Fast Connectivity Minimization on Large-Scale Networks

CHEN CHEN, Google Inc.
RUIYUE PENG, Translational MRI, LLC.
LEI YING, University of Michigan, Ann Arbor
HANGHANG TONG, University of Illinois at Urbana-Champaign

The connectivity of networks has been widely studied in many high-impact applications, ranging from immunization, critical infrastructure analysis, social network mining, to bioinformatic system studies. Regardless of the end application domains, connectivity minimization has always been a fundamental task to effectively control the functioning of the underlying system. The combinatorial nature of the connectivity minimization problem imposes an exponential computational complexity to find the optimal solution, which is intractable in large systems. To tackle the computational barrier, greedy algorithm is extensively used to ensure a near-optimal solution by exploiting the diminishing returns property of the problem. Despite the empirical success, the theoretical and algorithmic challenges of the problems still remain wide open. On the theoretical side, the intrinsic hardness and the approximability of the general connectivity minimization problem are still unknown except for a few special cases. On the algorithmic side, existing algorithms are hard to balance between the optimization quality and computational efficiency. In this paper, we address the two challenges by (1) proving that the general connectivity minimization problem is NP-hard and $1 - 1/e$ is the best approximation ratio for any polynomial algorithms, and (2) proposing the algorithm CONTAIN and its variant CONTAIN+ that can well balance optimization effectiveness and computational efficiency for eigen-function based connectivity minimization problems in large networks.

CCS Concepts: • **Mathematics of computing → Paths and connectivity problems**.

Additional Key Words and Phrases: Graph mining, network connectivity

ACM Reference Format:
Chen Chen, Ruiyue Peng, Lei Ying, and Hanghang Tong. 2020. Fast Connectivity Minimization on Large-Scale Networks. *ACM Trans. Knowl. Discov. Data.* 37, 4, Article 111 (August 2020), 25 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Network connectivity has been extensively studied in a myriad of high-impact domains such as immunization, critical infrastructure analysis, social network mining, bioinformatic system studies, etc. Varying by the applications, the connectivity of the network may take different forms as well. For example, in the immunization-related applications, epidemic threshold [45] is commonly used to measure the connectivity for disease propagation; while in the critical infrastructure scenario, natural connectivity [24] is considered as a good measure to evaluate the robustness of the system. Despite the various forms, the connectivity minimization problem has always been a

Authors' addresses: Chen Chen, chenannie45@gmail.com, Google Inc., Mountain View, California, 94043; Ruiyue Peng, rpeng8@asu.edu, Translational MRI, LLC., Los Angeles, California, 90034; Lei Ying, leiying@umich.edu, University of Michigan, Ann Arbor, Ann Arbor, Michigan, 48109; Hanghang Tong, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801.

1556-4681/2020/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

fundamental task in most of the applications, in which a less connected network or subnetwork is more preferred [29]. The primary goal for connectivity minimization is to find a set of nodes/edges whose removal may lead to the destruction of the underlying network. For example, in the critical infrastructure construction scenario, high impact facilities and links can be identified by the connectivity optimization algorithms that minimize the natural connectivity [24] of the network. The selected facilities and links can be viewed as the backbone of the network, which are essential to ensure the full functioning of the entire system. While in the immunization scenario, disease control centers need to vaccine high impact entities and cut down highly contagious connections to prevent the spread of the disease. To identify those high impact entities and contagious connections in the first place, connectivity optimization algorithms can be applied to locate the critical nodes or edges whose removal would minimize the epidemic threshold of the network [45].

The main computation obstacle for the connectivity minimization problems lies in its combinatorial nature. Specifically, for the global connectivity minimization problem, suppose the number of nodes and edges in the network is n and m respectively, then the number of all possible node sets of size k would be $\binom{n}{k}$ and the number of all possible edge sets is $\binom{m}{k}$. Such exponential complexity would make exhaust search intractable even in mid-sized networks.

To reduce the exponential time complexity, existing algorithms predominantly rely on the greedy scheme. Taking the node deletion based connectivity minimization problem for an example, the greedy scheme would iteratively collect the node that has the largest impact on the pre-defined connectivity in the network until the budget is used up. In virtue of the diminishing returns property on a wide-range of the connectivity minimization problems [8], the greedy scheme can secure a near-optimal approximated solution with an approximation ratio of $1 - 1/e$ [42]. A key step in the greedy scheme is to calculate the impact of each candidate node/edge on the given connectivity measure, which often involves eigen-decomposition operations with *polynomial* complexity w.r.t. the size of the network. Obviously, a polynomial algorithm still can not handle large-scale networks efficiently. To further accelerate the algorithm, matrix perturbation based methods are frequently used to approximate the impact of a node/edge [11]. Such approximation algorithms have been proved to scale linearly w.r.t. the network size, while exhibiting empirical superiority over other alternative methods.

Although the above-mentioned methods are empirically effective for some specific connectivity minimization problems, two main challenges for the general connectivity minimization problem still remain largely open. On the theoretical side, the *hardness* of the general connectivity minimization problem has never been systematically justified except for a few special instances (e.g. epidemic threshold [12] and triangle capacity [37]). Furthermore, although the greedy scheme can guarantee a $1 - 1/e$ approximation ratio for the connectivity minimization problem, it still remains unknown if a better approximation ratio can be achieved within polynomial time. On the algorithmic side, exact greedy algorithms often bear polynomial time complexity, which is not scalable in large-scale networks. Although matrix perturbation based approximation methods can simplify the complexity down to the linear scale, their optimization quality is highly dependent on the spectrum of the underlying network (the optimization quality would deteriorate quickly in networks with small eigen-gaps [10, 30]).

In this paper, we address the theoretical and algorithmic challenges of the connectivity minimization problem. The main contributions of the paper can be summarized as follow.

- *Revealing the Fundamental Limits.* We prove that for the connectivity minimization problem on a wide-range of connectivity measures, (1) is NP-hard and (2) $(1 - 1/e)$ is the best approximation ratio for any polynomial algorithms, unless $NP \subseteq DTIME(n^{O(\log \log n)})$[1].

[1] $DTIME(t(n))$: the collection of languages that are decidable by $O(t(n))$ time deterministic Turing machine[47].

- *Developing New Algorithms.* We propose an effective algorithm (CONTAIN) for eigen-function based network connectivity optimization. The centerpieces of the proposed method include (a) an effective impact score approximation method and (b) an efficient eigen-pair update method by leveraging partial QR decomposition and the sparse, low-rank property of the network perturbation matrix. The proposed CONTAIN algorithm bears three distinct advantages over the existing methods, including (1) *effectiveness*, being able to handle small eigen-gap networks, consistently outperforming the state-of-the-art methods over a diverse set of real networks; (2) *scalability*, with a linear complexity w.r.t. the network size; and (3) *generality*, applicable to a variety of different eigen-function based network connectivity measures (e.g., leading eigenvalue, triangle capacity and natural connectivity) as well as network operations (node vs. edge deletion). In addition, we also propose a variation of CONTAIN (CONTAIN+) which can further simply the computational complexity by deriving a closed-form approximation on node/edge impact scores, which saves the step for calculating the updated eigen-pairs.

The rest of the paper is organized as follows. In Section 2, we give the formal definition of the connectivity minimization problem. In Section 3, we prove the hardness and approximability of the problem. In Section 4, we describe the minimization algorithm CONTAIN and its variant CONTAIN+. In Section 5, we evaluate the effectiveness and efficiency of the proposed method on real datasets. In Section 6, we present an overview of the related literature and then conclude in Section 7.

This paper is a significant extension of [9], where we have studied the fundamental limits of connectivity optimization problem and introduced the CONTAIN algorithm. Although the CONTAIN algorithm is efficient for optimizing the eigen-function based connectivity measures as shown in [9], it needs to perform updated eigen-pairs approximation on every candidate node/edge in order to calculate its connectivity impact. Such scheme would make the complexity of the algorithm strongly dependent to the efficiency of the updated eigen-pair approximation step. In this extension, we propose a closed-form approximation algorithm CONTAIN+ which can effectively infer the node/edge connectivity impact without performing eigen-pair approximation, and compare it with CONTAIN regarding their effectiveness and efficiency both theoretically and empirically.

2 PROBLEM DEFINITION

In this section, we formally introduce the network connectivity optimization problem and review the general strategy of greedy algorithms.

Table 1 gives the main symbols used throughout the paper. Following the convention, we use bold upper-case for matrices (e.g. $\mathbf{A}$), bold lower-case for vectors (e.g. $\mathbf{a}$) and calligraphic for sets (e.g. $\mathcal{A}$). We use $\tilde{}$ to denote the notations after node/edge deletion, and Δ to denote the perturbations (e.g. $\Delta\mathbf{A} = \tilde{\mathbf{A}} - \mathbf{A}$). $C(G)$ represents the network connectivity measure to be optimized in G; o indicates an element (a node/edge) in network G; $I(o)$ denotes the impact score of element o on $C(G)$; $\mathbf{\Lambda}$ and $\mathbf{U}$ denote the eigenvalue matrix and eigenvector matrix for the adjacency matrix $\mathbf{A}$ of the network.

2.1 Network Connectivity Measures

Many network connectivity measures can be defined as

$$C(G) = \sum_{\pi \in G} f(\pi) \tag{1}$$

Table 1. Main Symbols.

Symbol	Definition and Description
$G(V, E)$	an undirected network
$\mathbf{A}, \mathbf{B}$	the adjacency matrices (bold upper case)
$\mathbf{a}, \mathbf{b}$	column vectors (bold lower case)
$\mathcal{A}, \mathcal{B}$	sets (calligraphic)
$\mathbf{A}(i, j)$	the element at the i^{th} row and the j^{th} column in $\mathbf{A}$
$\mathbf{a}(i)$	the i^{th} element of vector $\mathbf{a}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\Delta\mathbf{A}$	perturbation of $\mathbf{A}$
$\tilde{\mathbf{A}}$	the adjacency matrix after node/edge deletion on $\mathbf{A}$
m, n	number of edges and nodes in network G
$C(G)$	connectivity measure of network G
$C_{\mathcal{T}}(G)$	the local connectivity of subgraph $\mathcal{T}$ on network G
$F(\mathbf{\Lambda}^{(r)})$	associated eigen-function for $C(G)$
o	a network element in G (a node/edge)
$I(o)$	connectivity impact score of o on $C(G)$
$I_{\mathcal{T}}(o)$	local connectivity impact score of o on $C_{\mathcal{T}}(G)$
$\lambda, \mathbf{u}$	the leading eigenvalue and eigenvector of $\mathbf{A}$ (in magnitude)
$\mathbf{\Lambda}, \mathbf{U}$	the eigenvalue and eigenvector matrix of $\mathbf{A}$
$\mathbf{\Lambda}^{(r)}, \mathbf{U}^{(r)}$	the top-r eigen-pairs of $\mathbf{A}$ (in magnitude)
k	the budget

with

$$f(\pi) = \begin{cases} w_\pi > 0 & \text{if } \pi \text{ is a valid subgraph} \\ 0 & \text{otherwise.} \end{cases}$$

where π is a subgraph of G, f is a non-negative function that maps any subgraph in G to a non-negative real number w_π (i.e. $f : \pi \rightarrow \mathbb{R}^+$), which can be customized by different application scenarios. In [8], various weight functions are discussed for different connectivity measures. Specifically, we have $f(\phi) = 0$ for empty set ϕ; when $f(\pi) > 0$, we call subgraph π as a *valid subgraph*. In other words, the network connectivity $C(G)$ can be viewed as a weighted aggregation of the connectivities of all valid subgraphs in the network.

