Increasing and decreasing network efficiency by connecting or disconnecting communities

Zachary D. Eager and Anthony A. Harkin

School of Mathematical Sciences, Rochester Institute of Technology, Rochester, NY 14623

ABSTRACT

Given a handful of new edges with which to augment an already established network, how should one place these new edges in order to most increase the overall efficiency of the entire network? Conversely, given an existing network, which few edges should be removed in order to most decrease the efficiency of the network? Here we propose a novel algorithm for adding a set of k new edges to an existing network in such a way that increases the network efficiency, in the sense defined by V. Latora and M. Marchiori. The algorithm we propose can also suggest which edges should be removed from the network in order to decrease overall network efficiency. The heuristic motivation underlying our proposed algorithm is that if some "efficient communities" contained within a network can be detected, then intelligently adding brand new edges which further interconnect these communities should significantly improve the efficiency of the whole network. To this end, we show how communities within the network having efficient structure can be automatically detected by employing a concept we refer to as "efficiency-modularity", which is a term borrowed from the well-known network modularity. We demonstrate the effectiveness of this new algorithm for increasing, and decreasing, network efficiency with experiments on both real-world and synthetic networks.

Keywords: network efficiency, graph augmentation, community detection, modularity

1. INTRODUCTION

Latora and Marchiori proposed "network efficiency" as a precise mathematical measure to describe how well a system exchanges information in parallel, every vertex sends information concurrently along the network through its edges.[1] They examined neural networks, man-made communication and transportation systems and showed that the underlying general principle of their construction is a small-world principle of high efficiency. Therefore, we expect that an effective algorithm for increasing network efficiency by adding new edges could be useful in diverse settings, such as communication or transportation systems. Network efficiency has also been used to study the effects of errors and attacks on scale free networks.[2]

Consider a simple graph $\mathbf{G}$ with n vertices and m edges. The $n \times n$ adjacency matrix $\mathbf{A}$ can be used to obtain the distance matrix, $\mathbf{D}$, of the network consisting of the shortest path lengths D_{ij} between two vertices i and j. We note that $D_{ij} = 0$ when $i = j$ and $D_{ij} = \infty$ when there exists no path from i to j. As defined by Latora and Marchiori, the entries of the efficiency matrix $\mathbf{E}$ are inversely proportional to the entries of the distance matrix, $E_{ij} = \frac{1}{D_{ij}}$. When there is no path connecting nodes i and j then $E_{ij} = 0$. In addition, when $i = j$ then $E_{ij} = 0$. The efficiency is then defined as[1]

$$E(\mathbf{G}) = \frac{1}{n(n-1)} \sum_{i,j \in \mathbf{G}} E_{ij}.$$

The efficiency of a graph having n nodes is $0 \leq E(\mathbf{G}) \leq 1$, where $E = 0$ when the graph has no edges, and $E = 1$ when the graph is complete.

In this paper we explore the following question. Given k additional edges with which to augment the graph $\mathbf{G}$, how should they be placed in order to most increase the network efficiency? Obviously, given k edges with which to increase the efficiency of a network, one could use brute force to explore all possible ways to add the edges to the graph and report the maximum efficiency possible to attain. This brute force approach would be computationally infeasible for all but the smallest networks.

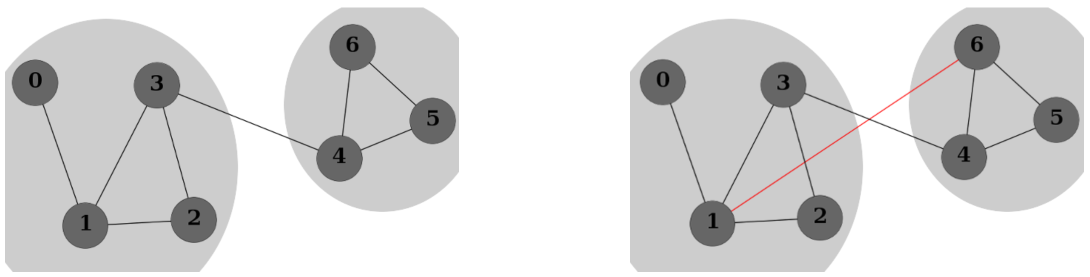


Figure 1: (Left) The original graph with efficiency $E(\mathbf{G}) = 0.627$. (Right) The augmented graph with $k = 1$ new edge added $E(\mathbf{G}_{aug}) = 0.690 > E(\mathbf{G})$. Note that the added edge connects what appear to be two different clusters, or communities of nodes.

Although there are is no previous work that specifically addresses graph augmentations which aim to increase network efficiency as defined by Latora and Marchiori, there are several papers discussing augmentation problems for other notions of "connectivity," which have been shown to be NP-complete.[3,4] The work of Demaine and Zadimoghaddam focuses on edge augmentations minimizing the diameter of the network.[5] Meyerson and Tagiku tackle a similar problem of approximately minimizing average shortest path lengths from a single source.[6] Papagelis et al. develop a graph augmentation algorithm designed for minimizing the characteristic path length.[7] Their path screening algorithm for minimizing the characteristic path length can also employ graph coarsening in order to perform augmentations on very large graphs.[8,9] Wang and Mieghem implement a strategy based on the Fiedler vector for adding edges to graphs in order to optimize algebraic connectivity.[10]

We propose an algorithm for edge augmentation that is tailor-made to produce a large increase in network efficiency, while at the same time being much more computationally feasible than a brute force approach. The edge augmentation algorithm we present is motivated by the intuition that, if a large increase in network efficiency is desired, then one should add new edges in such a way that the community structure within the network becomes less pronounced. More specifically, in a previous paper, the authors have shown that the concept of network efficiency can be adapted to detect community structure within networks such that the communities found each have higher efficiency than expected based on random chance.[11] One might then reasonably assume that deliberately adding new edges which further connect highly efficient communities would increase the overall network efficiency even more. For example, in Fig. 1 (left) a network with $n = 7$ nodes and $m = 8$ edges is shown, and it appears to contain two communities (shaded). In Fig. 1 (right), we added $k = 1$ edge connecting nodes 1 and 6. This new edge further connects the two communities together, and the overall network efficiency has increased from $E = 0.627$ to $E = 0.690$, which is the optimal increase attainable with one new edge. We will formalize this algorithm in the next section.

