Hypothesis generation for rare and undiagnosed diseases through clustering and classifying time-versioned biological ontologies

Michael S. Bradshaw [1], Connor P. Gibbs [2], Skylar Martin [1], Taylor Firman [3], Alisa Gaskell [3], Bailey K. Fosdick [4], Ryan M. Layer [1*]

[1]Department of Computer Science, University of Colorado Boulder, Boulder, CO, United States of America

[2]Department of Statistics, Colorado State University, Fort Collins, CO, United States of America

[3]Precision Medicine Institute, Children's Hospital Colorado, Aurora, CO, United States of America

[4]Department of Biostatistics & Informatics, Colorado School of Public Health, Aurora, CO, United States of America

[*]Correspondence: ryan.layer@colorado.edu (RL)

Supplemental Network Clustering

Considering the generalizability of networks, many fields have devoted efforts to modeling network structure including statistics, physics, computer science, biology, and social sciences. Within biology, networks have been used successfully for well over a decade now. The applications have been far and wide including disease variant prioritization [1, 2], drug-target identification [3, 4], disease gene detection [5]. One sub-field of network science commonly used within a biological context is network clustering.

Network clustering is the general process of assigning nodes to collections whose members are densely connected within the collection yet sparsely connected to the rest of the network. These relatively dense subsets of nodes are called clusters (or communities) [6]. The utility of network clustering was arguably established when cluster members were shown to embody structural and functional similarities [7]. As a result, a flurry of algorithms designed to partition a graph to create this disparity in connectedness within and between partitions was proposed [8, 9].

The rapid growth of literature prompted several comparative studies in the late 2000s [10, 11] and generalizations in the 2010s [12, 13, 14]. While the literature is expansive, common approaches to network clustering include those

algorithms 1) based on topological distance or random walks, 2) based on the optimization of some objective function such as modularity or surprise, and 3) based on statistical models or principles. While there are many available options for network clustering, the algorithms chosen for this study cover each of the three general approaches and have been shown to outperform competing methods in comparative studies.

We focus on methods designed to uncover disjoint and overlapping clusters (e.g., see Fig 1A and B). The disjoint clustering methods considered include a greedy modularity maximization method [15], walktrap [16], and infomap [**?** 17]. While each method seeks to partition the node set, the objective differs across each method, leading to fundamentally different clusters. CESNA (communities from edge structure and node attributes) [14] is an overlapping clustering method that leverages both the topology of the network and node metadata to form clusters. Unlike the greedy, walktrap, and infomap methods, CESNA distinguishes between genes and phenotypes, treating the biological network as a heterogeneous graph. A summary of each method is included in the following subsections.

Greedy Proposed in [15], this clustering method seeks to optimize a modularity score. In particular, a partition creating the largest disparity between the fraction of edges internal to the partition and the expected fraction from a random graph with the same degree sequence is desired. Since maximizing this quantity is NP-complete in the strong sense [18], a greedy heuristic is used to find reasonable clusters. To start, each node is considered a community. At each iteration of the algorithm, two communities that contribute maximum positive value to the global modularity score are merged. This process continues until no such increase in modularity is possible. The estimated complexity of this method on sparse networks is $\mathcal{O}(nlog^2n)$ where n denotes the number of nodes in the network. Greedy is implemented in the Python networkx package [19].

Walktrap Walktrap was introduced in [16] as a means for clustering a graph using random walks, positing that short random walks tend to remain within a community. To start, each node is considered a community. Distances between communities are computed via random walks, and communities are merged such that there are shorter walks within a community and larger walks between communities. This process is repeated $n-1$ times implying a complexity of $\mathcal{O}(mn^2)$ where m denotes the number of edges in the graph, or $\mathcal{O}(n^2logn)$ for sparse graphs [20]. Walktrap is implemented for Python in the CDlib library [21].

