# Scalable Computing of Betweenness Centrality based on Graph Reduction with a Case Study on Breast Cancer Analytics

**Chung-Hsien Chou**
University of California Irvine
**Shaoting Wang**
University of California Irvine
**Hsiang-Shun Shih**
University of California Irvine
**Phillip C-Y. Sheu** ( psheu@uci.edu )
University of California Irvine https://orcid.org/0000-0003-2036-850X

**Research article**

**Keywords:** Graph, Betweenness Centrality, Graph Reduction

**Posted Date:** November 11th, 2020

**DOI:** https://doi.org/10.21203/rs.3.rs-72273/v1

# Scalable Computing of Betweenness Centrality based on Graph Reduction with a Case Study on Breast Cancer Analytics

Chung-Hsien Chou, Shaoting Wang, Hsiang-Shun Shih, Phillip C-Y. Sheu
*Department of Electrical Engineering and Computer Science*
*University of California, Irvine*
*Contact: psheu@uci.edu*

## Abstract

### Background

Graph theory has been widely applied to the studies in biomedicine such as structural measures including betweenness centrality. However, if the network size is too large, the result of betweenness centrality would be difficult to obtain in a reasonable amount of time.

### Result

In this paper, we describe an approach, 1+ε lossy graph reduction algorithm, to computing betweenness centrality on large graphs. The approach is able to guarantee a bounded approximation result. We use GSE48216, a breast cancer cell line co-expression network, to show that our algorithms can achieve a higher reduction rate with a trade-off of some bounded errors in query results. Furthermore, by comparing the betweenness centrality of the original graph and the reduced graph, it can be shown that a higher reduction rate does not sacrifice the accuracy of betweenness centrality when providing faster execution time.

### Conclusions

Our proposed 1+ε lossy graph reduction algorithm is validated by the experiment results which show that the approach achieves a faster execution within a bounded error rate**.**

***Keywords – Graph, Betweenness Centrality, Graph Reduction***

## I. BACKGROUND

Graph theory has been applied to biomedical researches in many ways. For example, scientists have applied betweenness to search for the most important genes in genetic networks [1-3, 4-6]. Another common graph problem that has been applied to biomedical research is the reachability problem to determine if a specific gene can influence another gene with a certain factor [7]. Yet another example is to use the minimum spanning tree to identify clusters from microarray data based on their similarity [8]. Unfortunately, when a graph is huge scientists usually have to spend long time executing combinatorial graph algorithms such as shortest path algorithm and betweenness centrality. Therefore, how to reduce the response time with a tolerable accuracy when dealing with large graphs becomes important, especially when the graph and its selected subgraphs are queried repeatedly.

Data reduction [9] is useful for many applications that involve big data. In this paper, we apply data reduction to speed up graph algorithms with better execution time and better performance. Existing approaches to graph data reduction may be syntactic or semantic, which may be lossless [10-12] or lossy [13-14]. The difference between syntactical graph reduction and semantical data reduction is that traditional methods are syntactic and rely on data compression, which means they focus on graph structures by serialization or redundancy removal. On the other hand, semantic graph data reduction is query-based that can be applied on top of syntactic data compression to provide additional data reduction. For example, query-based graph data reduction may further reduce a graph by preserving only the information relevant to the queries needed by an application.

Our hypothesis is that further reduction may be possible even on top of semantic reductions. Sometimes the accuracy of query results may not be strictly required (or expected such as with a search engine). Specifically, if we allow approximate results, where the difference between the approximate results and the precise results is bounded, a graph may be further reduced. In the past, to our knowledge, the only work related to this approach is [15] where Bader et al. propose an approximate approach to enhance the performance of betweenness centrality. They speed up the performance by reducing computation for some irrelevant single-source shortest paths. However, based on different graph data, the error rate varies from 10% to 46.7%, which could be considered too high. In this study, we use a co-expression network of breast cancer as a case study to prove our hypothesis. We can demonstrate that our approach not only speeds up the total computational time since we not only reduce the graph size, but also maintains the accuracy with an acceptable error rate bounded by 10%. Unfortunately, in [15] the authors do not show the exact speedup result, but they only show the percentage of shortest paths that were computed versus shortest paths sampled/ all shortest paths. Our dataset is much larger than theirs (n=17,055 and m=4,109,069 >> n=9,914 and m=36,854), and our betweenness reduction has 100% accuracy which is better than 89.69% as reported in [15].

In the following sub-sections, we summarize the computational problems associated with betweenness centrality. We also discuss the methods we employ to reduce graphs.

### A. *Betweenness centrality*

Betweenness centrality analysis is a widely-applied methodology which aims to distinguish the importance of every node in the graph. In the case of genetic graph, the gene with a higher betweenness rank can be considered to exert higher influence on the whole network; and usually the genes found are considered as the backbone and basis that represent a genetic graph [3]. For example, to target the most influential genes in a cancer-gene graph, finding the betweenness centrality is a common approach for biologists.

The fundamental approach to computing betweenness centrality is based on the shortest path algorithm [16]. Brandes [17] propose to compute betweenness centrality without computing all-pair shortest paths. The improved time complexity, which is $O(mn + n^2 logn)$, is still considered to be expensive, especially when the input graph is huge and dense. In the past two decades, computer scientists have been seeking for a more efficient algorithm. Brandes et al. [18], based on the original Brandes' algorithm, propose a heuristic that implements different strategies to compute betweenness centrality. Their experiment result suggests that randomly selecting the source nodes can achieve higher accuracy than a deterministic approach. A parallel algorithm is proposed by Bader et al. [19]. In addition to that, Bader et al. [15] proposes an approximation approach with an adaptive sampling technique in order to compute the betweenness centrality of a given node. Also, Ahmet et al. [20] propose several graph reduction methods to make Brandes algorithm run faster, including the inclusion of bridges, degree-1 nodes, and identical vertices. The majority of these previous works use a sampling method or graph reduction to reduce the running time of traversing all the nodes. For the works utilizing sampling technique, they face the trade-off between performance and accuracy. Even for a graph reduction method, it still spends much time traversing the nodes which are relatively irrelevant. All of them focus on finding all-pair shortest paths to obtain the value of betweenness centrality for all the nodes. We propose an approach that reduces the input graph to achieve a better execution performance without loss of accuracy to find the node with the highest betweenness centrality.

