OPEN ACCESS Freely available online

Journal of Proteomics & Bioinformatics

Research Article

GraphBreak: Tool for Network Community Based Regulatory Medicine, Gene Co-expression, Linkage Disequilibrium Analysis, Functional Annotation and More

Abhishek Narain Singh*

Schiller International University, Heidelberg Campus, Zollhofgarten 1, 69115 Heidelberg, Germany

ABSTRACT

Graph network science is becoming increasingly popular, notably in big-data perspective where understanding individual entities for individual functional roles is complex and time consuming. It is likely when a set of genes are regulated by a set of genetic variants, the genes set is recruited for a common or related functional purpose. Grouping and extracting communities from network of associations becomes critical to understand system complexity, thus prioritizing genes for disease and functional associations. Workload is reduced when studying entities one at a time. For this, we present GraphBreak, a suite of tools for community detection application, such as for gene co-expression, protein interaction, regulation network, etc. Although developed for use case of eQTLs regulatory genomic net-work community, study-results shown with our analysis with sample eQTL data-GraphBreak can be deployed for other studies if input data has been fed in requisite format, including but not limited to gene co-expression networks, protein-protein interaction network, signaling pathway and metabolic network. GraphBreak showed critical use case value in its downstream analysis for disease association of communities detected. If all independent steps of community detection and analysis are a step-by-step sub-part of the algorithm. GraphBreak can be considered a new algorithm for community based functional characterization. Combination of various algorithmic implementation modules into a single script for this purpose illustrates GraphBreak's novelty. Compared to other similar tools, with GraphBreak we can better detect communities with over representation of its member genes for statistical association with diseases, therefore target genes which can be prioritized for drug-positioning or drug-repositioning as the case be.

Keywords: eQTL–expression quantitative trait loci; SNP–Single Nucleotide Polymorphism; LD–Linkage Disequilibrium

INTRODUCTION

By signaling pathway networks, bio-based chemical molecular networks-such as protein–protein interaction networks, gene co-expression networks, gene regulatory networks, metabolic networks- provide a graphical representation of cellular and tissue systems. A bioarXiv preprint of this paper was published in early 2021 [1] and the abstract was published in the book of abstract of BIOCOMP 2020 [2]. To guide biological experiments, networks must be analyzed by means of community detections, if every gene transcript, gene product, protein, genotypic variants, such as SNPs, were to be characterized individually they could be very difficult to perform. To increase understanding of network phenomena, network scientists deploy models. For this, we introduce GraphBreak, a network analysis tool to understand variant based regulation of expression of genes as a community. It conducts community analysis of overrepresentation association with disease and other function

Correspondence to: Abhishek Narain Singh, Schiller International University, Heidelberg Campus, Zollhofgarten 1, 69115 Heidelberg, Germany, E-mail: abhishek.narain@iitdalumni.com

Received: 04-Feb-2023, Manuscript No. JPB-23-21703; **Editor assigned:** 06-Feb-2023, Manuscript No. JPB-23-21703 (PQ); **Reviewed:** 21-Feb-2023, Manuscript No. JPB-23-21703; **Revised:** 28-Feb-2023, Manuscript No. JPB-23-21703 (R); **Published:** 08-Mar-2023, DOI: 10.35248/0974-276X. 23.16.625

Citation: Singh AN (2023) GraphBreak: Tool for Network Community Based Regulatory Medicine, Gene Co-expression, Linkage Disequilibrium Analysis, Functional Annotation and More. J Proteomics Bioinform.16:625.

phenotypes, along with linkage association of SNP variants for causative prioritization. This network-based community detection has been examined in detail from genomic variants, such as SNPs with expression of genes, but GraphBreak can be used for any network analysis, such as those discussed above. For example, WGCNA [3] has been widely used for gene co-expression analysis. Compared to WGCNA, where a weight is assigned for any connections between two genes, GraphBreak assumes a weight of 1 for every connection. Condor, a similar tool for community detection in gene regulation context Platig et al., [4] is used for the purpose that models' network assume it to be a bipartite graph.

Motivation for developing a new community detection tool

GraphBreak was inspired by Condor [4], which implements modularity detection in bipartite network [5]. Regulatory connections between genomic variants and associated gene expressed is bipartite by de-fault, given that there were no connections between any two genomic variants (such as SNPs), nor any connections between any two genes expressed. As simple community detection in a connected graph is performant in obtaining the communities, there was no need to classify community detection problem as a bipartite graph if data itself is arranged in such a way. For this, our approach was to find an implementation module for various community detection algorithms with preferably parallel computing capability, which can then be used for our own purpose, such as for regulatory genomic network analysis.