By choosing an appropriate $f()$ function (please refer to [8] for details), Eq. (1) includes several prevalent network connectivity measures, e.g., path capacity (i.e., sum of weighed paths in the network, which is in close relation to the epidemic threshold), triangle capacity (i.e., total number of triangles in the network, which is rooted in social balance theory) and natural connectivity (i.e., sum of weighted loops in the network, which is closely related to network robustness). In terms of computation, it is often much more efficient to either approximate or compute these connectivity measures by the associated eigen-function $F(\mathbf{\Lambda}^{(r)})$, where $\mathbf{\Lambda}^{(r)}$ represents the top-r eigenvalues of $\mathbf{A}$. For example, the path capacity converges to the leading eigenvalue of the adjacency matrix of the network [5], the triangle capacity can be approximated by the sum of cubes of the eigenvalues [51], and the natural connectivity is calculated by the sum of exponentials of the eigenvalues [24]. On the other hand, the number of connected components in the network can be calculated by the number of zero eigenvalues of its Laplacian matrix [48], which can also be viewed as an eigen-function of the network.

It is worth noting that the connectivity measure defined in Eq. (1) can also be extended to measure the local connectivity of a subset of nodes $\mathcal{T}$, where we define $f(\pi) > 0$ iff π is incident to the node set $\mathcal{T}$. Local connectivity plays an important role when the connectivity analysis is targeted on a specific subset of nodes. For example, in the targeted immunization scenario, the key goal is to protect the selected target entities from disease infection. Constrained by the budget, we may only allowed to cut part of the contagious connections around the target entities. In this setting, it is crucial to identify key connections within the budget so that the connectivity of the target entities is minimized.

2.2 Network Connectivity Minimization

With the network connectivity measure in Eq. (1), we formally define network connectivity optimization problem as follows.

Problem 1. **Network Connectivity Minimization (NETCOM)**

Given: *(1) a network G; (2) a connectivity mapping function $f : \pi \rightarrow \mathbb{R}^+$ which defines $C(G)$; (3) a type of network operation (node deletion vs. edge deletion) and (4) an integer budget k with $1 < k < \min\{|\mathcal{S}_\pi|, K\}$ where $\mathcal{S}_\pi = \{\pi | f(\pi) > 0\}$ denotes the set of valid subgraphs and K denotes the number of valid network elements.*

Output: *a set of network elements $\mathcal{X}$ of size k, whose removal from G would minimize connectivity $C(G)$.*

It is worth noting that depending on the definition of $C(G)$, the valid subgraphs in $\mathcal{S}_\pi$ may have various structures. In the triangle minimization scenario, $\mathcal{S}_\pi$ contains all the triangles in the network. When the valid subgraph shares the same form as the operation type (i.e. a valid subgraph is a single node in node-level operation scenario, or a valid subgraph is an edge in edge-level operation scenario), we call this kind of valid subgraphs as *singletons*. In Problem 1, we also require that the budget $1 < k < \min\{|\mathcal{S}_\pi|, K\}$. This is a fairly generic constraint which can be easily met. For example, for the node deletion operation, the set of valid network elements is simply the entire node set of the input network (i.e., $K = n$); for a connected network with its connectivity measure $C(G)$ defined as the path capacity, we have that $|\mathcal{S}_\pi| > n$. Therefore, the above constraint simply means that we cannot delete all the nodes from the input network, which would make the problem trivial. On the other end of the spectrum, we require that the budget $k > 1$. Otherwise (with $k = 1$), the problem can be easily solved in polynomial time (e.g., by choosing the valid network element with the largest impact score). Problem 1 provides a general definition of the network connectivity optimization problem, which can be in turn instantiated into different instances, depending on (1) the specific choice of the connectivity measure $C(G)$ (or equivalently the choice of the $f()$ function), and (2) the type of network operation (node deletion vs. edge deletion). For example, in the robustness analysis of the power grid, we might choose the natural connectivity as $C(G)$ to evaluate the robustness of the system, and we are interested in identifying k most critical power transmission lines whose failure would cause a cascading failure of the entire grid. To abstract it as a network connectivity optimization problem, we have the input network set as the topological structure of the power grid; the connectivity to optimize as the natural connectivity; the operation type as edge deletion; and the valid network elements as all the edges (i.e., $K = m$ in this case).

The NETCOM problem can be easily extended to local connectivity measures. Specifically, the local connectivity minimization problem can be defined as follows.

Problem 2. **Local Connectivity Minimization**

Given: *(1) a network G; (2) a subset of target nodes $\mathcal{T}$; (3) a connectivity mapping function $f : \pi \rightarrow \mathbb{R}^+$ which defines the local connectivity measure $C_\mathcal{T}(G)$; (3) a type of network operation (node deletion vs. edge deletion) and (4) an integer budget k with $1 < k < \min\{|\mathcal{S}_\pi|, K\}$ where $\mathcal{S}_\pi = \{\pi | f(\pi) > 0\}$ denotes the set of valid subgraphs and K denotes the number of valid network elements.*

Output: *a set of network elements $\mathcal{X}$ of size k with $\mathcal{X} \cap \mathcal{T} = \Phi$, whose removal from G would minimize connectivity $C_\mathcal{T}(G)$.*

Note that in Problem 2, the restriction $\mathcal{X} \cap \mathcal{T} = \Phi$ on $\mathcal{X}$ is used to avoid the trivial solution under node deletion operations, in which target nodes $\mathcal{T}$ are removed for local connectivity minimization.

2.3 Greedy Strategy for NETCOM

Due to the combinatorial nature of Problem 1, it is computationally infeasible to solve it in a brute-force manner. Thanks to the diminishing returns property of NETCOM, the greedy strategy has become a prevalent choice for solving Problem 1 with a guaranteed $(1 - 1/e)$ approximation ratio. For the ease of following discussions, we present the outline of such greedy strategy in Algorithm 1. In Algorithm 1, the solution set $\mathcal{X}$ is initialized with an empty set. At each iteration (step 2 to step 8), the element (a node or an edge) with the highest impact score is added to the solution set $\mathcal{X}$ until the budget is reached. The returned solution set $\mathcal{X}$ in step 9 guarantees a $(1 - 1/e)$ approximation ratio. For more details and proofs, please refer to [8].

Algorithm 1 A Generic Greedy Strategy for NETCOM [8]

Input: (1) A network G; (2) a connectivity mapping function $f : \pi \rightarrow \mathbb{R}^+$ which defines $C(G)$; (3) a type of network operation and (4) a positive integer k
Output: a set of network elements $\mathcal{X}$ of size k.

1: initialize $\mathcal{X}$ to be empty
2: **for** $i = 1$ to k **do**
3: **for** each valid network element o in G **do**
4: calculate $I(o) \leftarrow C(G) - C(G \setminus \{o\})$
5: **end for**
6: add the element $\tilde{o} = \text{argmax}_o I(o)$ to $\mathcal{X}$
7: remove the element $\{\tilde{o}\}$ from network G
8: **end for**
9: return $\mathcal{X}$

3 FUNDAMENTAL LIMITS

In this section, we start with detailing the theoretic challenges of the network connectivity optimization (NETCOM) problem, and then reveal two fundamental limits, including its hardness and its approximability.

3.1 Theoretic Challenges of NETCOM

The first theoretic challenge of NETCOM lies in its hardness. Since the NETCOM problem has various instances, intuitively, the hardness of those instances might vary dramatically from one to another. For example, if the elements in the valid subgraph set $\mathcal{S}_\pi$ are all singletons w.r.t. the corresponding operation type (i.e., $\mathcal{S}_\pi$ is the node set of the input network for the node-level optimization problem, or $\mathcal{S}_\pi$ is the edge set for the edge-level optimization problem), we can simply choose the top–k nodes/edges with the highest $f(\pi)$ scores, which immediately gives the optimal solution. However, if NETCOM is instantiated as an edge minimization problem under node deletion operations (i.e. the valid subgraph $\mathcal{S}_\pi$ consists of all the edges, the valid network element set is the entire node set) , the problem would become the (weighted) max-k vertex cover problem, which is known to be NP-hard. Such observations naturally give rise to the following question, *what is the key intrinsic property of valid subgraph set $\mathcal{S}_\pi$ in conjunction with the network operation type that determines whether or not the corresponding NETCOM instance is polynomially solvable?* To date, the hardness of the general NETCOM problem has largely remained unvalidated, except for a few special instances (see Section 6 for details). The second theoretic challenge of NETCOM lies in its approximability. The greedy algorithm outlined in Section 2 has a provable $(1 - 1/e)$ approximation ratio [8]. However, we still do not know if such an approximation ratio

is *optimal*. In other words, it remains unknown if there exists any polynomial algorithm with an approximation ratio better than $(1 - 1/e)$ for NETCOM.

3.2 Fundamental Limit #1: NP-Hardness

We reveal the hardness result of the NETCOM problem in Theorem 1. It states that the NETCOM problem defined in Problem 1 are in general NP-hard, unless the valid subgraphs in set $\mathcal{S}_\pi$ are mutually independent to each other[2].

Theorem 1. **NP-Hardness of NETCOM.** *The NETCOM problem with non-independent valid subgraphs in Problem 1 is NP-hard.*

Proof. As NETCOM problem admits two possible network operations, including node deletions and edge deletions, we present our proof for each scenario in the following two lemmas. Lemma 1 together with Lemma 2 would prove that NETCOM problem is NP-hard. □

lemma 1. *The k-node connectivity minimization problem is NP-hard.*

Proof. By Eq. (1), the connectivity of network G is defined as $C(G) = \sum_{\pi \in G} f(\pi)$. We have function f defined as

$$f(\pi) = \begin{cases} w_\pi > 0 & \text{if } \pi \text{ is a valid subgraph} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Hence, we formulate the k-node minimization problem as follows.

Problem 3. k-Node Minimization Problem: $NodeMin(G, k)$

Given: *(1) A network $G =< V, E >$; (2) the connectivity function f as defined in Eq.* (2)*; and (3) the budget k.*

Output: *A set with k nodes $V' \subseteq V$, such that $C(G \setminus V')$ (i.e. the connectivity in $G \setminus V'$) is minimized.*

Here we prove that $NodeMin(G, k)$ is NP-hard by constructing a polynomial reduction from a well-known NP-hard problem, the max k-coverage problem ($MaxCover(n, m, k)$) [27]. The max k-coverage problem $MaxCover(n, m, k)$ is defined as follows.