2. EFFICIENCY MODULARITY

The central idea we will employ to augment a graph in order to increase network efficiency is that we will endeavour to detect two communities which each have a higher than expected efficiency, and then intelligently connect them in order to make the community structure within the entire network less pronounced. In order to make use of this idea, we need to first review the method of efficiency-modularity for detecting communities in networks, as developed in a previous work of the authors.[11]

Let $\mathbf{E}$ be the efficiency matrix of an undivided network $\mathbf{G}$. Forming the $n \times n$ efficiency matrix $\mathbf{E}$ requires all-pairs shortest paths between nodes to be computed.[12] We define the efficiency-modularity for dividing a network into two partitions ($\mathbf{G}_1$ and $\mathbf{G}_2$) as

$$Q_{eff} = \frac{1}{4\widetilde{m}} \sum_{i,j \in \mathbf{G}_1} \left(E_{ij} - \widetilde{P}_{ij} \right) + \frac{1}{4\widetilde{m}} \sum_{i,j \in \mathbf{G}_2} \left(E_{ij} - \widetilde{P}_{ij} \right).$$

This quantity represents the difference between the actual and expected efficiencies within communities. We define $\widetilde{P}_{ij}$ to be the expected efficiency between nodes i and j, given by

$$\widetilde{P}_{ij} = \frac{\widetilde{k}_i \widetilde{k}_j}{2\widetilde{m}}$$

where $\widetilde{k}_i = \sum_{j\in\mathbf{G}} E_{ij}$ and $\widetilde{m} = \frac{1}{2}\sum_{i,j\in\mathbf{G}} E_{ij}$. Assigning nodes into either $\mathbf{G}_1$ or $\mathbf{G}_2$ such that Q_{eff} is maximized will partition the network into two communities whose efficiencies will be higher than expected based on random chance.

We can assign nodes into the two partitions by defining a vector $\mathbf{s}$ such that $s_i = +1$ if node i is in $\mathbf{G}_1$, and $s_i = -1$ if node i is in $\mathbf{G}_2$. Observing that the quantity $\frac{1}{2}(s_i s_j + 1)$ is 1 if i and j are in the same group and 0 otherwise, we can then express efficiency-modularity in quadratic form as

$$\begin{aligned} Q_{eff} &= \frac{1}{4\widetilde{m}} \sum_{i,j\in\mathbf{G}} \left(E_{ij} - \frac{\widetilde{k}_i \widetilde{k}_j}{2\widetilde{m}} \right) (s_i s_j + 1) \\ &= \frac{1}{4\widetilde{m}} \sum_{i,j\in\mathbf{G}} \left(E_{ij} - \widetilde{P}_{ij} \right) s_i s_j \\ &= \frac{1}{4\widetilde{m}} \mathbf{s}^T \widetilde{\mathbf{B}} \mathbf{s} \end{aligned}$$

where the efficiency-modularity matrix is $\widetilde{\mathbf{B}} = \mathbf{E} - \widetilde{\mathbf{P}}$. We note that $\widetilde{\mathbf{B}}$ is a real, symmetric matrix where the elements of each of its rows and columns sum to zero, so that it always has an eigenvector $\langle 1, 1, 1, \cdots, 1\rangle$ with eigenvalue zero.

Next we proceed by writing $\mathbf{s}$ as a linear combination of the normalized eigenvectors $\mathbf{u}_i$ of $\widetilde{\mathbf{B}}$ so that $s = \sum_{i=1}^{n} a_i \mathbf{u}_i$ with $a_i = \mathbf{u}_i^T \cdot \mathbf{s}$. Then we have

$$Q_{eff} = \frac{1}{4\widetilde{m}} \sum_i a_i \mathbf{u}_i^T \widetilde{\mathbf{B}} \sum_j a_j \mathbf{u}_j = \frac{1}{4\widetilde{m}} \sum_{i=1}^{n} (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i \tag{1}$$

where β_i is the eigenvalue of $\widetilde{\mathbf{B}}$ corresponding to eigenvector $\mathbf{u}_i$.

We wish to maximize $Q_{eff} = \frac{1}{4\widetilde{m}} \sum_{i=1}^{n} (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i$ by choosing an appropriate division of the vertices into partitions $\mathbf{G}_1$ and $\mathbf{G}_2$, or equivalently by choosing the values of the index vector $\mathbf{s}$. This means choosing $\mathbf{s}$ so as to concentrate as much weight as possible on the terms in equation (1) involving the largest positive eigenvalues. The power method can be used for obtaining the leading eigenvalue β_1 and eigenvector $\mathbf{u}_1$. Following Newman's approach for optimizing modularity, since the largest term in the sum is $(\mathbf{u}_1^T \cdot \mathbf{s})^2 \beta_1$, we maximize the dot product by setting $s_i = 1$ if the corresponding element of $\mathbf{u}_i$ is positive and $s_i = -1$ otherwise.[13] In other words, all vertices whose corresponding elements in the leading eigenvector are positive go in one partition, $\mathbf{G}_1$, and all of the rest in the other, $\mathbf{G}_2$.