Existing algorithms for community detection influencing GraphBreak development

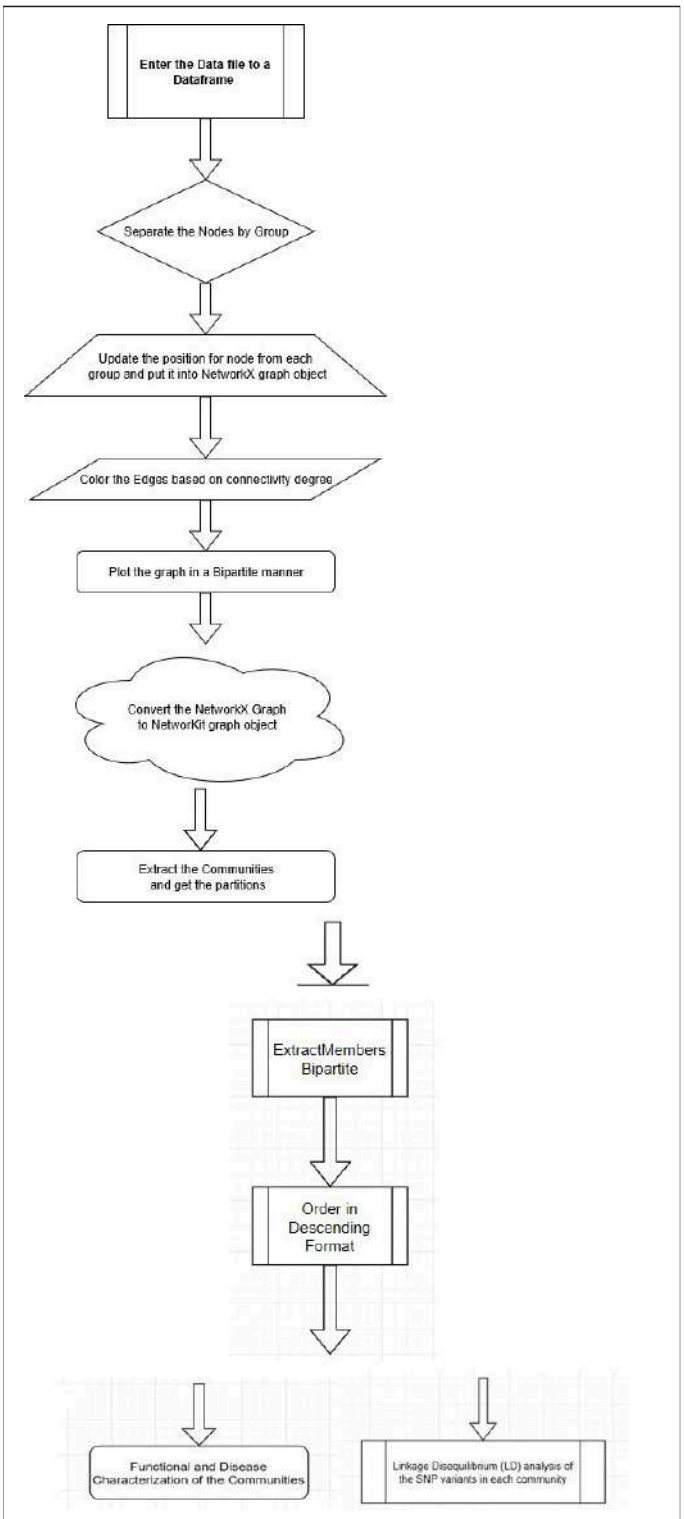
Sets of nodes are partitioned by a class of network analysis methods into sub-sets depending on graph structure. For instance, all nodes in a connected component are reachable from each other. Using breadth-first search, a network's connected components can be computed in linear time. Community detection is the task of identifying groups of nodes in the network that are significantly more densely connected among another than to the rest of nodes. Various definitions of the structure to be discovered community is a data mining problem. NP-hard optimization problem defines this task by community quality measures-first and foremost modularity [6]. The goal of GraphBreak was for end users to choose from popular community detection algorithms, such as Louvain's algorithm [7] and Girvan-Newman algorithm [6].

NetworKit [8] was chosen for community detection because with parallel script in place, we should be ready in the future for high volume of data. NetworKit is a Python Application Programming Interface (API) that approaches community detection from the perspective of modularity maximization and engineer parallel heuristics, which deliver a good balance between solution quality and running time. NetworKit API has been previously used for solving biological network problems such as Protein-Protein Interaction (PPI) network [9], therefore reassuring this API's deployment for biochemical application. PLP algorithm implements community detection by extracting communities from a labeling of the node set, or label propagation [10]. Community detection with Louvain method [7] can be classified as a locally greedy, bottom-up multilevel algorithm. NetworKit authors recommend, PLM algorithm with optional refinement step as the default choice for modularity-driven community detection in large networks. PLP delivers a better time to solution for very large network in the range of billions of edges, but with a qualitatively different solution and worse modularity. Therefore Girvan-Newman and Louvain algorithm for community detection using Networkx [11] module in Python and the parallel version of Louvain algorithm (PLM) using the NetworKit module in Python was implemented, as we felt it was illogical to use PLP to save time but compromise modularity solutions obtained, as for the time being our data sizes are not in the order of terabytes that would require us to see the problem from a really 'Big Data' perspective.

