mathbf{9.665 \times 10^4}$ (1242.03)	$\mathbf{5.737 \times 10^4}$ (99.92)	1.879×10^5 (192.96)
sticky: 1×10^5	7.948×10^4 (7178.42)	3.125×10^4 (1212.25)	$\mathbf{4.041 \times 10^3}$ (57.82)	1.117×10^4 (52.45)
renga: 2×10^5	– –	1.933×10^5 (9134.35)	1.035×10^5 (376.59)	1.931×10^5 (397.58)
sticky: 3×10^5	– –	– –	6.119×10^3 (168.19)	1.841×10^4 (178.70)
renga: 4×10^5	– –	– –	1.511×10^5 (1263.19)	1.984×10^5 (833.61)
sticky: 5×10^5	– –	– –	7.177×10^3 (315.74)	2.104×10^4 (317.97)
renga: 6×10^5	– –	– –	1.719×10^5 (2332.04)	1.989×10^5 (1301.62)
sticky: 7×10^5	– –	– –	7.649×10^3 (522.69)	2.192×10^4 (492.75)
renga: 9×10^5	– –	– –	1.837×10^5 (4789.74)	1.993×10^5 (2038.93)
sticky: 9×10^5	– –	– –	8.462×10^3 (661.34)	2.458×10^4 (649.72)
renga: 1×10^6	– –	– –	1.863×10^5 (5743.19)	1.993×10^5 (2278.35)
sticky: 3×10^6	– –	– –	1.332×10^4 (2670.93)	3.968×10^4 (2635.17)

Table 4.6: Example 4.3: Estimated energy and CPU time in seconds (in brackets) of `eig.m`, Algorithms 2.2, 3.1, and 3.2 for nonsymmetric adjacency matrices created by the MATLAB toolbox CONTEST and (4.3), where "–" denotes an algorithm fails to compute the graph energy within 12 hours. Here we also list the size n of the adjacency matrices.

Model: n	`eig.m`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
geo: 1×10^4	1.240×10^4 (167.06)	$\mathbf{1.798\times10^3}$ (3.15)	$\mathbf{2.157\times10^3}$ (3.97)	1.083×10^4 (11.03)
renga: 1×10^4	2.692×10^4 (165.16)	$\mathbf{7.134\times10^3}$ (3.33)	1.013×10^4 (9.64)	1.258×10^4 (3.30)
sticky: 1×10^4	2.299×10^3 (320.77)	$\mathbf{8.226\times10^2}$ (3.03)	$\mathbf{6.381\times10^2}$ (9.05)	1.940×10^3 (8.44)
renga: 9×10^4	– –	6.418×10^4 (1227.15)	5.062×10^4 (85.35)	1.809×10^5 (170.44)
sticky: 9×10^4	– –	7.417×10^3 (1090.82)	9.011×10^2 (70.19)	2.594×10^3 (61.54)
renga: 1×10^5	– –	7.134×10^4 (1574.58)	5.564×10^4 (103.57)	1.817×10^5 (193.68)
sticky: 1×10^5	– –	7.986×10^3 (1498.75)	9.192×10^2 (72.88)	2.813×10^3 (66.99)
renga: 2×10^5	– –	1.426×10^5 (11589.31)	9.814×10^4 (383.93)	1.869×10^5 (404.37)
sticky: 3×10^5	– –	– –	1.193×10^3 (257.66)	4.126×10^3 (277.98)
renga: 4×10^5	– –	– –	1.444×10^5 (1223.92)	1.922×10^5 (842.48)
sticky: 5×10^5	– –	– –	1.327×10^3 (479.43)	4.576×10^3 (491.66)
renga: 6×10^5	– –	– –	1.645×10^5 (2323.83)	1.925×10^5 (1327.46)
sticky: 7×10^5	– –	– –	1.499×10^3 (743.97)	4.969×10^3 (757.72)
renga: 9×10^5	– –	– –	1.777×10^5 (4588.21)	1.928×10^5 (1997.88)
sticky: 9×10^5	– –	– –	1.682×10^3 (957.55)	5.875×10^3 (952.09)
renga: 1×10^6	– –	– –	1.786×10^5 (5690.75)	1.929×10^5 (2361.80)
sticky: 3×10^6	– –	– –	2.524×10^3 (3799.34)	9.019×10^3 (3596.05)