### B. *Computing betweenness centrality*

As discussed in Section 2.1, betweenness centrality is one common approach used to discover the most important nodes in a graph. Given a graph G, betweenness centrality is computed by applying an all-pair shortest path algorithm to find the nodes that have the most shortest paths passing through. In this paper, without losing generality we assume one variation of the Dijkstra algorithm that includes both endpoints in each path [21] is applied to compute betweenness centrality. The procedure is summarized as follows:

*1)* Obtain all-pair shortest paths. Find shortest paths from each node to all the other nodes. Repeat this step n times so that every node takes turns to be the source.

*2)* After all-pair shortest paths are obtained, compute the number of shortest paths pass each node. The value is the betweenness centrality of the node.

For example, in the directed graph shown in Figure 1, nodes represent genes (e.g., P1, P2, etc), and edges represent diseases such that two nodes are connected if they are both related to a disease and the weight carried by an edge represents the influence factor. To find the most important gene we can find the node that has the highest betweenness centrality value.

## II. RESULTS

### A. *GRSP and LGRSP on large graphs*

We apply both GRSP and LGRSP to the graph corresponding to the dataset which is composed of 4,109,069 edges and 17,055 nodes.

#### *1) GRSP*

After applying the GRSP algorithm to the graph, 3,184,625 edges are reduced. It takes 2575.4 seconds to finish the betweenness centrality algorithm on the reduced graph, which requires 12252 seconds on the original graph. Thus, the reduction ratio is 77.5%, and the speedup is 4.76.

#### *2) LGRSP ($\varepsilon$ = 0.2)*

After applying LGRSP ($\varepsilon$ = 0.2) to the graph, the reduction ratio becomes 83.5%, which is higher than that of GRSP, while the speedup can reach 6.1. Since it takes 2009.9 seconds to finish the betweenness centrality algorithm on the reduced graph, which requires 1225.2 seconds on the original graph. Therefore, the ability of LGRSP to speed up is more eminent than that of GRSP.

#### *3) Evaluation using betweenness centrality*

We compute the betweenness centrality of the graph to set a ground truth, and after the reduction algorithm is applied, we compute the betweenness centrality again to compare the difference and compute the accuracy. Since the betweenness centrality algorithm provides a rank of the most important nodes in the graph, we apply the common rank measure approach that calculates accuracy as the following formula [22], where *tp* stands for the number of true positives, *tn* stands for the number of true negatives, and n is the number of items on the list.

$$\text{Accuracy} = (tp + tn) \,/\, n\,, \tag{4}$$

In Table 1, we can see the result for GRSP. We apply the accuracy formula to compare the rank of betweenness for the original graph and the reduced graph after applying the GRSP algorithm. We can see that the accuracy is 100%.

The result for LGRSP is shown in Table 2. In Table 2, we calculate a rank for betweenness centrality for a set of nodes

(i.e., nodes with node id 155249, 154415, etc.) which is the ground truth that we will need when verifying the accuracy for the results of LGRSP. Second, for each graph generated by LGRSP with different error rates ($\varepsilon$ = 0.1 – 1.0), we also calculate the rank of betweenness. We then calculate the accuracy of the ranks using the accuracy formula, and we obtain the result that the accuracy for each error rate is in the range between 90% and 100%.

*4) Sensitivity to ε*

As it takes very long to run LGRSP on the original graph for every ε, to save the time of computation we randomly extract a smaller graph dataset which is a subgraph of the original graph for sample values of ε from 0.2 to 1.0 (and $\varepsilon = 0$ for GRSP of course), in order to know that how fast the reduced graph can speed up using GRSP without sacrificing the accuracy. The subgraph contains 10,000 edges and 3720 nodes. We then apply the GRSP algorithm to this subgraph, and it can reduce 1,938 edges. After the reduction, we apply Dijkstra's shortest path algorithm on the reduced graph and compare the execution time before and after the reduction, based on the formula shown previously in the evaluation section. The reduction ratio is 19.38%, and the speedup is 1.3.

We then apply the 1+ε LGRSP algorithm to the subgraph. When ε is 0.2, the reduction ratio is 21.08% and the speedup is 1.341. We can see that with the error rate limited to 20% ($\varepsilon$ =0.2), the reduction ratio could be higher than that of GRSP, and the Dijkstra's shortest path algorithm runs faster on the graph reduced by LGRSP.

We run experiments on the same graph using different values of $\varepsilon$, the result of reduction ratio is shown in Figure 3, and the speedup is shown in Figure 4.

## III. DISSCUSSION

The results show that with a higher reduction ratio, LGRSP can speed up more, and it can preserve a pretty good accuracy (90%~100%). Compared to [15], because their best case varies from 10%~46%, the $\varepsilon$=0.1 in our approach means it is the error rate for the shortest path, but the corresponding accuracy for betweenness centrality that applies LGRSP (ε=0.1) first is 90% which is better than their approach. For other values of ε, when it goes higher the accuracy can still be kept at least 90%.