NetworkX has functionalities to help convert the graph data from one format to another. NetworkX [11], a de-facto standard for the analysis of small to medium networks in a Python environment, is not suitable for massive networks due to its pure Python implementations. Even though our data size was not of the category very 'Big Data', it was large enough to be classified as 'Big Data', so using algorithmic implementations in NetworkX for community detection was clearly time and resource demanding. Girvan-Newman and Louvain algorithm were implemented with the help of NetworkX Python module for small and medium size data, we strongly emphasize use of GraphBreak by parallel implementation of Louvain's algorithm (PLM) for large-scale data, which we developed using NetworKit Python module. Data sizes are currently not extremely large, for practical purposes, so the choice of deploying Louvain algorithm or Girvan-Newman will not have a detrimental effect on execution time.

GraphBreak pseudocode and workflow

Flowchart for GraphBreak workflow, followed by pseudocode is shown below



Flow chart for workflow of GraphBreak.

MATERIALS AND METHODS

GTEx [12] V7 data was used for analysis, cis-eQTLs of which was generated by FastQTL [13] by GTEx consortium as the following downloadable file GTEx_Analysis_v7_eQTL.tar.gz and GTEx_Analysis_v8_eQTL.tar.gz. We took only sample data from artery – aorta from GTEx V7 and artery-coronary tissue from GTEx V8 as a test case given the significance of this tissue in coronary artery disease. GraphBreak and Condor were executed to detect communities and see the efficacy by downstream analysis of genes in each of communities for their statistical significance with disease. The computing facility from 'Bioinformatics Centre at University of Eastern Finland', Kuopio campus was used.

Execution of GraphBreak and Condor for community detection

GraphBreak was used to plot connections between variants and genes. Figure 1 below shows a sample GraphBreak plot for 3 genes as an example.

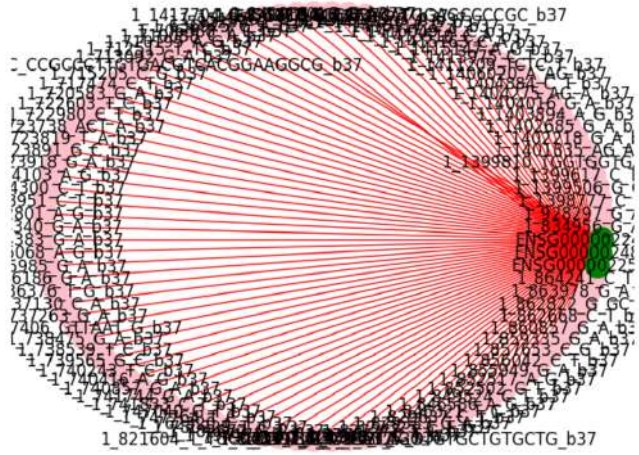


Figure 1: GraphBreak association plot for 3 genes with the genomic variants.

Connections between genomic variants and genes were plotted using Condor as in Figure 2.

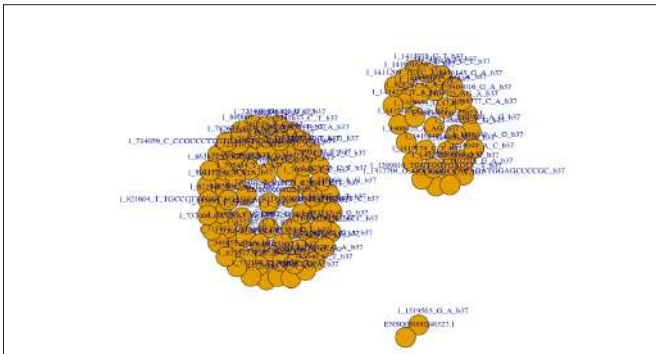


Figure 2: Condor association connections plot of sample 3 genes with the genomic variants.

Community detection methods were called once connections were plotted into memory of data structures. Correspondences with Condor authors lead to them developing community plots and 16 communities were detected by Condor as shown in Figures 3 and 4.

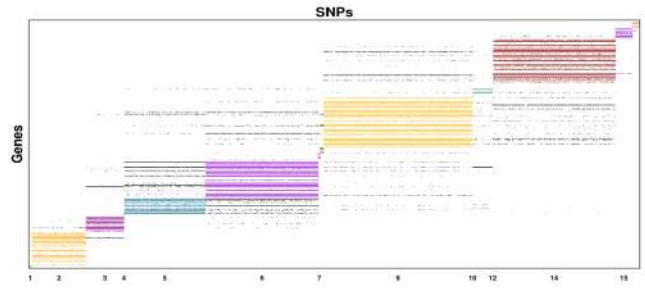


Figure 3: Condor communities for sample tissue data Artery-Aorta GTEx V7.

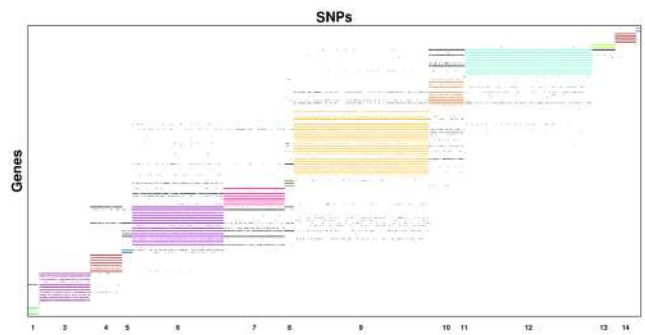


Figure 4: Condor communities for sample tissue data Artery-Coronary GTEx V8.