Infomap Rooted in information theory, infomap discovers communities by minimizing the description length of an information flow on a graph, a variant of a coding problem [22]. Information flows are measured across a network using random walks where groups of nodes for which information flows easily "can be aggregated and described as a single

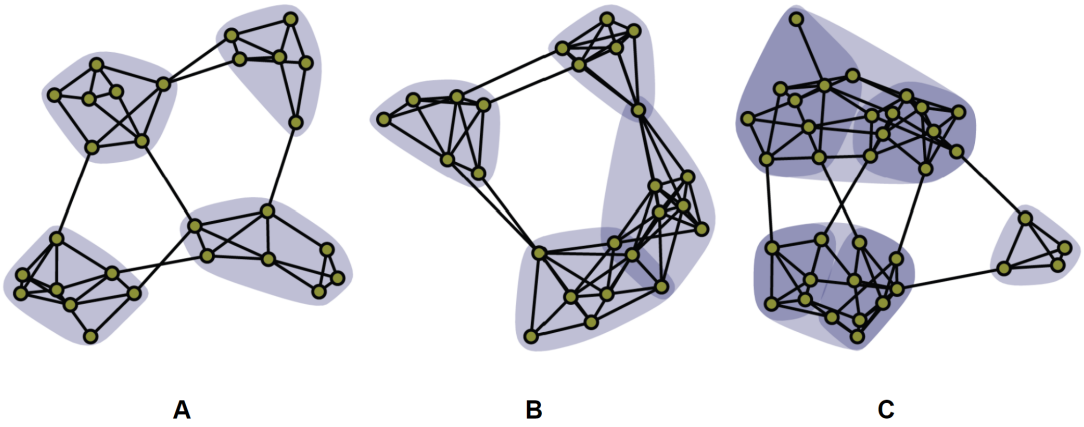

Fig. 1. A contrived example illustrating various types of community structure including a) disjoint, b) overlapping, and c) hierarchical communities. Adapted from [26].

well-connected module" [**?**]. The authors demonstrate these modules are synonymous with communities. The map equation, introduced in [17], provides the theoretical basis for the infomap algorithm, describing how well information about the original network is transferred through a given network partition. [23] estimates that the complexity of infomap is $\mathcal{O}(m)$. This method is available through the infomap package in Python [24].

CESNA Distinct from its competing methods, CESNA (communities from edge structure and node attributes) discovers communities using the structure of the edges within a graph and the available node attributes. In particular, CESNA treats the biological network as a heterogeneous network, distinguishing genes from phenotypes rather than treating them fundamentally the same. The method posits that a graph arises from its nodes' attributes and its nodes' community structure before inferring the latent community structure via maximum likelihood estimation under a statistical model [14]. Discovered communities should be topologically dense and maintain similar node attributes. One iteration of CESNA has an estimated complexity of $\mathcal{O}(m+nk)$ where k is the number of node attributes. For this study, the node type (i.e. gene or phenotype) is the only node attribute considered implying a computational complexity of $\mathcal{O}(m+n)$. This method is implemented using the Stanford Network Analysis Platform (SNAP) [25].

Supplemental Network Subclustering

Clusters in real networks tend to exhibit hierarchical structure such that larger clusters are assumed composed of smaller clusters that can be further divided (e.g., see 1C). For this study, we leverage hierarchical clustering methods to subdivide clusters attained from the network clustering methods discussed in Section 1 into "subclusters" of manageable size using Paris [27]. This method produces a dendrogram which can be cut to produce subclusters. The dendrogram is cut such that the largest resulting subcluster maintains no more than 100 members. This subclustering method is described in the following subsections and implemented in the Python scikit-network package [28].

Paris Paris is an agglomerative hierarchical clustering method based on node pair sampling. In particular, the method proposes a distance measure that compares the joint probability of sampling a pair of nodes, say i and j, to the product of the marginal probabilities of sampling i and j. In particular, two nodes are relatively close if the probability of sampling j given that i has been sampled is large compared to the probability of sampling j. To start, each node is assigned to its cluster. Recursively, the two closest clusters are combined resulting in a dendrogram. The algorithm is a modification of the Louvain algorithm where the iterative step is replaced by a single merge. The complexity of paris is $\mathcal{O}(m)$.

