Noise-Enhanced Community Detection

Reyhaneh Abdolazimi
rabdolaz@data.syr.edu
Data Lab, EECS Department
Syracuse University

Shengmin Jin
shengmin@data.syr.edu
Data Lab, EECS Department
Syracuse University

Reza Zafarani
reza@data.syr.edu
Data Lab, EECS Department
Syracuse University

ABSTRACT

Community structure plays a significant role in uncovering the structure of a network. While many community detection algorithms have been introduced, improving the quality of detected communities is still an open problem. In many areas of science, adding noise improves system performance and algorithm efficiency, motivating us to also explore the possibility of adding noise to improve community detection algorithms. We propose a noise-enhanced community detection framework that improves communities detected by existing community detection methods. The framework introduces three noise methods to help detect communities better. Theoretical justification and extensive experiments on synthetic and real-world datasets show that our framework helps community detection methods find better communities.

CCS CONCEPTS

• **Information systems** → **Data mining**;

KEYWORDS

Community Detection, Noise-Enhanced Methods, Graph Mining.

ACM Reference Format:
Reyhaneh Abdolazimi, Shengmin Jin, and Reza Zafarani. 2020. Noise-Enhanced Community Detection. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media (HT '20), July 13–15, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3372923.3404788

1 INTRODUCTION

Communities are observed in many real-world networks: sets of nodes with higher internal density within each set than between them [6]. Communities carry various insights. In biological networks, communities represent functional units of cells [38]; in scientific collaboration networks, communities denote scientists with similar research interests [36]; and in social networks, communities are groups of friends with similar interests or backgrounds [11].

To detect communities more accurately and efficiently in networks, research has focused on designing new *community detection* algorithms [3, 6, 11, 28, 29, 35, 37]. Instead of designing new algorithms, an unexplored alternative to improve communities detected is to modify the input data to such algorithms: the network.

HT '20, July 13–15, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7098-1/20/07…$15.00
https://doi.org/10.1145/3372923.3404788

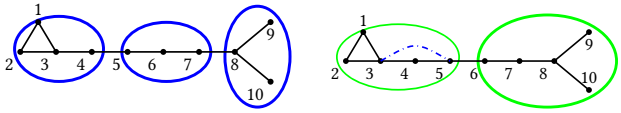

Figure 1: Detected Communities (ovals) before (a) and after (b) adding noise (dashed edge) using the same community detection algorithm (Leading Eigenvector method). Adding noise edge $(3, 5)$ **in (b) helps find better communities, as observed by** 30% **decrease in the value of community detection objective function (edge cut, here).**

A natural approach to modify data is to introduce noise. While noise is often unwanted and uncontrollable, and attempts are made to remove or reduce its effects, it has been shown beneficial in many areas of science, especially in nonlinear information processing systems [5]. Noise enhancement has long been used in physical systems as *stochastic resonance* and has also shown promise in areas such as stochastic optimization, image processing, and machine learning [2, 5, 32, 34]. Such benefits of adding noise have motivated us to explore the possibility of enhancing community detection by adding noise. Adding noise introduces an extra step to the existing algorithms. This extra *noise injection* step introduces a degree of randomization to the algorithms. A natural way to introduce noise in a network is to add edges as it allows one to systematically compare the detected communities in noisy and noiseless networks.

To provide some intuition on how adding noise can improve community detection, we provide an example. Consider the graph in Figure 1a with 10 nodes and 10 edges. In this graph, we can detect three communities (shown with ovals) using the Leading Eigenvector community detection method [29]: $\{1, 2, 3, 4\}$, $\{5, 6, 7\}$, and $\{8, 9, 10\}$. We can evaluate these communities using a community detection objective function. Here, we use *edge cut* [19] and obtain an edge cut value of 1.3. We add a single noise edge $(3, 5)$ to the graph (the dashed line) to get the *noisy graph* in Figure 1b. The same Leading Eigenvector method now detects two communities in this noisy graph: $\{1, 2, 3, 4, 5\}$ and $\{6, 7, 8, 9, 10\}$. The added noise not only leads to finding fewer communities but also better ones, as the edge cut value for these new communities in the original graph is 1, a %30 decrease over the initial value of 1.3.

Noise-Enhanced Community Detection. In this paper, we investigate noise-enhanced community detection. We propose a simple framework to improve communities in an undirected unweighted network by adding noise, as outlined in Algorithm 1. Our approach is iterative (to account for noise randomness). In each iteration,

Algorithm 1 Noise-Enhanced Community Detection

Input: Graph G, Noise Injection Method `Noise`
Community Detection Method `CD`,
Objective Function `Obj`, `Iterations`.
Output: Noise-enhanced communities C_{best}
$C_0 \leftarrow \texttt{CD}(G)$
$O_0 \leftarrow \texttt{Obj}(C_0,\ G)$
$O_{\text{best}} \leftarrow O_0$
$C_{\text{best}} \leftarrow C_0$
for $i = 1$ `to Iterations` **do**
  $\tilde{G}_i \leftarrow \texttt{Noise}(G)$
  $C_i \leftarrow \texttt{CD}(\tilde{G}_i)$
  $O_i \leftarrow \texttt{Obj}(C_i, G)$
  if O_i improves compared to O_{best} **then**
    $O_{\text{best}} \leftarrow O_{\text{i}}$
    $C_{\text{best}} \leftarrow C_{\text{i}}$
return C_{best}

we build a noisy network by adding noise (edges) to the original graph, and we detect communities in this noisy network. We then evaluate the communities detected in the original graph using some objective function. We iterate a few times and return the best communities detected, e.g., with the highest objective function value, as the *noise-enhanced communities*. Compared to communities detected in the original graph, our goal is to detect better communities (in terms of some objective function) while injecting limited noise, i.e., without significantly increasing the community detection execution time. Overall, we aim to answer two questions:

Q1. Does adding noise improve the performance of community detection algorithms? If it does, to what extent injecting noise will improve detected communities?

Q2. Injecting noise increases the cost of finding communities. Are the improvements justifiable relative to the potential improvements to detect communities? What is the trade-off?

By addressing these questions, at a high level, our framework makes the following contributions:

- We introduce *noise-enhanced community detection*, a framework to improve current community detection methods by introducing noise;
- We introduce three methods to add noise to a graph. These methods can be used as a preprocessing step to improve existing community detection algorithms (Section 3);
- We provide a theoretical foundation for noise-enhanced community detection by proving that the suggested noise injection methods improve common community detection objective functions under different scenarios (Section 5); and
- We evaluate our framework on various real-world and synthetic networks using well-established community detection methods. Our results show that adding noise to networks allows one to detect better communities compared to those detected in the original graphs (Section 6).

2 LITERATURE REVIEW

While to the best of our knowledge, there has been no attempts to enhance community detection by adding noise, our framework broadly relates to research on (I) community detection and (II) noise-enhanced systems.

Community Detection. There are many techniques to detect communities in networks. Here, we focus on the more common community detection methods and group them into the following four categories (refer to Ref. [8] for a comprehensive review).

I. Hierarchical methods are suitable for networks with hierarchical structure, and can be grouped into: (1) agglomerative and (2) divisive methods [8]. Agglomerative methods iteratively merge communities with sufficiently high similarity and divisive algorithms iteratively split communities by removing edges that connect vertices with low similarity. A well-known divisive algorithm is the one proposed by Girvan and Newman [31], which splits communities by computing *edge betweenness*. The algorithm is computationally expensive, and cannot be applied to large networks. A well-known agglomerative method is FastGreedy, which iteratively merges groups of nodes by using a greedy technique [28]

II. Modularity-based methods optimize *modularity*, a community quality measure, to obtain better communities [8]. These methods mostly belong to one of four categories: (1) greedy methods [28], or those that are based on (2) simulated annealing [12], (3) extremal optimization [7], or (4) spectral optimization [29]. A fast greedy method from this category is the *Louvain* method [3], which hierarchically optimizes modularity.

III. Spectral methods rely on the rich foundation in spectral graph theory to detect communities [8]. A well-known example is the *Leading Eigenvector* method [29], which using spectral bisection splits a network into groups while minimizing edges between them.

IV. Dynamic methods detect communities by running a dynamical process on the network. These algorithms are mostly based on random walks [35], spin dynamics [39], or synchronization [1]. WalkTrap [35], a popular method from this category, relies on random walks to compute similarities between nodes, which in turn are used to detect communities via hierarchical clustering.

As our goal is to add an extra *noise injection* step to the community detection process, we experiment with well-known representatives from each community detection category as we will detail in our experiments.

Noise-Enhanced Systems. Adding noise enhances performance in many areas [5]. We review some here.

I. Stochastic Resonance (SR) is observed when increasing random noise improves signal detection performance [22]. SR is frequently used in noise-enhanced information systems with examples in biological, physical, and engineered systems [10, 23, 24].