Although Algorithm 3.1 also works for large-scale graphs, we see from Tables 4.5 and 4.6 that this algorithm may suffer from the difficulty of low accuracy, i.e. the magnitude of its results can be lower than the “real” values obtained from `eig.m`. Indeed, the estimations obtained from both Algorithms 2.2 and 3.1 can be one magnitude lower than the “real” value obtained from `eig.m`. As a comparison, all the approximations from Algorithm 3.2 are about in the same order, and they meet the upper and lower bounds shown in Table 4.2. On the other hand, Algorithms 3.1 and 3.2 are comparable for matrices whose size are less than 5×10^5 in terms of CPU time. However, Algorithm 3.2 converges faster than Algorithm 3.1 as the size of the graph is larger than 10^6, and it is suitable to evaluate energy of extremely large-scale graphs.

In summary, all the numerical results show that if the adjacency matrices are of medium size, say, whose order are below 10^4, Algorithms 2.2 and 3.1 often run faster than Algorithm 3.2, however, both of them may suffer from underestimating the graph energy. As a comparison, all the results obtained from Algorithm 3.2 are in the same magnitude as the “exact” value computed by using `eig.m`. When the size of the graph is extremely large, say, over 10^6, all of the three algorithms `eig.m`, Algorithms 2.2 and 3.1 may fail to work, while Algorithm 3.2 works quite well. Therefore, the restarted and randomized algorithm is competitive to evaluate energy of extremely large-scale graphs.

Example 4.4. To our knowledge, MATLAB is not the fastest way for eigenvalue problems, and C, C++ or FORTRAN can be much faster. For instance, ARPACK is a software written in FORTRAN for solving large-scale eigenvalue problems. SPECTRA stands for Sparse Eigenvalue Computation Toolkit as a Redesigned ARPACK (`https://spectralib.org/`)[1]. It is a C++ library for large-scale eigenvalue problems, built on top of Eigen, an open source linear algebra library. SPECTRA calculates a specified number of eigenvalues/eigenvectors of a large square matrix. However, in the computation of graph energy, one has to compute all the eigenvalues of an adjacency matrix. In this case, SPECTRA is unsuitable and may converge slowly.

As a compromise, we present a comparison of numerical results under different programming environment in this example. More precisely, we show the numerical behavior of Algorithms 2.2, 3.1 and 3.2 implemented on Python for evaluating energy of both large-scale directed and undirected graphs. As a comparison, we run Python function `np.linalg.eigvals` on these problems. The data are available from `https://sparse.tamu.edu/`. The settings of the parameters involved in Algorithms 2.2, 3.1 and 3.2 are similar to those in Example 4.2.

To compare with the results of Example 4.2, which are obtained from using MATLAB, we choose some of the same undirected and directed graphs in this example. Table 4.7 lists the numerical results. It is seen from Tables 4.7, 4.3 and 4.4 that, whether we use MATLAB or Python, the values of the graph energy got from the four algorithms are about the same. On the other hand, for both directed and undirected graphs, the running times of Algorithms 2.2, 3.1, and 3.2 on Python are comparable to those of their counterparts on MATLAB. However, for computing graph energy of undirected graphs, we see that the Python function `np.linalg.eigvals` is much slower than the MATLAB built-in function `eig.m`. Indeed, no matter what computing environment is utilized, as the sizes of eigenproblems we solve in our randomized algorithms are much smaller than n, the proposed algorithms will run much faster than the classical algorithms in which all the eigenvalues of a graph need to compute.