Problem 4. Max k-Coverage Problem: $MaxCover(n, m, k)$

Given: *(1) the universal set of n elements $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$; (2) a collection $\mathcal{S} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ of m distinct subsets of $\mathcal{U}$, which are not mutually exclusive; (3) the non-negative weights $\mathcal{W} = \{w_i, \ldots, w_n\}$ associated to the corresponding elements in $\mathcal{U}$; and (4) a positive integer k.*

Output: *A set $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| \leq k$, s.t. $\sum_{e_i \in \mathcal{U}'} w_i$ is maximized, where $\mathcal{U}'$ is the set of elements covered by the sets in $\mathcal{S}'$.*

We aim to prove that $MaxCover(n, m, k)$ is polynomially reducible to $NodeMin(G, k)$ (or equivalently $MaxCover(n, m, k) \leq_p NodeMin(G, k)$). Without loss of generality, we assume that $1 < k < m$. The rationality behind this assumption is that when $k = 1$, $MaxCover(n, m, 1)$ can be trivially solved by picking the set in $\mathcal{S}$ that contains the elements with the largest weight sum (i.e. S'=$\{\arg\max_{\mathcal{B} \in \mathcal{S}} \sum_{e_i \in \mathcal{B}} w_i\}$); while for $k \geq m$, we may just take all the subsets in $\mathcal{S}$ into $\mathcal{S}$' to guarantee a maximum coverage.

Given an instance of $MaxCover(n, m, k)$ with $1 < k < m$, we can construct a network G with m nodes, each corresponds to one subset in $\mathcal{S}$. For each element e_i in $MaxCover(n, m, k)$, we construct a valid subgraph G_i as follows. First, we scan set $\mathcal{S}$ and obtain all the sets $\{\mathcal{B}_1^i, \ldots, \mathcal{B}_l^i\} \subseteq \mathcal{S}$ that contain element e_i. Then we map $\{\mathcal{B}_1^i, \ldots, \mathcal{B}_l^i\}$ into the nodes in G and get the corresponding l

[2]Two valid subgraphs are independent to each other if they do not have common valid network element.

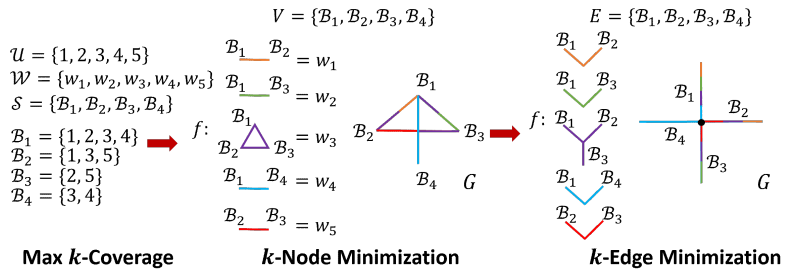


Fig. 1. An illustration of polynomial reduction from Max k-Coverage problem.

nodes. By connecting those l node with edges, we get a subgraph G_i in G with connectivity score $f(G_i) = w_i$. In this way, removing any nodes from G_i would break the subgraph completeness. Repeating the above process for all the elements in $\mathcal{U}$, the final graph we get is $G = G_1 \cup \ldots \cup G_n$, and the connectivity function is defined as $f(G_i) = w_i$. Since the sets in $\mathcal{S}$ are distinct and not mutually exclusive, the resulting valid subgraphs are guaranteed to be non-independent. Therefore, the solution of $MaxCover(n, m, k)$ would be equivalent to the solution of $NodeMin(G, k)$, which completes the proof. □

Figure 1 gives an illustration of the reduction from an instance of $MaxCover(n, m, k)$ to an instance of $NodeMin(G, k)$, in which different valid subgraphs are marked with different colors. Edges with multiple colors indicate that they are involved in multiple valid subgraphs.

LEMMA 2. *The k-edge connectivity minimization is NP-hard.*

PROOF. We still use the connectivity measure defined Eq. (2) to complete the proof. The corresponding k-edge minimization problem can be defined as follows.

PROBLEM 5. k-Edge Minimization Problem ($EdgeMin(G, k)$)

Given: *(1) A network $G =< V, E >$; (2) the connectivity function f as defined in Eq. (2); and (3) the budget k.*

Output: *A set with k edges $E' \subseteq E$, such that $C(G \setminus E')$ (i.e. the connectivity in $G \setminus E'$) is minimized.*

We prove that $EdgeMin(G, k)$ is NP-hard by constructing a polynomial reduction from the $MaxCover(n, m, k)$ problem. Similar to the rationale in the previous proof, we assume that $1 < k < m$.

Given an instance of $MaxCover(n, m, k)$, we construct a m-edge star-shaped network G (i.e. all the m edges share one common endpoint). Specifically, each subset in $\mathcal{S}$ corresponds to an edge in G and each element e_i in $\mathcal{U}$ represents a valid subgraph G_i in the connectivity function. To construct subgraph G_i, we first locate all the subsets in $\mathcal{S}$ that contain element e_i, and map them into the corresponding edges in G. Then the sub-star formed by those edges can be viewed as a valid subgraph G_i with $f(G_i) = w_i$. The removal of any edge from G_i would destroy the completeness of the valid subgraph. Consequently, we have n valid subgraphs in G. Similarly, as the sets in $\mathcal{S}$ are distinct and not mutually exclusive, the resulting valid subgraphs are guaranteed to be non-independent. Therefore, the solution of $MaxCover(n, m, k)$ would be equivalent to the solution of $EdgeMin(G, k)$, which completes the proof. □

Figure 1 gives an illustration of the reduction from an instance of $MaxCover(n, m, k)$ to $EdgeMin(G, k)$. Again, edges with multiple colors indicate their participation in multiple valid subgraphs.

3.3 Fundamental Limit #2: Approximability

Based on the hardness result of NETCOM, we further reveal the approximability of NETCOM in Theorem 2, which says that $(1-1/e)$ is indeed the best approximation ratio a polynomial algorithm can achieve unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).

Theorem 2. **Approximability of NETCOM.** *$(1-1/e)$ is the best approximation ratio for the NETCOM problem with non-independent valid subgraphs in polynomial time, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).*

Proof. We prove this by contradiction. In the proof of Theorem 1, we show that max k-Coverage problem is polynomially reducible to the NETCOM problem, which implies that if there is an α-approximation algorithm that can solve NETCOM in polynomial time with $\alpha > (1-1/e)$, there will be an α-approximation algorithm for max k-Coverage as well. However, it has been proved in [26] that the maximum k-coverage problem can not be approximated with a factor better than $(1-1/e)$ unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$), which contradicts with our assumption. Hence, we conclude that there is no polynomial algorithm for the NETCOM problem with an approximation ratio greater than $(1-1/e)$, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$). □

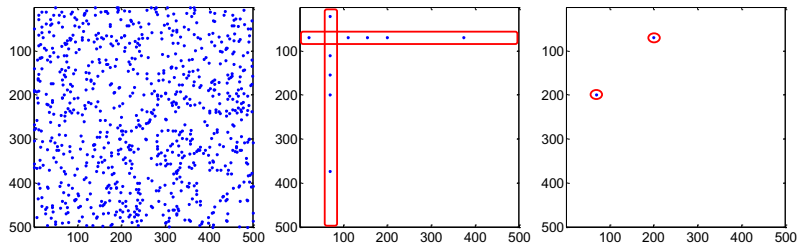
Since the greedy strategy in Algorithm 1 guarantees a $(1-1/e)$ approximation ratio, Theorem 2 implies that the greedy algorithm is the best polynomial algorithm for NETCOM in terms of its approximation ratio unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).

4 ALGORITHM AND ANALYSIS

In this section, we start with detailing the algorithmic challenges of the network connectivity optimization (NETCOM) problem, and then present an effective algorithm, followed by some analysis in terms of its effectiveness and efficiency.

4.1 Algorithmic Challenges of NETCOM

In the greedy strategy (Algorithm 1), a key step is to calculate the impact score of each network element, i.e., $I(o) = C(G) - C(G \setminus \{o\})$ (Step 4). As we have mentioned in Section 2, the network connectivity measures $C(G)$ studied in this paper can be calculated or well approximated by a function of top-r eigenvalues of its adjacency matrix (i.e. $C(G) = F(\mathbf{\Lambda}^{(r)})$, where $F()$ is the function of eigenvalues). Therefore, the core step of calculating $I(o)$ is to compute $\mathbf{\Lambda}^{(r)}$ on $G \setminus \{o\}$, which takes $O(m)$ time (say using the classic Lanczos method). Consequently, simply recomputing $C(G \setminus \{o\})$ for each network element from scratch would make the entire algorithm $O(mn)$ for node-level optimization problems and $O(m^2)$ for edge-level optimization problems, neither of which is computationally feasible in large networks. To address this issue, existing literature often resorts to matrix perturbation theory. Its key idea is to view the deletion of a network element o as a perturbation to the original network (i.e. $\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$). Thus, the new eigenvalues (and hence the new connectivity measure $C(G \setminus \{o\})$) can be approximated from the eigenvalues and eigenvectors of the original network in constant time, making the overall algorithm linear w.r.t. the size of the input network [11, 12]. However, for networks with small eigen-gaps, the approximation accuracy of matrix perturbation theory based methods might deteriorate quickly, if not collapse at all. This issue might persist even if we switch to computationally more expensive high-order matrix perturbation theory [11, 12]. Thus, the main algorithmic challenge is how to accurately approximate the top-r eigenvalues of the input network after a node/edge deletion.



(a) Random (b) Node Deletion (c) Edge Deletion

Fig. 2. Illustrations and comparison of random perturbation matrix (a), which is dense and potentially full-rank, vs. perturbation matrices by node deletion (b) and edge deletion (c), both of which are sparse and low-rank.

4.2 CONTAIN: The Proposed Algorithm

We propose a new updating algorithm for the top-r eigenvalues after node/edge deletion. In order to maintain the linear complexity of the entire greedy algorithm, we seek to update the top-r eigenvalues in *constant* time for each node/edge deletion operation.

Our key observation is as follows. In classic matrix perturbation theory (whether the first-order matrix perturbation theory or its high-order variants), a fundamental assumption is that the perturbation matrix $\Delta\mathbf{A}$ is a random matrix whose spectrum is well-bounded as illustrated in Figure 2(a). However, such assumption does not hold in the node/edge deletion scenario (Figure 2(b) and (c)), in which the perturbation matrix $\Delta\mathbf{A}$ is *sparse* and *low-rank*. Armed with this observation, we propose an effective eigen-pair update algorithm for node/edge deletion based on partial-QR decomposition. Unlike matrix perturbation based methods, which would inevitably introduce approximation error in the procedure, the proposed algorithm does not introduce any additional error when computing the impact score $I(o)$, and it runs in constant time for each node/edge operation.

The proposed CONTAIN algorithm is presented in Algorithm 2. Overall, it follows the greedy strategy (Algorithm 1). In detail, We first compute the top-r eigen-pairs of the network and compute the connectivity score of the original network (step 2-3). From step 4 to step 19, we iteratively select the element with the highest impact score. When evaluating the impact of each valid element, we first construct the perturbation matrix $\Delta\mathbf{A}$ for the corresponding element and then perform eigen decomposition on it (step 6-7). Particularly, for node deletion operation, suppose the removed node v has a set of neighbor nodes $\mathcal{N}_v$. Then the resulting perturbation matrix $\Delta\mathbf{A}$ has $\Delta\mathbf{A}(v, \mathcal{N}_v) = \Delta\mathbf{A}(\mathcal{N}_v, v) = -1$, which is a rank-2 sparse matrix. Therefore, $\mathbf{U}_\Delta$ and $\mathbf{\Lambda}_\Delta$ can be directly expressed as an $n \times 2$ matrix and a 2×2 matrix respectively. Moreover, let $n_v = |\mathcal{N}_v|$, the non-zero entries in the eigenvector matrix of $\Delta\mathbf{A}$ are

$$\begin{aligned} \mathbf{U}_\Delta(v, 1) &= \frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(v, 2) = \frac{1}{\sqrt{2}} \\ \mathbf{U}_\Delta(\mathcal{N}_v, 1) &= -\frac{1}{\sqrt{2n_v}}, \mathbf{U}_\Delta(\mathcal{N}_v, 2) = \frac{1}{\sqrt{2n_v}} \end{aligned} \tag{3}$$

and the eigenvalue matrix of $\Delta\mathbf{A}$ is

$$\mathbf{\Lambda}_\Delta = \begin{bmatrix} \sqrt{n_v} & 0 \\ 0 & -\sqrt{n_v} \end{bmatrix} \tag{4}$$

Algorithm 2 The CONTAIN Algorithm

Input: (1) The adjacency matrix of the network $\mathbf{A}$; (2) the associated eigen-function $F()$ for connectivity $C(G)$; (3) rank r; (4) the network operation (node vs. edge deletion); and (5) a positive integer k.