## IV. CONCLUSIONS

In this paper, we describe the use of graph reduction to compute betweenness centrality in large graphs. Our hypothesis is that if we allow approximate results, where the difference between the approximate results and the precise results is bounded, a graph may be further reduced, and the speed of graph algorithms can be improved. We conduct experiments on betweenness computation using lossy graph reduction to compare the reduction rates of our algorithms on different sizes of a breast cancer dataset. The results show that our algorithms can achieve a higher reduction rate with a trade-off of some bounded errors in query results. We compare the betweenness centrality of the original graph and the reduced graphs with respect to different reduction rates, and show that, for the accuracy of betweenness centrality, a higher reduction rate does not sacrifice the accuracy of betweenness centrality when providing faster execution time.

## V. METHODS

### A. The query-based graph reduction problem (QGRP)

The query-based graph reduction problem (QGRP) can be described as follows: The inputs of QGRP are a weighted graph $G$ and a set of possible queries $Q$ for a given graph problem $P$. The answer is a reduced graph $G'$ which is the smallest subgraph of $G$ such that for all $q \in Q$, $q(G) = q(G')$, $q(G)$ represents the answer to query $q$ in graph $G$, and $q(G')$ represents the answer to $q$ in graph $G'$. In other words, QGRP aims to find the smallest reduced graph G' which preserves the answers of every query that comes from a specific graph problem.

A separate QGRP can be defined for each graph problem. For the shortest path problem, we can define the Graph Reduction Shortest Path (GRSP) problem as follows: Taking a graph G, find the smallest reduced graph G' which preserves the set of all-pair shortest paths of $G$. Our GRSP algorithm based on Dijkstra's shortest algorithm is given in [23].

### B. The lossy graph data reduction problem (LGRP)

Given a weighted graph $G$ and a set of possible queries $Q$ for a given graph problem $P$, the Lossy Graph Reduction Problem (LGRP) computes $G'$, the smallest subgraph of $G$, with which we can answer the queries $q \in Q$, or $q' \in Q'$ (either in the original form or in a transformed form) within a bounded error rate, $q(G) \leq (1 + \varepsilon) * q(G')$.

Therefore, the goal of lossless graph data reduction can be summed up as:

*1)* Computing corresponding query $q' \in Q'$ for any $q \in Q$.

*2)* For a loseless reduction, ε is zero. That's is, for $q \in Q$, $q(G) = q'(G')$, $q(G)$ represents the answer to $q$ in graph $G$ such as a subgraph, sum of edge weights, etc.

Accordingly, the wrongly answered queries of lossy graph data reduction can be divided into two types:

*1)* $\exists\, q \in Q$ *s.t.* $G' \notin dom(q)$. In other words, some query cannot be answered in $G'$.

*2)* $\exists\, q \in Q$ *s.t.* $(G) \neq q'(G')$, In other words results of some queries may be inconsistent between G and G'.

### C. Methods for lossy graph data reduction (LGR)

If an application is able to indulge the trade-off between graph reduction and accuracy for some queries within an error bound, we call such a reduction 1+ε lossy reduction, where $\varepsilon$ *(0* $< \varepsilon <$ *1)* is the error bound for the application.

*1) LGRSP*

Take a graph $G$ as input and assume that the application only focuses on finding the shortest paths between any two nodes on

the graph. The answer to the lossy graph reduction problem includes not only the smallest subgraph $G'$ of the original graph $G$ that preserves the set of all-pair shortest paths, but also a further reduction $G''$ of $G'$ which preserves most shortest paths in the set of all-pair shortest paths within a designated preservation ratio. We call this problem the Lossy Graph Reduction-Shortest Path (LGRSP) problem.

*2) The 1+ε LGRSP algorithm*

Given $\varepsilon$ $(0 \prec \varepsilon \prec 1)$ and a graph $G$, the basic idea of the algorithm is as follows: If a shortest path $p$ between two nodes is found, we can remove any edge e connecting these two nodes if $weight(p) \leq (1+\varepsilon) * weight(e)$. That is, we can observe two cases to conduct graph reduction. First, if a shortest path is shorter than an edge and both of them share the same endpoints, removing the edge from the graph will not cause an error because the edge is not on the shortest path tree. Second, if the shortest path is longer than the edge but does not exceed the pre-specified bound, we can still conduct graph reduction which guarantees that the adjusted answers of shortest path queries would be no greater than 1+$\varepsilon$. The details of the 1+$\varepsilon$ LGRSP algorithm are discussed in [23]. This paper demonstrates the usefulness of the 1+$\varepsilon$ LGRSP algorithm in the context of biomedicine.

As the advantage of the graph decomposition approach is obvious, to evaluate the efficiency of the graph reduction algorithms and the scalable algorithm, we experimented the GRSP and LGRSP with several genetic datasets.

*D. Setup*

*1) Experiment environment*

| | |
|---|---|
| Database | Neo4j Community Edition: 3.0.6 |
| Language | Java (jdk-8u02) |
| IDE | Eclipse (neon) |
| OS | Windows server 2012 |
| CPU | Intel Xeon E5-2670 v3, 2.30GHz |
| RAM | 256 GB |

*2) Evaluation function*

Below are the definition of reduction ratio and speedup:

$$\text{ReductionRatio} = 1 - \frac{Num\ of\ Edges\ after\ Reduction}{Num\ of\ Edges\ before\ Reduction} \quad (1)$$

$$\text{Speedup} = \frac{Execution\ Time\ of\ shortest\ path\ before\ Reduction}{Execution\ Time\ of\ shortest\ path\ after\ Reduction} \quad (2)$$

*3) Datasets*

In systems biology, correlation networks have been widely used. Take gene co-expression network analysis as an example, it is a system biology approach aims to characterize the correlation of gene expressions in microarray data. Correlation network analysis is able to not only identify the genes that occur frequently but also help finding potential therapeutic molecules with gene-screening methods. For example, Tang [24] use gene co-expression datasets from the GEO database to screen potential biomarkers related to the progression and prognosis of breast cancer. In this study, we use co-expression networks. Such networks have been considered by a majority of correlation network applications.