The entries of $\mathbf{u}_1$ that are closest to either 1 or -1 represent nodes that are most strongly in $\mathbf{G}_1$ or $\mathbf{G}_2$, respectively. In the same light, the entries of $\mathbf{u}_1$ closest to 0 correspond to nodes not strongly in either community. Our strategy for increasing the efficiency of the network will be to add an edge connecting two nodes which most strongly belong to their respective communities. Therefore, we add an edge between the nodes whose corresponding entries of $\mathbf{u}_1$ are the most positive and most negative (i.e. the nodes that are most strongly in $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively).

As a simple example, consider the graph $\mathbf{G}$ given in Fig. 1 (Left). We observe two communities given by nodes $\{0, 1, 2, 3\}$ and $\{4, 5, 6\}$. We assume that to increase the efficiency of $\mathbf{G}$ we should add an edge between the most central nodes in the two communities. Calculating the leading eigenvector of the efficiency-modularity matrix $\widetilde{\mathbf{B}}$ for the graph $\mathbf{G}$ yields:

$$\mathbf{u}_1 = \begin{bmatrix} -0.37179189 \\ -0.42672136 \\ -0.32626109 \\ -0.14886969 \\ 0.34148924 \\ 0.4660774 \\ 0.4660774 \end{bmatrix}$$

The smallest and largest elements of the leading eigenvector (red) correspond to nodes 1 and either 5 or 6 respectively. It follows that these nodes belong most strongly in opposite communities. Thus, adding an edge between nodes 1 and 5 or nodes 1 and 6 should result in a greater increase in $E(\mathbf{G})$ than would we would expect from adding an edge between two nodes at random. In Fig. 1 (right) we see that the efficiency increases from $E(\mathbf{G}) = 0.627$ to $E(\mathbf{G}_{aug}) = 0.690$. As a point of comparison, adding an edge between nodes 0 and 2 of $\mathbf{G}$ yields an efficiency increase to only $E(\mathbf{G}_{aug}) = 0.651$.

In order to add k new edges to a network, our algorithm iterates through the edges and adds them one at a time in the following manner. At each iteration we compute the leading eigenvector $\mathbf{u}_1$ of the efficiency-modularity matrix $\widetilde{\mathbf{B}}$. We then determine the elements i and j of $\mathbf{u}_1$ whose entries have the largest difference in magnitude. We then connect node i to node j with a new edge. Note that if nodes i and j are already connected by an edge, we choose the i, j entries from $\mathbf{u}_1$ with the largest difference in magnitude that are not already connected with an edge. This edge augmentation algorithm takes k iterations to complete and requires $O(n^3)$ computations at each iteration.

The same algorithm can be adapted to suggest which edges to remove from a network in order to decrease overall efficiency. The heuristic motivation for decreasing efficiency is that, instead of further connecting efficient communities by adding edges, we try to disconnect efficient communities by intelligently removing edges. We can remove an existing edge corresponding to the connected nodes whose leading eigenvector entries differ by the most. For example, in Fig. 1 then entries in the leading eigenvector corresponding to nodes 3 and 4 have a difference of 0.49035893, which is the largest difference among nodes connected by an edge. Therefore, the algorithm would suggest to remove the edge connecting node 3 to node 4 in order to most decrease the overall network efficiency.

3. COMPARISON OF EDGE AUGMENTATION ALGORITHMS

In this section, we will compare different approaches for suggesting edge augmentations to improve network efficiency. For each network considered in this section, we will apply our proposed algorithm based on efficiency-modularity in order to increase network efficiency. We also implement a slightly faster variation of our Efficiency Modularity algorithm which employs a path-length cutoff on the algorithm that finds the all-pairs shortest path lengths.

To obtain an upper bound on how much efficiency can be increased by adding k new edges, we will apply a brute force greedy algorithm where all possible edge augmentations are computed. For the brute force approach, we consider adding one new edge at a time to the graph. For each new edge added, we find in a brute force manner the two nodes to connect such that the network efficiency is increased the most. Since computing the network efficiency requires $O(n^3)$ computations, the overall complexity of this Greedy algorithm to add a single new edge is $O(n^5)$. We note that any edge augmentation to a network will increase its efficiency. Thus, as reasonable lower bound on how much efficiency will be increased, we will also consider a brute force greedy algorithm where we add k edges that increase efficiency the least. We refer to the greedy algorithm that increases efficiency the most as *Greedy-Best*, and the greedy algorithm that increases efficiency the least as *Greedy-Worst*. In addition to *Greedy-Best* and *Greedy-Worst*, four other benchmark comparisons are considered,

- *Random Insertion*, average efficiency increase obtained by connecting random edges,
- *Degree Centrality*, connecting hubs by finding the highest degree nodes which aren't already connected,

- *Newman Modularity*, finding edges which connect communities found by Newman modularity,[13]
- *Papagelis et al.*, finding "ghost edges" which maximize the utility of the shortest path length of the graph.[7]

We will now compare the eight algorithms described above to various synthetic (12 K_5 Ring, Erdős-Rényi, LFR, Barabási-Albert) and real-world (Karate, Dolphins) networks that represent undirected, unweighted graphs (Table 1). In Table 1, we present the properties of the various test networks.

Name	Nodes	Edges	Possible Augmentations	Density	E($\mathbf{G}$)
LFR	65	92	1988	0.044	0.240
Erdős-Rényi	65	220	1860	0.106	0.477
K_5 Ring	60	132	1638	0.075	0.244
Barabási-Albert	100	99	4851	0.020	0.262
Karate	34	78	483	0.139	0.492
Dolphins	62	159	1732	0.084	0.379

Table 1: Networks used to compare algorithms for increasing network efficiency.