Output: a set of network elements $\mathcal{X}$ of size k.

1: initialize $\mathcal{X}$ to be empty
2: compute $[\mathbf{U}^{(r)}, \mathbf{\Lambda}^{(r)}] \leftarrow$top-$r$ eigen-pairs of matrix $\mathbf{A}$
3: compute $C(G) \leftarrow F(\mathbf{\Lambda}^{(r)})$
4: **for** $i = 1$ to k **do**
5: **for** each valid element o in G **do**
6: $\Delta\mathbf{A} \leftarrow$ the perturbation matrix by element o's deletion
7: $[\mathbf{U}_\Delta, \mathbf{\Lambda}_\Delta] \leftarrow$eigen-pairs of $\Delta\mathbf{A}$
8: $\mathbf{R} \leftarrow$upper triangular matrix from $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$'s partial-QR decomposition
9: $\mathbf{\Lambda}_z \leftarrow$ eigenvalues of $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$
10: compute $I(o) \leftarrow C(G) - F(\mathbf{\Lambda}_z)$
11: **end for**
12: add $\tilde{o} = \text{argmax}_o I(o)$ to $\mathcal{X}$
13: update $C(G) \leftarrow C(G) - I(\tilde{o})$ and set $I(\tilde{o}) \leftarrow -1$
14: $\Delta\mathbf{A} \leftarrow$ the perturbation matrix by element $\tilde{o}$'s deletion
15: $[\mathbf{U}_\Delta, \mathbf{\Lambda}_\Delta] \leftarrow$ eigen-pairs of $\Delta\mathbf{A}$
16: $[\mathbf{Q}, \mathbf{R}] \leftarrow$ partial-QR decomposition of $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$
17: $[\mathbf{U}_z, \mathbf{\Lambda}_z] \leftarrow$ eigen-pairs of $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$
18: update $\mathbf{U}^{(r)} \leftarrow (\mathbf{Q}\mathbf{U}_z)^{(r)}$, $\mathbf{\Lambda}^{(r)} \leftarrow \mathbf{\Lambda}_z^{(r)}$, $\mathbf{A} \leftarrow \mathbf{A} + \Delta\mathbf{A}$
19: **end for**
20: return $\mathcal{X}$

In the edge deletion scenario, the perturbation matrix $\Delta\mathbf{A}$ corresponding to the removal of edge $\langle u, v\rangle$ has only two non-zero entries $\Delta\mathbf{A}(u, v) = \Delta\mathbf{A}(v, u) = -1$ and $u \neq v$, which is also a rank-2 matrix. Then, the only non-zero entries in $\mathbf{U}_\Delta$ are

$$\begin{aligned} \mathbf{U}_\Delta(u, 1) &= \frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(u, 2) = \frac{1}{\sqrt{2}} \\ \mathbf{U}_\Delta(v, 1) &= -\frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(v, 2) = \frac{1}{\sqrt{2}} \end{aligned} \tag{5}$$

And the eigenvalue matrix $\mathbf{\Lambda}_\Delta$ is

$$\mathbf{\Lambda}_\Delta = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{6}$$

With the eigenvector matrix of $\Delta\mathbf{A}$, we proceed to perform partial-QR decomposition on $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$ in step 8. As $\mathbf{U}^{(r)}$ is already orthonormal, the $\mathbf{Q}$ matrix in the decomposition can be written as the concatenation of $\mathbf{U}^{(r)}$ and two orthogonal vectors in unit length as follows

$$\mathbf{Q} = [\mathbf{U}^{(r)}, \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|}, \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|}] \tag{7}$$

By the Gram-Schmidt process, we have

$$\begin{aligned}\mathbf{q}_1 &= \mathbf{U}_\Delta(:,1) - \mathbf{U}^{(r)}\mathbf{r}_1 \\ \mathbf{q}_2 &= \mathbf{U}_\Delta(:,2) - \mathbf{U}^{(r)}\mathbf{r}_2 + \mathbf{r}_1'\mathbf{r}_2\frac{\mathbf{q}_1}{\|\mathbf{q}_1\|^2}\end{aligned} \tag{8}$$

where $\mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,1)$ and $\mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,2)$.
For node-level operations, we have

$$\begin{aligned}\mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,1) &= \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(v,:) - \frac{1}{\sqrt{n_v}}\sum_{u\in\mathcal{N}_v}\mathbf{U}^{(r)}(u,:))' \\ \mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,2) &= \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(v,:) + \frac{1}{\sqrt{n_v}}\sum_{u\in\mathcal{N}_v}\mathbf{U}^{(r)}(u,:))'\end{aligned} \tag{9}$$

While for edge-level operations, we have

$$\begin{aligned}\mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,1) &= \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(u,:) - \mathbf{U}^{(r)}(v,:))' \\ \mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:,2) &= \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(u,:) + \mathbf{U}^{(r)}(v,:))'\end{aligned} \tag{10}$$

Correspondingly, the upper-triangular matrix $\mathbf{R}$ can be written as

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{r}_1 & \mathbf{r}_2 \\ 0 & \|\mathbf{q}_1\| & -\frac{\mathbf{r}_1'\mathbf{r}_2}{\|\mathbf{q}_1\|} \\ 0 & 0 & \|\mathbf{q}_2\| \end{bmatrix} \tag{11}$$

By the definition of $\mathbf{q}_1, \mathbf{q}_2$ in Eq. (8) together with the orthonormal property of the eigenvectors, the norms of $\mathbf{q}_1$ and $\mathbf{q}_2$ can be computed indirectly with two $r \times 1$ vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ as

$$\begin{aligned}\|\mathbf{q}_1\| &= \sqrt{1 - \|\mathbf{r}_1\|^2} \\ \|\mathbf{q}_2\| &= \sqrt{1 - \|\mathbf{r}_2\|^2 - \frac{(\mathbf{r}_1'\mathbf{r}_2)^2}{1 - \|\mathbf{r}_1\|^2}}\end{aligned} \tag{12}$$

This enables us to compute $\|\mathbf{q}_1\|$ and $\|\mathbf{q}_2\|$ without explicitly constructing $\mathbf{q}_1$ and $\mathbf{q}_2$, which reduces the cost of step 8 from $O(nr)$ to $O(r)$. It can be proved that by setting $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$, the eigenvalues of $\mathbf{Z}$ are just the top eigenvalues of the perturbed matrix $\mathbf{A} + \Delta\mathbf{A}$, and the top eigenvectors of $\mathbf{A} + \Delta\mathbf{A}$ can be calculated by $\mathbf{Q}\mathbf{U}_z$ (step 18). Therefore, we only need $\mathbf{\Lambda}_z$ to compute the impact score of element o (step 10). After scanning all the valid elements in the current network, we choose the one with the largest impact score and add it to the element set $\mathcal{X}$ (step 12-13). Then, we update the network and its eigen-pairs (step 14-18). The procedure to update eigen-pairs is similar to that of computing the impact score for a given network element (step 6-9), with the following subtle difference. In order to just compute the impact score of a given network element, we only need the updated eigenvalues. This is crucial as it saves the computation of (1) constructing $\mathbf{q}_1$ and $\mathbf{q}_2$, (2) finding the eigenvectors of $\mathbf{Z}$, and (3) updating the eigenvectors of perturbed matrix $\mathbf{A} + \mathbf{\Delta A}$, which in turn helps maintain constant time complexity for each inner for-loop (step 5-11).

Algorithm 2 can be easily extended to address the local connectivity minimization problem by properly approximate the local connectivity impact score at step 10. It is worth noting that for some complex local connectivity measures like local natural connectivity or local length-t path capacity, it is often time consuming to directly calculate local connectivity $C_{\mathcal{T}}(G)$ and element impact $I_{\mathcal{T}}(o)$ on $C_{\mathcal{T}}(G)$ than the global connectivity $C(G)$ and $I(o)$. This is mainly because that

$C_{\mathcal{T}}(G)$ and $I_{\mathcal{T}}(o)$ can not be directly calculated with the eigen-pairs of the network. To efficiently address this problem, we propose the following heuristic to measure $C_{\mathcal{T}}(G)$ and $I_{\mathcal{T}}(o)$.

LEMMA 3. Local Impact Computation. *Let $C(G \setminus \mathcal{T})$ and $I^{G\setminus\mathcal{T}}(o)$ denotes the global connectivity of graph $G \setminus \mathcal{T}$ and impact of element o on $C(G \setminus \mathcal{T})$, then $C_{\mathcal{T}}(G) = C(G) - C(G \setminus \mathcal{T})$, and $I_{\mathcal{T}}(o) = I(o) - I^{G\setminus\mathcal{T}}(o)$.*

PROOF. The first equation naturally holds by the definition of connectivity measures. Here we proceed to prove the second part. By the definition of $I_{\mathcal{S}}(o)$, we have

$$I_{\mathcal{T}}(o) = C_{\mathcal{T}}(G) - C_{\mathcal{T}}(G \setminus \{o\})$$

By the fact that $C_{\mathcal{T}}(G) = C(G) - C(G \setminus \mathcal{T})$, the above equation can be re-write as

$$\begin{aligned} I_{\mathcal{T}}(o) &= (C(G) - C(G \setminus \mathcal{T})) - (C(G \setminus \{o\}) - C(G \setminus \{o\} \cup \mathcal{T}) \\ &= (C(G) - C(G \setminus \{o\})) - (C(G \setminus \mathcal{T}) - C(G \setminus \{o\} \cup \mathcal{T}) \\ &= I(o) - I^{G\setminus\mathcal{T}}(o) \end{aligned} \tag{13}$$

□

4.3 Proof and Analysis

In this subsection, we analyze the proposed CONTAIN algorithm w.r.t. its effectiveness and efficiency.

4.3.1 Effectiveness. The effectiveness of CONTAIN is summarized in Lemma 4, which says that the computation of the impact score for each valid network element in the inner for-loop does not introduce any extra approximation error.