The workflow of the gene expression data process in this study is shown in Figure 2. The gene expression profiles of GSE48216 submitted by Daemen A et al. [25] is downloaded from the Gene Expression Omnibus (GEO) database. The GSE48216 is an expression profiling based on GPL10999 Illumina Genome Analyzer IIx (Homo sapiens) and contains 56 samples. It includes 56 breast cancer cell lines which are profiled to identify patterns of gene expression associated with subtypes and responses to therapeutic compounds. The Robust Multi-Array Average (RMA) algorithm in the affy package within Bioconductor in R is used to preprocess the gene expression profile data. After background correction, quantile normalization, and probe summarization, the dataset with 15,485 genes is further processed, and the top 50% most variant genes by analysis of variance (9,133 genes) are selected for WGCNA analysis [26].

The expression data profile of these 9,133 genes are constructed into a gene co-expression network using the WGCNA package in R. The network is composed of 4,109,069 edges and 17,055 nodes.

The adjacency matrix $a_{ij}$ corresponding to the connection strength between each pair of nodes is calculated as follows:

$$s_{ij} = correlation\ (x_i, x_j) | a_{ij} = S_{ij}^{\beta}, \quad (3)$$

where $X_i$ and $X_j$ are vectors of expression value for genes *i* and *j*, $s_{ij}$ represents the Pearson correlation coefficient of gene *i* and gene *j*, and $a_{ij}$ encodes the network connection strength between gene *i* and gene *j*.

## ABBREVIATIONS

QGRP: The Query-Based Graph Reduction Problem
GRSP: Graph Reduction Shortest Path
LGRP: The Lossy Graph Data Reduction Problem
LGR: lossy graph data reduction
LGRSP: Lossy Graph Reduction-Shortest Path

## DECLARATIONS

### ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not Applicable

### CONSENT FOR PUBLICATION

Not Applicable

### AVAILABILITY OF DATA AND MATERIALS

The datasets during and/or analyzed during the current study

available from the corresponding author on reasonable request.

## COMPETING INTERESTS

The authors declare that they have no competing interests


## FUNDING

This research was supported in part by NEC Solution Innovators, Ltd., Japan.


## AUTHORS' CONTRIBUTIONS

SW developed the original idea of lossy graph reduction. CHC studied how the computation of betweenness centrality may be benefited from graph reduction, analyzed the data, ran the experiments, analyzed the results, and prepared the manuscript. HSS strengthened the manuscript with additional insights, and PCYS supervised the research and edited the manuscript. All authors have read and approved the manuscript.


## ACKNOWLEDGMENTS

The breast cancer dataset was provided by Professor Charles C.N. Wang of Asia University, Taiwan.


## REFERENCES


[1] Li, Bi-Qing, et al., Identification of lung-cancer-related genes with the shortest path approach in a protein-protein interaction network, BioMed research international 2013.

[2] Koschützki, Dirk, and Falk Schreiber, Centrality analysis methods for biological networks and their application to gene regulatory networks, Gene Regulation and Systems Biology, 2008; Volume 2, pp. 193-201.

[3] Shao-Ting Wang, Jennifer Jin, Pete Rivett, and Atsushi Kitazawa, Graph databases and applications, International Journal of Semantic Computing, 2015; 9(4), pp. 523-545.

[4] Jovelin R, Phillips PC, Evolutionary rates and centrality in the yeast gene regulatory network, Genome Biol. 2009; 10(4): R35.

[5] Mistry, Divya, Roger P. Wise, and Julie A. Dickerson, DiffSLC: A graph centrality method to detect essential proteins of a protein-protein interaction network, PloS One 2017, 12(11): e0187091

[6] Joyce, Karen E., et al. A new measure of centrality for brain networks, PLoS One 2010, 5(8): e12200

[7] Pavlopoulos, G. A., Secrier, M., Moschopoulos, C. N., Soldatos, T. G., Kossida, S., Aerts, J., ... & Bagos, P. G., Using graph theory to analyze biological networks, BioData Mining 2011, 4(1): 10

[8] Wan, Raymond, et al., HAMSTER: visualizing microarray experiments as a set of minimum spanning trees, Source Code for Biology and Medicine 2009, 4(1): 8

[9] Y. Choi and W. Szpankowski, Compression of graphical structures: fundamental limits, algorithms, and experiments, IEEE Transactions on Information Theory, February 2012, 58(2):620–638.

[10] Ahlswede, Rudolf, E-H. Yang, and Zhen Zhang, Identification via compressed data, IEEE Transactions on Information Theory, 1997, 43(1): pp. 48-70.

[11] U. N. Raghavan, R. Albert, and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Physical Review E, 2007, 76(3): 036106.

[12] S. Chen and J. Reif, Efficient lossless compression of trees and graphs, in Proceedings of the IEEE Data Compression Conference (DCC '96), Mar 1996, pp. 428–437.

[13] S. Navlakha, R. Rastogi, and N. Shrivastava, Graph summarization with bounded error, in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, New York, NY, USA: ACM, 2008, pp. 419–432

[14] W. Fan, X. Wang, and Y. Wu, Querying big graphs within bounded resources, in SIGMOD, 2014, pp. 301-312.

[15] Bader, David A., et al., Approximating betweenness centrality, International Workshop on Algorithms and Models for the Web-Graph. Springer, Berlin, Heidelberg, 2007. p. 124-137.

[16] Pavlopoulos, Georgios A., et al., Using graph theory to analyze biological networks, BioData Mining, 2011, 4(1): 10.