3.1 Synthetic Networks

In 2008, Lancichinetti et al. introduced a class of benchmark networks, known as LFR benchmark networks, that account for the heterogeneity in the distributions of node degrees and of community sizes often found in real-world networks.[14] LFR benchmark networks are becoming increasingly popular in recent literature involving community detection.[15–18] In Fig. 2 we apply the algorithms to an LFR benchmark network. Specifically, the LFR network has the following parameters: nodes $N = 65$, ave. degree $\langle k \rangle = 3$, mixing parameter $\mu = 0.1$, min. community size $s_{min} = 5$, and max. community size $s_{max} = 20$. The LFR network has a low edge density and low initial efficiency, however, there is community structure which indicates the Efficiency Modularity algorithm should suggest augmentations that increase efficiency quite well, as is evident in Fig. 2 (Left).

The next synthetic network we consider is an Erdős-Rényi random graph with $N = 65$ nodes and a rewiring probability of $p = 0.1$ (Fig. 3). Random graphs built according to the Erdős-Rényi model, exhibit a small average shortest path length and as a result are highly efficient.[1,19] Since there is little to no community structure to

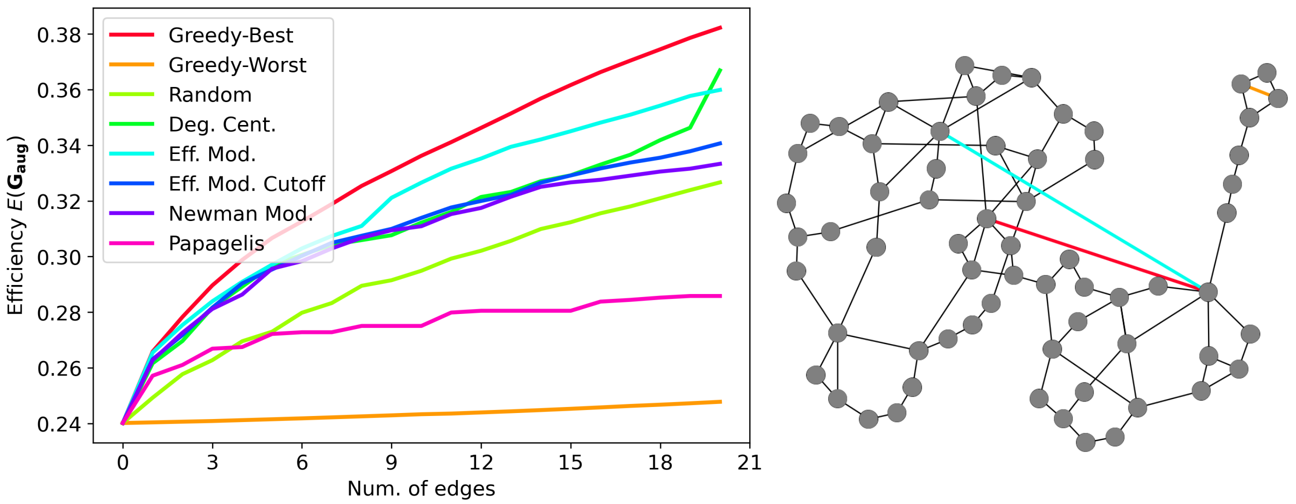


Figure 2: LFR Network. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for an LFR network with $N = 65$ nodes found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.

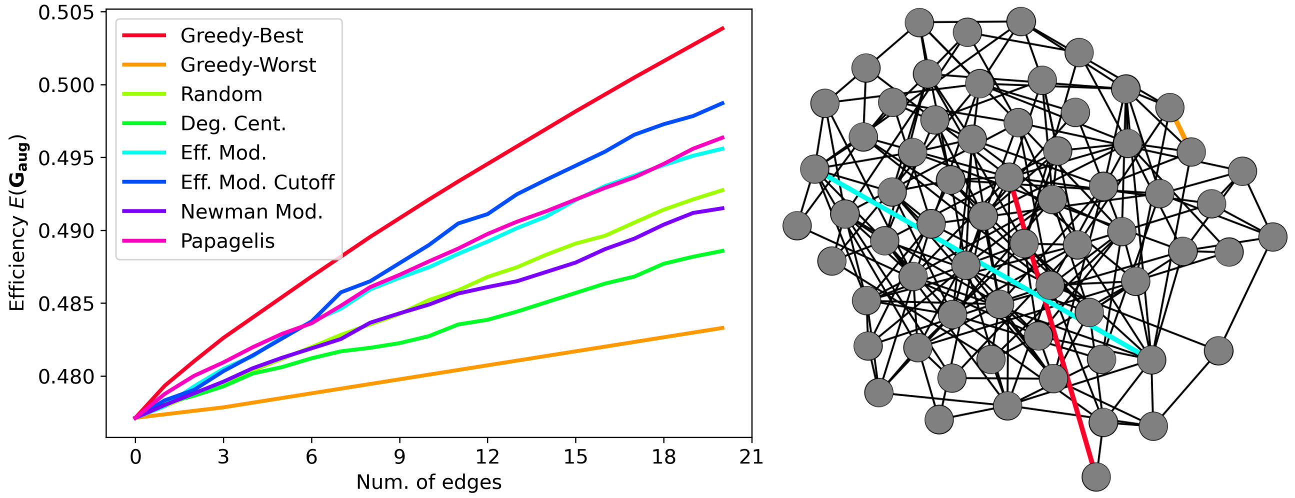


Figure 3: Random Network. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for an Erdős-Rényi random graph with $N = 65$ nodes found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.

begin with, we see in Fig. 3 (Left) that Efficiency Modularity is only slightly better at increasing the efficiency over Random Insertion.

We emphasize that adding edges via Efficiency Modularity was designed to be most effective when community structure is present. If the network has no community structure, such as the example in Fig. 3, then placing k new edges at random may increase the efficiency in a manner comparable to our proposed algorithm, although in that case one still needs to perform an $O(n^3)$ computation to obtain the before and after values of the network efficiency. Whereas, for networks where community structure is present, such as the LFR network shown in Fig. 2, there is always a chance that placing k random edges will result in many, or all, of the k new edges being inserted into the same community. This outcome will not be the most effective increase in efficiency since it will not interconnect the communities as our proposed algorithm is specifically designed to do.