Example 4.5. In this example, we pay attention to the energy of large-scale complete and path graphs. The adjacency matrix A of a complete graph K_n with n vertices is an $n \times n$

[1] We thank a reviewer for reminding us of this.

Table 4.7: Example 4.4: Estimated energy and CPU time in seconds (in brackets) of Algorithms 2.2, 3.1, 3.2, and `np.linalg.eigvals` for undirected and directed graphs with Python.

Graphs	Matrix: n	`np.linalg.eigvals`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
Undirected graphs	oregon-1: 11492	7.493×10^3 (178.58)	5.009×10^3 (7.26)	3.517×10^3 (9.04)	8.201×10^3 (6.34)
	as-22july06: 22963	1.525×10^4 (1221.27)	1.026×10^4 (47.32)	$\mathbf{4.704 \times 10^3}$ (13.56)	1.155×10^4 (13.83)
	cond-mat-2003: 21163	4.242×10^4 (3528.27)	2.246×10^4 (114.01)	$\mathbf{7.686 \times 10^3}$ (29.32)	2.584×10^4 (33.26)
	cond-mat-2005: 40421	5.607×10^4 (7436.01)	3.071×10^4 (232.12)	$\mathbf{9.058 \times 10^3}$ (38.25)	3.065×10^4 (43.75)
Directed graphs	wiki-Vote: 8297	2.905×10^3 (190.37)	2.135×10^3 (3.57)	1.933×10^3 (22.26)	4.218×10^3 (8.44)
	p2p-Gnutella04: 10879	4.420×10^3 (177.18)	1.065×10^3 (7.29)	$\mathbf{7.853 \times 10^2}$ (16.00)	2.896×10^3 (22.54)
	p2p-Gnutella24: 26518	6.450×10^3 (1908.61)	2.538×10^3 (78.41)	$\mathbf{8.861 \times 10^2}$ (68.12)	2.223×10^3 (51.39)
	cit-HepTh: 27770	4.281×10^3 (12132.50)	6.376×10^3 (91.54)	3.077×10^3 (18.36)	9.935×10^3 (27.36)

symmetric matrix with all off-diagonal entries equal to 1 and all diagonal entries equal to 0. The eigenvalues of the adjacency matrix are given by [10]:

- A simple (multiplicity one) eigenvalue: $\lambda_1 = n - 1$.
- A repeated eigenvalue of multiplicity $n-1$: $\lambda_i = -1,\ i = 2, 3, \ldots, n$.

Hence, the energy of the complete graph K_n is

$$E(K_n) = 2n - 2. \tag{4.4}$$

On the other hand, the adjacency matrix of a path graph P_n with n vertices is a symmetric tridiagonal matrix, with ones on the sub- and super-diagonals, and zeros elsewhere. Thus, the eigenvalues of the adjacency matrix of P_n are given by [10]

$$\lambda_k = 2\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \ldots, n.$$

The eigenvalues are real, distinct, and lie strictly within the interval $(-2, 2)$. Therefore, the energy of the path graph P_n is equal to

$$E(P_n) = 2\sum_{k=1}^{n}\left|\cos\left(\frac{k\pi}{n+1}\right)\right|. \tag{4.5}$$

Despite the fact that explicit expressions for the energies of complete and path graphs can be readily derived from their known spectral properties, it is interesting to see performances of the new algorithms on the graphs where the graph energy is easy to compute. We employ the proposed algorithms to evaluate the energy estimation on these two type of graphs. The numerical results are summarized in Table 4.8.

Table 4.8: Example 4.5: Estimated energy and CPU time in seconds (in brackets) of Algorithms 2.2, 3.1, 3.2 and `eig.m` or `np.linalg.eigvals` with MATLAB and Python for complete and path graphs, where "–" denotes an algorithm fails to compute the graph energy within 12 hours.