II. Image Processing also benefits from noise enhancement. Adding noise to images before thresholding can improve the human brain's ability to perceive noisy visual patterns [42]. Adding noise can also improve image segmentation [16], image re-sampling detection [26], and image resizing detection [25]. As an example, adding an appropriate amount of noise can result in more accurate detection

of micro-calcifications in mammograms, which in turn can lead to early breast cancer diagnosis [44].

III. Signal Detection. Noise can help improve the detectability of signals. For example, when detecting a constant signal in a Gaussian mixture noise background, some white Gaussian noise can improve the performance of the sign detector [15]. Additive noise can also help detect a weak sinusoid signal more efficiently [46].

IV. Optimization. In search algorithms, when searching for an optimum is likely to get trapped in local minima, randomization helps finding optimal or near-optimal solutions. For example, the randomization in both crossover and mutation steps of Genetic Algorithms (GA) [43] helps avoid self-similarity in the population, i.e., helps avoid local minima [5]. The role of mutation is similar to adding noise and often a suitable mutation rate can improve performance.

V. Machine Learning. Noise can help reduce the convergence time in many clustering and competitive learning algorithms [32]. It can also decrease the convergence time of backpropagation algorithm, when training convolutional neural network [2]. This happens as backpropagation and some clustering algorithms such as k-means can be thought of as special cases of Expectation-Maximization (EM) algorithm [33], which improves by adding noise.

3 NOISE INJECTION METHODS

According to Algorithm 1, our framework follows three steps: first, some noisy edges are added to the graph. Then, communities are identified in the noisy graph using a community detection algorithm. Finally, the detected communities are evaluated using an objective function. Note that the objective function is evaluated on the original graph instead of the noisy graph to determine improvements due to noise (line 10 in Algorithm 1). To systematically analyze noise enhancement in community detection, we experiment with various community detection methods and objective functions. To add noise to graphs, we introduce three general ways.

Our noise injection methods focus on high degree nodes. In Section 5, we theoretically justify this decision where we show the importance of [disconnected] high degree nodes in detecting communities. Furthermore, empirical findings have shown that communities are more likely to contain more high degree nodes [27]. Informally, our results show that if a high degree node is not connected to another high degree node, by connecting it, we strengthen the within-community connections, and make it easier for community detection algorithms to detect the communities.

Therefore, in the proposed noise methods, we perform the following common steps: (1) we sort nodes based on their degrees, (2) we select the top p percent of sorted nodes as *candidates*, and (3) we add edges within candidates if those edges do not exist. To add edges, we select pairs of nodes (edge endpoints) from candidates. The proposed methods vary in how such pairs of nodes are selected.

I. Random Noise (`Random`). Edge endpoints are randomly chosen from the candidates. Before adding a noise edge, we ensure that it is not in the graph. If the number of candidates equals the number of nodes, `Random` simply connects nodes irrespective of their degrees.

II. Weighted Noise (`Weighted`). We select node (edge endpoint) v_i with probability $P_{\texttt{Weighted}}(v_i)$ that depends on its degree and that of other candidate nodes:

$$P_{\texttt{Weighted}}(v_i) = \frac{d_i}{\sum_{j=1}^{n} d_j},$$

where d_i denotes the degree of node v_i, and n is the number of candidate nodes.

III. Frequency Noise (`Frequency`). Nodes are selected based on the degree distribution of the candidates, where nodes whose degrees are more frequent, are less likely to be selected:

$$P_{\texttt{Frequency}}(v_i) = \frac{1 - f_{d_i}/n}{f_{d_i} \times \sum_{d=1}^{k}(1 - f_d/n)}, \tag{1}$$

where f_{d_i} is the frequency of degree d inside candidates and k is the maximum degree.

Time Complexity. The proposed noise methods include the following steps in a graph with $|V|$ nodes and $|E|$ edges: Computing node degrees in $O(|E|)$, sorting nodes based on their degrees in $O(|V|\log|V|)$, selecting *candidates* in $O(1)$, calculating node probabilities in $O(|V|)$, and adding noise edges based on node probabilities in $O(|E|)$ (The number of noise edges is at most $|E|$). So, the final time complexity introduced by adding noise is $O(|V|\log|V|)$.

Example 3.1. *Consider the graph shown in Figure 2 with* 6 *nodes* $\{a, b, c, d, e, f\}$ *with degrees* $\{4, 1, 3, 2, 2, 2\}$ *and* 7 *edges. We sort nodes based on their degrees* $\{a, c, d, f, e, b\}$. *Assume we select the top* 70% *of these sorted nodes as candidates:* $\{a, c, d, f\}$. *Finally, we select pairs of nodes among candidates as follows:*

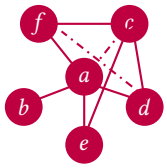

Figure 2: Sample Graph

`I. Random.` *Randomly selects pairs of nodes and connects them, e.g., it may add a noise edge between nodes* d *and* f *(dashed line in Figure 2).*

`II. Weighted.` *Nodes with higher degrees are more likely to be selected so node* a *with degree* 4 *has the highest probability. The probabilities* $P_{\texttt{Weighted}}(v_i)$ *for* v_i *in* $\{a, c, d, f\}$ *are* $\{4/11, 3/11, 2/11, 2/11\}$.

`III. Frequency.` *Nodes with less frequent degrees in candidates are more probable to be selected. Hence, edge* (a, c) *is the most likely noise edge as nodes* a, *and* c *with degrees* 4 *and* 3 *and* $f_4 = f_3 = 1$ *have the highest probability* $P_{\texttt{Frequency}}(a) = P_{\texttt{Frequency}}(c) = 3/8$ *to be chosen. The frequency of other candidate nodes are as follows:* $P_{\texttt{Frequency}}(d) = P_{\texttt{Frequency}}(f) = 1/8$.

4 EXPERIMENTAL SETUP

In this section, we detail the datasets, how noise quantity was controlled, candidates sizes selected, Q1 community detection methods used, objective functions, and evaluation metrics.

I. Datasets. We study the impact of noise on community detection in both synthetic and real-world networks:

(1) Synthetic Networks. To systematically verify our noise-enhanced framework, we should be able to analyze it in networks with different community structures. So, we use the well-established benchmark [17] proposed by Lancichinetti et al., which generates graphs with different degree distributions and community sizes distributions. To create such benchmark graphs, we need to set the value of several parameters, where we specifically followed suggestions provided by [17] and created 32 datasets. The parameters are in Table 2, where n is the number of nodes, γ is the exponent of the

Table 1: Real-world Datasets Statistics

Type	Network	$\|V\| = n$	$\|E\| = m$	Average Degree	Clustering Coefficient
Biological Network	Bio-Dmela [40]	7,393	25,569	6.917	0.0119
	Bio-Grid-Yeast [40]	6,008	156,945	104	0.163
Collaboration Network	CAGrqc [18]	5,242	14,496	5.526	0.5296
	Ca-HepTh [18]	9,877	25,998	5.259	0.4714
	CA-HepPh [18]	12,008	118,521	19.74	0.6115
Social Network	Fb-Athletes [18]	13,866	86,858	12.52	0.348
	Fb-Government [18]	7,057	89,455	25.35	0.4534
	Fb-Politician [18]	5,908	41,729	14.12	0.482
	Fb-Company [18]	14,113	52,310	7.41	0.399
	BlogCatalog [45]	10,312	333,983	64.77	0.4838
Road Network	Euro roads [40]	1,174	1,417	2.4	0.0167
	Minnesota roads [40]	2,642	3,303	2.5	0.01596
	California roads [20]	21,048	21,693	2.06	0.25

Table 2: Synthetic Datasets Statistics

Graph size (n)	β, γ	Mixing Parameter (μ)	Average Degree (K)
1,000	{[1, 2], [2, 3]}	{0.1, 0.5}	{5, 10, 15, 20}
10,000	{[1, 2], [2, 3]}	{0.1, 0.5}	{15, 20, 25, 30}

power law degree distribution, β is the exponent of the power law distribution for community sizes, μ is the mixing parameter where $1-\mu$ determines the fraction of links that each node shares with the other nodes in its community, and K is the average node degree.

(2) Real-world Networks: We also evaluate our framework with real-world networks. For systematic analysis, we use 13 real-world networks from four general category of networks: biological networks, collaboration networks, social networks, and road networks. Table 1 provides the statistics of these real-world networks.

II. Noise Proportion. We characterize noise in terms of the proportion of noise edges added, where noise increases the number of edges in the graph by e percent. For example, if there are 1,000 edges in the graph, we can add 2% of current edges (20 edges) to the graph. We vary e values from 1% to 10% with 1% increments.

III. Candidates Size. As noted in Section 3, we select the top p percent of sorted nodes as candidates. We vary p values from 10% to 100% with 10% increments. When $p = 100\%$, all nodes are candidates.