Each clustering and subclutering method is summarized in terms of its class, complexity, and objective in Table 1, along with relevant references.

Table 1. Description of clustering and subclustering methods considered in this study. The complexity and method objectives are provided where n denotes the number of nodes, m denotes the number of edges, and k denotes the number of node attributes. References are provided for each method.

Class	Method	Complexity	Objective	References
Clustering	Greedy	$\mathcal{O}(n \log^2 n)$	Modularity	[15]
	Walktrap	$\mathcal{O}(n^2 \log n)$	Distance based on random walks	[16, 20]
	Infomap	$\mathcal{O}(m)$	Map equation	[22, 17, 23]
	CESNA	$\mathcal{O}(m + nk)$	Latent space likelihood	[14]
Subclustering	Paris	$\mathcal{O}(m)$	Distance based on node pair sampling	[27, 29]

Nontrivial Clusters

Applying greedy, paris, infomap and cesna to our network of the 2021 data resulted in very few clusters being identified, most of which were very large in size Fig 2. The cesna clusters are an exception to this generalization, all of which are ¡ 200 members in size. Our goal is to recommend a set of genes and HPO terms to clinicians and experimentalists for further investigation, providing such users with a list of several thousand items is not a useful product. After discussion with clinicians, we determined that to be useful clusters would need to have fewer than 100 nodes. We later found articles [30] from a DREAM challenge which required that all clusters be in the range of 3-100 nodes because "[clusters] with over 100 genes are typically less useful to gain specific biological insights" [31]. To meet this new requirement, we re-clustered all previously identified clusters using Paris hierarchical clustering [27]. Paris allows an upper limit to cluster size to be set, this resulted in many more clusters than previously identified all ≤ 100 nodes in size (Fig 2).

Data Sources

Two main data sources have been used in this study, namely STRING [32] and Human Phenotype Ontology (HPO) [33]. Deserving additional mention is the genes-to-phenotype annotation file produced by HPO which aggregates information from Online Mendelian Inheritance in Man (OMIM) [34] and Orphanet [35]. These data sources were gathered for the years 2019-2021. STRING was gathered easily using their portal (https://string-db.org/cgi/access?sessionId=b8wDXZhU8UYD&footer_active_subpage=archive) for accessing older versions of the data, specifically only the Homo sapien (species code 9606) data was gathered and used. Collecting older versions of HPO was less straightforward. Older version of HPO were gathered from git commit histories of the Github repositories: obophenotype/human-phenotype-ontology and drseb/HPO-archive. The genes-to-phenotype annotation files were gathered from those same Github repositories, the Monarch Initiative Jenkins servers (current ones and an older one accessed via the Wayback Machine [36], Monarch's Jenkins servers appear to be no longer publicly available). Specific URLs used for each of these data sources can be found in the reproducible pipeline Snakemake file [37]. The publication dates (or date ranges in the case of STRING) can be seen in Table 2. Due to the unavailability of g2p annotations prior to 2019, we opt to use only data 2019 - 2022.