GraphBreak obtained about 56 communities comprising of expressed genes and associated genomic variants for each, GraphBreak plots for 5 communities with only gnes, shown in Figure 5.

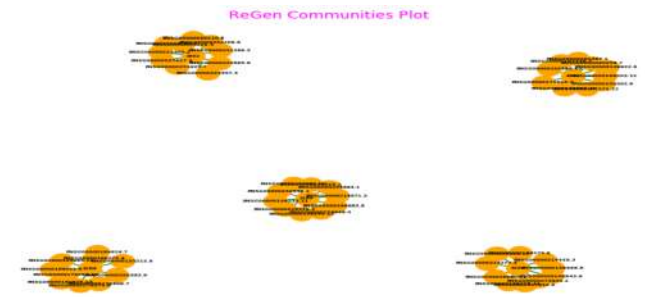


Figure 5: GraphBreak (also named as ReGen-for regulatory genomic work) communities plot for 5 of the communities.

Plotting 57 communities, using GraphBreak would have made the plot non-aesthetic, so a few were plotted to ensure that we received expected result. Sum of the number of genes in each of the communities obtained from Condor and GraphBreak were plotted. Condor does not use Louvain algorithm, but igraph, for the purpose of finding the communities Barber [5], another method which implements modularity detection in bipartite network. Figures 6 and 7 show the sum of the genes in each community obtained from Condor and GraphBreak, respectively.

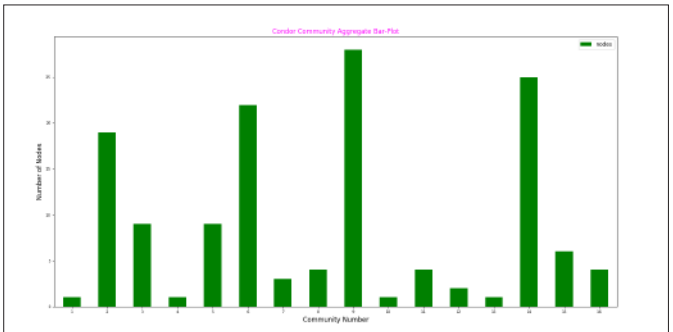


Figure 6: Bar plot of sum of genes in each of the communities obtained by Condor. **Note:** (▬) Nodes.

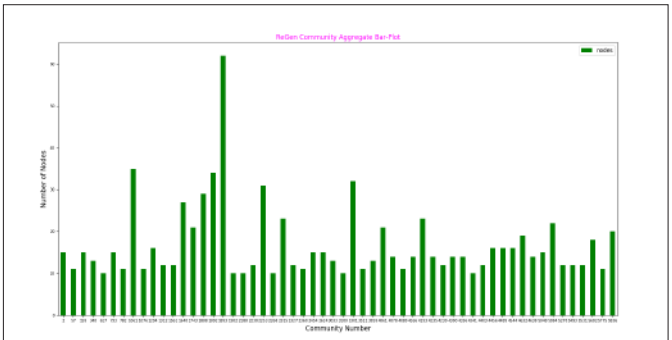


Figure 7: Bar plot for sum of genes in each of the communities obtained from GraphBreak. **Note:** (▬) Nodes.

Each community obtained by Condor and GraphBreak had genomic variants, such as SNPs with them, which can be plotted in a similar way. For our current illustration purpose, we focus only on set of genes derived for each of the communities as downstream analysis.

Downstream analysis methods

Over-representation analysis of genes obtained for each of the communities: Set of genes obtained from communities by Condor or GraphBreak can be tested for their significance of association for functionalities and diseases by means of over-representation analysis as detailed in the following paper Drăghici et al., [14], high performance implementation of which was published here Fabregat et al., [15] are based on collection of knowledge bases from experimental findings. Reacfoam [15] improvised the graphical representation that uses 'Voronoid Diagram' to depict statistically significant associations for over-representation analysis with functionalities and diseases and uses open source knowledge of biochemical pathways. Figure 8 shows statistically significant results for one community's (1061) set of genes being significantly associated with diseases, metabolism of proteins, immune system, metabolism, and cell cycle.

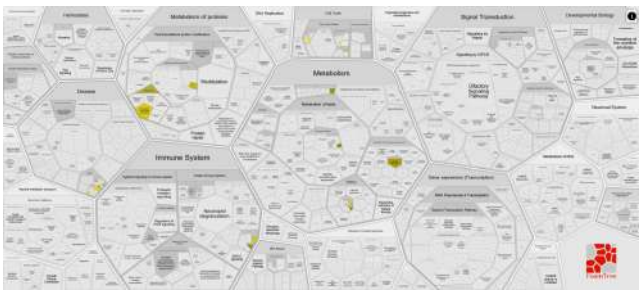


Figure 8: GraphBreak Community 1061 genes 'Voronoi Diagram' Reacfoam p-value association.