IV. Community Detection Methods. Based on the review in Section 2, we select four algorithms, each representing a category of community detection methods: (1) FastGreedy from hierarchical methods, (2) Louvain from modularity-based methods, (3) Leading EigenVector from spectral methods, and (4) Walktrap from dynamic methods. All selected methods have shown great performance in extracting high quality communities.

V. Objective Functions. There are two groups of objectives functions for evaluating quality of communities (see Ref. [19] for details): (1) *multi-criterion* scores, which consider both edges inside the communities and those crossing communities, and (2) *single criterion* scores, which either consider inside edges, or crossing edges. To evaluate the quality of communities, we select two objective functions from each of the aforementioned categories; conductance [14, 41] and normalized cut [41] from multi-criterion scores, modularity [30] and edge cut [19] from single criterion scores.

VI. Evaluation Metrics. To assess noise enhancement, we measure the following for each objective function:

- `Expected First Success (EFS)` is the expected number of times (`Iterations` in Algorithm 1) that we need to add noise to the network to ensure that we improve communities at least once. For example, if we run Algorithm 1 for 100 iterations and improve communities in 34 of these iterations, the Expected First Success is 3 as $\frac{100}{34} \simeq 2.94$. Formally, for objective function obj:
$$\texttt{EFS}_{obj} = \left\lceil \frac{\text{number of iterations}}{\text{noise-enhanced iterations}} \right\rceil$$
- `Relative Objective Improvement (ROI)` is the relative objective value improvement after adding noise:
$$\texttt{ROI}_{obj} = \frac{obj_{\text{noise-enhanced}} - obj_{\text{original}}}{obj_{\text{original}}} \times 100$$

Before performing experiments, we show that these objective functions can in theory be improved next.

5 THEORETICAL ANALYSIS

In this section, we theoretically analyze noise-enhancement benefits in community detection. We demonstrate that (1) adding a noise edge between two high degree nodes will increase the chance of re-partitioning a graph by assigning the two high degree nodes to the same community in terms of the minimum normalized cut, for which we provide a spectral analysis. As a result, we show that (2) once a high degree node moves to the community of the other, all objective functions in this study improve under some constraints. Our setting can be generalized to multiple communities and multiple nodes in these communities.

5.1 Spectral Analysis

We first provide a spectral analysis of adding an edge to a graph. Before delving into the details, we quickly review the concepts of the normalized Laplacian and the random walk transiton matrix.

Normalized Laplacian Matrix. For an undirected graph $G = (V, E)$, the normalized Laplacian of G is the matrix $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where A is its adjacency matrix and D is its degree matrix. A normalized Laplacian has a bounded spectrum, i.e. $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n \leq 2$, where λ_i's are the eigenvalues of L. The Laplacian matrix have been used to investigate many useful properties of a graph. Especially, by Cheeger's inequality [4], the minimum normalized cut of a graph is bounded by the second smallest eigenvalue of the normalized Laplacian, i.e. λ_2. More specifically, the Cheeger constant (the subset with the smallest conductance) $h(G)$ satisfies:

$$\lambda_2 \leq h(G) = \min_{S \subset V} \frac{|(x, y) \in E, x \in S, y \notin S|}{min(vol(S), vol(V \setminus S))} \leq \sqrt{2\lambda_2} \quad , \qquad (2)$$

where $vol(S)$ is the volume of S, the sum of degrees of nodes in S.

Random Walk Transition Matrix. The transition matrix of the random walk on $G = (V, E)$ is matrix $P = AD^{-1}$. As P is a stochastic matrix, its spectrum is also bounded: $1 = \mu_1 \geq \mu_2 \geq \cdots \geq \mu_{n-1} \geq \mu_n \geq -1$, where μ_i's are the eigenvalues of P. Matrix P is *similar* to $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ (i.e., they have the same eigenvalues) and $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, so $\mu_i = 1 - \lambda_i$, for $1 \leq i \leq n$. In this work, we define

the ℓ-th spectral moment m_ℓ of a graph G using the spectrum of its P, $m_\ell = \mathbb{E}(\lambda^\ell)$, as $\frac{1}{n}\sum_{i=1}^{n}\lambda_i{}^\ell = \mathbb{E}(\lambda^\ell)$. The spectral moments of P have a property: m_ℓ is equal to the expected return probability of a random walk of step ℓ starting at a node i where i is chosen uniformly at random from V. In our previous work [13], we prove that $m_2 = \mathbb{E}(d_i)\,\mathbb{E}(\frac{1}{d_i d_j})$, where $\mathbb{E}(d_i)$ denotes the average degree in the graph and $d_i d_j$ follows the joint degree distribution $p(d_i, d_j)$: the probability that a node with degree d_i is connected to one with degree d_j (refer to Ref. [13] for the proof details). To show how adding a noise edge impacts the minimum normalized cut, we first demonstrate how the spectral moment changes:

Theorem 5.1. *Let graph $G' = (V', E')$ be obtained from graph $G = (V, E)$ by connecting nodes u and v. Then, $m_2' - m_2 = \frac{2}{(d_u+1)(d_v+1)} \cdot (1 - \mathbb{E}_{x:x\sim u}(\frac{d_v+1}{d_x}) - \mathbb{E}_{y:y\sim v}(\frac{d_u+1}{d_y}))$, where d_u, d_v, d_x, d_y denote the degree of the nodes u, v, x and y in G, respectively, and $x \sim u$ denotes that x is u's neighbor in G and $y \sim v$ denotes that y is v's neighbor.*

Proof. Clearly, $|V'| = |V|$ and $|E'| = |E| + 1$. Based on our previous work [13], $m_2' = \mathbb{E}'(d_i)\,\mathbb{E}'(\frac{1}{d_i d_j})$ and $m_2 = \mathbb{E}(d_i)\,\mathbb{E}(\frac{1}{d_i d_j})$. As only one edge is added, $\mathbb{E}'(d_i) = \frac{2|E'|}{|V'|} = \frac{2|E'|}{|V|}$. For $\mathbb{E}'(\frac{1}{d_i d_j})$, we notice that the change is in two parts: (1) the newly added edge (u, v) which contributes $\frac{1}{(d_u+1)(d_v+1)}$, and (2) for those edges which are incident to u, as the degree of u turns from d_u to $d_u + 1$, the overall difference is $\sum_{x:x\sim u}(\frac{1}{d_x d_u} - \frac{1}{d_x(d_u+1)}) = \sum_{x:x\sim u}(\frac{1}{d_u(d_u+1)d_x}) = d_u\,\mathbb{E}_{x:x\sim u}(\frac{1}{d_u(d_u+1)d_x}) = \frac{\mathbb{E}_{x:x\sim u}(\frac{1}{d_x})}{d_u+1}$. Similarly, for those edges which are incident to v, the differences are $\frac{\mathbb{E}_{y:y\sim v}(\frac{1}{d_y})}{d_v+1}$. Therefore, $\mathbb{E}'(\frac{1}{d_i d_j}) = \frac{|E|\,\mathbb{E}(\frac{1}{d_i d_j}) + \frac{1}{(d_u+1)(d_v+1)} - (\frac{\mathbb{E}_{x:x\sim u}(\frac{1}{d_x})}{d_u+1} + \frac{\mathbb{E}_{y:y\sim v}(\frac{1}{d_y})}{d_v+1})}{|E'|}$. By simplifying the algebra and using $m_2' = \mathbb{E}'(d_i)\,\mathbb{E}'(\frac{1}{d_i d_j})$, we get $m_2' - m_2 = \frac{2}{(d_u+1)(d_v+1)} \cdot (1 - \mathbb{E}_{x:x\sim u}(\frac{d_v+1}{d_x}) - \mathbb{E}_{y:y\sim v}(\frac{d_u+1}{d_y}))$. □

Obviously, if one adds a noise edge connecting two nodes u and v which are in the same subset of the current minimum cut, its conductance ($\phi(S) = \frac{|(x,y)\in E, x\in S, y\notin S|}{min(vol(S), vol(V\setminus S))}$) decreases, making the Cheeger constant of G': $h(G') \le h(G)$. However, from Theorem 5.1, when $1 - \mathbb{E}_{x:x\sim u}(\frac{d_v+1}{d_x}) - \mathbb{E}_{y:y\sim v}(\frac{d_u+1}{d_y}) < 0$, m_2 decreases. Notice that the term $\mathbb{E}_{x:x\sim u}(\frac{d_v+1}{d_x})$ is greater than 1 if the degree of node v is generally larger than that of the neighbors of u. As a special case satisfying the condition, connecting two high degree nodes will decrease m_2. As we only look into undirected graphs without self-loops, it is easy to see that the first spectral moment m_1 is 0. The decrease of m_2 indicates μ_i's going towards zero. As a result, μ_2 will more likely decrease while $\lambda_2 = 1 - \mu_2$ will increase, which makes the bounds of the Cheeger constant in Equation 2 (λ_2 and $\sqrt{2\lambda_2}$) greater, which is the opposite to the decrease from $h(G)$ to $h(G')$. This observation indicates that *two disconnected high degree nodes are more likely to be in different subsets of the minimum cut.*

Therefore, connecting high degree nodes increases the conductance of the current minimum cut, but decreases conductance of cuts which assign the high degree nodes into the same subset. Hence, our noise-enhanced method increases the chance of re-partitioning a graph by assigning these high degree nodes into the same subset in terms of minimum normalized cut. Next, we show if this happens, i.e., one high degree node moving to the community of the other, all objective functions in this study improve under some constraints.