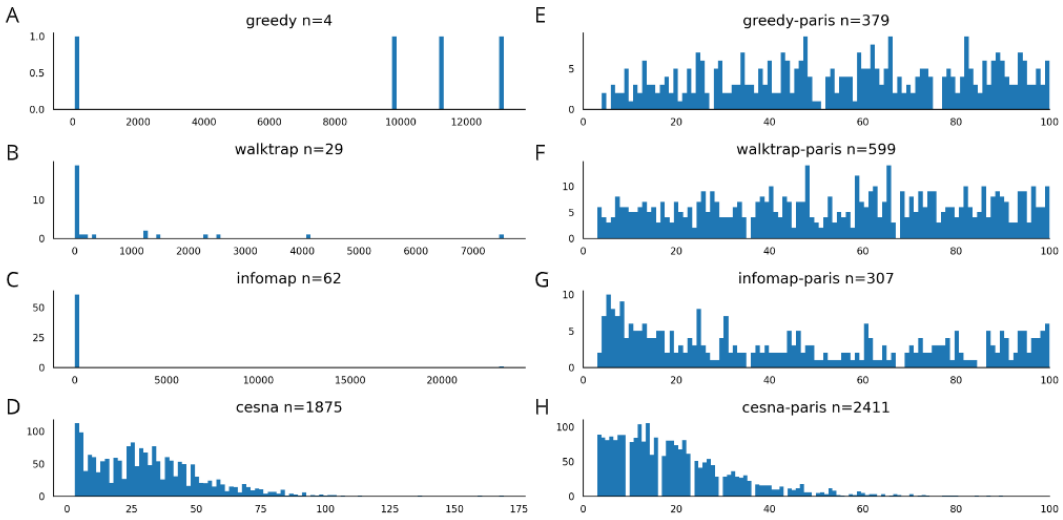

Fig. 2. Shown are distributions of cluster size using A. greedy B. walktrap C. infomap and D. censa. The second column contains the subcluster size distributions, where every cluster from panels A-D were clustered again with the paris-hierarchical method E. greedy-paris F. walktrap-paris G. infomap-paris and H. censa-paris. In all of these plots the x-axis is cluster-size and the y-axis is the number of clusters. The first three of the clustering algorithms on their own, have a tendency to produce very few clusters that are all very large, some with a membership larger than 20,000 nodes, in the case of infomap. Our end goal is to use these clusters to provide sets of genes and phenotypes that are likely to have yet-to-be-discovered clinically meaningful relationships. These clusters in A-D are far too large to be useful for hypothesis generation in clinical or experimental settings. A second layer of cluster with the paris method is applied and shown in F-H, setting an upper limit of 100 on cluster size results in many more clusters of a size manageable for human curation.

Null Model Comparison

When choosing the snowball sampling null model we compared it to two other null models, random clusters and edge shuffle, and found snowball sampling to be a far more rigorous model. We compared the of empirical p-values in the 2019 greedy-paris cluster p-values calculated using the three null models, see and their resulting distributions in Fig S3. With the random clusters null model 39% of clusters achieved significance ($p < 0.05$), and the edge shuffle null model resulted in 48%. Unlike these two other null models where a substantial portion of clusters were significant,

Table 2. The three data sources gathered and used in this study and the accompanying dates each version was originally published (or published and active in the case of STRING). All dates are in year-month-day format. Genes-to-phenotype 2019 is taken from the way-back machine. It cannot be automatically downloaded so I copy and pasted the information and saved it in a file in the git repository. Previous versions of STRING and well preserved and documented. The availability of previous versions of HPO and the G2P annotations do not have a central or documented location for previous versions, we use what is available and closest to the end of each calendar year. HPO and g2p date are time stamps associated with each files in the GitHub history. Of note is that genes-to-phenotypes 2019 is taken from the way-back machine. Automatically downloading it proved difficult so the information was copied, pasted, and saved in a file in the git repository. Additionally, genes-to-phenotypes 2018 could not be found in any GitHub repository histories or the way back machine, for reason we do not use data further back than the end of 2019.

Year	HPO	Genes-to-phenotype	STRING
2019	2019-11-08	2019-9-2	2019-1-19 to 2020-10-17
2020	2020-12-07	2020-8-25	2020-10-17 to 2021-8-12
2021	2021-10-10	2021-10-10	2021-8-12 to present
2022	2022-12-06	2022-12-06	2021-8-12 to present

we found snowball sampling to be very conservative in comparison, only 6% were significant and 70% of the clusters had $p = 1.00$.

The random cluster null model works by:

- Start with a real cluster, of size X.
- Choose X node uniformly at random from the network without replacement to create a synthetic cluster.
- Perform rediscovery on the synthetic cluster.
- repeat steps 2 and 3 1,000 times
- Calculate the empirical p-value of the number of rediscoveries in the real cluster versus the number of rediscoveries in it's 1,000 synthetic clusters.