LD Prioritization analysis of genomic variants obtained for each of the communities: Using existing database for variant classifications, such as RegulomeDB [16], which classifies genomic variants based on already known properties, known as an eQTL, or having transcription factor binding site, DNase peak, etc., genomic variants grouped in each community can then be annotated for functionalities. Next, variants can be evaluated for LD amongst each other to reduce possible causative variants from those shortlisted in step one. Figure 9 below shows a simple LD association plot obtained by LDMatrix web-applicationt part of LDlink package [17].

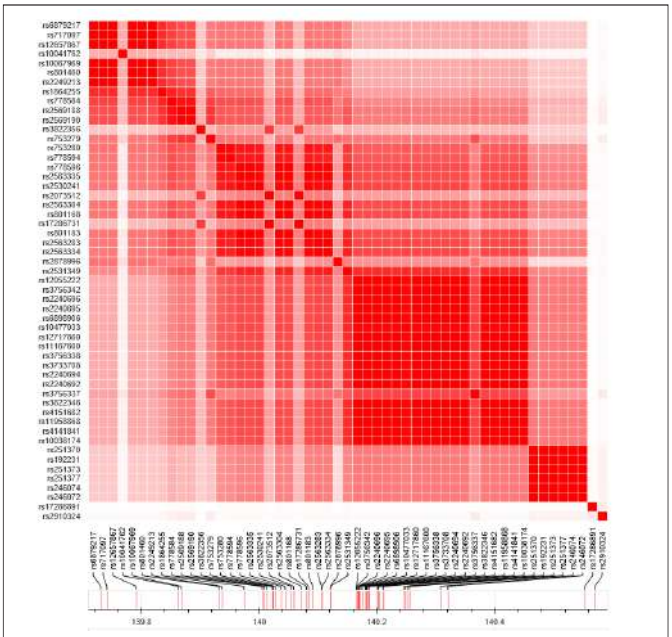


Figure 9: Classifying variants based on their LD associations.

Benchmarking

Igraph [18] and graph-tool's [19] functionality and target use cases are similar to NetworKit, confirmed by authors of NetworKit. Packaged as Python modules, they provide a broad feature set for network analysis workflows and active user communities. They address scalability issues by implementing core data structures and algorithms in C or C++; graph-tool builds on Boost Graph Library and parallelizes some kernels using Open MP.

Gephi [20], a GUI application for Java platform with a strong focus on visual network exploration, is geared towards network science but differs in important aspects from NetworKit. Pajek [21], a proprietary GUI application for Windows operating systemoffers visualization features with analysis capabilities like NetworKit. Pajek XXL, a variant, uses less memory and focuses on large datasets. SNAP [22] net-work analysis package has recently adopted the hybrid approach of C++ core and Python interface.

Related efforts from algorithm engineering community are KDT [23] (algebraic built, distributed parallel backend), GraphCT [24] (focused on massive multithreading architectures like Cray XMT), STINGER (a dynamic graph data structure with some analysis capabilities) [25] and Ligra [26] (a recent shared memory parallel library). These offer high performance through native, parallel implementations of certain kernels. For NetworKit to characterize a complex network, it would need a substantial set of analytics, which frameworks currently do not provide. These tools can be further studied for their usability in current regulatory genomic network perspective if efficiency was improved. NetworKit module was preferred in the creation of GraphBreak as it already offers parallel implementations of various community detection algorithms and based on its benchmark performance analysis with closest alternative tools, such as igraph and graph tool.

As GraphBreak uses NetworKit [8] and NetworkX [11] Python modules, its benchmarking and comparative benchmarking depends on benchmarking of these modules, which were done by the authors of these tools (cited in the references). NetworKit outperformed igraph and graph-tool in a comparative benchmarking, making it the preference for our 'Big Data' eQTL biological information. Previously, Kanter et al., [27] used igraph for community detection. Algorithms used for these tools would be similar, but the optimized implementation differs, as that has an effect on the computational resource utilization, parallel compute capability, and other performance metrics, apart from the fact that an easy to use programming interface is also needed.

Another scalability factor is memory footprint of the graph data structure. NetworKit, with 260 M edges of the uk-2002 web graph occupies 9 GB with a lean implementation, compared to igraph (93 GB) and graph-tool (14 GB), benchmarked by the authors NetworKit increasing our confidence in deploying this module in Python for developing GraphBreak.

Because of NetworKit's competitive disk I/O, the parser is significantly faster for non-attributed graphs. NetworkX was used to load the data to memory for initial calculations and graph plotting then a feature was used to convert to NetworKit format. Although this used additional time for conversion, it was convenient while developing GraphBreak. Comparing NetworKit and the various algorithms implemented by these tools, Figure 10, shows that it outperforms others for community detection-making it our preference as a Python module.