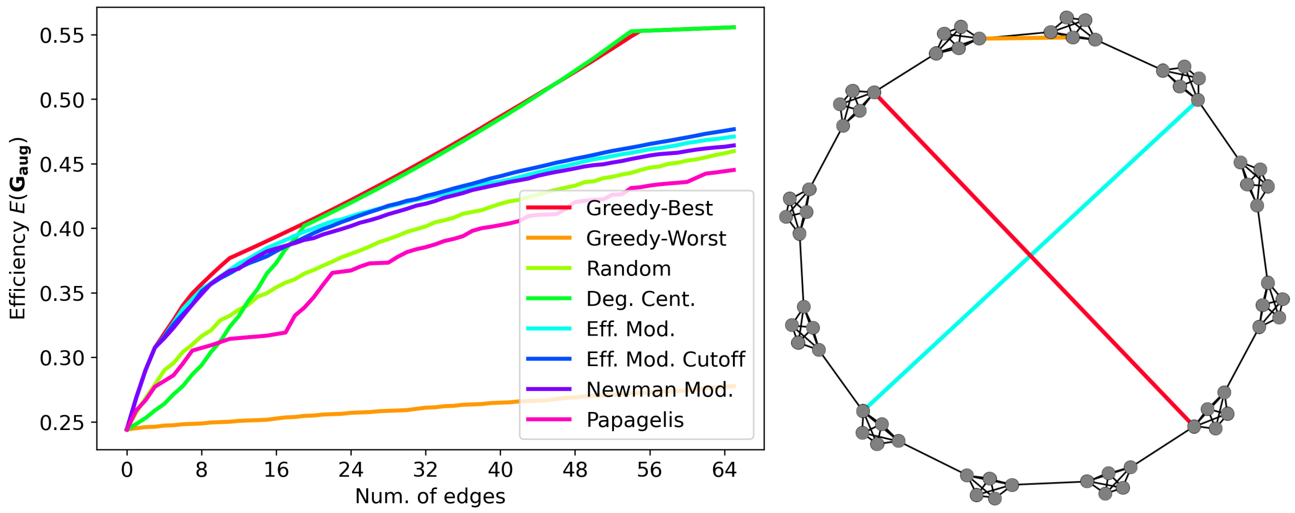
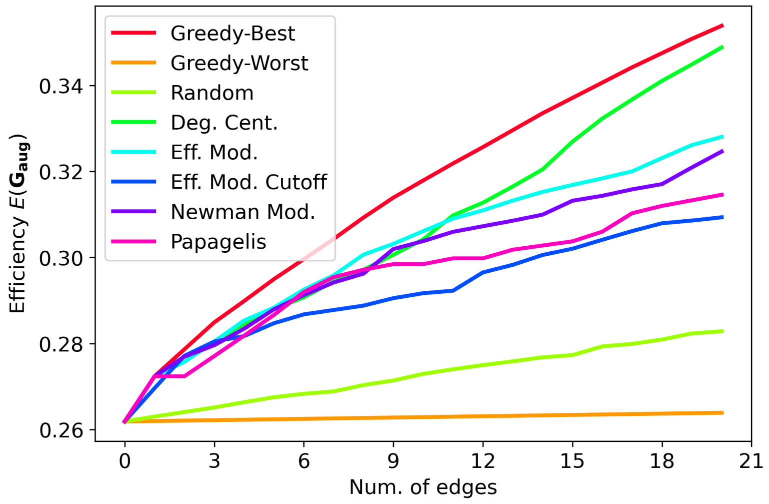


Figure 4: K_5 Ring. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for a ring of twelve K_5 graphs found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.



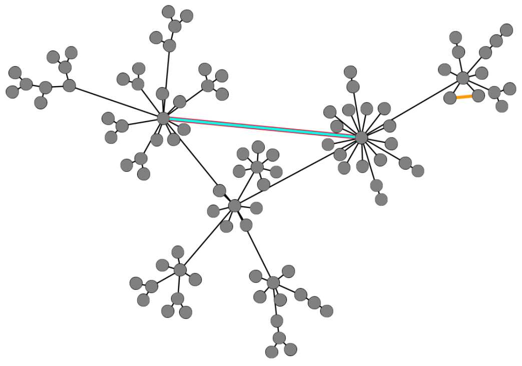


Figure 5: Scale-free Network. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for a Barabási-Albert random graph with $N = 100$ nodes found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.

The next synthetic network we consider is a series of complete K_5 graphs connected to form a ring, i.e. each K_5 is connected to its two neighboring K_5 graphs via a single edge (Fig. 4). The significance of this network is two-fold. First, the network has a low edge density initially. And secondly, even though there is clear community structure, it is a known issue that modularity quality functions such as efficiency-modularity cannot detect each complete K_5 as a separate community.[20] The results in Fig. 4 (Left) are initially similar to those of the LFR network. That is, Efficiency Modularity performs well initially but once enough new edges are added which destroy the community structure (approx. $k = 20$) then the Efficiency Modularity approach performs similarly to Random Insertion. We also note that, since there are many unconnected nodes having the same highest degree, that the Degree Centrality algorithm chooses two of those at random which results in poor performance in this particular instance.

The last synthetic network we consider is a Barabási-Albert random graph with $N = 100$ nodes (Fig. 5). The Barabási-Albert model produces scale-free networks which are widely observed in natural and human-made systems.[21] We observe that augmentations suggested by Efficiency Modularity and Degree Centrality yield far more effective results than those suggested via a Random Insertion approach.