Edge shuffle null model tests the effect of randomized edge for rediscovery rather than randomized clusters. This model works by:

figure for plos submission

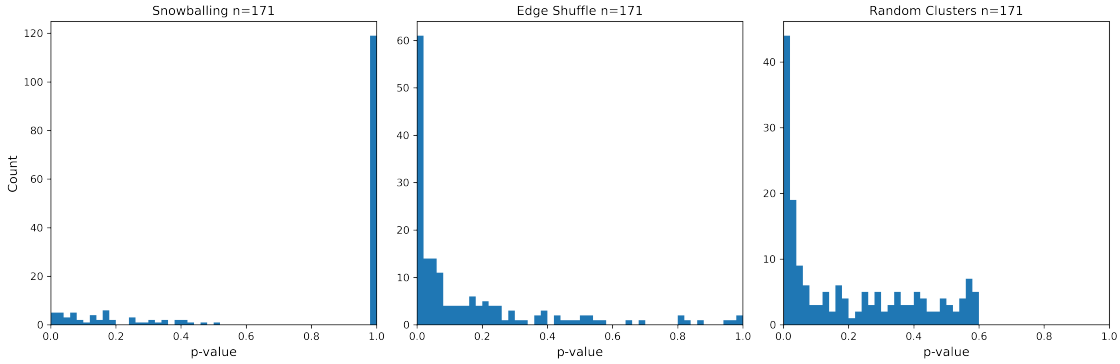

Fig. 3. Distribution of empirical p-values of the 2019 greedy-paris clusters using three different null models: snowballing sampling, edge shuffle, and random clusters. All figures share the same y-axis which is also on a log scale. The proportion of clusters with $p < 0.05$ in each model is 6%, 48%, and 39%, for snowballing, edge-shuffle, and random clusters respectively. Snowballing has 70% of its clusters with $p = 1.00$, whereas the other two models have zero clusters falling into this category.

- For each real 2019 cluster
- Take the list of new edges added in 2020 and shuffle which nodes constitute a new synthetic edge - synthetic edges are comprised of the same nodes as the real new edges but in randomized pairings.
- Performed rediscovery on the real cluster with the list of synthetic edges.
- repeat steps 2 and 3 1,000 times
- Calculate the empirical p-value of the number of rediscoveries for each cluster with the real list of new edges versus the number of rediscoveries with 1,000 synthetic new edge list.

References

[1] van der Velde, K. J. *et al.* A pipeline-friendly software tool for genome diagnostics to prioritize genes by matching patient symptoms to literature (2020).

[2] Singleton, M. V. *et al.* Phevor combines multiple biomedical ontologies for accurate identification of disease-causing alleles in single individuals and small nuclear families. *Am. J. Hum. Genet.* **94**, 599–610 (2014).

[3] Sun, Y. *et al.* Identification of hub genes and potential molecular mechanisms in patients with HBV-Associated acute liver failure. *Evol. Bioinform. Online* **16**, 117693432094390 (2020).

[4] Liang, W., Sun, F., Zhao, Y., Shan, L. & Lou, H. Identification of susceptibility modules and genes for cardiovascular disease in diabetic patients using WGCNA analysis. *Journal of Diabetes Research* **2020** (2020).

[5] Ghiassian, S. D., Menche, J. & Barabási, A.-L. A DIseAse MOdule detection (DIAMOnD) algorithm derived from a systematic analysis of connectivity patterns of disease proteins in the human interactome. *PLoS Computational Biology* **11**, e1004120 (2015).

[6] Girvan, M. & Newman, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* **99**, 7821–7826 (2002).

[7] Newman, M. E. & Girvan, M. Finding and evaluating community structure in networks. *Physical Review E* **69**, 026113 (2004).

[8] Wu, F. & Huberman, B. A. Finding communities in linear time: A physics approach. *The European Physical Journal B* **38**, 331–338 (2004).

[9] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. & Parisi, D. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. U. S. A.* **101**, 2658–2663 (2004).