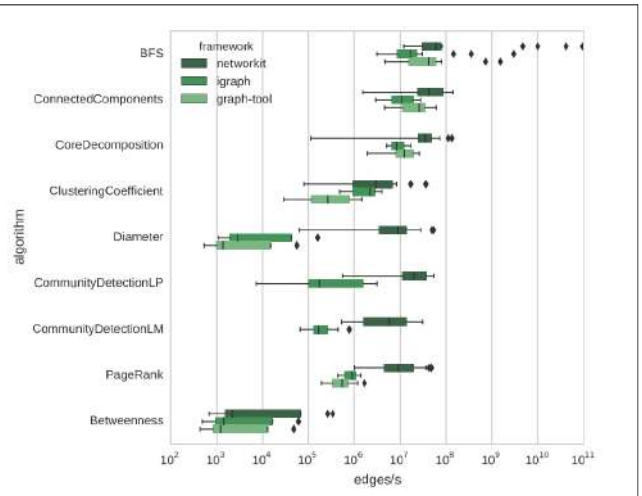


Figure 10: Processing rates of typical analytics tasks: NetworKit in comparison with igraph and graph-tool. **Note:** (▬) networkit,(▬) igraph, (▬) graph-tool.

NetworKit, igraph and graph-tool rely on hybrid architecture of C/C++ implementations with a Python interface. Igraph uses non-parallel C code while graph-tool features parallelism. Graph-tool's approach to community detection is different; hence the comparison is between igraph and NetworKitfor this functionality, which is used in GraphBreak. NetworKit's authors performed the benchmark [8]. GraphBreak would deploy corresponding Python modules to igraph or another community detection tool if drastics improvements were made. NetworKit was the only option for community detection, as closest alternatives, i.e, igraph and graph-tool; require a significantly higher amount of memory and computational time.

I/O should be taken in account for real workflows, as getting a large graph from hard disk to memory often takes longer than actual analysis. For benchmark, authors of NetworKit chose GML graph file format for input files because it is supported by all three frameworks. They observed that NetworKit parser is significantly faster for these non-attributed graphs in Figure 11.

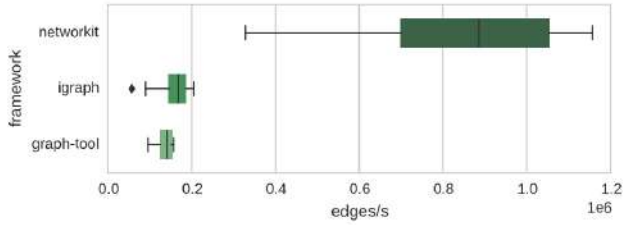


Figure 11: I/O rates of reading a graph from a GML file: NetworKit in comparison to igraph and graph-tool. **Note:** (▬) networkit, (▬) igraph, (▬) graph-tool.

This feature can be considered for future improvements in GraphBreak. Graph data is currently read using NetworkX Python module for initial computation work, such as graph plotting, and uses a converter for NetworKit format for community detection.

GraphBreak was compared to CONDOR [4], results in comparison chart in Figures 12 and 13.

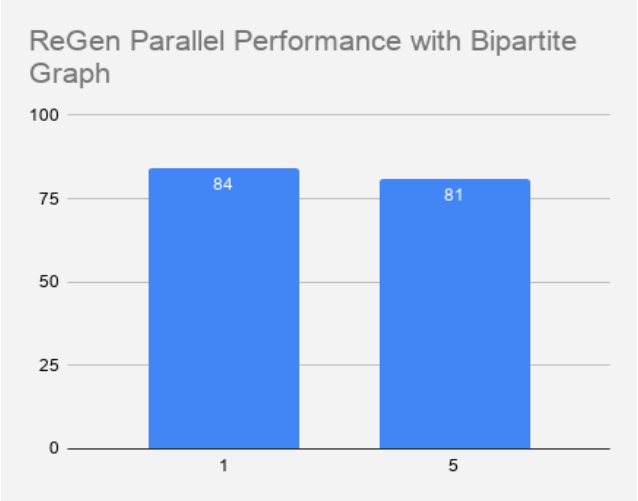


Figure 12: Parallel performance of GraphBreak (in this case regulatory network analysis so also called ReGen) is not greatly benefitted when the bipartite graph also needs to be plotted. X-axis number of computing cores, Y-axis time in minutes. (Data tested: Artery-Coronary tissue eQTL GTEx V8.).

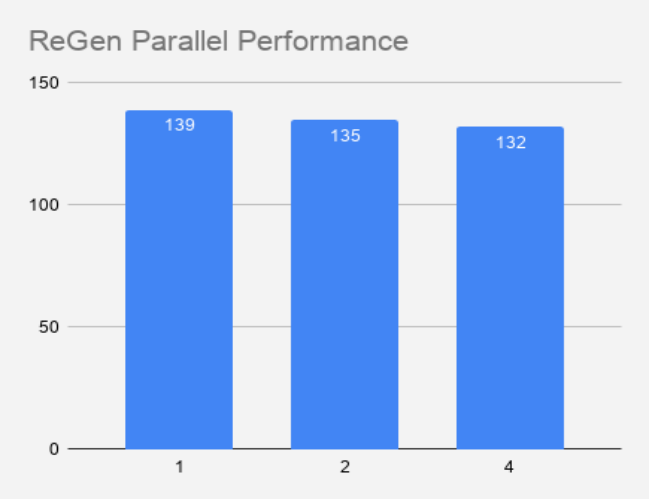


Figure 13: Execution time for GraphBreak (in this case regulatory network analysis so also called ReGen), Y-axis (in seconds), is far less when community detection is done without bipartite plotting. X-axis is number of cores used for computing. (Data tested: Artery-Coronary tissue eQTL GTEx V8.).