3.2 Real-World Networks

Fig. 6 (Left) shows the results for the real-world karate club network of Zachary[22] when using each of the algorithms to suggest edge augmentations. The network shows the pattern of friendships between the members of a karate club at an American university in the 1970s. Shortly after the formation of the network, the club split into two as a result of an internal dispute. Since there are two distinct communities within the karate club network along with two hub nodes which connect these communities, it is expected that Efficiency Modularity and Degree Centrality are more effective than Random Insertion in suggesting edge augmentations.

The second real-world network we consider is a social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand[23] (Fig. 7). There are two communities of dolphins, each of which is well interconnected but not fully interconnected, therefore Degree Centrality tends to suggest adding edges which further connect nodes within communities, whereas Efficiency Modularity suggests adding edges across communities.

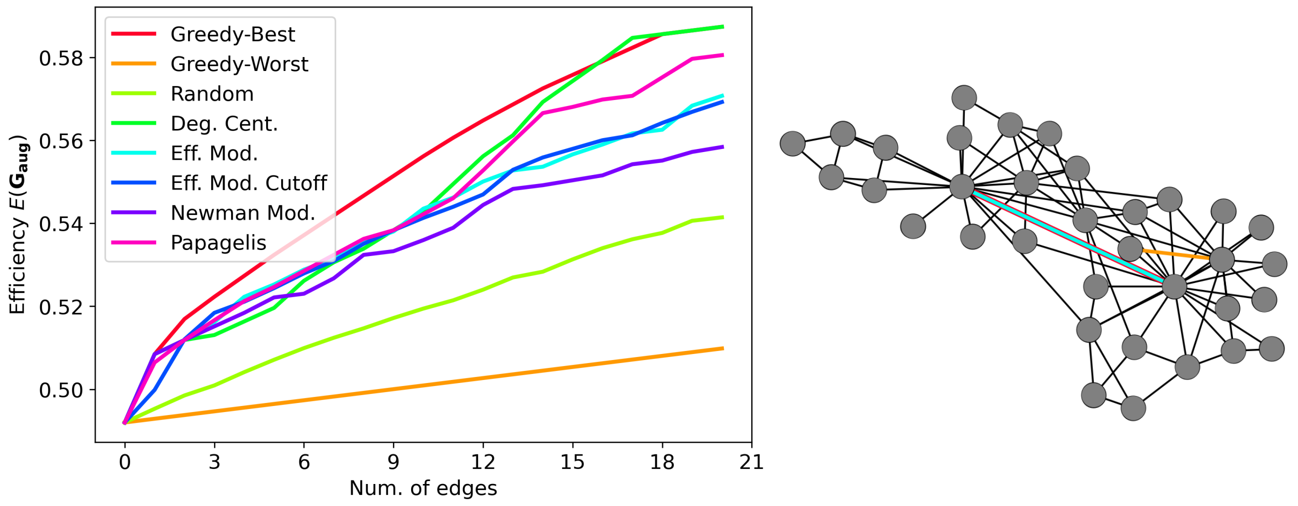


Figure 6: Karate Club. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for the karate club network found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.

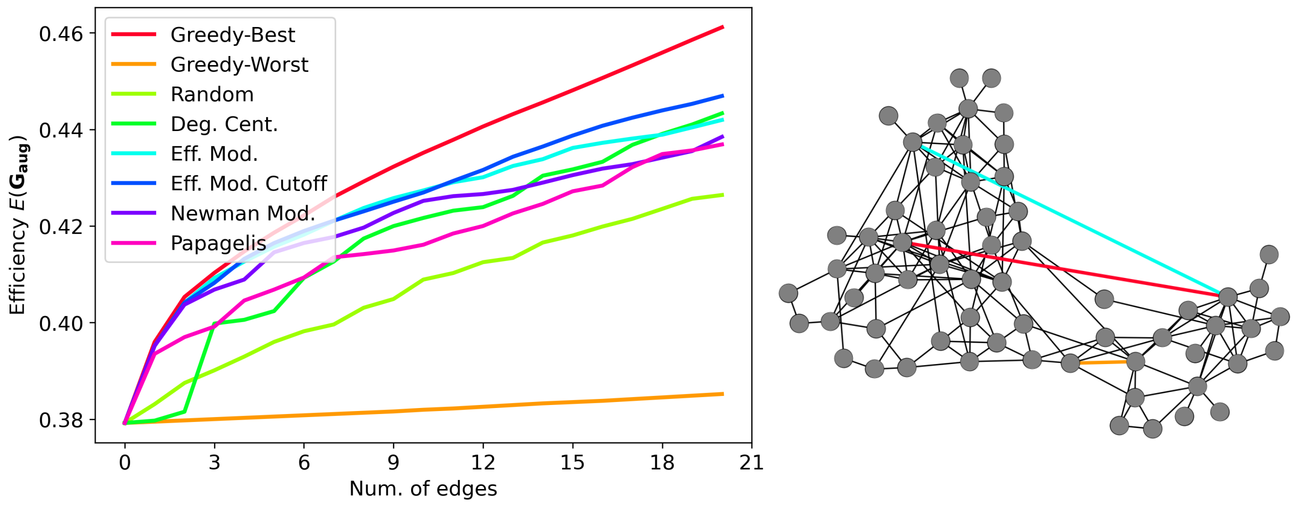


Figure 7: Dolphin Network. (Left) Efficiency of the augmented network, $E(\mathbf{G_{aug}})$, versus the number of augmented edges for the dolphin network found using each of the algorithms. (Right) The initial network configuration is shown along with the first augmented edge placement suggested by each method.