[10] Lancichinetti, A. & Fortunato, S. Community detection algorithms: A comparative analysis. *Physical Review E* **80**, 056117 (2009).

[11] Fortunato, S. Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010).

[12] Traag, V. A. & Bruggeman, J. Community detection in networks with positive and negative links. *Physical Review E* **80**, 036115 (2009).

[13] Psorakis, I., Roberts, S., Ebden, M. & Sheldon, B. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E* **83**, 066114 (2011).

[14] Yang, J., McAuley, J. & Leskovec, J. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, 1151–1156 (IEEE, 2013).

[15] Clauset, A., Newman, M. E. & Moore, C. Finding community structure in very large networks. *Physical Review E* **70**, 066111 (2004).

[16] Pons, P. & Latapy, M. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, 284–293 (Springer, 2005).

[17] Rosvall, M., Axelsson, D. & Bergstrom, C. T. The map equation. *The European Physical Journal Special Topics* **178**, 13–23 (2009).

[18] Brandes, U. *et al.* Maximizing modularity is hard. *arXiv preprint physics/0608255* (2006).

[19] Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. Rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

[20] Xie, J. & Szymanski, B. K. Community detection using a neighborhood strength driven label propagation algorithm. In *2011 IEEE Network Science Workshop*, 188–195 (IEEE, 2011).

[21] Rossetti, G., Milli, L. & Cazabet, R. CDLIB: a python library to extract, compare and evaluate communities from complex networks. *Applied Network Science* **4**, 1–26 (2019).

[22] Rosvall, M. & Bergstrom, C. T. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences* **104**, 7327–7331 (2007).

[23] Mukherjee, A., Choudhury, M., Peruani, F., Ganguly, N. & Mitra, B. *Dynamics on and of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems*, 209 (Springer New York, New York, NY, 2013), 2013 edn.

[24] Edler, D., Eriksson, A. & Rosvall, M. The mapequation software package. Tech. Rep., The MapEquation software package (2022). URL https://mapequation.org.

[25] Leskovec, J. & Sosič, R. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)* **8**, 1–20 (2016).

[26] Karataş, A. & Şahin, S. Application areas of community detection: A review. In *2018 International congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, 65–70 (IEEE, 2018).

[27] Bonald, T., Charpentier, B., Galland, A. & Hollocou, A. Hierarchical graph clustering using node pair sampling. *arXiv preprint arXiv:1806.01664* (2018).

[28] Bonald, T., de Lara, N., Lutz, Q. & Charpentier, B. Scikit-network: Graph analysis in python. *Journal of Machine Learning Research* **21**, 1–6 (2020). URL http://jmlr.org/papers/v21/20-412.html.

[29] Charpentier, B. Multi-scale clustering in graphs using modularity (2019).

[30] Tripathi, B., Parthasarathy, S., Sinha, H., Raman, K. & Ravindran, B. Adapting community detection algorithms for disease module identification in heterogeneous biological networks. *Front. Genet.* **10**, 164 (2019).

[31] Choobdar, S. *et al.* Assessment of network module identification across complex diseases. *Nature Methods* **16**, 843–852 (2019).

[32] Szklarczyk, D. *et al.* STRING v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.* **47**, D607–D613 (2019).

[33] Köhler, S. *et al.* The human phenotype ontology in 2021. *Nucleic Acids Res.* **49**, D1207–D1217 (2021).

[34] Amberger, J. S., Bocchini, C. A., Scott, A. F. & Hamosh, A. OMIM.org: leveraging knowledge across phenotype–gene relationships. *Nucleic Acids Res.* **47**, D1038–D1043 (2018).

[35] Orphanet. Orphanet. https://www.orpha.net/consor/cgi-bin/index.php?lng=EN (2022). Accessed: 2022-1-25.

[36] Machine, I. A. W. Internet archive: Wayback machine. https://archive.org/web/ (2022). Accessed: 2022-1-26.

[37] Mölder, F. *et al.* Sustainable data analysis with snakemake. *F1000Res.* **10**, 33 (2021).