5.2 Objective Function Analysis

To simplify, consider graph G with communities S_1 and S_2. Denote disconnected high degrees node v_i in S_1 with degree d_i and v_j in S_2 with degree d_j. Assume $d_i = d_{i,in} + d_{i,out}$ (similarly, $d_j = d_{j,in} + d_{j,out}$), where $d_{i,in}$ ($d_{j,in}$) is the number of edges between v_i (v_j) and nodes in S_1 (S_2), and $d_{i,out}$ ($d_{j,out}$) is the number of edges between v_i (v_j) and nodes in S_2 (S_1). Let m_{S_2} denotes the number of edges in S_2, and m_{S_1} the number of edges in S_1. We use prime symbol ($\prime$) to denote the updated objective function value, e.g., *conductance*$'$. We assume that the high degree node that moves to the community of the other is the one that has comparatively less within-community connections than cross-community connections, e.g., v_i moves to S_2 when $d_{i,in} < d_{i,out}$. We show that all objective functions can be improved under this condition.

Theorem 5.2 (Modularity Change). *If $d_{i,in} < d_{i,out}$, moving v_i from S_1 to S_2 increases modularity.*

Proof. Modularity [30] measures the density of edges inside communities compared to edges between communities, and can be obtained by $\frac{1}{2m}\sum_{ij}[A_{ij} - \frac{d_i d_j}{2m}]\delta(c_i, c_j)$, where A is the adjacency matrix of graph G, d_i and d_j are degrees of nodes i and j, m is the number of edges in the graph, and $\delta(c_i, c_j) = 1$ if both vertices i and j belong to the same community; otherwise, it is 0. If v_i moves to S_2, the modularity of each community changes as follows: (1) S_2: the modularity of v_i's neighbors in S_2 will increase from zero to a positive value $1 - \frac{d_i d_j}{2m}$ as the delta function now becomes $\delta(c_i, c_j) = 1$ for all such neighbors j. On the other hand, the modularity of nodes that are not connected to v_i changes from zero to negative values $-\frac{d_i d_j}{2m}$. When $d_{i,out} > m_{S_2}$, the overall modularity of nodes in S_2 will be increased. (2) S_1: the modularity of nodes which are connected to v_i will be decreased (positive values are changed to zero), but modularity of nodes that were not connected to v_i are increased (negative values are changed to zero). When $d_{i,in} < m_{S_1}$, the modularity of nodes in S_1 will be increased. Since adding v_i to S_2 only affects the modularity of S_1 and S_2, and they are increased, the overall modularity of G will be also increased. □

Theorem 5.3 (Edge cut Change). *If $d_{i,in} < d_{i,out}$, moving v_i from S_1 to S_2 decreases edge cut.*

Proof. Edge cut, c_S, is the number of crossing edges between community S and other nodes in G [19]. As we only consider S_1 and S_2, $c_{S_1} = c_{S_2}$, which we simplify as c_S. If we move v_i to S_2, $d_{i,out}$ edges will be counted as edges in S_2, and $d_{i,in}$ edges will be considered as new crossing edges between S_1 and S_2. So, the new edge cut values are $c'_{S_1} = c'_{S_2} = c_S - d_{i,out} + d_{i,in}$. When $d_{i,in} < d_{i,out}$, edge cut is decreased. □

Theorem 5.4 (Conductance Change). *If $d_{i,in} < d_{i,out}$, moving v_i from S_1 to S_2 decreases conductance.*

Proof. Conductance for community S can be calculated as: $Conductance_S = \frac{c_s}{2m_s + c_s}$ [14, 41]. If we move v_i to S_2, $d_{i,out}$ edges

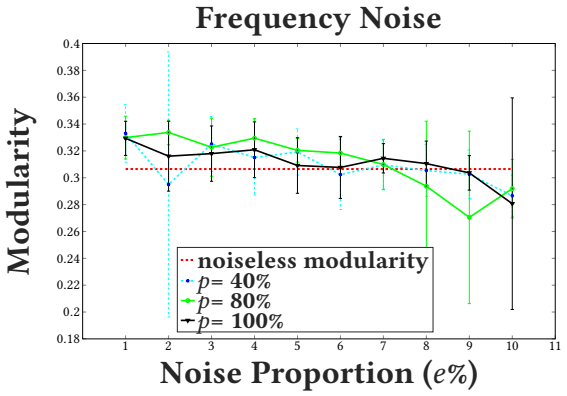

Figure 3: Modularity of noise-enhanced Leading EigenVector on Bio-Dmela dataset when using frequency noise and with candidates size $p \in \{40\%, 80\%, 100\%\}$.

will be counted as edges in S_2, so $m'_{S_2} = m_{S_2} + d_{i,out}$, and $m'_{S_1} = m_{S_1} - d_{i,in}$. So, $Conductance'_{S_2} = \frac{c'_{S_2}}{2m'_{S_2}+c'_{S_2}} = \frac{c_S - d_{i,out} + d_{i,in}}{2m_{S_2}+c_S+d_{i,out}+d_{i,in}}$ and $Conductance'_{S_1} = \frac{c'_{S_1}}{2m'_{S_1}+c'_{S_1}} = \frac{c_S - d_{i,out} + d_{i,in}}{2m_{S_1}+c_S+d_{i,out}-d_{i,in}}$. When $d_{i,in} < d_{i,out}$, conductances of S_1 and S_2 will decrease. □

Theorem 5.5 (Normalized Cut Change). *If $d_{i,in} < d_{i,out}$, moving v_i from S_1 to S_2 decreases cut size.*

Proof. Normalized cut (Ncut) for community S can be calculated as $Ncut_S = \frac{c_s}{2m_s+c_s} + \frac{c_s}{2(m-m_s)+c_s}$ [41]. The first term is basically conductance, and conductance is decreased based on Theorem 5.4. So, $Ncut'_{S_2} = \frac{c'_{S_2}}{2m'_{S_2}+c'_{S_2}} + \frac{c'_{S_2}}{2(m-m'_{S_2})+c'_{S_2}} = Conductance'_{S_2} + \frac{c_{S_2} - d_{i,out} + d_{i,in}}{2(m-m_{S_2})+c_{S_2}-3d_{i,out}+d_{i,in}}$, and $Ncut'_{S_1} = \frac{c'_{S_1}}{2m'_{S_1}+c'_{S_1}} + \frac{c'_{S_1}}{2(m-m'_{S_1})+c'_{S_1}} = Conductance'_{S_1} + \frac{c_{S_1} - d_{i,out} + d_{i,in}}{2(m-m_{S_1})+c_{S_1}-d_{i,out}+3d_{i,in}}$. Hence, when $kd_{i,out} < d_{i,in} < d_{i,out}$ and $1/3 \le k < 1$, Ncut for S_1 decreases. □

6 EXPERIMENTAL ANALYSIS

We evaluate the impact of adding noise on communities detected in real-world and synthetic networks.

6.1 Noise-enhanced Community Detection in Real-World Networks.

We start with an example. Figure 3 shows the modularity of noise-enhanced Leading EigenVector method on Bio-Dmela dataset using Frequency method for various candidate sizes, $p \in \{40\%, 80\%, 100\%\}$, and noise proportions e. For stability, for each e, we run the experiments 10 times and compute the average modularity value. Error bars denote one standard deviation. The dashed horizontal red line shows the original modularity obtained in the noiseless Bio-Dmlea. As shown, most obtained modularity values are above the original modularity in the noiseless graph, implying that better communities are found with respect to modularity by adding noise.

Figure 3 demonstrates the feasibility of noise-enhanced community detection. Hence, we further design experiments to systematically assess the impact of noise on all graphs. For each graph and each candidate size p, we use noise methods to add different proportions of noise e and measure our evaluation metrics (EFS and ROI) for all objective functions and community detection methods. For each e, we run the experiments 10 times to assess stability of the results. As p varies from 10% to 100% with 10% increments and e varies from 1% to 10% with 1% increments, for each p, we perform 100 experiments, and for all p, we perform 1,000 experiments. For each dataset and objective function, these results with respect to both evaluation metrics (ROI and EFS) can be summarized using 6 plots as shown as an example in Figure 4. The figure shows the effect of all three proposed noise methods on conductance of communities detected in HepPH. As shown in the Figure, EFS is on average 2 for Louvain, FastGreedy, and Leading Eigenvector community detection methods, so on average, one only needs to add noise twice to improve detected communities. ROI for conductance is also shown in Figure 4. As all these noised-enhanced methods can improve conductance (negative ROI value), for clarity, we show the absolute values of ROI in the figure. We observe that the conductance of communities detected by Leading Eigenvector is much higher than that of those detected by other methods. Overall, our experiments lead to $13 \times 3 \times 4 = 156$ figures for all datasets, community detection methods, noise methods, and objective functions.