[17] Brandes, U., A faster algorithm for betweenness centrality, J. Mathematical Sociology, 2001, 25(2), pp. 163–177.

[18] Brandes, Ulrik, and Christian Pich. Centrality estimation in large networks, International Journal of Bifurcation and Chaos, 2007, 17(7), pp. 2303-2318.

[19] Bader, David A., and Kamesh Madduri. Parallel algorithms for evaluating centrality indices in real-world networks, Parallel Processing International Conference on. IEEE, 2006, pp. 539-550.

[20] Ahmet Erdem Sariyüce, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. 2017. Graph Manipulations for Fast Centrality Computation. ACM Trans. Knowl. Discov. Data 11, 3, Article 26 (April 2017), 25 pages. DOI:https://doi.org/10.1145/3022668

[21] Brandes, Ulrik. "On variants of shortest-path betweenness centrality and their generic computation." Social Networks 30.2 (2008): 136-145.

[22] Huang, Jin, and Charles X. Ling, Rank measures for ordering, European Conference on Principles of Data Mining and Knowledge Discovery, Springer, Berlin, Heidelberg, 2005. p. 503-510.

[23] Wang, S., Zhang, G., Sheu, P., et al., Lossy graph data reduction. International Journal of Semantic Computing,2018; Vol. 12, No. 03, pp. 425-456.

[24] Tang, Jianing, et al, Prognostic genes of breast cancer identified by gene co-expression network analysis, Frontiers in oncology, 2018, 8: 374.

[25] Daemen, Anneleen, et al., Modeling precision treatment of breast cancer, Genome Biology, 2013, 14(10): R110.

[26] Langfelder, P., & Horvath, S., WGCNA: An R package for weighted correlation network analysis, BMC Bioinformatics, 2008, 9(1), 559.


## FIGURES AND TABLES

Figure 1. This is a graph example. Nodes represent genes and edges represent common diseases with influence factors.

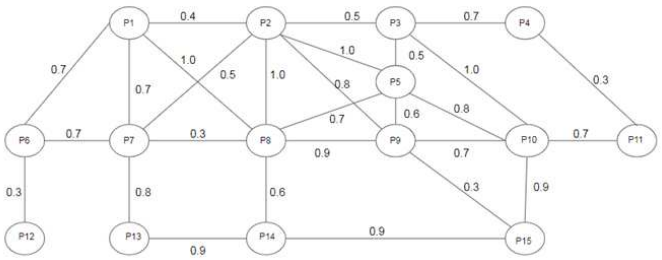



Figure 2. Data preparation and co-expression network construction.

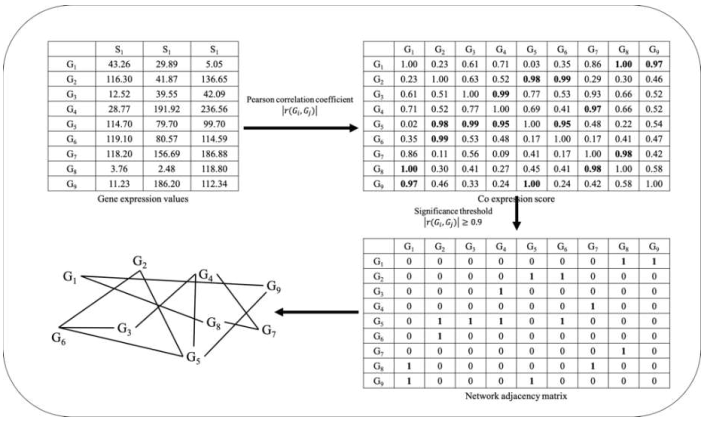


Figure 3. Reduction Ratio of 1+ε LGRSP as ε increases.

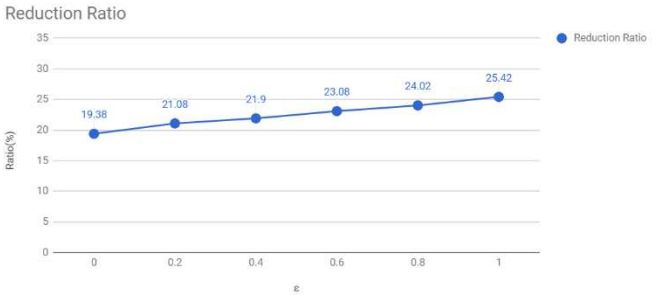



Figure 4. Speedup of 1+ε LGRSP as ε increases.

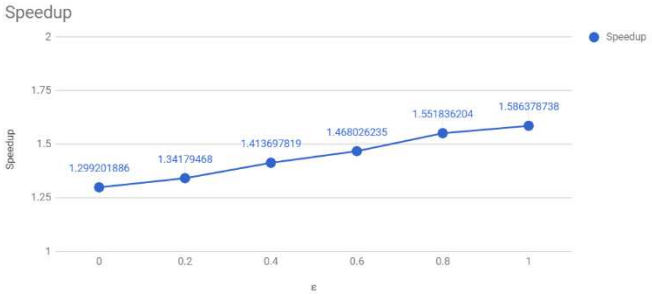



Table 1. Accuracy for betweenness centrality after the GRSP algorithm is applied. We calculate accuracy based on the comparison of the rank of betweenness (rank of btw.) for the graph before (original graph) and after applying the GRSP algorithm (after the GRSP is applied).

