Community Detection of Superimposed Relational Structures

Abstract

Algorithms for "community" or cohesive subgraph detection are frequently used to analyze network representations of complex systems, and constitute an active area of methodological research. Algorithms for overlapping community detection are of particular interest, because it is often unrealistic to form strict partitions of natural systems, with at least some nodes participating in multiple clusters. However, the literature on overlapping communities does not distinguish different sources of potential overlap. This paper begins by discussing randomness, bridging nodes, and superimposed relational structures as three distinct sources of overlapping communities. It then focuses on the third source (neglected in the literature), by constructing test graphs that merge two distinct relational structures and comparing the communities identified by eight partition and overlapping community algorithms. Results show that most algorithms focus on the same source structure in each graph, ignoring the other, although a few algorithms find combined or nested communities from both structures.

1. Introduction

When modeling natural systems as networks of nodes and links (represented as graphs of vertices and edges), "community detection algorithms" comprise a significant component of our analytic toolkit. It would perhaps be better to call these "cohesive subgraph detection algorithms" to avoid sociological connotations of "community" and to reflect what these algorithms do: find regions of a graph that cohere under some definition (to be discussed) based purely on structural characteristics (the connectivity the edges provide to the vertices) without consideration of exogenous attributes or properties of the domain being modeled. However, "community detection" is the term common in the literature, which we will therefore use. Community detection algorithms are significant analytic tools because often the structural coherence that they detect reflects, and therefore helps us discover, the organization of the natural systems being modeled and associated processes [7, 24].

Even when restricted to graph-theoretic terms (rather than, for example, sociological concepts), there are many possible definitions of cohesive subgraphs or "communities". A common intuition is that a community is a connected subgraph that has more internal than external connections. Variations are possible in how this intuition is instantiated, including "weak" and "strong" communities [9]. Some algorithms try to maximize the actual linkage found within each proposed partition to those expected at random based on the degrees of the nodes involved [17]. Other definitions are based on random traversals of the graph, for example letting such a "walker" walk and seeing where it gets stuck [20], or on information theoretic methods of minimizing the length of such a walk [23]. Indeed, research proposing alternative community detection algorithms has exploded over the past two decades, resulting in hundreds of alternatives [7] to the point that one gets the impression (especially in the early literature) that the definition of a community is "whatever my algorithm produces". The field both suffers and benefits from a lack of shared definition of what constitutes a community: suffering in that it makes comparing alternatives more difficult, but benefiting in that distinct definitions (perhaps implicit in algorithms) may capture uniquely useful perspectives on natural systems. We will consider this potential benefit in this paper.

Most work on community detection has posed the problem as one of partitioning the vertices of the graph to maximize some measure. A partition forces each node into one and only one community, but it is easy to come up with examples of natural systems in which singular membership is unrealistic. For example, human individuals participate in distinct but possibly overlapping work, home, and recreational communities; polysemic words may be related to multiple other groups of words (e.g., "bright" as light intensity or mental intelligence); and a protein may be associated with distinct groups of other proteins via the metabolic processes in which they co-participate [18]. Accordingly, a subset of the fast-expanding literature on community detection has included "overlapping community detection" algorithms, which we will abbreviate as "OCD" (with apologies to clinical psychologists). These algorithms seek principled ways to allow vertices to be classified into more than one community.

However, the OCD literature lacks an analysis of the origins of overlapping communities, and hence the forms they may take. As a first step, this paper proposes

three origins of overlapping communities: randomness, bridging, and superimposed relational structures. Then, focusing on superimposed relational structures, we define a method for generating benchmark test cases and conduct an empirical study to compare major algorithms (both partitioning and overlapping) on their ability to discern one or both of the superimposed structures.

2. Origins

2.1. Randomness

The first origin of overlapping network communities has been well studied in the literature: random "noise". In this paradigm, we assume that the true underlying community structure is (or approximates) a partition, but there is some randomness in the connectivity that causes links to cross partitions. This randomness is tunable via a mixing parameter, such as μ in the Girvan- Newman [10] and LFR [14] benchmarks. It would be beyond our scope to fully discuss what is meant by "randomness" as a cause of overlapping communities. Perhaps one believes that the universe is deterministic, but some processes that we are unaware of produce connections that we cannot explain, so they appear to be random; or perhaps one views the universe as having an intrinsic nondeterminism that cannot be modeled even with full information. Regardless of one's metaphysics, this view of overlapping network communities assumes that there is some most-accurate partition, and the job of the OCD algorithm is to find this partition in spite of the random "noise" in the data. This concept is not intrinsically about overlapping communities, and is well studied in the literature with studies comparing algorithms based on benchmarks varying the mixing parameter μ [e.g., 13], so we won't focus on it here.

2.2. Bridging

The second origin we consider is the situation where (like in the random case) the underlying network communities are basically a partition of nodes, but there are some exceptional nodes that participate in more than one network community [26]. For example, we have seen this in the community structure of a large online network of education professionals, Tapped In, in which chat rooms and discussion forums are generally partitioned according to membership in online classes, professional development efforts, etc.; but Tapped In volunteer facilitators make an appearance in multiple chat rooms and discussion forums as they assist the local facilitators [25]. These volunteers participate in multiple groups to be helpful to each individually, but may also play a role in coordination, spread of information, etc. between groups.