For space reasons, we summarize our results in Table 3, which provides the average EFS and ROI of noise-enhanced community detection methods on all real-world networks. For each network, there are four rows, one for each objective function. For a given data set, noise method, and community detection method, we provide both EFS and ROI. Positive ROI values for modularity indicate improvements (better communities) and negative ROI values for conductance, edge cut, and normalized cut indicate improvements. Gray cells indicate that the specific community detection method did not improve with the specific noise method, and pink cells indicate that the community detection method was unable to identify communities (often due to computational complexity). We summarize the findings in Table 3 as follows:

- Noise-enhanced community detection obtains an average EFS = 43 and ROI = 10.5 (absolute ROI values are considered for calculating the averages) over all networks, community detection methods, noise methods, and objective functions, indicating that noise often improves detected communities;
- Weighted Noise is the best noise method for improving communities in biological networks, social networks, and collaboration networks, and Random Noise is the best choice for road networks;
- Louvain methods improves the most by adding noise compared to other community detection methods;
- While all community detection method in general improve, each method improves best with a specific type of noise: (1) Louvain improves more with Weighted Noise and Frequency Noise, where for Weighted: (EFS = 21, ROI = 3.45), and for Frequency: (EFS = 29, ROI = 5.3). These numbers are the average [absolute] values for modularity, conductance, edge cut, and normalized cut; (2) Leading EigenVector improves more with Random Noise and Weighted Noise, where for Random: (EFS = 25, ROI = 27.2), and for Weighted: (EFS = 37, ROI = 31.4); (3) Fastgreedy improves more with Weighted Noise and Frequency Noise, where for Weighted: (EFS = 29, ROI = 5.05), and for Frequency: (EFS =

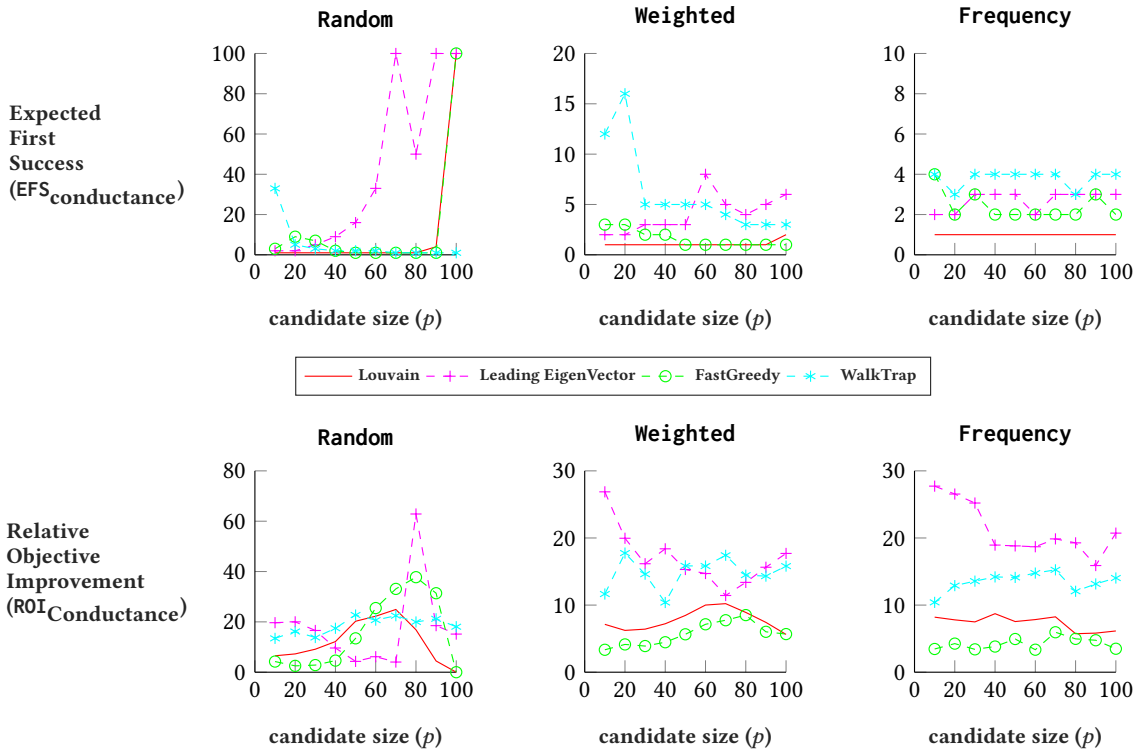

Figure 4: Effect of all three proposed noise methods on the conductance of communities detected in HepPH. The first and the second row show EFS and ROI for conductance. For HepPH, EFS=2 on average for Louvain, FastGreedy, and Leading Eigenvector and ROI of communities detected by Leading Eigenvector is much higher than that of those detected by other methods.

40, ROI = 6.5); and (4) WalkTrap improves best with Random Noise, where for Random: (EFS = 40, ROI = 5.05); and

- Adding noise often cannot help detect better communities in road networks. This happens as there are not as many high degree nodes in road networks as there are in other types of networks, where the proposed noise methods rely on high degree nodes.

Based on results obtained in Section 5.1, we also added a constraint when selecting candidates where we enforced the degree of each node among candidates to be higher than the average degree of its neighbors. Then we sort nodes in candidates based on their degrees and add noise edges by connecting pairs of candidates. Table 4 shows the average EFS and ROI of adding this new constraint to our noise enhanced community detection methods on 4 real-world datasets (for each category in Table 1, one dataset is chosen). As the results are similar to the results in Table 3 and connecting nodes with high degrees is easier to implement, the results following are based on connecting high degree nodes.

Impact of Limited Noise. We note that our experiments show that even when we add a few edges to large networks, one can detect better communities. As an example, EFS and ROI for modularity after adding 1,000 edges to Facebook-Company network is shown in Table 5. The Table shows that if we use any of Louvain, FastGreedy, or Leading EigenVector methods where noise is on average added four times to this dataset (EFS=4), we can detect better communities: ROI value will be at least %0.23.

6.2 Noise-enhanced Community Detection in Synthetic networks.

We start with an example. Figures 5 and 6 provide the EFS of modularity and edge cut after applying Louvain on synthetic networks with $n = 1,000$ nodes and $n = 10,000$ nodes, respectively. The points with EFS> 100 are shown at the top of the charts. As Figure 5 shows, when $\mu = 0.5$ (a balance between number of edges inside communities and outside of them), Louvain is able to improve modularity with low values of EFS, while when $\mu = 0.1$, as 90% of edges are inside communities and our approach is based on adding noise edges, Louvain is less likely to improve modularity. This figure shows when graphs become denser (increasing K), EFS for edge cut increases. Figure 6 shows when graphs become larger ($n = 10,000$) and denser (increasing K), EFS for modularity decreases on average. All chart in both Figures 5 and 6 show that noise enhanced Louvain is able to highly improve modularity and edge cut EFS for synthetic networks with $\mu = 0.5$, $\beta = 1$, and $\gamma = 2$. Table 6 provides the average of EFS and ROI for synthetic networks. The first and second four rows show statistics on objective functions for $n = 1,000$ and $n = 10,000$, respectively. Each element in Table 6 shows the average EFS or ROI for 16 synthetic networks with the same number of nodes (N) and different μ, β, and γ values. As Table 6 shows noise-enhanced community detection is able to improve all objective functions in terms of both EFS and ROI. WalkTrap

Table 3: Expected First Success (EFS) and Relative Objective Improvement (ROI) of noise-enhanced community detection methods on real-world networks. These numbers show that: (1) Louvain improves more with Weighted Noise and Frequency Noise, (2) Leading EigenVector improves more with Random Noise and Weighted Noise, (3) FastGreedy improves more with Weighted Noise and Frequency Noise, and (4) WalkTrap improves best with Random Noise.