| Two Graphs / Rank of btw. | original graph | after GRSP applied |
|---|---|---|
| 1 | ENSG00000155249 | ENSG00000155249 |
| 2 | ENSG00000154415 | ENSG00000154415 |
| 3 | ENSG00000155158 | ENSG00000155158 |
| 4 | ENSG00000217598 | ENSG00000217598 |
| 5 | ENSG00000166710 | ENSG00000166710 |
| 6 | ENSG00000154646 | ENSG00000154646 |
| 7 | ENSG00000154645 | ENSG00000154645 |
| 8 | ENSG00000196785 | ENSG00000196785 |
| 9 | ENSG00000154451 | ENSG00000154451 |
| 10 | ENSG00000214088 | ENSG00000214088 |
| **Accuracy** | | **100%** |

Table 2. Accuracy for betweenness centrality after the LGRSP algorithm is applied. We calculate the accuracy based on the comparison of the rank of betweenness (rank of btw.) for the graph before (original) and after applying the LGRSP algorithm with different error rates (ε=0.1 – 1.0). We only keep the last 6 digits of each gene Ensemble in this table.

| Error rate / Rank of btw. | original | ε=0.1 | ε=0.2 | ε=0.3 | ε=0.4 | ε=0.5 | ε=0.6 | ε=0.7 | ε=0.8 | ε=0.9 | ε=1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 | 155249 |
| 2 | 154415 | 154415 | 155158 | 154415 | 155158 | 154415 | 155158 | 154415 | 155158 | 154415 | 155158 |
| 3 | 155158 | 155158 | 154415 | 155158 | 154415 | 155158 | 154415 | 155158 | 154415 | 155158 | 154415 |
| 4 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 | 217598 |
| 5 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 | 166710 |
| 6 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 | 154646 |
| 7 | 154645 | 154645 | 154645 | 154645 | 154645 | 154645 | 154645 | 154451 | 154645 | 154451 | 154645 |
| 8 | 196785 | 196785 | 196785 | 196785 | 196785 | 196785 | 196785 | 154645 | 196785 | 154645 | 196785 |
| 9 | 154451 | 154451 | 154451 | 154451 | 174332 | 154451 | 154451 | 196785 | 174332 | 196785 | 174332 |
| 10 | 214088 | 154736 | 214088 | 214088 | 154451 | 214088 | 214088 | 214088 | 154451 | 134326 | 154451 |
| **Accuracy** | | **90%** | **100%** | **100%** | **90%** | **100%** | **100%** | **100%** | **90%** | **90%** | **90%** |

# Figures

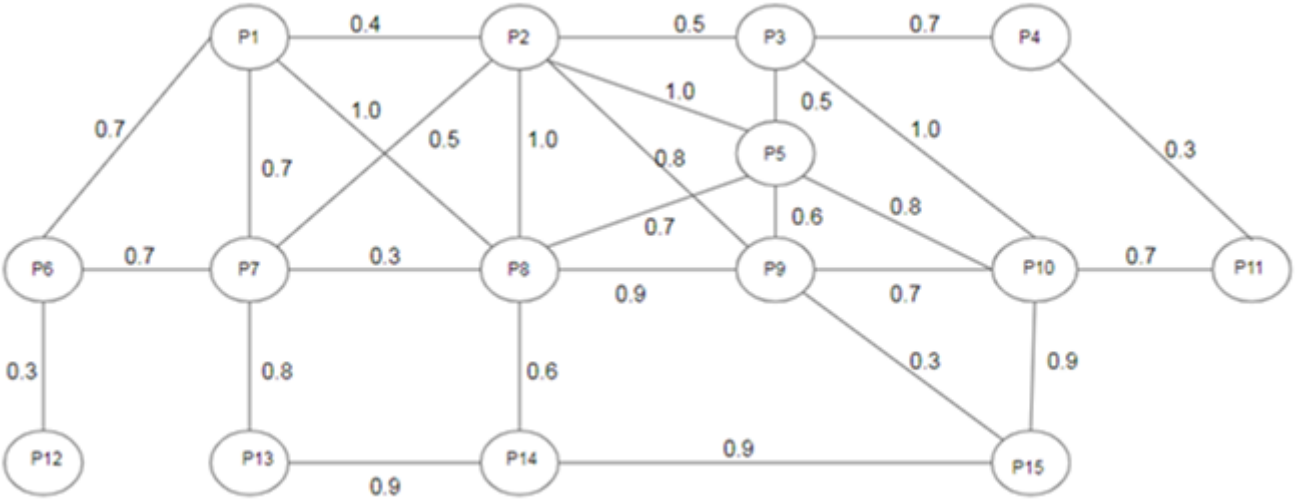
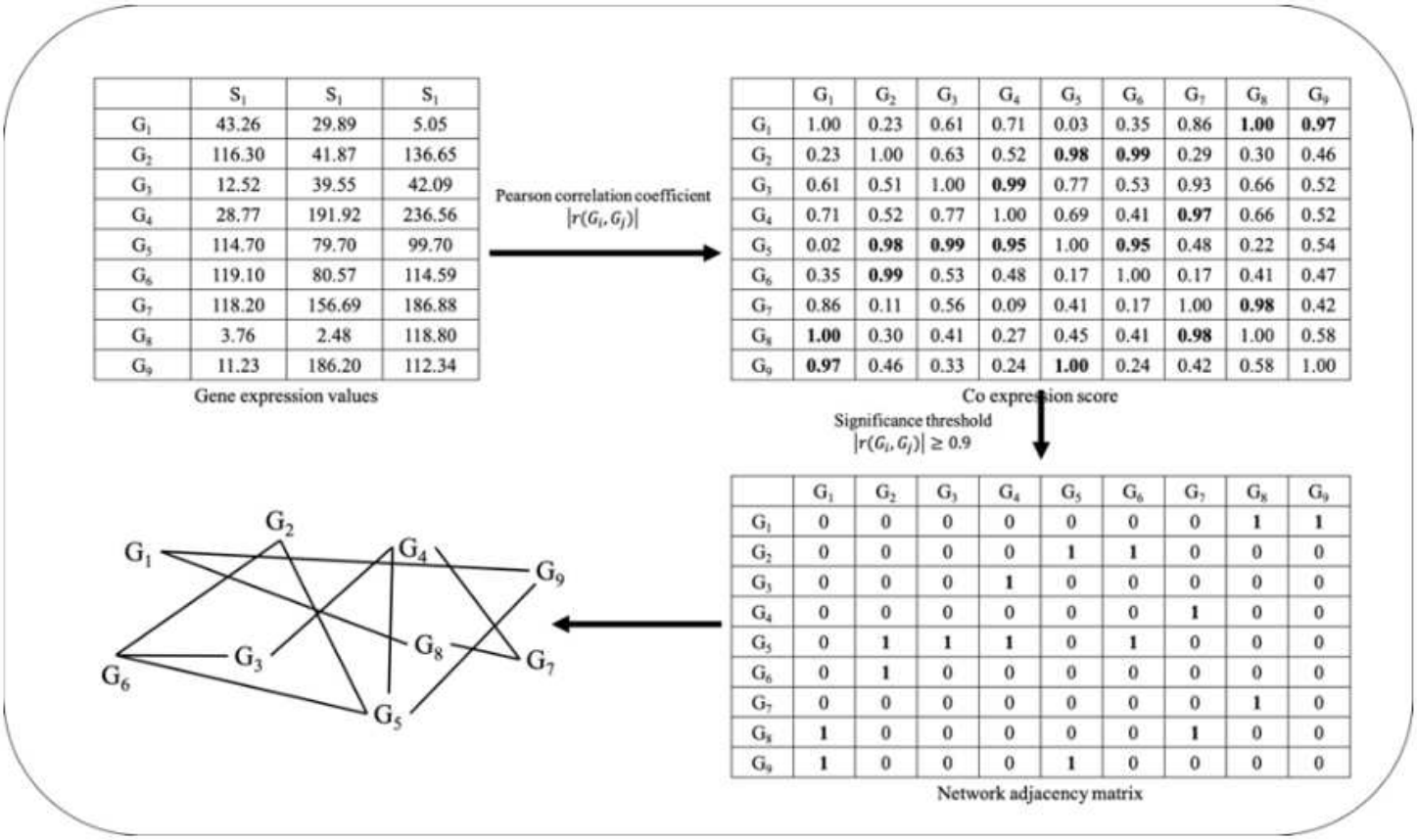