LEMMA 4. **Effectiveness of CONTAIN.** *Suppose $\mathbf{A}$ is approximated with its top-r eigen-pairs with approximation error $\mathbf{E}$ (i.e. $\mathbf{A} = \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{E}$), then the $\mathbf{\Lambda}_z$ and $\mathbf{QU}_z$ returned in Algorithm 2 can be used to approximate $\tilde{\mathbf{A}}$ as its top eigen-pairs with no extra error.*

PROOF. As $\mathbf{A} = \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{E}$ and $\Delta\mathbf{A} = \mathbf{U}_\Delta\mathbf{\Lambda}_\Delta\mathbf{U}_\Delta'$, then $\tilde{\mathbf{A}}$ can be expressed as

$$\begin{aligned} \tilde{\mathbf{A}} &= \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{U}_\Delta\mathbf{\Lambda}_\Delta\mathbf{U}_\Delta' + \mathbf{E} \\ &= [\mathbf{U}^{(r)}, \mathbf{U}_\Delta] \begin{bmatrix} \mathbf{\Lambda}^{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_\Delta \end{bmatrix} [\mathbf{U}^{(r)}, \mathbf{U}_\Delta]' + \mathbf{E} \end{aligned} \tag{14}$$

Perform partial-QR decomposition on $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$ as $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta] = \mathbf{QR}$, we get orthonormal basis for $\tilde{\mathbf{A}}$ and an upper triangular matrix $\mathbf{R}$. Then the perturbed matrix $\tilde{\mathbf{A}}$ can be rewritten as

$$\tilde{\mathbf{A}} = \mathbf{QR} \begin{bmatrix} \mathbf{\Lambda}^{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_\Delta \end{bmatrix} \mathbf{R}'\mathbf{Q}' + \mathbf{E} \tag{15}$$

Let $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$ and perform eigen decomposition on $\mathbf{Z}$ as $\mathbf{Z} = \mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}_z'$, $\tilde{\mathbf{A}}$ is now equivalent to

$$\tilde{\mathbf{A}} = \mathbf{QU}_z\mathbf{\Lambda}_z\mathbf{U}_z'\mathbf{Q}' + \mathbf{E} = (\mathbf{QU}_z)\mathbf{\Lambda}_z(\mathbf{QU}_z)' + \mathbf{E} \tag{16}$$

Since both $\mathbf{Q}$ and $\mathbf{U}_z$ are orthonormal, we have $(\mathbf{QU}_z)(\mathbf{QU}_z)' = \mathbf{I}$. Thus, $\mathbf{\Lambda}_z$ and $\mathbf{QU}_z$ can be viewed as the top eigen-pairs of $\tilde{\mathbf{A}}$. As the approximation error remains to be $\mathbf{E}$ in Eq. (16), it implies that no extra error is introduced in the procedure, which completes the proof. □

As the eigenvalues in real network are often skewed [20], the above impact scores can be approximated with top-r eigenvalues. Analysis in [51] and [6] show that the truncated approximations for triangle capacity and natural connectivity can achieve high accuracy with only top-50 eigenvalues, which enables a great acceleration on impact score approximation.

4.3.2 Efficiency. The complexity of the proposed CONTAIN algorithm is summarized in Lemma 5, which says it is linear in both time and space.

LEMMA 5. **Complexity of CONTAIN.** *The time complexity of CONTAIN for node-level connectivity optimization is $O(k(mr + nr^3))$. The time complexity of CONTAIN for edge-level connectivity optimization is $O(k(mr^3 + nr^2))$. The space complexity of CONTAIN is $O(nr + m)$.*

PROOF. In the CONTAIN algorithm, computing top-r eigen-pairs and connectivity $C(G)$ would takes $O(nr^2 + mr)$ and $O(r)$ respectively. To compute the impact score for each node/edge (step 5-11), it takes $O(d_v r)$ (d_v is the degree of node v) for node v, and $O(r)$ for each edge to get the upper triangular matrix $\mathbf{R}$ in step 8. Since performing eigen-decomposition on $\mathbf{Z}$ at step 9 takes $O(r^3)$, the complexity to collect impact scores for all the nodes/edges are $O(nr^3 + mr)$ and $O(mr^3)$ respectively. Picking out the node/edge with highest impact score in current iteration would cost $O(n)$ for node level operations and $O(m)$ for edge level operations. At the end of the iteration, updating the eigen-pairs of the network takes the complexity of $O(nr^2 + r^3)$. As we have $r \ll n$, the overall time complexity to select k nodes would be $O(k(mr + nr^3))$; and the complexity to select k edges would be $O(k(mr^3 + nr^2))$

For space complexity, it takes $O(n + m)$ to store the entire network, $O(nr)$ to calculate and store the top-r eigen-pair of the network, $O(n)$ to store the impact scores for all the nodes in node level optimization scenarios and $O(m)$ to store the impact scores for all the edges, the eigen-pair update requires a space of $O(nr)$. Therefore, the overall space complexity for CONTAIN is $O(nr + m)$. □

4.4 CONTAIN+: The Closed-form Heuristics

Here we provide the heuristics for the triangle capacity and natural connectivity optimization problems, which can be easily extended to other similar connectivity measures.

4.4.1 Impact Approximation. For triangle capacity optimization, the impact of a node/edge is the number of triangles that the node/edge participates in, which can be directly approximated with the eigen-pairs of the current network.

LEMMA 6. Closed-Form Impact Score for Triangle Capacity. *Given a network G with adjacency matrix $\mathbf{A}$ and eigen-pair $(\mathbf{U}, \mathbf{\Lambda})$. The number of triangles that node v participates in is $I(v) = \sum_{i=1}^{n} \frac{\lambda_i^3 \mathbf{u}_i^2(v)}{2}$; the number of triangles that edge $\langle u, v\rangle$ participates in is $I(\langle u, v\rangle) = \sum_{i=1}^{n} \lambda_i^2 \mathbf{u}_i(u)\mathbf{u}_i(v)$.*

PROOF. The first part of the lemma has been proved in [51]. Here we proceed to prove the second part.

The number of triangles that edge $\langle u, v\rangle$ involves in equals to the number of length-2 paths from node u to node v, which equals $\mathbf{A}^2(u, v)$. As $\mathbf{A} = \mathbf{U\Lambda U}'$, we have $\mathbf{A}^2 = \mathbf{U\Lambda}^2\mathbf{U}'$. Therefore, $\mathbf{A}^2(u, v) = \sum_{i=1}^{n} \lambda_i^2 \mathbf{u}_i(u)\mathbf{u}_i(v)$. □

LEMMA 7. Closed-Form Impact Score for Natural Connectivity. *Given a network G with adjacency matrix $\mathbf{A}$ and eigen-pair $(\mathbf{U}, \mathbf{\Lambda})$. The impact of node v on natural connectivity is $I(v) = \sum_{i=1}^{n} e^{\lambda_i} \mathbf{u}_i^2(v)$; the impact of edge $\langle u, v\rangle$ is $I(\langle u, v\rangle) = \sum_{i=1}^{n} \frac{e^{\lambda_i} - 1}{\lambda_i} \mathbf{u}_i(u)\mathbf{u}_i(v)$.*

PROOF. Natural connectivity can be viewed as an aggregation of weighted closed-walks [24]. For node v, its impact on the number of length-t closed-walks is proportional to $\mathbf{A}^t(v, v)$, so its overall impact on natural connectivity can be expressed as $I(v) = \sum_{j=0}^{\infty} \frac{\mathbf{A}^j(v,v)}{j!}$. As we have $\mathbf{A} = \mathbf{U\Lambda U}'$ and

$\mathbf{A}^j(v,v) = \sum_{i=1}^n \lambda_i^j \mathbf{u}_i^2(v)$, we have

$$I(v) = \sum_{j=0}^{\infty} \frac{1}{j!} \sum_{i=1}^{n} \lambda_i^j \mathbf{u}_i^2(v) = \sum_{i=1}^{n} \sum_{j=1}^{\infty} \frac{1}{j!} \lambda_i^j \mathbf{u}_i^2(v) \tag{17}$$
$$= \sum_{i=1}^{n} e^{\lambda_i} \mathbf{u}_i^2(v)$$

For edge $\langle u, v\rangle$, the number of length-t closed-walks it participates in equals to the number of length-$(t-1)$ walks from node u to node v, which can be expressed as $\mathbf{A}^{t-1}(u,v)$. Therefore, its overall impact on natural connectivity can be written as $I(\langle u,v\rangle) = \sum_{j=1}^{\infty} \frac{\mathbf{A}^{j-1}(u,v)}{j!}$. Let $\mathbf{T} = \sum_{j=1}^{\infty} \frac{\mathbf{A}^{j-1}}{j!}$, then we have

$$\mathbf{AT} = \sum_{j=1}^{\infty} \frac{\mathbf{A}^j}{j!} = \sum_{j=0}^{\infty} \frac{\mathbf{A}^j}{j!} - \mathbf{I} = e^{\mathbf{A}} - \mathbf{I}$$

Based on the above equation, we have

$$\mathbf{T} = \mathbf{A}^{-1}(e^{\mathbf{A}} - \mathbf{I}) = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}'(\mathbf{U}(e^{\mathbf{\Lambda}} - \mathbf{I})\mathbf{U}') \tag{18}$$
$$= \mathbf{U}\mathbf{\Lambda}^{-1}(e^{\mathbf{\Lambda}} - \mathbf{I})\mathbf{U}'$$

Thus, $I(\langle u,v\rangle) = \mathbf{T}(u,v) = \sum_{i=1}^{n} \frac{e^{\lambda_i}-1}{\lambda_i} \mathbf{u}_i(u)\mathbf{u}_i(v)$ □

4.4.2 Effectiveness of CONTAIN+. As we have mentioned in the previous subsection, the impact score of a node/edge is often approximated with the top-r eigen-pairs of the network. Let $(\mathbf{U}, \mathbf{\Lambda})$ be the eigen-pairs of network G, $(\tilde{\mathbf{U}}, \tilde{\mathbf{\Lambda}})$ be the eigen-pairs of network $G \setminus \{v\}$. To estimate the impact of node v on the connectivity of the network, CONTAIN needs to utilize the eigenvalues from both the original network (i.e. $\mathbf{\Lambda}$) and the perturbed network (i.e. $\tilde{\mathbf{\Lambda}}$) for the calculation (Algorithm 2 step 10); while CONTAIN+ only relies on the eigen-pairs in the original network (i.e. $(\mathbf{U}, \mathbf{\Lambda})$). Take triangle capacity optimization under node operations as an example. the impact of node v can be approximated as $I(v)_{\text{CONTAIN}} = \sum_{i=1}^{r} \frac{\lambda_i^3}{6} - \sum_{i=1}^{r} \frac{\tilde{\lambda}_i^3}{6}$ by CONTAIN; or it can be approximated as $I(v)_{\text{CONTAIN+}} = \sum_{i=1}^{r} \frac{\lambda_i^3 \mathbf{u}_i^2(v)}{2}$ by CONTAIN+. Suppose the exact impact of node v is $I(v)_{Exact}$, then we can define the approximation error of CONTAIN as $err_{\text{CONTAIN}} = I(v)_{Exact} - I(v)_{\text{CONTAIN}}$ and the error of CONTAIN+ as $err_{\text{CONTAIN+}} = I(v)_{Exact} - I(v)_{\text{CONTAIN+}}$. In Lemma 8, we give the analysis on err_{CONTAIN} and $err_{\text{CONTAIN+}}$.

LEMMA 8. *The error of CONTAIN with top-r eigen-pair based triangle capacity impact approximation is* $err_{CONTAIN} = \frac{\sum_{i=r+1}^{n} \lambda_i^3 - \tilde{\lambda}_i^3}{6}$*; the approximation error for CONTAIN+ is* $err_{CONTAIN+} = \sum_{i=r+1}^{n} \frac{\lambda_i^3 \mathbf{u}_i^2(v)}{2}$.