Network Dataset	Objective Function	Random Noise								Weighted Noise								Frequency Noise							
		Louvain		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*	
		EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI
Bio-Dmela	Modularity	8	0.7	2	7.4	1000	0.01	50	1	3	0.76	2	6.7	500	0.04	27	1.1	5	0.9	2	6.3	1000	0.01	12	1.3
	Conductance	2	-4.5	11	-12	2	-3.1	26	-4.6	2	-5.3	16	-12.9	2	-3.4	23	-3.7	2	-6.8	20	-9.8	2	-4	17	-4.9
	Edge Cut	5	-6.4	2	-58.2	2	-16.5	2	-21.2	3	-7	2	-56.3	2	-21.7	2	-23.6	2	-9.4	3	-5.8	2	-26.8	2	-25
	NCut	4	-4.1	10	-6.6	8	-3.1	32	-5.2	2	-5	14	-10	2	-3.5	28	-4.7	2	-6.5	70	-1.08	2	-4.5	17	-4.4
Bio-Grid-Yeast	Modularity	13	0.22	5	0.68	37	1.1	9	0.9	5	0.22	8	0.8	50	0.67	12	0.92	4	0.25	4	1.15	37	0.76	15	1
	Conductance	2	-10.5	23	-4.4	47	-6.9	13	-0.34	2	-11	28	-2.8	45	-6.8	4	-0.07	2	-10.9	17	-2.8	44	-7	3	-0.09
	Edge Cut	4	-5.3	3	-5.6	2	-26.3	2	-12	3	-5.7	4	-5.9	2	-23.9	2	-12	3	-5.9	2	-5.3	2	-22.8	2	-11
	NCut	3	-9.3	50	-9	55	-2.8	28	-0.32	2	-9.6	51	-5	52	-6	5	-0.07	2	-9.4	200	-10	55	-3.6	4	-0.09
CAGrqc	Modularity	12	0.024			42	0.18	20	0.29	40	0.012			9	0.24	10	0.16	7	0.02			8	0.25	26	0.11
	Conductance	17	-4.2	27	-12	13	-7.3	20	-2.8	3	-4.01	3	-35	4	-6.7	65	-2.5	2	-4.6	3	-46	7	-5.8	500	-0.08
	Edge Cut	66.6	-0.036	34	-10.7	15	-1.6	14	-1.3	27	-0.24	110	-3.2	18	-2	12	-1.1	46	-0.34	25	-14.3	10	-2.3	25	-0.56
	NCut	37	-2.7	4	-35	24	-7.2	25	-2.2	8	-4.3	6	-17.1	5	-6.7	10	-1.22	3	-4.5	4	-44	8	-6.4	333	-0.28
Ca-HepTh	Modularity	27	0.11	10	113	10	0.39	500	0.04	15	0.15	2	100	4	0.38	334	0.1	56	0.13	4	117	5	0.39	334	0.03
	Conductance	11	-3.7	45	-17.7	4	-9.3	30	-3.4	7	-3.3	72	-35.1	2	-7.7	35	-3.5	15	-3.4	4	-12.7	2	-6.5	200	-1.4
	Edge Cut	25	-0.28	125	-7.2	58	-0.8	3	-3.8	70	-0.35	250	-28	28	-1	2	-5	3	-3.4	64	-12	8	-1.4	2	-4.8
	NCut	30	-2.5	40	-8.7	12	-10	28	-3.3	8	-3.2	110	-16.5	3	-8	60	-3.1	20	-3	6	-11.9	2	-6.6	500	-1.9
CA-HepPh	Modularity	4	0.45	70	0.48	7	0.96	16	1.15	2	0.5	250	0.19	3	1.2	50	0.55	2	0.64	51	0.25	3	0.95	100	0.04
	Conductance	3	-12	42	-12	12	-16.1	6	-18	2	-7	5	-17	3	-5.2	7	-15.1	2	-7.7	4	-19	3	-4	5	-14
	Edge Cut	70	-0.4	25	-1.2	50	-2.1	4	-9.6	30	-2.4	23	-5.6	4	-3.6	2	-13.9	11	-2.6	8	-6	2	-3.9	2	-13.9
	NCut	3	-13	36	-13.9	4	-17.4	13	-20.3	3	-7.4	6	-15.9	3	-5.7	8	-14.9	2	-7.1	4	-20	3	-3.7	5	-14.1
Fb-Athletes	Modularity	2	0.82	2	27.6	2	2.3	5	0.54	2	0.9	2	17.1	2	1.4	4	0.55	2	90	2	12.4	2	1.1	5	0.58
	Conductance	3	-3.1	47	-33	2	-5.7	22	-1.4	2	-3.2	21	-41	2	-5.3	15	-1.3	2	-3.7	42	-18	2	-3.8	16	-1.4
	Edge Cut	7	-4.9	3	-52.1	334	-0.86	4	-11.6	3	-5.7	2	-43.5	154	-2.3	2	-14.4	2	-9.7	2	-42.9	112	-1.53	2	-12.6
	NCut	4	-2.7	36	-31	3	-6.1	25	-1.5	3	-3.1	46	-58	2	-5.2	21	-1.5	2	-3.6	70	-20	2	-3.9	21	-1.1
Fb-Government	Modularity	15	0.02	28	0.85	2	2.7	4	0.43	10	0.02	40	0.81	2	2.6	8	0.34	9	0.02	41	107	2	2.6	9	0.3
	Conductance	3	-2.7	4	-17.9	3	-12.2	13	-2.8	3	-2.7	2	-16.2	3	-14.2	500	-0.31	3	-2.7	2	-15.4	3	-11.5	1000	-0.02
	Edge Cut	16	-2.7	5	-26.5	9	-16.3	2	-21.9	4	-2.6	15	-26.6	3	-16.5	2	-25.7	4	-2.8	11	-25.2	3	-12	2	-25.5
	NCut	3	-2.8	5	-10.2	3	-11.2	12	-2.88	3	-25	2	-9.8	3	-10.6	200	-1.2	4	-2.5	2	-10.5	3	-11.1	1000	-0.11
Fb-Politician	Modularity	14	0.01	8	30.3	5	0.44	5	0.46	13	0.01	5	26.4	4	0.44	16	0.21	10	0.01	2	34.6	2	0.43	44	0.11
	Conductance	4	-2.4	2	-16	14	-4.4	55	-0.46	2	-2.3	3	-15	10	-4.3			2	-2.7	5	-17	7	-3.1		
	Edge Cut	32	-1.2	20	-20.8	5	-15.2	5	-8	10	-1.5	20	-24	3	-16.6	2	-9.9	19	-1.5	5	-25	2	-13.1	2	-8.2
	NCut	5	-2.6	4	-12.1	15	-4.1	58	-0.39	2	-2.4	5	-11.5	10	-4.4			2	-2.6	7	-13.3	9	-3.8		
Fb-Company	Modularity	10	0.2	2	48.4	8	0.58	56	0.26	5	0.24	2	54.7	5	0.47	66	0.29	12	0.29	2	44.5	5	48	250	0.11
	Conductance	9	-2	100	-0.48	7	-9.8	20	-3.5	3	-2.4	40	-100	6	-5.7	19	-3	5	-3.1	11	-100	5	-1.9	11	-2.7
	Edge Cut	28	-4.1	3	-33	34	-2.2	2	-8.6	10	-4.3	2	-47	17	-3.9	2	-11	3	-12.4	2	-53	10	-3.1	2	-14.5
	NCut	24	-1.7	70	-34	11	-9.7	19	-4	5	-2	29	-92	7	-6.1	25	-3.7	5	-2.9	9	-98	6	-24	11	-2.9
BlogCatalog	Modularity	63	0.07	2	0.06	2	0.88	2	6.8	45	0.08	2	0.09	2	0.98	2	7.3	40	0.08	2	0.08	2	0.96	2	7.7
	Conductance	3	-6.2	2	-0.4	3	-2.5	3	-0.3	2	-6.5	3	-0.38	2	-2.9	3	-0.25	2	-6.6	2	-0.8	2	-3.2	3	-0.32
	Edge Cut	30	-8.6	6	-0.03	15	-13.7	4	-18	3	-7.4	3	-0.05	29	-6.5	2	-19	3	-8.2	5	-0.04	49	-6	2	-21
	NCut	3	-3.1	4	-0.04	22	-1.3	8	-0.28	2	-3	3	-0.05	2	-1.8	3	-0.25	2	-2.1	4	-0.05	2	-1.9	2.6	-0.27
California-roads	Modularity			3	120							3	129							3	36.2				
	Conductance			26	-100							47	-100							1000	-1				
	Edge Cut			37	-70			2	-9.4			44	-85			2	-9.6			334	-3			2	-13.3
	NCut			49	-100							40	-100							334	-3				
Euro-roads	Modularity	500	0.002			32	0.23	66	0.24	155	0.015			16	0.25	84	0.11	13.2	0.17			9	0.27	56	0.16
	Conductance	13	-7.6			22	-4.2	31	-2.1	32	-4.1	40	-12.2	15	-4.4	41	-2.6	20	-3.6	33	-9.6	10	-4.2	62	-3.2
	Edge Cut	33	-2.4	58	-72	144	-0.81	40	-1.6	70	-0.9	77	-50	91	-1.17	28	-1.8	100	-1.4	250	-20.6	200	-0.29	10	-2.6
	NCut	18	-6.1	10	-33	28	-5.5	50	-1.7	84	-1.19	250	-13.3	22	-4.7	42	-2.5	7	-5.4	24	-22.2	9	-4.7	46	-2.8
Minnesota-roads	Modularity	100	0.004	29	30224	63	0.38			144	0.08	2	102000	72	0.09			100	0.006	2	95000	91	0.04		
	Conductance	156	-0.18			55	-2.8	500	-0.96	60	-1.9			50	-2.4	500	-1.25	100	-0.04			55	-33	125	-1.7
	Edge Cut	65	-0.9			52	-1.4	2	-15.6					50	-1.9	2	-16.7	500	-0.28			52	-2.3	3	-7.3
	NCut					65	-2.1			65	-1.2			58	-1.2			200	-0.79			54	-1.9	200	-1.16

often does not highly improve when adding noise to large graphs, especially for modularity objective function.