Under this view of the origin of overlapping network communities, there is again some true or most accurate partition, but there are exceptions at the individual node level, and the job of the OCD algorithm is to both find the partitions and provide a means of identifying these bridging nodes, even if they are forced by the core algorithm to be classified in one partition. Bridging may be a matter of degree. For example, the R igraph package provides a means to identify 'crossing' edges: those that cross between two partitions. From this one can derive the number of crossing edges a given vertex participates in, a basic measure of bridging that can be normalized for comparative purposes. Also in R, the linkcomm package provides node-level 'community centrality' metrics that can weight nodes according to how similar (overlapping) or dissimilar (modular) the communities they belong to are. While worthy of further study, we do not focus on bridging in this paper, instead addressing what may be the biggest challenge.

2.3. Superimposed Relational Structures

The third origin of overlapping network communities we consider is fundamentally different: rather than assuming there is some underlying partition with either random edge or node-level exceptions, there may be multiple simultaneous community structures present, corresponding to multiple domain processes or relations. Consider for example social media. When members of a certain generation first began to use Facebook, there was some uncertainty about who to "friend". One might have "friended" family members and friends in the colloquial sense, but then gone to a conference where colleagues also asked to connect. One ends up with a mish-mash of multiple types of relations. This kind of "context collapse" is common in social media [15]. A motivating example for the empirical study that follows is to consider an idealized primary public school, where students are involuntarily placed in equal sized groups (classes) for most of the day, but may also have out of school relations that reflect typical social network structures in which both node degree and community size are heterogeneous (if not strictly "scale free").

One might argue that this kind of overlapping community structure is best understood as poor data collection and modeling rather than a community detection challenge: one should not construct a network model that is based on multiple relations, but rather should separate them out either by excluding all but one relational structure or by representing them separately in multilayer networks [6]. However, this critique begs the question: we may not know what the multiple structures

are when we first approach a natural phenomenon with network characteristics; and we need OCD precisely to identify the structures so we can know them and subsequently choose to separate them.

Under this view of the origin of overlapping network communities, there are truly different network structures imposed over the same nodes, and the job of the OCD algorithm is to identify these structures. A matter requiring further analysis is that it may be mathematically impossible to separate superimposed structures, so that the best we might expect is for an OCD algorithm to identify one structure that is present, and for another OCD algorithm to identify the other structure. This suggests that the proliferation of OCD algorithms may be an advantage rather than a disadvantage.

The remainder of this paper takes an empirical look at this sense of overlapping network communities. We define a benchmark that superimposes two distinct relational structures, and apply selected partitioning and OCD algorithms to the resulting networks to see whether they can at least discriminate one of these base structures, if not both.

3. Community Detection Algorithms

Community detection algorithms considered in this research are summarized below. This is necessarily an incomplete review: more comprehensive reviews are available [7, 9, 16, 26]. The following algorithms were selected to be representative of distinct approaches, with the restriction that usable implementations must be available.

3.1. Disjoint Community Detection Algorithms

The algorithms in this section partition nodes into disjoint sets. We tested Fast-Greedy, Louvain, InfoMap, Label Propagation, Spin Glass, and Walktrap.

Fast-Greedy [5] and **Louvain** [3] algorithms are based on modularity [16, 17]. Finding partitions that maximize modularity is NP-Hard [4], so most modularity-based algorithms use heuristics to approximate the optimum partition. The Fast-Greedy method runs in time O(M h log N), where N=|V|, M=|E|, and h is the height of the community dendrogram. It works best on hierarchical networks with balanced dendrograms [5]. The Louvain method was designed to be tractable on large networks, taking a hierarchical strategy of iterative coarsening that runs in O(N log N) [3]. Modularity optimization methods suffer from a resolution limit: they fail to detect clusters smaller than some scale that depends on the size of the network [8].

InfoMap is a flow-based community detection algorithm that runs in O(N log N) time. Unlike modularity based methods implemented in this experiment, Infomap also uses edge directions. InfoMap is implemented with the same underlying hierarchical algorithm as Louvain [2], but tries to minimize the "map equation" instead of maximizing modularity [23]. The map equation indicates the average number of bits needed to describe the walk of a random walker, using a "map" of community partitions to shorten the description: a better-quality partition yields shorter descriptions.

Label Propagation is a fast, nearly linear time algorithm that works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex [21]. In this research, the extended implementation of Label Propagation in igraph-R is utilized. This extension implements edge weights.

Spin Glass tries to find communities by employing methods of statistical mechanics. The community structure of the network is interpreted as the spin configuration that minimizes the energy of an infinite range spin glass, with the spin states being the community indices [22]. This method applies to weighted and directed networks.

Walktrap is based on the intuition that random walks on a graph tend to get trapped into densely connected parts corresponding to communities. This algorithm's runtime is $O(NM^2)$ in the worst case and O(NMh) in sparse networks where h is the height of the dendrogram representing hierarchical community structure [20].