Broad level comparison of two-regulation based community detection tools, Figure 12, shows GraphBreak can detect a greater number of communities that are over-represented for disease.

Parallel mode choice and performance: Speedup would be negatively affected by substantial initial section of the algorithm that is serial in nature. Additionally, speedup would also depend on how well parallelism has been exploited for PLM algorithm. We tested GraphBreak's performance for following system:

Linux 3.10.0-957.12.2.el7.x86_64#1 SMP Tue May 14 21:24:32 UTC 2019 x86_64 x86_64x86_64 GNU/Linux

For data Artery Coronary significant association in GTEx: rwxrwxr-. 1 domain users 90M Oct 16 17:10 Artery_Coronary.v8.signif_variant_gene_pairs.txt*

For varying number of compute cores (Parallel Louvain Algorithm) with bipartite plots (Note: This should be done when data is small since bipartite graph plot-ting takes substantial amount of time, and more importantly, the graph plotted would be uninformative and unaesthetic for large amount of data).

For varying number of compute cores (Parallel Louvain Algorithm) without bi-partite plots we see a significant reduction in execution time.

RESULTS AND DISCUSSION

17 of 56 communities detected using GraphBreak for given dataset of Artery-Aorta eQTL associations for GTEx V7 were able to significantly associate its set of genes with disease; for given dataset, 0 of 16 communities detected using Condor showed any statistically significant association. While Condor could predict 2 Artery-Aorta eQTL associations for GTEx V8, GraphBreak was able to predict 8 communities with significant disease association. This illustrates that community detection by different algorithmic approaches lead to different set of results and can have a different impact on downstream analysis. Despite the algorithm being at the disposal of the computer science world, we find GraphBreak novel role in the field of regulation of gene expressions in this gap in research. The data on hand dictates the results of communities. With another dataset, the results from GraphBreak show significant association with diseases and other pathway functionalities while Condor's results underperform.

Many genomic variants were classified based on their regulation information in prior knowledge and could prioritize those which are already known to be an eQTL by some other study for functional validation. They were also investigated for LD. New eQTLs could be found. This paper is software and methods paper, rather than a scientific results paper, so reporting those genes and variants would defeat the purpose. Detailed biological and biomedical significance of results obtained using GraphBreak and Condor for artery-aorta and other relevant tissues for our research interest in coronary artery disease would need to be presented in a separate publication. GraphBreak could aid other scientists with their research interests in other tissues and samples of their interests for GTEx or any other datasets [28,29].

CONCLUSION

GraphBreak lies at the intersection of network, regulatory genomic science and other analysis such as gene-coexpression. State-of-the-art algorithms for network analysis tasks were incorporated into Python modules, and then coding implementation was done as read-to-use software. This produced a tool suite of network analytics based, community generators and utility software to explore and characterize regulatory networks, such as that of eQTLs network data sets on typical multicore processor.

Scalability with fast parallel algorithms is limited by size of shared memory: A standard multicore workstation with 256 GB RAM can process up to 1010 edge graphs or functional associations between a genomic variant; this value is large enough even if the p-value for associations between the genomic variant and gene expression is kept high as a threshold. In the future, it could assign different weights per edge connections, such as weight being auto-assigned based on p-value of the associations.

This article shows GraphBreak's application for targeting genes by overexpression analysis and ability to find the corresponding associated genomic regulatory variants. GraphBreak suite could incorporate other community detection algorithms. Although it takes more time than Condor, in its current state GraphBreak performs well for detecting communities. Difference in algorithms used for Condor and GraphBreak impacts community size and members making it unfair to compare the performances of these tools. GraphBreak's actual performance should be benchmarked with its dependency of NetworkX and NetworKit their authors already showed superior performances in their papers. Newer algorithms could be developed given the NP-hard nature of community detection algorithms, which would be useful in improvising GraphBreak suite to incorporate more algorithmic implementation and options.

From the medical biological perspective, statistical association of community members to any known disease would be valuable. From test data taken for analysis, GraphBreak is better than Condor for this. Many of communities derived from GraphBreak leveraged the importance of community detection for statistically significant disease associations for Artery-Aorta tissue analyzed for GTEx V7, compared to none by Condor; while 2 communities came up with disease association for Artery-Coronary tissue analyzed for GTEx V8 data compared to 8 by GraphBreak. Both can be applied for other relevant tissues for detecting common set of communities having a set of genomic variants regulating a set of gene expression from medical-biological perspective. This paper showed how genes from each community can be analyzed downstream for their statistical associations with disease.

Changing community detection algorithm from Louvain and Girvan-Newman to Leiden algorithm, which may be better than Louvain algorithm as recently published Traag VA et. al., could lead to future improvements. As NetworkX module functionality, independent of community detection task, performs graph plotting for bipartite nature of connections, it creates an opportunity of parallelism that can be exploited in future improvisation of GraphBreak code. Future work of GraphBreak would also make a comparison of Girvan-Newman and Leiden algorithm implementation as well for benchmarking purpose. Future development can also see automated downstream analysis of the individual genomic variants such as SNPs in each community with respect to their linkage disequilibrium such as those in cis relation, and also automation to classify the SNPs based on database of pre-existing regulatory information known.