6.3 Ground-truth Communities

While community detection methods based on modularity optimization have shown to be effective in identifying communities in real-world and synthetic networks, modularity optimization may fail to detect communities that are smaller than a scale due to its *Resolution Limit* [9]. Hence, we also use *Normalized Mutual information (NMI)* as the objective function to compare the communities detected by our noise-enhanced framework and ground-truth communities. We use two datasets with ground truth communities: (1) email-Eu-core [18] with 1K nodes and 13K edges and (2) DBLP [21] with 13K nodes and 56K edges. Table 7 provides the EFS and ROI for the NMI. The results show improvements in NMI for the proposed noise enhanced community detection framework compared to the existing community detection methods. In particular, all proposed noise methods can help detect better communities, where Random Noise yields EFS=7, and ROI=2.42, Weighted Noise results in EFS=7, and ROI=2.9, and Frequency Noise obtains EFS=16, and ROI=4.8.

7 CONCLUSION

We introduced a framework to enhance community detection by adding noise to networks. The approach adds a preprocessing step to the current community detection methods as a noise injection step. For noise injection, three methods were proposed that randomly add noise edges to the network, focusing on high degree nodes. Our theoretical and extensive empirical results show that this approach leads to finding better communities using current community detection methods not only by detecting communities that are better in terms of an objective functions but also by detecting communities that are more similar to the ground-truth.

Table 4: Expected First Success (EFS) and Relative Objective Improvement (ROI) of noise-enhanced community detection methods on 4 real-world networks after adding the constraint that the degree of each node in candidates is higher than the average degree of its neighbors.

Network Dataset	Objective Function	Random Noise								Weighted Noise								Frequency Noise							
		Louvain		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*	
		EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI
Bio-Dmela	Modularity	4	0.68	2	7.3			8	1.4	4	0.99	2	6.7	251	0.01	210	1.5	6	0.91	2	6.4			10	1.3
	Conductance	2	-5.4	14	-9.9	2	-3.6	17	-3.4	2	-6.8	16	-8.5	2	-3.7	18	-6.09	2	-3.3	18	-7.5	3	-2.8	9	-3.7
	Edge Cut	2	-9.5	2	-57.3	2	-28.1	2	-24	2	-10.5	2	-60.4	2	-30.5	2	-26	3	-5.8	2	-56.2	2	-28.6	2	-23.2
	NCut	2	-5.8	8	-6.4	2	-3.8	10	-4.5	2	-6.8	11	-8.3	2	-4.25	13	-4.7	2	-3.8	66	-1	3	-2.9	8	-4.2
CAGrqc	Modularity	36	0.08			30	0.25	13	0.16	13	0.01			18	0.25	39	0.13	8	0.02			16	0.28	501	0.05
	Conductance	2	-4.3	2	-43.3	7	-4.9	6	-2	2	-4.1	3	-43.4	9	-6.3	20	-0.72	3	-3.1	2	-51.7	11	-6.4		
	Edge Cut	23	-0.21	15	-25	11	-1.5	3	-2.1	10	-0.24	21	-2.1	9	-1.8	4	-1.4	17	-0.12	17	-2.6	24	-1.35	12	-0.84
	NCut	2	-3.5	3	-32	9	-5.9	6	-1.8	2	-4.2	6	-15.6	9	-5.7	26	-0.97	4	-2.8	4	-39.3	12	-6.3		
Fb-Athletes	Modularity	2	0.88	3	12.1	2	1.12	6	0.51	2	0.85	3	13.8	2	1.2	6	0.55	2	0.72	3	12.8	2	0.97	7	0.67
	Conductance	3	-3.03	6	-26.4	2	-4.4	18	-1.5	2	-3.5	16	-15.3	2	-4.3	15	-1.09	3	-3.05	42	-0.69	3	-3.6	14	-1.2
	Edge Cut	3	-6.7	2	-47.6	501	-0.06	2	-14.2	2	-7.7	2	-49.1	72	-0.86	2	-14.4	4	-6.4	2	-40.7			2	-5
	NCut	3	-2.9	38	-25.1	2	-4.6	32	-1.6	2	-3.3	40	-53.9	2	-3.6	17	-1.4	3	-3.02	73	-24.6	3	-3.4	12	-1.2
Euro-roads	Modularity	126	0.02			8	0.22	42	0.23	72	0.03			7	0.34	51	0.27	100	0.02			9	0.27	32	0.23
	Conductance			11	-49.7	6	-5.8	22	-2.4			40	-12.2	5	-5.7	63	-1.16			12	-25.6	11	-4.8	20	-2.75
	Edge Cut	32	-1.23	46	-95	100	-1.01	7	-2.4	28	-1.08	27	-67.2	167	-0.59	5	-2.1	72	-0.8	167	-60	152	-0.27	9	-2.3
	NCut	72	-0.09	11	-37.6	7	-4.2	20	-3.1	167	-3.1	11	-46.5	11	-4.6	30	-1.1			15	-29.1	10	-5.1	16	-2.7

Table 5: Expected First Success (EFS) and Relative Objective Improvement (ROI) of noise-enhanced community detection methods for modularity when adding $1,000$ noise edges to Fb-Company. Applying any of Louvain, FastGreedy, or Leading EigenVector after adding $1,000$ noise edges to Fb-Company leads to detecting better communities: EFS=4, and ROI=%0.23.

Network	Random Noise								Weighted Noise								Frequency Noise							
	Louvain		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*	
	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI
Fb-Company	5	0.23	2	22.9	3	0.39			3	0.25	2	57	3	0.5			4	0.31	2	61	2	0.45		

Table 6: Expected First Success (EFS) and Relative Objective Improvement (ROI) of noise-enhanced community detection methods on synthetic networks. Each number shows the average EFS or ROI for 16 synthetic networks with the same number of nodes (n) and different μ, β, and γ values.

Network Size	Objective Function	Random Noise								Weighted Noise								Frequency Noise							
		Louvain		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*	
		EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI
$n = 1,000$	Modularity	11	0.1	4	4.04	9	0.58	7	0.138	11	0.1	4	4.05	8	0.61	9	0.12	12	0.1	4	4.19	7	0.64	13	0.11
	Conductance	4	-0.49	3	-8.77	4	-1.57	5	-0.6	3	-0.52	4	-8.32	4	-1.65	7	-0.39	3	-0.58	3	-9.2	3	-1.76	8	-0.3
	Edge Cut	7	-1.78	4	-16.82	4	-5.15	5	-1.94	7	-1.83	4	-15.9	4	-5.51	4	-2.43	8	-1.69	3	-18.2	3	-5.95	4	-2.81
	NCut	4	-0.36	3	-5.09	5	-1.22	5	-0.56	4	-0.37	3	-6.97	4	-1.31	7	-0.39	3	-0.39	3	-5.58	4	-1.39	9	-0.31
$n = 10,000$	Modularity	6	0.095	4	6.53	8	0.7	100	0.003	5	0.18	4	8.85	6	0.76	272	0.0006	4	0.25	4	9.3	5	0.76	3200	0.0002
	Conductance	3	-0.13	3	-11.95	5	-0.95	20	-0.16	3	-0.13	3	-14.83	4	-1.18	65	-0.1	2	-0.23	2	-16.98	3	-1.39	889	-0.06
	Edge Cut	11	-1.32	3	-33.54	5	-4.45	6	-2.75	5	-2.02	3	-34.5	4	-5.05	6	-3.29	5	-1.88	2	-38.69	3	-5.68	5	-4.24
	NCut	5	-0.06	3	-6.3	7	-0.79	20	-0.15	4	-0.07	3	-7.5	4	-0.93	72	-0.1	2	-0.14	3	-9.41	3	-1.2	1143	-0.04

Table 7: Expected First Success (EFS) and Relative Objective Improvement (ROI) of noise enhanced community detection methods for NMI improvement based on ground truth communities. All noise methods can improve NMI, where Random Noise obtains EFS=7, and ROI=2.42, Weighted Noise yields EFS=7, and ROI=2.9, and Frequency Noise results in EFS=16, and ROI=4.8.