MATLAB					
Graphs	Matrix: n	`eig.m`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
Complete graphs	$n = 2 \times 10^4$	3.999×10^4 (81.96)	2.199×10^4 (46.97)	2.017×10^4 (6.04)	2.017×10^4 (6.12)
	$n = 4 \times 10^4$	7.999×10^4 (517.71)	4.399×10^4 (450.64)	4.016×10^4 (20.94)	4.018×10^4 (21.35)
	$n = 6 \times 10^4$	1.200×10^5 (1649.94)	$\mathbf{6.599} \times \mathbf{10^4}$ (1490.72)	$\mathbf{6.015} \times \mathbf{10^4}$ (45.20)	$\mathbf{6.016} \times \mathbf{10^4}$ (46.51)
	$n = 1 \times 10^5$	2.000×10^5 (6653.72)	1.100×10^5 (6485.21)	1.001×10^5 (119.26)	1.001×10^5 (122.11)
Path graphs	$n = 2 \times 10^4$	2.546×10^4 (6.72)	$\mathbf{3.802} \times \mathbf{10^3}$ (13.55)	$\mathbf{1.376} \times \mathbf{10^3}$ (42.07)	$\mathbf{1.922} \times \mathbf{10^3}$ (25.99)
	$n = 4 \times 10^4$	5.093×10^4 (27.09)	$\mathbf{7.604} \times \mathbf{10^3}$ (98.10)	$\mathbf{1.134} \times \mathbf{10^3}$ (89.59)	$\mathbf{1.401} \times \mathbf{10^3}$ (53.94)
	$n = 6 \times 10^4$	7.639×10^4 (58.37)	1.141×10^4 (321.23)	$\mathbf{9.993} \times \mathbf{10^2}$ (134.26)	$\mathbf{1.152} \times \mathbf{10^3}$ (73.22)
	$n = 1 \times 10^5$	1.273×10^5 (103.35)	$\mathbf{1.901} \times \mathbf{10^4}$ (1341.20)	$\mathbf{8.306} \times \mathbf{10^2}$ (268.31)	$\mathbf{8.951} \times \mathbf{10^2}$ (132.37)
Python					
Graphs	Matrix: n	`np.linalg.eigvals`	Algorithm 2.2	Algorithm 3.1	Algorithm 3.2
Complete graphs	$n = 2 \times 10^4$	3.999×10^4 (719.28)	2.199×10^4 (67.37)	2.019×10^4 (14.16)	2.019×10^4 (14.80)
	$n = 4 \times 10^4$	7.999×10^4 (5724.37)	4.399×10^4 (473.06)	4.018×10^4 (57.98)	4.019×10^4 (61.12)
	$n = 6 \times 10^4$	1.199×10^5 (19380.86)	$\mathbf{6.599} \times \mathbf{10^4}$ (1504.56)	$\mathbf{6.018} \times \mathbf{10^4}$ (132.13)	$\mathbf{6.018} \times \mathbf{10^4}$ (139.14)
	$n = 1 \times 10^5$	– –	1.099×10^5 (6686.17)	1.001×10^5 (372.21)	1.001×10^5 (396.68)
Path graphs	$n = 2 \times 10^4$	2.546×10^4 (1164.68)	$\mathbf{3.802} \times \mathbf{10^3}$ (36.17)	$\mathbf{1.360} \times \mathbf{10^3}$ (101.36)	$\mathbf{1.958} \times \mathbf{10^3}$ (43.10)
	$n = 4 \times 10^4$	5.092×10^4 (7801.71)	$\mathbf{7.604} \times \mathbf{10^3}$ (257.59)	$\mathbf{1.142} \times \mathbf{10^3}$ (241.66)	c$\mathbf{1.446} \times \mathbf{10^3}$ (89.51)
	$n = 6 \times 10^4$	7.639×10^4 (25697.05)	1.141×10^4 (824.69)	$\mathbf{1.001} \times \mathbf{10^3}$ (366.49)	$\mathbf{1.174} \times \mathbf{10^3}$ (123.72)
	$n = 1 \times 10^5$	– –	$\mathbf{1.901} \times \mathbf{10^4}$ (3641.90)	$\mathbf{8.329} \times \mathbf{10^2}$ (620.53)	$\mathbf{9.030} \times \mathbf{10^2}$ (210.62)

In this example, we run the algorithms in different programming environment including MATLAB and Python. As is shown in Table 4.8, the values of graph energy computed by the same algorithms in both programming environments are in the same order of magnitude.