**Figure 1**

This is a graph example. Nodes represent genes and edges represent common diseases with influence factors.




| | S1 | S1 | S1 |
|---|---|---|---|
| G1 | 43.26 | 29.89 | 5.05 |
| G2 | 116.30 | 41.87 | 136.65 |
| G3 | 12.52 | 39.55 | 42.09 |
| G4 | 28.77 | 191.92 | 236.56 |
| G5 | 114.70 | 79.70 | 99.70 |
| G6 | 119.10 | 80.57 | 114.59 |
| G7 | 118.20 | 156.69 | 186.88 |
| G8 | 3.76 | 2.48 | 118.80 |
| G9 | 11.23 | 186.20 | 112.34 |

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 |
|---|---|---|---|---|---|---|---|---|---|
| G1 | 1.00 | 0.23 | 0.61 | 0.71 | 0.03 | 0.35 | 0.86 | **1.00** | **0.97** |
| G2 | 0.23 | 1.00 | 0.63 | 0.52 | **0.98** | **0.99** | 0.29 | 0.30 | 0.46 |
| G3 | 0.61 | 0.51 | 1.00 | **0.99** | 0.77 | 0.53 | 0.93 | 0.66 | 0.52 |
| G4 | 0.71 | 0.52 | 0.77 | 1.00 | 0.69 | 0.41 | **0.97** | 0.66 | 0.52 |
| G5 | 0.02 | **0.98** | **0.99** | **0.95** | 1.00 | **0.95** | 0.48 | 0.22 | 0.54 |
| G6 | 0.35 | **0.99** | 0.53 | 0.48 | 0.17 | 1.00 | 0.17 | 0.41 | 0.47 |
| G7 | 0.86 | 0.11 | 0.56 | 0.09 | 0.41 | 0.17 | 1.00 | **0.98** | 0.42 |
| G8 | **1.00** | 0.30 | 0.41 | 0.27 | 0.45 | 0.41 | **0.98** | 1.00 | 0.58 |
| G9 | **0.97** | 0.46 | 0.33 | 0.24 | **1.00** | 0.24 | 0.42 | 0.58 | 1.00 |

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 |
|---|---|---|---|---|---|---|---|---|---|
| G1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | **1** |
| G2 | 0 | 0 | 0 | 0 | **1** | **1** | 0 | 0 | 0 |
| G3 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| G4 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| G5 | 0 | **1** | **1** | **1** | 0 | **1** | 0 | 0 | 0 |
| G6 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| G8 | **1** | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| G9 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 |

**Figure 2**

Data preparation and co-expression network construction.