Another possibility of future work would be to use the concept of shingling and hashing as proposed by Gibson, to generate

graph networks where for each of the nodes the values it contains could be a set of expressed genes, SNPs, known disease, tissue observed, etc., in order to get a consensus of set of genes, SNPs, tissues affected in a disease or trait of interest.

REFERENCES

1. Singh AN. GraphBreak: Network Community detection and annotation in biomedical regulatory informatics context. bioRxiv. 2021.
2. Singh AN. GraphBreak, Book of abstract, The 21st International Conference on Bioinformatics and Computational Biology (BIOCOMP 2020) as part of American Council on Science & Education / CSCE 2020, ISBN # 1-60132-512-6.
3. Langfelder P, Horvath S. WGCNA: An R package for weighted correlation network analysis. BMC Bioinform. 2008;9(1):1-3.
4. Platig J, Castaldi PJ, deMeo D, Quackenbush J. Bipartite community structure of eQTLs. PLoS Comput Biol. 2016;12(9):e1005033.
5. Barber MJ. Modularity and community detection in bipartite networks. Phys Rev E. 2007;76(6):066102.
6. Girvan M, Newman ME. Community structure in social and biological networks. PNAS. 2002;99 (12):7821-7826.
7. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp.2008(10):P10008.
8. Staudt CL, Meyerhenke H. Engineering parallel algorithms for community detection in massive networks. IEEE Trans Parallel Distrib Syst. 2015;27(1):171-184.
9. Flick P. Analysis of human tissue-specific protein-protein interaction networks. Master's thesis, KIT. 2014.
10. Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E. 2007;76(3):036106.
11. Hagberg A, Swart P, S Chult D. Exploring network structure, dynamics, and function using NetworkX. LANL, Los Alamos, NM (United States); 2008.
12. Genet N. The Genotype-Tissue Expression (GTEx) project. Nat Genet. 2013;45(6):580-585.
13. Ongen H, Buil A, Brown AA, Dermitzakis ET, Delaneau O. Fast and efficient QTL mapper for thousands of molecular phenotypes. Bioinform. 2016;32(10):1479-1485.
14. Drăghici S, Khatri P, Martins RP, Ostermeier GC, Krawetz SA. Global functional profiling of gene expression. Genomics. 2003;81(2):98-104.
15. Fabregat A, Jupe S, Matthews L, Sidiropoulos K, Gillespie M, Garapati P, et al. The reactome pathway knowledgebase. Nucleic Acids Res. 2018;46(D1):D649-55.
16. Boyle AP, Hong EL, Hariharan M, Cheng Y, Schaub MA, Kasowski M, et al. Annotation of functional variation in personal genomes using RegulomeDB. Genome Res. 2012;22(9):1790-17977.
17. Machiela MJ, Chanock SJ. LDlink: A web-based application for exploring population-specific haplotype structure and linking correlated alleles of possible functional variants. Bioinform. 2015;31(21):3555-3557.
18. Csardi G, Nepusz T. The igraph software package for complex network research. Int J Complex Syst. 2006;1695(5):1-9.
19. Harmon OR, Mercier D, Guala B, Brown M, Burdick C. Graph tool. J Econ Educ. 2012;43(1):107-108.
20. Bastian M, Heymann S, Jacomy M. Gephi: An open source software for exploring and manipulating networks. IProc Int AAAI Conf Weblogs Soc Media 2009;361-362.
21. Batagelj V, Mrvar A. Pajek—analysis and visualization of large networks. Springer Berlin Heidelberg; 2004.
22. Leskovec J, Sosic R. SNAP: A general purpose network analysis and graph mining library in C++.
23. Lugowski A, Alber D, Buluç A, Gilbert JR, Reinhardt S, Teng Y, et al. A flexible open-source toolbox for scalable complex graph analysis. InProceedings of the 2012 SIAM International Conference on Data Mining 2012;930-941. SIAM J Appl Math.
24. Ediger D, Jiang K, Riedy EJ, Bader DA. Graphct: Multithreaded algorithms for massive graph analysis. IEEE Trans. Parallel Distrib Syst. 2012;24(11):2220-2229.
25. Riedy DE, Bader DA. STINGER: High performance data Structure for streaming graphs. In 2012 IEEE Conference on High Performance Extreme Computing.
26. Shun J, Blelloch GE. Ligra: A lightweight graph processing framework for shared memory. Proc ACM SIGPLAN Symp Princ Pract Parallel Program. 2013;135-146.
27. Kanter I, Yaari G, Kalisky T. Applications of community detection algorithms to large biological datasets. Deep Sequencing Data Analysis. 2021:59-80.
28. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: Guaranteeing well-connected communities. Sci Rep. 2019;9(1):5233.
29. Gibson D, Kumar R, Tomkins A. Discovering large dense subgraphs in massive graphs. In proceedings of the 31st international conference on very large data bases. 2005;721-732.