In addition, in terms of CPU time, all algorithms implemented in MATLAB run faster than their counterparts in Python. These observations are consistent with the observations drawn in Example 4.4.

We present some remarks on the numerical results from using MATLAB. On the one hand, for the complete graphs listed in Table 4.8, the graph energies calculated by Algorithms 2.2, 3.1 and 3.2 are all at the same magnitude as the "real value" obtained from `eig.m`, except for the case when $n = 6 \times 10^4$. Furthermore, Algorithms 3.1 and 3.2 are comparable in terms of computation time. Both algorithms run at least 20 times faster than Algorithm 2.2 and `eig.m`. On the other hand, for the path graphs, the numerical results of our three randomized algorithms are not satisfactory. It is observed that the energy values estimated by the new algorithms are all one order of magnitude lower than those computed by `eig.m`. The reason is that the decaying property of the eigenvalues in this type of adjacency matrix is relatively poor.

5. Concluding Remarks

Graph energy is an invariant that is calculated from the eigenvalues or singular values of the adjacency matrix of a graph. Most of the existing work focuses on establishing upper or lower bounds for estimating graph energy. However, these results may not be sharp and can be far away from the exact values. To the best of our knowledge, efficient algorithms for computing energy of extremely large-scale graphs are still lacking.

To fill in this gap, we propose three randomized algorithms based on low-rank approximations of adjacency matrices, for evaluating energy of extremely large-scale undirected and directed graphs. The motivations are that large-scale adjacency matrices are often approximately low (numerical) rank, as well as the estimation and the "exact" graph energy are only necessary to be in the same magnitude in practical use. Theoretical results are established to show the rationality and effectiveness of our new strategies. Numerical experiments are performed on some extremely large-scale real-world and synthetic graphs, which demonstrate the feasibility and efficiency of the proposed algorithms. Moreover, the proposed strategies also apply to the computation of Laplacian energy which comes from graph energy [11–13, 56].

Given prior information on adjacency matrices (including directed/undirected and rank), an interesting question is it possible to automatically select between Algorithms 2.2, 3.1, and 3.2? Indeed, the numerical performances of our proposed algorithms strongly rely on the size the (numerical) rank of the adjacency matrix of a directed/undirected graph. Given the rank r of a adjacency matrix, if the adjacency matrix is of medium size and $r < 5000$, we can choose Algorithm 2.2 for the graph energy. If the matrix is large and $5000 \leq r \leq 10000$, it is preferable to use Algorithm 3.1. When the adjacency matrix is extremely large and $r > 10^4$, Algorithm 3.2 is a good choice.

There are still some problems need to study further. In this paper, we only consider the convergence of Algorithms 3.1 and 3.2 for undirected graphs, however, the convergence of the two algorithms for directed graphs is still an open problem. Moreover, we find that the numerical performances of Algorithms 2.2 and 3.1 strongly rely on the choice of s, i.e. the size of the initial block for sampling, and a skillful choice of this parameter is very interesting. Finally, the proposed algorithms are based on the assumption that adjacency matrices are approximately low (numerical) rank. Although this is generally true in practice, it is not the case for some typical adjacency matrices. For instance, it is known that almost all eigenvalues of a random graph on n vertices are $\mathcal{O}(\sqrt{n}/2)$, and there is no (strongly) decaying property for the eigenvalues or singular values of this type of adjacency matrices. How to deal with these problems deserves further investigation and are definitely a part of our future work.

Acknowledgments. We would like to express our sincere thanks to the anonymous referees and our editor for insightful comments and suggestions that greatly improved the quality of this paper.