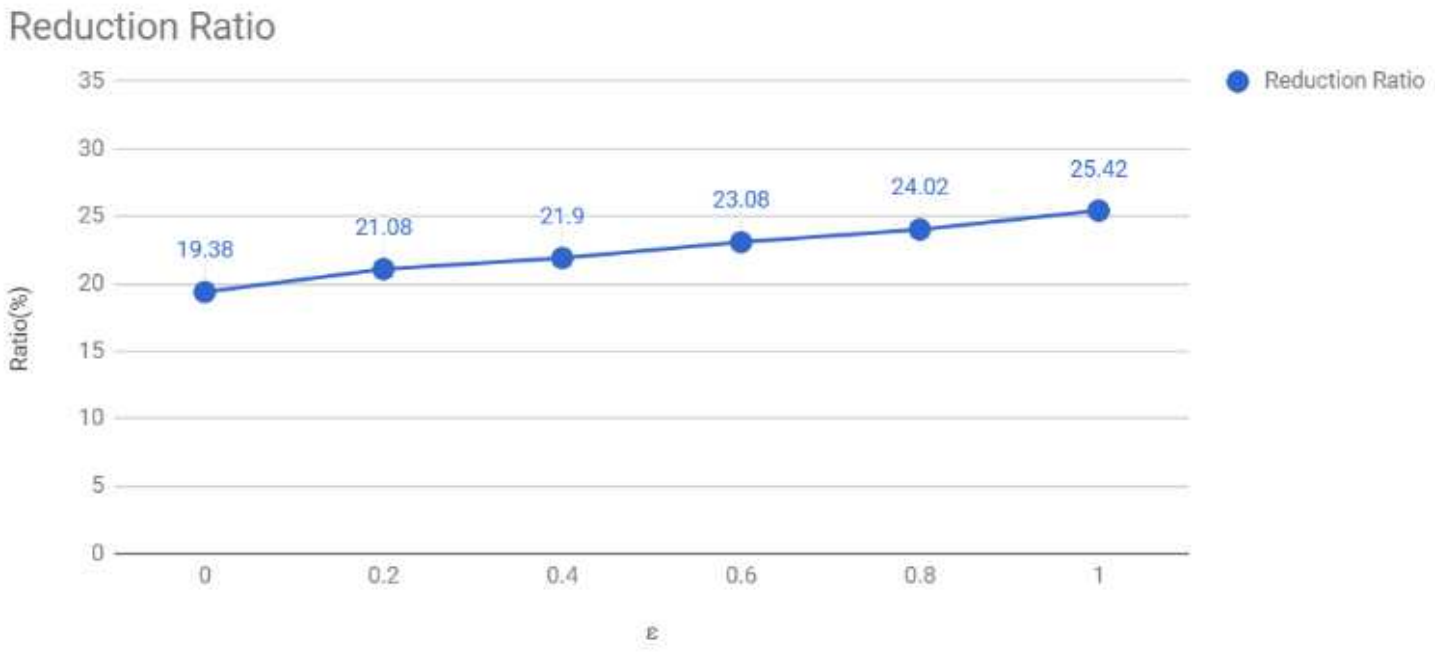



**Figure 3**

Reduction Ratio of 1+ LGRSP as  increases.

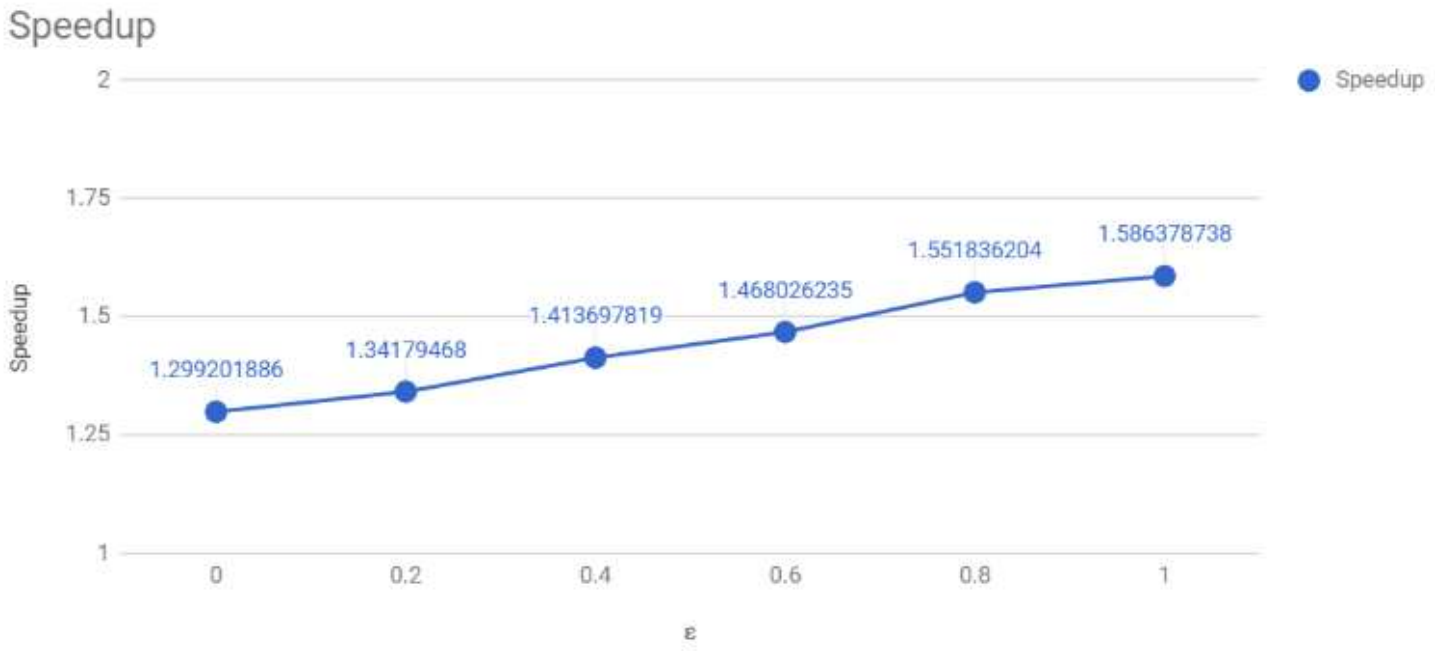



**Figure 4**

Speedup of 1+ LGRSP as  increases.