PROOF. As we have $I(v)_{Exact} = \sum_{i=1}^{n} \frac{\lambda_i^3 \mathbf{u}_i^2(v)}{2} = \frac{\sum_{i=1}^{n} \lambda_i^3 - \tilde{\lambda}_i^3}{6}$, Lemma 8 naturally holds when $I(v)_{\text{CONTAIN}}$ and $I(v)_{\text{CONTAIN+}}$ are subtracted from $I(v)_{Exact}$ respectively. □

Lemma 8 implies that when the removed node has small effect to the bottom-$(n-r)$ eigenvalues of the underlying network, CONTAIN is preferred as err_{CONTAIN} would be small. While in networks with very skewed eigenvalue distributions, CONTAIN+ is preferred as the bottom-$(n-r)$ eigenvalues are small in magnitude. Similar analysis can be derived for edge-level operation scenarios and natural connectivity optimization scenarios, which is omitted for brevity.

4.4.3 Complexity of CONTAIN+.

Theorem 3. *The time complexity of CONTAIN+ for node level connectivity optimization is* $O(nr^2 + mr + knr^2)$. *The time complexity for edge level connectivity optimization is* $O(k(mr + nr^2))$. *The space complexity of CONTAIN+ is* $O(nr + m)$.

Proof. The CONTAIN+ algorithm is generally similar to the CONTAIN algorithm except for the node/edge impact calculation part. Therefore, CONTAIN+ also need to take $O(nr^2 + mr)$ to compute the top-r eigen-pairs. In each iteration, it takes CONTAIN+ $O(nr)$ time to get all the impact scores for each node or $O(mr)$ for the scores of each edge. After picking out the highest impact node/edge with $O(n)$ or $O(m)$ complexity, we update the eigen-pairs of the network with the method used in CONTAIN with $O(nr^2 + r^3)$ complexity. Therefore, the overall complexity for node level connectivity minimization is $O(nr^2 + mr + knr^2)$ and the complexity for edge level connectivity minimization is $O(nr^2 + mr + k(mr + nr^2))$, which can be simplified as $O(k(mr + nr^2))$.

As for the space complexity, CONTAIN+ also need space to store all the top-r eigen-pairs and the impact scores for nodes/edges. Since no extra computation space is introduced in CONTAIN+ compared to CONTAIN, the overall space complexity for CONTAIN+ is still $O(nr + m)$. □

5 EVALUATIONS

In this section, we evaluate the proposed CONTAIN algorithm. All experiments are designed to answer the following two questions:

- **Effectiveness.** How effective is the proposed CONTAIN algorithm in minimizing various connectivity measures?
- **Efficiency.** How efficient and scalable is the proposed CONTAIN algorithm?

5.1 Experiment Setup

5.1.1 Datasets. We perform experiments on 10 different datasets from 4 different domains, including Airport: an air traffic network that represents the direct flight connections between internal US airports[3]; Oregon: an autonomous system network which depicts the information transferring relationship between routers from [32]; Chemical: a network based on [18] that shows the similarity between different chemicals; Disease: a network that depicts the similarity between different diseases [18]; Gene: a protein-protein interaction network based on [18]; Astrph: a collaboration network between authors whose papers were submitted to Astro Physics category on Arxiv [33]; Hepth: a collaboration network between authors whose papers were submitted to High Energy Physics (Theory category) on Arxiv [32]; Aminer: a collaboration network between researchers in the Aminer datasets [49]; Eucore: the email correspondence network from a large European research institution [33]; Fb: a social circle network collected from Facebook [39]; and Youtube: a friendship network among youtube users [50]. The statistics of those datasets are listed in Table 2.

Table 2. Statistics of Datasets.

Domain	Dataset	#Nodes	#Edges	Avg Degree
Infrastructure	Airport	2,833	7,602	5.37
	Oregon	5,296	10,097	3.81
Biology	Chemical	6,026	69,109	22.94
	Disease	4,256	30,551	14.36
	Gene	7,604	14,071	3.7
Collaboration	Astrph	18,772	198,050	21.1
	Hepth	9,877	25,985	5.26
	Aminer	1,211,749	4,756,194	7.85
Social	Eucore	1,005	16,064	31.97
	Fb	4,039	88,234	43.69
	youtube	1,138,499	2,990,443	5.25

[3]http://www.levmuchnik.net/Content/Networks/NetworkData.html.

5.1.2 Comparing Methods. We compare the proposed algorithm with the following methods. (1) *Degree*: selecting top−k nodes (edges) with the largest degrees; specifically, for edge $\langle u, v\rangle$, let d_u and d_v denote the degrees for its endpoints respectively, the score for $\langle u, v\rangle$ is $\min\{d_u, d_v\}$[4]. (2) *PageRank*: selecting top−k nodes (edges) with the largest PageRank scores [44] (the corresponding edge score is the minimum PageRank score among its two endpoints); (3) *Eigenvector*: selecting top−k nodes (edges) with the largest eigenvector centrality scores [43] (the corresponding edge score is the minimum eigenvector centrality score among its endpoints); (4) *Netshield/Netmelt*: selecting top−k nodes (edges) that minimize the leading eigenvalue of the network [11, 12]; (5) *MIOBI*: a greedy algorithm that employs first-order matrix perturbation method to estimate element impact score and update eigen-pairs [6]; (6) *MIOBI-S*: a variant of MIOBI that selects top−k nodes (edges) in one batch without updating the eigen-pairs of the network; (7) *MIOBI-H*: a variant of MIOBI that employs high order matrix perturbation method to update eigen-pairs [10]; (8) *Exact*: a greedy algorithm that recomputes the top-r eigen-pairs to estimate the impact score for each candidate node/edge. It is worth to note that the *NetShield/NetMelt*, *MIOBI* and CONTAIN algorithms are all based on the greedy algorithm, which has the same approximation ratio in theory. The main difference lies in their accuracy to approximate the perturbed eigen-pairs.

For the results reported in this paper, we set rank $r = 80$ for all the top−r eigen-pairs based approximation methods (methods (5)-(8) and the proposed CONTAIN method).

5.1.3 Evaluation Metrics. The performance of the algorithm is evaluated by the impact of its selected elements $I(\mathcal{X}) = C(G) - C(G \setminus \mathcal{X})$. The larger the $I(\mathcal{X})$ is, the more effective the algorithm is. For a given dataset, connectivity measure and network operation, we normalize $I(\mathcal{X})$ by that of the best method, so that the results in different datasets are comparable in the same plot.

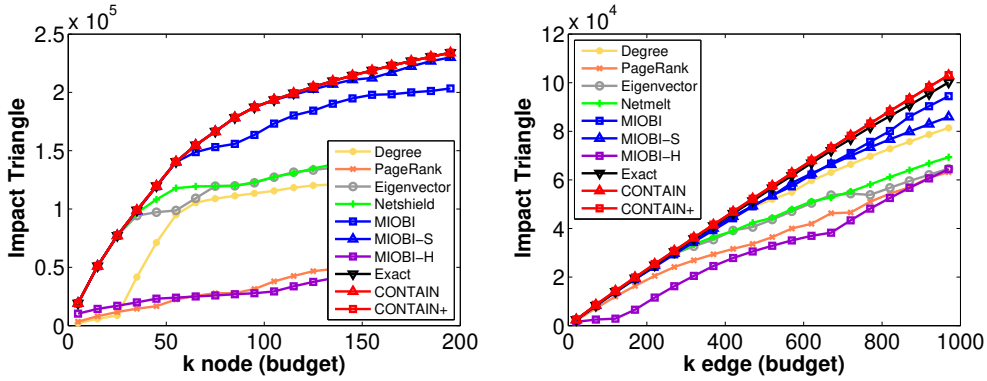
5.1.4 Machine and Repeatability. All the experiments in the paper are performed on a machine with 2 processors (Intel Xeon 3.5GHz) with 256GB of RAM. The algorithms are programmed with MATLAB using a single thread.

5.2 Effectiveness

5.2.1 Effectiveness of CONTAIN and CONTAIN+. We compare the proposed algorithm and the baseline methods on three connectivity measures (leading eigenvalue, number of triangles, and natural connectivity) by both node-level operations and edge-level operations on all datasets in our experiment. Since the *Exact* method needs to recompute the top-r eigen-pairs for each candidate node/edge which is very time-consuming, its results would be absent on some large datasets (e.g., AMINIER and ASTRPH) where it does not finish the computation within 24 hours. In our experiment, the budget for node-level operations is $k = 20$, the budget for edge-level operations is $k = 200$. The results are shown from Figure 4 to Figure 9[5]. Specifically, the Y-axis is the impact score achieved by the method normalized by the maximum impact score among all methods on the same dataset. We can see that the proposed CONTAIN (the red solid bar) and CONTAIN+ (the red hollow bar) (1) are very close to the *Exact* method (the black hollow bar); and (2) consistently outperforms all the other alternative methods. In the meanwhile, the proposed CONTAIN and CONTAIN+ algorithms are much faster than *Exact*, as will shown in the next subsection.

To study the effectiveness of CONTAIN and CONTAIN+ for the local connectivity minimization problem. We experiment on the CHEMICAL data and compare the performance of different methods for minimizing the local triangle capacity in the network. From Figure 3, it is obvious to see that

[4]We use $\min\{d_u, d_v\}$ as edge score to ensure that both ends of the top ranked edges are high degree nodes.
[5]The impact of Degree and PageRank method on leading eigenvalue and natural connectivity in Chemical and Aminer datasets are nearly zero.



(a) Node operations. (b) Edge operations.

Fig. 3. The optimization results on the number of local triangles on the CHEMICAL dataset.

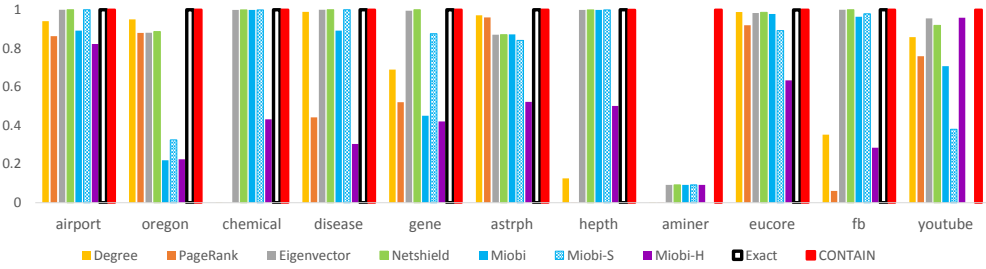


Fig. 4. The optimization results on leading eigenvalue with node-level operations.

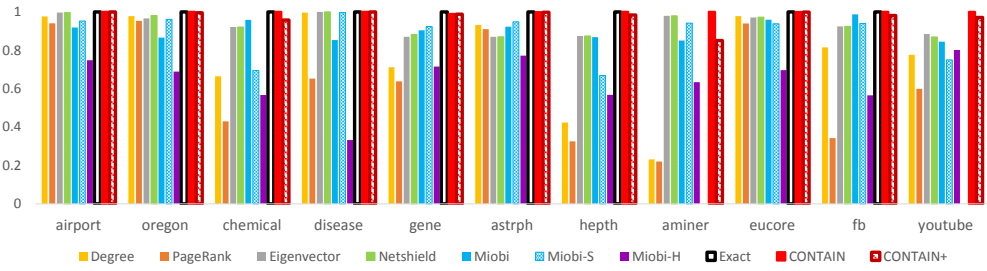


Fig. 5. The optimization results on the number of triangles with node-level operations.