This work is supported by the National Natural Science Foundation of China (Grant No. 12271518), by the Fujian Natural Science Foundation (Grant No. 2023J01354), and by the Open Project of Key Laboratory of Data Science and Intelligence Education of the Ministry of Education (Grant No. DSIE202504).

References

[1] C. Adiga and M. Smitha, On maximum degree energy of a graph, *Int. J. Contemp. Math. Sci.*, **4** (2009), 385–396.

[2] X. Ai, Node importance ranking of complex networks with entropy variation, *Entropy*, **19**:7 (2017), 303.

[3] S. Akbari, A. Ghodrati, and M. Hosseinzadeh, Some lower bounds for the energy of graph, *Linear Algebra Appl.*, **591** (2020), 205–214.

[4] N. Alawiah, N. Rad, A. Jahanbani, and H. Kamarulhaili, New upper bounds on the energy of a graph, *MATCH Commun. Math. Comput. Chem.*, **79** (2017), 287–301.

[5] F. Alinaghipour and B. Ahmadi, On the energy of complement of regular line graph, *MATCH Commun. Math. Comput. Chem.*, **60** (2008), 427–434.

[6] G. Arizmendi and O. Arizmendi, The graph energy game, *Discrete Appl. Math.*, **330** (2023), 128–140.

[7] R. Balakrishnan, The energy of a graph, *Linear Algebra Appl.*, **387** (2004), 287–295.

[8] G. Caporossi, D. Cvetković, I. Gutman, and P. Hansen, Variable neighborhood search for extremal graphs. 2. Finding graphs with extremal energy, *J. Chem. Inf. Comput. Sci.*, **39** (1999), 984–996.

[9] C. Coulson, B. O'Leary, and R. Mallion, *Hückel Theory for Organic Chemists*, Academic Press, 1978.

[10] D. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Application*, Academic Press, 1980.

[11] K. Das and S. Mojallal, Upper bounds for the energy of graphs, *MATCH Commun. Math. Comput. Chem.*, **70** (2013), 657–662.

[12] K. Das and S. Mojallal, On Laplacian energy of graphs, *Discrete Math.*, **325** (2014), 52–64.

[13] K. Das, S. Mojallal, and I. Gutman, On energy and Laplacian energy of bipartite graphs, *Appl. Math. Comput.*, **273** (2016), 759–766.

[14] J. Day and W. So, Graph energy change due to edge deletion, *Linear Algebra Appl.*, **428** (2008), 2070–2078.

[15] F. Dopico, J. González, D. Kressner, and V. Simoncini, Projection methods for large T-Sylvester equations, *Math. Comp.*, **85** (2016), 2427–2455.

[16] W. Du, X. Li, and Y. Li, The energy of random graphs, *Linear Algebra Appl.*, **435** (2015), 2334–2346.

[17] N. Entezari, S. Al-Sayouri, A. Darvishzadeh, and E. Papalexakis, All you need is low (rank): Defending against adversarial attacks on graphs, in: *Proceedings of the 13th International Conference on Web Search and Data Mining*, ACM, (2020), 169–177.

[18] E. Estrada, *The Structure of Complex Networks*, Oxford University Press, 2011.

[19] E. Estrada and M. Benzi, What is the meaning of the graph energy after all?, *Discrete Appl. Math.*, **230** (2017), 71–77.

[20] B. Feng and G. Wu, Revisiting the low-rank eigenvalue problem, *Appl. Math. Lett.*, **112** (2021), 106706.

[21] H. Ganie and B. Chat, Bounds for the energy of weighted graphs, *Discrete Appl. Math.*, **268** (2019), 91–101.

[22] H.A. Ganie, U. Samee, S. Pirzada, and A.M. Alghamdi, Bounds for graph energy in terms of vertex covering and clique numbers, *Electron. J. Graph Theory Appl.*, **7** (2019), 315–328.

[23] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 2013.