3.3 Decreasing Network Efficiency

The examples shown in this section, up to this point, have all focused on how well the Efficiency Modularity algorithm can suggest k new edges to add to a network if the goal is to increase network efficiency the most. On the other hand, in Fig. 8, we demonstrate how well Efficiency Modularity can suggest k edges for removal in order to most decrease the overall network efficiency. The idea for suggesting edge removal, to most decrease network efficiency, is to find the largest pairwise difference between two nodes in the leading eigenvector of the efficiency-modularity matrix which are connected by an edge, and remove that edge.

The example shown in Fig. 8 is the same LFR network that was used in Fig. 2. We observe a rapid decline in network efficiency due to the fact that removing the edges suggested by the Efficiency Modularity algorithm results in the network quickly becoming disconnected. Fig 9. demonstrates decreasing efficiency in the dolphin network by strategically removing edges using Efficiency Modularity.

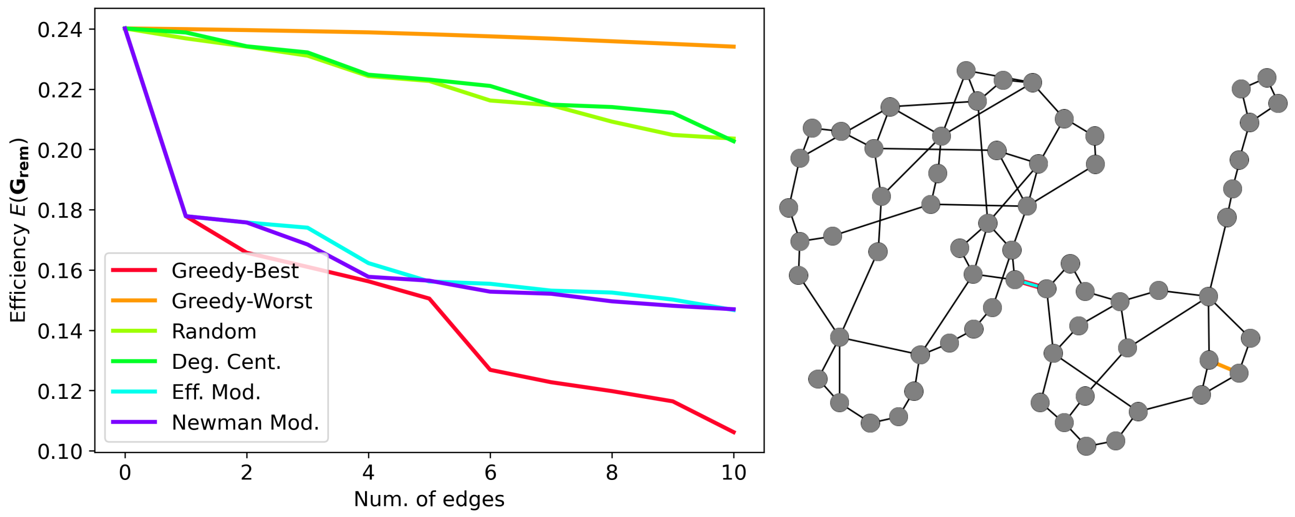


Figure 8: LFR Network. (Left) Efficiency of the augmented network, $E(\mathbf{G}_{\mathbf{rem}})$, versus the number of removed edges for the LFR network found using each of the algorithms. (Right) The initial network configuration is shown along with the first removed edge suggested by each method.

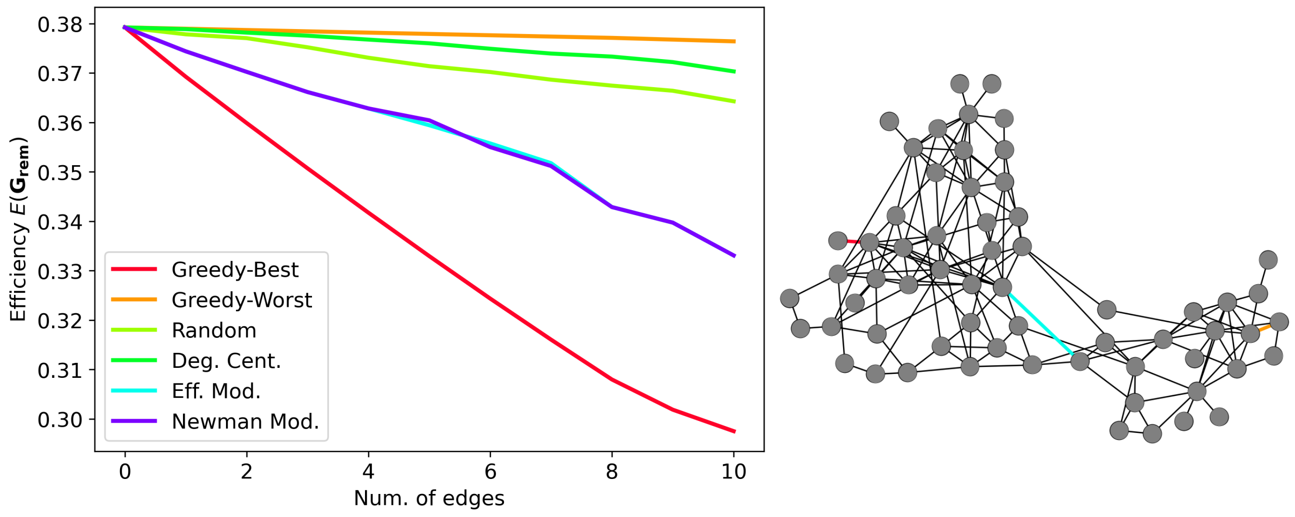


Figure 9: Dolphin Network. (Left) Efficiency of the augmented network, $E(\mathbf{G}_{\mathbf{rem}})$, versus the number of removed edges for the dolphin network found using each of the algorithms. (Right) The initial network configuration is shown along with the first removed edge suggested by each method.