Network	Random Noise								Weighted Noise								Frequency Noise							
	Louvain		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*		*Louvain*		*EigenVector*		*FastGreedy*		*WalkTrap*	
	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI	EFS	ROI
email-Eu-core	2	2.1	25	0.73	6	0.69	5	3	2	2.4	31	0.76	4	0.83	4	3.01	2	2.3	100	0.06	4	0.92	5	2.4
DBLP	2	4.6	5	2.9	8	2.08	2	3.3	2	6.3	3	3.4	6	2.64	2	4.07	2	13.9	2	12.1	6	2.5	2	4.3

REFERENCES

[1] Alex Arenas, Albert Díaz-Guilera, and Conrad J Pérez-Vicente. 2006. Synchronization reveals topological scales in complex networks. *Phys. Rev. letters* 96, 11 (2006).

[2] Kartik Audhkhasi, Osonde Osoba, and Bart Kosko. 2016. Noise-enhanced convolutional neural networks. *Neural Networks* 78 (2016), 15–23.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *JSTAT* 2008, 10 (2008), P10008.

[4] Jeff Cheeger. 1969. A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*. 195–199.

[5] Hao Chen, Lav R Varshney, and Pramod K Varshney. 2014. Noise-enhanced information systems. *PIEEE* (2014).

[6] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Phys. Rev. E* 70, 6 (2004), 066111.

[7] Jordi Duch and Alex Arenas. 2005. Community detection in complex networks using extremal optimization. *Phys. Rev. E* 72 (Aug 2005), 027104. Issue 2.

[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[9] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *Proceedings of the national academy of sciences* 104, 1 (2007), 36–41.

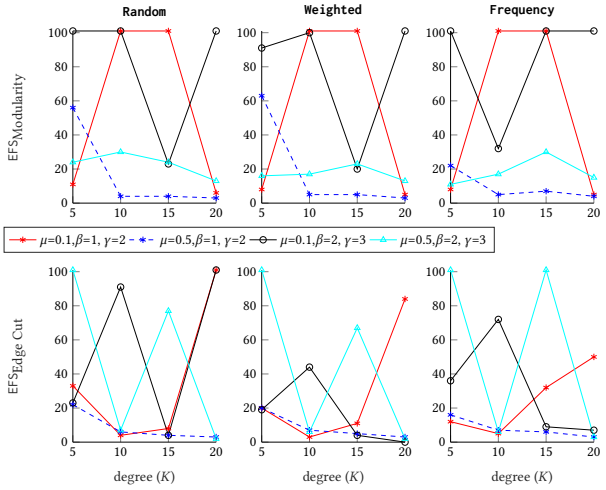

Figure 5: Expected First Success (EFS) of modularity and edge cut after applying Louvain on noisy communities of synthetic networks with $n = 1,000$. When $\mu = 0.5$, Louvain is able to improve modularity with low values of EFS, while when $\mu = 0.1$, Louvain is less likely to improve modularity. By increasing K, EFS for edge cut also increases.

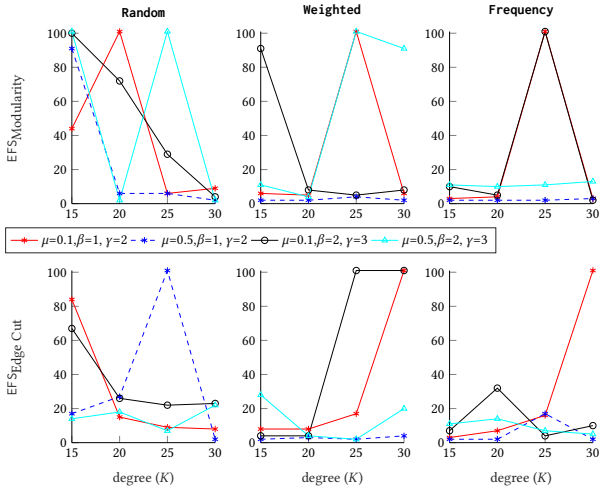

Figure 6: Expected First Success (EFS) of modularity and edge cut after applying Louvain on noisy communities of synthetic networks with $n = 10,000$. By increasing n and K, EFS for modularity decreases on average.

[10] Luca Gammaitoni, Peter Hänggi, Peter Jung, and Fabio Marchesoni. 1998. Stochastic resonance. *Reviews of modern physics* 70, 1 (1998), 223.

[11] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002).

[12] Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. 2004. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E* 70 (Aug 2004), 4. Issue 2.

[13] Shengmin Jin and Reza Zafarani. 2020. The Spectral Zoo of Networks: Embedding and Visualizing Networks with Spectral Moments. In *Proceedings of the KDD.*

[14] Ravi Kannan, Santosh Vempala, and Adrian Vetta. 2004. On clusterings: Good, bad and spectral. *JACM* 51, 3 (2004).

[15] Steven Kay. 2000. Can detectability be improved by adding noise? *IEEE signal processing letters* 7, 1 (2000), 8–10.

[16] O. Krishna, R. K. Jha, A. K. Tiwari, and B. Soni. 2013. Noise induced segmentation of noisy color image. In *2013 NCC*. 1–5.

[17] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78, 4 (2008), 046110.

[18] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection.

[19] Jure Leskovec, Kevin J Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proc. of WWW*. 631–640.

[20] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. 2005. On trip planning queries in spatial databases. In *SSTD*. 273–290.

[21] Xueyu Mao, Purnamrita Sarkar, and Deepayan Chakrabarti. 2017. Estimating mixed memberships with sharp eigenvector deviations. *arXiv preprint arXiv:1709.00407* (2017).

[22] Mark D McDonnell and Derek Abbott. 2009. What is stochastic resonance? Definitions, misconceptions, debates, and its relevance to biology. *PLoS comp. bio.* 5, 5 (2009).

[23] Mark D McDonnell and Lawrence M Ward. 2011. The benefits of noise in neural systems: bridging theory and experiment. *Nature Reviews Neuroscience* 12, 7 (2011), 415.

[24] Frank Moss, Lawrence M Ward, and Walter G Sannita. 2004. Stochastic resonance and sensory information processing: a tutorial and review of application. *Clinical neurophysiology* 115, 2 (2004), 267–281.

[25] L. Nataraj, A. Sarkar, and B. S. Manjunath. 2009. Adding Gaussian noise to "denoise" JPEG for detecting image resizing. In *ICIP*. 1493–1496.

[26] Lakshmanan Nataraj, Anindya Sarkar, and Bangalore S Manjunath. 2010. Improving re-sampling detection by adding noise. In *Media Forensics and Security II*, Vol. 7541.

[27] Mark Newman. 2018. *Networks*. Oxford university press.

[28] Mark EJ Newman. 2004. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* 69, 6 (2004).

[29] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* 74, 3 (2006), 036104.

[30] Mark EJ Newman. 2006. Modularity and community structure in networks. *PNAS* 103, 23 (2006), 8577–8582.

[31] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69, 2 (2004), 026113.

[32] Osonde Osoba and Bart Kosko. 2013. Noise-enhanced clustering and competitive learning algorithms. *Neural Networks* 37 (2013), 132–140.

[33] Osonde Osoba, Sanya Mitaim, and Bart Kosko. 2013. The noisy expectation–maximization algorithm. *Fluctuation and Noise Letters* 12, 03 (2013), 1350012.

[34] Renbin Peng, Hao Chen, and Pramod K Varshney. 2009. Noise-enhanced detection of micro-calcifications in digital mammograms. *IEEE JSTSP* 3, 1 (2009), 62–73.

[35] Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *ISCIS*.

[36] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *PNAS* 101, 9 (2004).

[37] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* 76, 3 (2007).

[38] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. 2002. Hierarchical organization of modularity in metabolic networks. *science* 297, 5586 (2002), 1551–1555.

[39] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Phys. Rev. E* 74, 1 (2006), 016110.

[40] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization.

[41] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Departmental Papers (CIS)* (2000), 107.

[42] Enrico Simonotto, Massimo Riani, Charles Seife, Mark Roberts, Jennifer Twitty, and Frank Moss. 1997. Visual Perception of Stochastic Resonance. *Phys. Rev. Lett.* 78 (Feb 1997), 0. Issue 6.

[43] Kit-Sang Tang, Kim-Fung Man, Sam Kwong, and Qun He. 1996. Genetic algorithms and their applications. *IEEE signal processing magazine* 13, 6 (1996), 22–37.

[44] Ted C Wang and Nicolaos B Karayiannis. 1998. Detection of microcalcifications in digital mammograms using wavelets. *IEEE trans. on medical imaging* 17, 4 (1998).

[45] R. Zafarani and H. Liu. 2009. Social Computing Data Repository. http://socialcomputing.asu.edu

[46] Steeve Zozor and Pierre-Olivier Amblard. 2002. On the use of stochastic resonance in sine detection. *Signal Proc.* 82, 3 (2002).