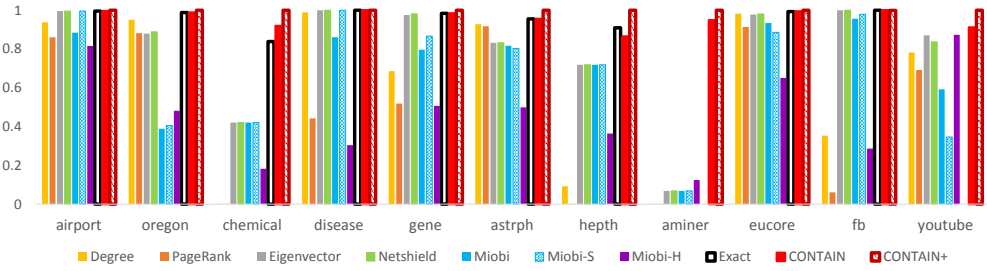


Fig. 6. The optimization results on natural connectivity with node-level operations.

both CONTAIN and CONTAIN+ can achieve similar performance with the *Exact* algorithm and outperform all other heuristic methods.

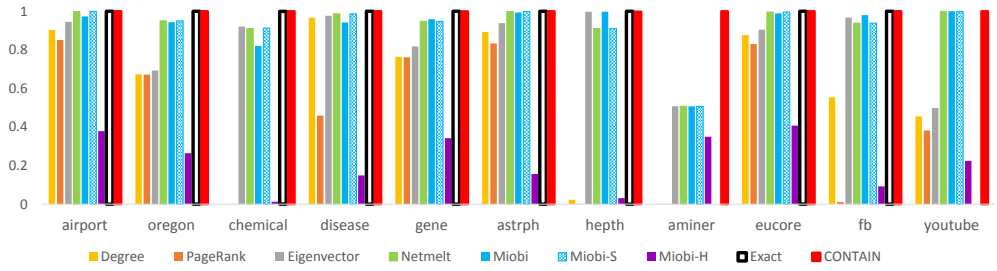


Fig. 7. The optimization results on leading eigenvalue with edge-level operations.

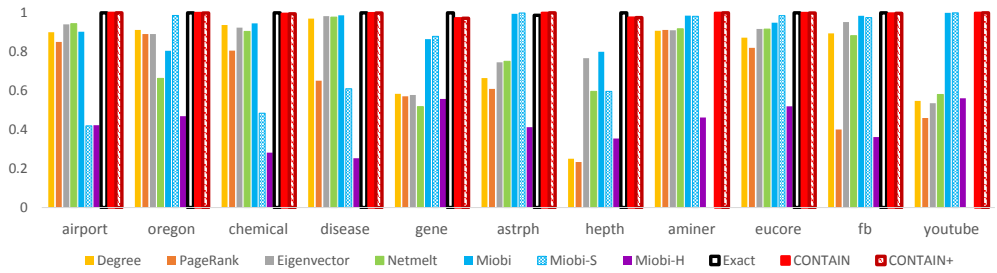


Fig. 8. The optimization results on the number of triangles with edge-level operations.

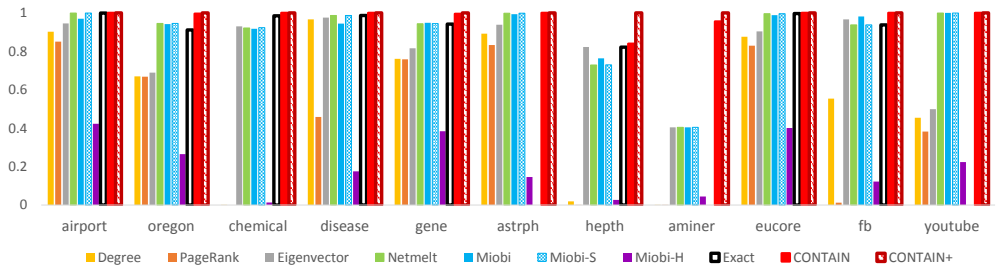


Fig. 9. The optimization results on natural connectivity with edge-level operations.

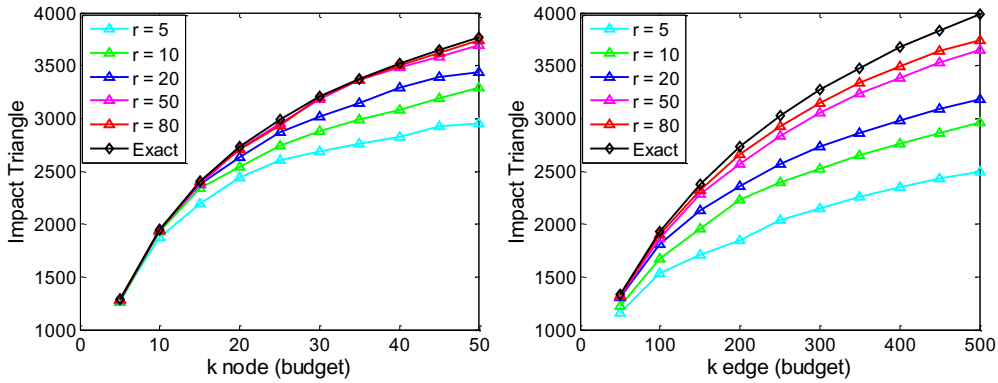
5.2.2 Effect of Rank r. The main parameter that affects the performance of CONTAIN is the rank r. To study the effect of r, we change r from 5 to 80 to minimize the number of triangles on the CHEMICAL dataset and compare them with the *Exact* method. The results are shown in Figure 10. From Figure 10, it is obvious to see that as r increases, the performance of CONTAIN increases accordingly, which is consistent with our effectiveness analysis. With $r = 80$, the performance of CONTAIN is very close to the *Exact* method with different k.

5.3 Efficiency

5.3.1 Efficiency of CONTAIN. In Table 3 and Table 4, we have compared the time complexity of all the methods in its base versions and we assume the budge $k < \log n$ for node operations and $k < \log m$ for edge operations.

Table 3. Time Complexity Comparison for Node Operations.

	Degree	PageRank	Eigenvector	NetShield	MIOBI	Exact	CONTAIN
Time Complexity	$O(m+nk)$	$O(m+nk)$	$O(m+nk)$	$O(m+nk^2)$	$O(k(mr+nr^2))$	$O(k(n^2r^2+nmr))$	$O(k(mr+nr^3))$

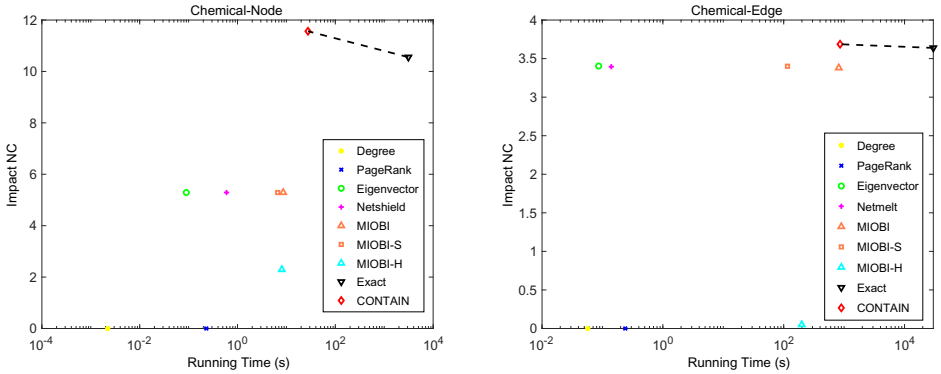


(a) Node operations. (b) Edge operations.

Fig. 10. The effect of r on optimizing the number of triangles on CHEMICAL dataset.

Table 4. Time Complexity Comparison for Edge Operations.

	Degree	PageRank	Eigenvector	NetMelt	MIOBI	Exact	CONTAIN
Time Complexity	$O(mk)$	$O(mk)$	$O(mk)$	$O(n+mk)$	$O(k(mr+nr^2))$	$O(k(nmr^2+m^2r))$	$O(k(mr^3+nr^2))$

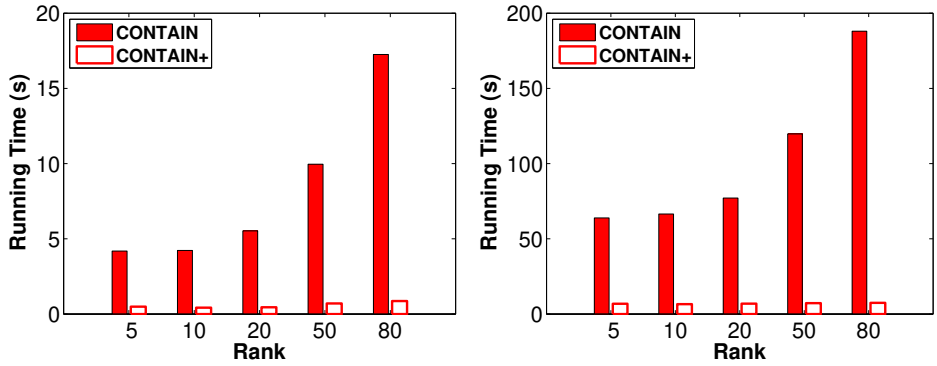


(a) Node operations (b) Edge operations

Fig. 11. The quality vs. running time trade-off on CHEMICAL. The budget for node operations is $k = 20$, the budget for edge operations is $k = 200$.

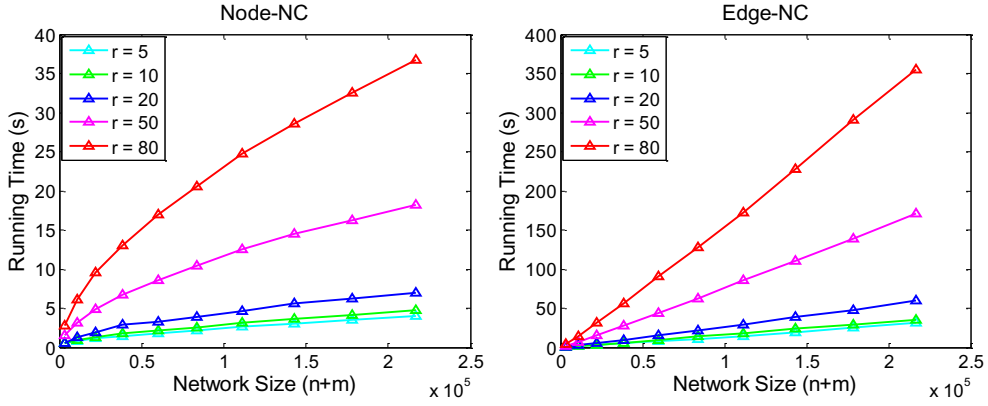
With that, we have compared the quality vs. running time trade-off of different methods for optimizing the natural connectivity (the most complicated connectivity measure) on the CHEMICAL dataset as presented in Figure 11. In both node-level and edge-level optimization scenarios, the proposed CONTAIN achieves a very similar performance as *Exact*. In terms of the running time, CONTAIN is orders of magnitude faster than *Exact*. Although the running time of other baseline methods is similar to CONTAIN, their performance (y-axis) is not as good as CONTAIN.