4. CONCLUSION

The concept of network efficiency introduced by Latora and Marchiori can be naturally combined with the well-known network modularity to create what we refer to as efficiency-modularity, Q_{eff}. Maximizing the efficiency modularity can be used to partition the network into two efficient communities. Nodes which contribute most strongly to maximizing the efficiency of each of their respective communities can then be connected together by a new edge in order to improve the efficiency of the entire network. This heuristic produces an algorithm for augmenting a network with k new edges. This same algorithm can also be employed for intelligently suggesting edges which, when removed, will quickly decrease network efficiency.

The effectiveness of this algorithm for increasing, or decreasing, the efficiency of networks via intelligent edge augmentations, or removals, was demonstrated and compared with six other benchmark approaches. It is seen that the Efficiency Modularity approach can be especially effective when applied to networks having existing community structure. In the experiments performed, we often observed that the edge augmentations suggested by the Efficiency Modularity algorithm rapidly improved the network efficiency during the initial augmentations, and then the improvements slowed down once edge saturation caused the community structures to be less pronounced.

It should be noted that in general, to calculate network efficiency an all-pairs shortest path algorithm must be employed. In this work, an all-pairs shortest path calculation is required to form the efficiency-modularity matrix, which can be computationally slow, $O(n^3)$. For larger scale networks, one could compute the approximate all-pairs shortest path matrix,[12, 24] allowing the Efficiency Modularity algorithm to run faster. In addition, computation of the all-pairs shortest path is highly parallelizable.[25]

5. ACKNOWLEDGMENTS

Research was supported by the National Science Foundation under award DMS-1358583.

REFERENCES

[1] Latora, V. and Marchiori, M., "Efficient behavior of small-world networks," *Phys. Rev. Lett.* **87**, 198701 (Oct. 2001).

[2] Crucitti, P., Latora, V., Marchiori, M., and Rapisarda, A., "Efficiency of scale-free networks: error and attack tolerance," *Physica A: Statistical Mechanics and its Applications* **320**, 622–642 (2003).

[3] Eswaran, K. and Tarjan, R., "Augmentation problems," *SIAM Journal on Computing* **5**(4), 653–665 (1976).

[4] Watanabe, T., "Edge-connectivity augmentation problems," *Journal of Computer and System Sciences* **35**(1), 96 – 144 (1987).

[5] Demaine, E. D. and Zadimoghaddam, M., "Minimizing the diameter of a network using shortcut edges.," in [*SWAT*], Kaplan, H., ed., *Lecture Notes in Computer Science* **6139**, 420–431, Springer (2010).

[6] Meyerson, A. and Tagiku, B., "Minimizing average shortest path distances via shortcut edge addition.," in [*APPROX-RANDOM*], Dinur, I., Jansen, K., Naor, J., and Rolim, J. D. P., eds., *Lecture Notes in Computer Science* **5687**, 272–285, Springer (2009).

[7] Papagelis, M., Bonchi, F., and Gionis, A., "Suggesting ghost edges for a smaller world.," in [*CIKM*], Macdonald, C., Ounis, I., and Ruthven, I., eds., 2305–2308, ACM (2011).

[8] Papagelis, M., "Refining social graph connectivity via shortcut edge addition," *ACM Transactions on Knowledge Discovery from Data* **10**(2) (2015).

[9] Filippidou, I. and Kotidis, Y., "Effective and efficient graph augmentation in large graphs," *IEEE International Conference on Big Data* , 875–880, IEEE (2016).

[10] Wang, H. and Mieghem, P. V., "Algebraic connectivity optimization via link addition," *Bionetics* (2008).

[11] Bales, K. N., Eager, Z. D., and Harkin, A. A., "Efficiency-modularity for finding communities and anticommunities in networks," *Journal of Complex Networks* **5**, cnw012 (2016).

[12] Baswana, S. and Kavitha, T., "Faster algorithms for all-pairs approximate shortest paths in undirected graphs," *SIAM Journal on Computing* **39**(7), 2865–2896 (2010).

[13] Newman, M. E., "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E* **74**(3), 036104 (2006).

[14] Lancichinetti, A., Fortunato, S., and Radicchi, F., "Benchmark graphs for testing community detection algorithms," *Physical Review E* **78**(4), 046110 (2008).

[15] Lee, J., Gross, S. P., and Lee, J., "Improved network community structure improves function prediction," *Scientific Reports* **3** (2013).

[16] Aldecoa, R. and Marín, I., "Exploring the limits of community detection strategies in complex networks," *Scientific Reports* **3** (2013).

[17] Aldecoa, R. and Marín, I., "Surprise maximization reveals the community structure of complex networks," *Scientific Reports* **3** (2013).

[18] Traag, V. A., Krings, G., and Van Dooren, P., "Significant scales in community structure," *Scientific Reports* **3** (2013).

[19] Erdős, P. and Rényi, A., "On the evolution of random graphs," in [*Publication of the Mathematical Institute of the Hungarian Academy of Sciences*], 17–61 (1960).

[20] Fortunato, S. and Barthelemy, M., "Resolution limit in community detection," *Proceedings of the National Academy of Sciences* **104**(1), 36–41 (2007).

[21] Barabasi, A.-L. and Albert, R., "Emergence of scaling in random networks," *Science* **286**(5439), 509–512 (1999).

[22] Zachary, W. W., "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research* **33**, 452–473 (1977).

[23] Lusseau, D., "The emergent properties of a dolphin social network," *Proceedings of the Royal Society of London B: Biological Sciences* **270**(Suppl 2), S186–S188 (2003).

[24] Boldi, P. and Vigna, S., "In-core computation of geometric centralities with hyperball: A hundred billion nodes and beyond," *CoRR* **abs/1308.2144** (2013).

[25] Han, Y., Pan, V. Y., and Reif, J. H., "Efficient parallel algorithms for computing all pair shortest paths in directed graphs," (1997).