[24] A. Graovac et al., On statistics of graph energy, *Z. Naturforsch., A: Phys. Sci.*, **56** (2001), 307–311.

[25] I. Gutman, Bounds for total π-electron energy, *Chem. Phys. Lett.*, **24** (1974), 283–285.

[26] I. Gutman, The energy of a graph, *Ber. Math-Statist. Sekt. Forschungsz. Graz*, **103** (1978), 1–22.

[27] I. Gutman, The energy of a graph: Old and new results, in: *Algebraic Combinatorics and Applications*, Springer, (2001), 196–211.

[28] I. Gutman, Bounds for all graph energies, *Chem. Phys. Lett.*, **528** (2012), 72–74.

[29] I. Gutman, S. Firoozabadi and A. Rada, On the energy of regular graphs, *MATCH Commun. Math. Comput. Chem.*, **57** (2007), 435–442.

[30] N. Halko, P. Martinsson, and J. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Rev.*, **53** (2011), 217–288.

[31] N.J. Higham and T. Mary, A new preconditioner that exploits low-rank approximations to factorization error, *SIAM J. Sci. Comput.*, **41** (2019), A59–A82.

[32] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, 2012.

[33] C. Hsieh, K. Chiang, and I. Dhillon, Low rank modeling of signed networks, in: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data MiningAugust*, ACM, (2012), 507–515.

[34] K. Ikramov, Conditions for unique solvability of the matrix equation $AX + X^\top B = C$, *Dokl. Math.*, **81** (2009), 63–65.

[35] A. Ilić, The energy of unitary cayley graphs, *Linear Algebra Appl.*, **431** (2009), 1881–1889.

[36] A. Ilić and M. Bašić, New results on the energy of integral circulant graph, *Appl. Math. Comput.*, **218** (2011), 3470–3482.

[37] G. Indulal and A. Vijayakumar, A note on energy of some graphs, *MATCH Commun. Math. Comput. Chem.*, **59** (2008), 269–274.

[38] A. Jahanbani, Some new lower bounds for energy of graphs, *Appl. Math. Comput.*, **296** (2017), 233–238.

[39] A. Jahanbani, Lower bounds for the energy of graphs, *AKCE Int. J. Graphs Comb.*, **15** (2018), 88–96.

[40] T. Kanada, M. Onuki, and Y. Tanaka, Low-rank sparse decomposition of graph adjacency matrices for extracting clean clusters, in: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, (2018), 1153–1159.

[41] J. Koolen and V. Moulton, Maximal energy graphs, *Adv. Appl. Math.*, **26** (2001), 47–52.

[42] J. Koolen and V. Moulton, Maximal energy bipartite graphs, *Graphs Combin.*, **19** (2003), 131–135.

[43] S. Kumar, P. Sarkar, and A. Pal, A study on the energy of graphs and its applications, *Polycyclic Aromat. Compd.*, **44**:6 (2023), 4127–4136.

[44] A. Langville and C. Meyer, Deeper inside PageRank, *Internet Math.*, **1** (2004), 335–380.

[45] X. Li, Y. Shi, and I. Gutman, *Graph Energy*, Springer Science & Business Media, 2012.

[46] E. Liberty et al., Randomized algorithms for the low-rank approximation of matrices, *Proc. Natl. Acad. Sci. USA*, **104** (2017), 20167–20172.

[47] H. Liu, M. Lu, and F. Tian, Some upper bounds for the energy of graphs, *J. Math. Chem.*, **41** (2007), 45–57.

[48] X. Ma, A low bound on graph energy in terms of minimum degree, *MATCH Commun. Math. Comput. Chem.*, **81** (2019), 393–404.

[49] Y. Ma, *Research on Measurement Methods for Node Importance Based on Graph Energy*, Master

Thesis, Shandong University at Weihai, 2019. (in Chinese)

[50] P. Martinsson, G. Quintana-Orti, and N. Heavner, randUTV: A blocked randomized algorithm for computing a rank-revealing UTV factorization, *arXiv:1703.00998*, 2017.