5.3.2 Efficiency of CONTAIN+. To justify the efficiency of CONTAIN+, we compare the running time of CONTAIN and CONTAIN+ on the CHEMICAL dataset in Figure 12. We can see that the running time of CONTAIN+ is orders of magnitudes faster than the CONTAIN algorithm. Moreover, as rank r increases, the running time of CONTAIN would increase in polynomial order due to the



(a) Node operations (b) Edge operations

Fig. 12. The running time comparison between CONTAIN and CONTAIN+ on the CHEMICAL dataset. The budget for both node and edge operations is $k = 20$.



(a) Node operations (b) Edge operations

Fig. 13. The scalability of CONTAIN performed on ASTRPH. The budget for both operations is $k = 20$.

eigen-decomposition operation for node/edge impact score approximation; while the running time of CONTAIN+ only shows a slightly linear increase across different rank settings.

5.3.3 Scalability of CONTAIN. The scalability results of CONTAIN are presented in Figure 13. As we can see, the proposed CONTAIN algorithm scales linearly w.r.t. the size of the input network (i.e. both the number of nodes and edges), which is consistent with Lemma 5.

6 RELATED WORK

In this section, we review the related literature from the following two perspectives, including (a) connectivity measures, (b) network connectivity optimization algorithms, and (c) applications.

Connectivity Measures. At the macro-level, network connectivity can be viewed as a measure to evaluate how well the nodes are connected together. Examples include the size of giant connected component [4], graph diameter [1], the mixing time [22], the vulnerability measure [2], and the clustering coefficient [52]. At the micro-view level, network connectivity measures the capacity of

edges, paths, loops, some complex motifs [40] or even the centrality of the nodes. Examples include the epidemic threshold [5], the natural connectivity (i.e., the robustness) [24], degree centrality [21], total pair-wise connectivity [19], etc.

Connectivity Optimization Algorithms. State-of-the-art algorithms for network connectivity optimization are almost exclusively designed for a specific connectivity measure and/or with a specific network operation (node deletion vs. edge deletion). To name a few, Chen et al. have proposed both node-level and edge-level manipulation strategies to optimize leading eigenvalue and proved that the corresponding node-level optimization problem is NP-hard in [12] and [11], respectively. On the other hand, Le et al. have proposed an algorithm to minimize the leading eigenvalue for networks with small eigen-gaps in [30]. In [37], Li et al. show that both node-level and edge-level triangle minimization problem is NP-hard and have proposed several heuristic strategies for the triangle minimization problem. In [6] and [7], Chan et al. study the optimization problem for network robustness without analyzing its hardness. In [46], Shen et al. study the optimization problem for total pair-wise connectivity. In order to effectively and efficiently compute the impact scores of network elements, the proposed CONTAIN algorithm resorts to partial-QR decomposition, which has been successfully used in several dynamic network mining tasks [16, 23, 35, 36].

Applications. The connectivity of the network has played an important role in many applications. For the applications that focus on system-level studies, global connectivity measures are more commonly used. In the immunization studies [17], it is critical to select a group of nodes/edges to effectively contain the propagation process [10, 12]. In the biomedical domain, antibiotic drugs are developed to kill the bacteria by disrupting their molecular network to the max extend [28]. While in [38], connectivity optimization methods have been applied on the protein-protein interaction network to identity critical proteins in the network. In the critical infrastructure networks, facilities that may cause large-scale failures are retrieved and protected proactively to ensure the full-functioning of the entire system [13]. It is worth noting that finding a group of nodes/edges that have high impact on the connectivity of the network is similar to the influence maximization problem [15, 25, 41] and its variations (e.g. viral marketing [14, 31], outbreak detection [34], etc). The main difference between the two problems is that the influence maximization problem is highly dependent on the underlying diffusion model (e.g. independent cascade, linear threshold [25]), while the connectivity optimization problem is directly based on the backbone (i.e. the underlying topology) of the network. In addition to the global connectivity measures, the local connectivity of the network has also been applied to some critical tasks like graph clustering [53, 54], link prediction [3], etc.

7 CONCLUSIONS

In this paper, we study the network connectivity minimization problem by addressing two open challenges. On the theoretic side, we prove that a wide range of network connectivity optimization (NETCOM) problems are NP-hard and $(1 - 1/e)$ is the best approximation ratio that a polynomial algorithm can achieve for NETCOM problems unless $NP \subseteq DTIME(n^{O(\log \log n)})$. On the algorithmic aspect, we propose an effective, scalable and generalizable algorithm CONTAIN and its closed-form variant CONTAIN+. Extensive experimental evaluations on a variety of real networks demonstrate that the proposed algorithm (1) consistently outperforms alternative methods, and (2) scales linearly w.r.t. the network size.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under grant No. 1947135, and 2003924, and by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725. The content of the information in this document does not necessarily reflect the position or the policy of

the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 1999. Internet: Diameter of the world-wide web. *nature* 401, 6749 (1999), 130.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378–382.

[3] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *arXiv preprint arXiv:1802.06916* (2018).

[4] Béla Bollobás. 2001. *The Evolution of Random Graphs–the Giant Component* (2 ed.). Cambridge University Press, 130–159. https://doi.org/10.1017/CBO9780511814068.008

[5] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. 2008. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)* 10, 4 (2008), 1.

[6] Hau Chan, Leman Akoglu, and Hanghang Tong. 2014. Make it or break it: Manipulating robustness in large networks. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 325–333.

[7] Hau Chan, Shuchu Han, and Leman Akoglu. 2015. Where graph topology matters: the robust subgraph problem. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 10–18.

[8] Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong. 2015. On the Connectivity of Multi-layered Networks: Models, Measures and Optimal Control. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 715–720.

[9] Chen Chen, Ruiyue Peng, Lei Ying, and Hanghang Tong. 2018. Network Connectivity Optimization: Fundamental Limits and Effective Algorithms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1167–1176.

[10] Chen Chen and Hanghang Tong. 2017. On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10, 2 (2017), 121–135.

[11] Chen Chen, Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2016. Eigen-Optimization on Large Graphs by Edge Manipulation. *ACM Trans. Knowl. Discov. Data* 10, 4, Article 49 (June 2016), 30 pages. https://doi.org/10.1145/2903148

[12] Chen Chen, Hanghang Tong, B Aditya Prakash, Charalampos E Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. 2016. Node Immunization on Large Graphs: Theory and Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 113–126.

[13] Liangzhe Chen, Xinfeng Xu, Sangkeun Lee, Sisi Duan, Alfonso G Tarditi, Supriya Chinthavali, and B Aditya Prakash. 2017. Hotspots: Failure cascades on heterogeneous critical infrastructure networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1599–1607.

[14] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1029–1038.

[15] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 199–208.

[16] Xilun Chen and K Selcuk Candan. 2014. LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 987–996.

[17] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. 2003. Efficient immunization strategies for computer networks and populations. *Physical review letters* 91, 24 (2003), 247901.

[18] Allan Peter Davis, Cynthia J Grondin, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Thomas C Wiegers, and Carolyn J Mattingly. 2015. The Comparative Toxicogenomics Database's 10th year anniversary: update 2015. *Nucleic acids research* 43, D1 (2015), D914–D920.

[19] Thang N Dinh, Ying Xuan, My T Thai, Panos M Pardalos, and Taieb Znati. 2011. On new approaches of assessing network vulnerability: hardness and approximation. *IEEE/ACM Transactions on Networking* 20, 2 (2011), 609–619.

[20] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, Vol. 29. ACM, 251–262.

[21] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[22] Mark Jerrum and Alistair Sinclair. 1988. Conductance and the rapid mixing property for Markov chains: the approximation of permanent resolved. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM, 235–244.

[23] Ling Jian, Jundong Li, and Huan Liu. 2018. Toward online node classification on streaming networks. *Data Min. Knowl. Discov.* 32, 1 (2018), 231–257. https://doi.org/10.1007/s10618-017-0533-y
[24] WU Jun, Mauricio Barahona, Tan Yue-Jin, and Deng Hong-Zhong. 2010. Natural connectivity of complex networks. *Chinese physics letters* 27, 7 (2010), 078902.
[25] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.
[26] Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Inform. Process. Lett.* 70, 1 (1999), 39–45.
[27] Jon Kleinberg and Eva Tardos. 2006. *Algorithm design*. Pearson Education India.
[28] Michael A Kohanski, Daniel J Dwyer, and James J Collins. 2010. How antibiotics kill bacteria: from targets to networks. *Nature Reviews Microbiology* 8, 6 (2010), 423.
[29] Istvan A Kovacs and Albert-Laszlo Barabasi. 2015. Network science: Destruction perfected. *Nature* 524, 7563 (2015), 38–39.
[30] Long T Le, Tina Eliassi-Rad, and Hanghang Tong. 2015. MET: A fast algorithm for minimizing propagation in large graphs with small eigen-gaps. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 694–702.
[31] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. 2007. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* 1, 1 (2007), 5.
[32] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 177–187.
[33] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.
[34] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 420–429.
[35] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *Proceedings of CIKM 2017*. ACM, 387–396.
[36] Liangyue Li, Hanghang Tong, Yanghua Xiao, and Wei Fan. 2015. Cheetah: fast graph kernel tracking on dynamic graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 280–288.
[37] Rong-Hua Li and Jeffrey Xu Yu. 2015. Triangle minimization in large networks. *Knowledge and Information Systems* 45, 3 (2015), 617–643.
[38] Qiao Liu, Chen Chen, Annie Gao, Hang Hang Tong, and Lei Xie. [n. d.]. VariFunNet, an integrated multiscale modeling framework to study the effects of rare non-coding variants in genome-wide association studies: Applied to Alzheimer's disease. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2177–2182.
[39] Julian Mcauley and Jure Leskovec. 2014. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 1 (2014), 4.
[40] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.
[41] Flaviano Morone and Hernán A Makse. 2015. Influence maximization in complex networks through optimal percolation. *Nature* 524, 7563 (2015), 65.
[42] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.
[43] Mark EJ Newman. 2008. The mathematics of networks. *The new palgrave encyclopedia of economics* 2 (2008), 1–12.
[44] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report.
[45] B Aditya Prakash, Deepayan Chakrabarti, Nicholas C Valler, Michalis Faloutsos, and Christos Faloutsos. 2012. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and information systems* 33, 3 (2012), 549–575.
[46] Yilin Shen, Nam P Nguyen, Ying Xuan, and My T Thai. 2012. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking* 21, 3 (2012), 963–973.
[47] Michael Sipser. 1997. *Introduction to the theory of computation*. PWS Publishing Company.
[48] Daniel Spielman. 2012. Spectral graph theory. In *Combinatorial scientific computing*. Number 18. Citeseer.
[49] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.
[50] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 817–826.

[51] Charalampos E Tsourakakis. 2008. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 608–617.

[52] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.

[53] Hao Yin, Austin R Benson, and Jure Leskovec. 2019. The Local Closure Coefficient: A New Perspective On Network Clustering. *networks* 26, 41 (2019), 44.

[54] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 555–564. https://doi.org/10.1145/3097983.3098069