[51] P. Martinsson, V. Rokhlin, and M. Tygert, A randomized algorithm for the decomposition of matrices, *Appl. Comput. Harmon. Anal.*, **30** (2011), 47–68.

[52] B. McClelland, Properties of the latent roots of a matrix: The estimation of π-electron energies, *J. Chem. Phys.*, **54** (1971), 640–643.

[53] I. Milovanović, E. Milovanović, and I. Gutman, Upper bounds for some graph energies, *Appl. Math. Comput.*, **289** (2016), 435–443.

[54] Y. Nakatsukasa, The low-rank eigenvalue problem, *arXiv:1905.11490v1*, 2020.

[55] M. Oboudi, A new lower bound for the energy of graphs, *Linear Algebra Appl.*, **580** (2015), 384–395.

[56] X. Qi, E. Fuller, Q. Wu, Y. Wu, and C.-Q. Zhang, Laplacian centrality: A new centrality measure for weighted networks, *Inf. Sci.*, **194** (2012), 240–253.

[57] J. Rada and A. Tineo, Upper and lower bounds for the energy of bipartite graphs, *J. Math. Anal. Appl.*, **289** (2004), 446–455.

[58] Y. Saad, *Numerical Algorithms for Large Eigenvalus Problems*, SIAM, 2011.

[59] F. Safaei, J. Kashkooei, and S. Fathi, A method for computing local contributions to graph energy based on Estrada-Benzi approach, *Discrete Appl. Math.*, **260** (2018), 214–226.

[60] A. Samanta and M. Kannan, Bounds and extremal graphs for the energy of complex unit gain graphs, *Linear Algebra Appl.*, **721** (2025), 844–866.

[61] B. Savas and I. Dhillon, Clustered low rank approximation of graphs in information science applications, in: *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM, (2011), 164–174.

[62] I. Shparlinski, On the energy of some circulant graphs, *Linear Algebra Appl.*, **414** (2016), 378–382.

[63] Y. Song, P. Arbelaez, P. Hall, C. Li, and A. Balikai, Finding semantic structures in image hierarchies using Laplacian graph energy, in: *Computer Vision – ECCV 2010. Lecture Notes in Computer Science*, Vol. 6314, Springer, 694–707.

[64] A. Taylor and D. J. Higham, CONTEST: A controllable test matrix toolbox for MATLAB, *ACM Trans. Math. Software*, **35** (2009), 483–490.

[65] I. Triantafillou, On the energy of singular graphs, *Electron. J. Linear Algebra*, **26** (2013), 535–545.

[66] M. Udell and A. Townsend, Why are big data matrices approximately low rank?, *SIAM J. Math. Data Sci.*, **1** (2019), 144–160.

[67] S. Voronin and G. Martinsson, Efficient algorithms for cur and interpolative matrix decompositions, *Adv. Comput. Math.*, **43** (2017), 495–516.

[68] L. Wang and X. Ma, Bounds of graph energy in terms of vertex cover number, *Linear Algebra Appl.*, **517** (2017), 207–216.

[69] D. Wong D, X. Wang, and R. Chu, Lower bounds of graph energy in terms of matching number, *Linear Algebra Appl.*, **549** (2015), 276–286.

[70] L. Wu, X. Ying, and X. Wu, Reconstruction from randomized graph via low rank approximation, in: *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*, SIAM, (2010), 60–71.

[71] Q. Yu, Z. Miao, G. Wu, and Y. Wei, Lumping algorithms for computing Google's PageRank and its derivative, with attention to unreferenced nodes, *Inf. Retrieval*, **15** (2012), 503–526.

[72] B. Zhou and H. Ramane, On upper bounds for energy of bipartite graph, *Indian J. Pure Appl. Math.*, **39** (2008), 483–490.

[73] D. Zügner, A. Akbarnejad, and S. Günnemann, Adversarial attacks on neural networks for graph data, in: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, (2020), 2847–2856.