Asynchronous Fluid [19] and Girvan-Newman [17] were also tested but were excluded from this paper due to poor performance (the latter failed to complete after several hours on Test100).

3.2. Overlapping Community Detection Algorithms

We studied two well-known algorithms that allow nodes to be assigned to multiple clusters.

Clique Percolation, also called CFinder [18] is based on the intuition that communities are comprised of overlapping cliques (e.g., triads, 4-cliques, etc.). At level k, a k-clique community is defined to be the largest connected subgraph of k-cliques, each of which overlaps with some other member k-clique on k-1 nodes. This definition is in general too rigid, and specifically fails completely on bipartite graphs (although modifications have addressed this problem). The problem of finding maximum cliques in a graph is NP-Hard. However, it is possible to find cliques of fixed sizes in polynomial time [12]. K-clique "communities" emerge naturally in sufficiently dense random networks [2], so should be interpreted with caution.

Link Communities [1], implemented as linkcomm in R [11], provides a more flexible approach based on an appealing intuition: the communities that nodes (such as ourselves) participate in are based on our *relationships* to others (e.g., our family, work and recreational relationships); therefore, it is the relationships (links) that should be partitioned rather than the nodes, with multiple community membership induced on the nodes based on the partitioning of the links on which the nodes are endpoints. Hierarchical clustering of links is based on link similarity, the relative number of common neighbors each node pair has. The algorithm's total computational complexity is $O(N^{(2/(\gamma-1))} + M^2)$, where γ is the power law degree exponent [Barabasi, 2016]. For sparse graphs ($M = O(N)$), the latter term dominates, leading to $O(N^2)$. The Linkcomm method supports directed and weighted graphs.

4. Methods

To study overlapping communities due to superimposed relational structures, we constructed test graphs by combining two benchmark graphs on the same set of nodes. Nodes in each source benchmark graph were assigned "ground-truth" community membership for that graph. Hence, each node in the combined graph inherits two community membership labels. In general, the two source graphs are distinct not only in the specific assignments of nodes to communities but also in the pattern of overall connectivity. In general, the source graphs differ in the number and therefore sizes of ground-truth communities. Also, they may differ in the heterogeneity of community sizes, from uniform to variable. This variation in source networks was designed to present the algorithms with different challenges.

4.1. Technical Resources

We used the Lancichinetti, Fortunato & Radicchi (LFR) benchmark [14] to create graphs with ground-truth communities. LFR was designed to generate benchmark networks that resemble the heterogeneous (scale-free) distributions of node degree and community sizes of real-world networks. It is highly parameterized, including control over average degree, maximum degree, minimum and maximum community size, a parameter μ to control randomized mixing of communities, and power-law exponents for degree and community size distribution. It can generate directed or undirected, weighted or unweighted graphs. We generated directed weighted graphs, but converted these to undirected graphs as required by algorithm limitations.

We used igraph in R (https://igraph.org/r/) to manipulate and merge graphs, and for various community detection algorithms it provides: Fast-Greedy, Louvain, InfoMap, Label Propagation, Spin Glass, Walktrap and Girvan-Newman. We also used the R CliquePercolation and linkcomm packages for overlapping community detection. NetworkX (https://networkx.org) provided alternate implementations and several additional community detection algorithms, including Asynchronous Fluid. We used igraph in Python (https://igraph.org/python/) to connect the results obtained in igraph-R and NetworkX and for its difflib library (discussed below).

4.2. Test Graphs

Four test graphs of four different sizes were constructed, each by combining two distinct relational structures over the same set of nodes with designated ground-truth communities, as discussed above. In summary, the test cases $G = (V, E_1 \cup E_2)$ are constructed from source graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ as follows (see also Figure 1):

Test75: $|V| = 75$. G_1 has 142 edges and 3 ground-truth communities of equal size with no mixing between them. G_2 has 144 edges and 8 ground-truth communities, with community sizes following a power-law distribution and moderate mixing between communities.

Test100: $|V| = 100$. Similar to Test75, G_1 has 4 equal sized disjoint communities with 227 edges. G_2 has 8 variable sized disjoint communities with 279 edges.

Test400: $|V| = 400$. G_1 has 4955 edges with 48 communities of near uniform size and some mixing. G_2 has 726 edges and heterogeneous structure amongst 15 communities. This graph was motivated by the first author's familiarity with a small village, combining school-related organized activity affiliations and family relationships.

Test 1000: $|V| = 1000$. G_1 has 2662 edges and 39 disjoint communities of moderately varying size. G_2 has 2651 edges and 3 disjoint communities of equal size.

4.3. Procedure

For each test graph, we ran Fast-Greedy, Louvain, InfoMap, Label Propagation, Spinglass, Walktrap, Girvan-Newman (not reported), Asynchronous Fluid (not reported), Clique Percolation (k=3 and k=4) and Link Communities, and then compared their results to the two ground-truth communities embedded in each graph. We used the Python difflib.SequenceMatcher (https://docs.python.org/3/library/difflib.html), an $O(n^2)$ worst case sequential matching algorithm that

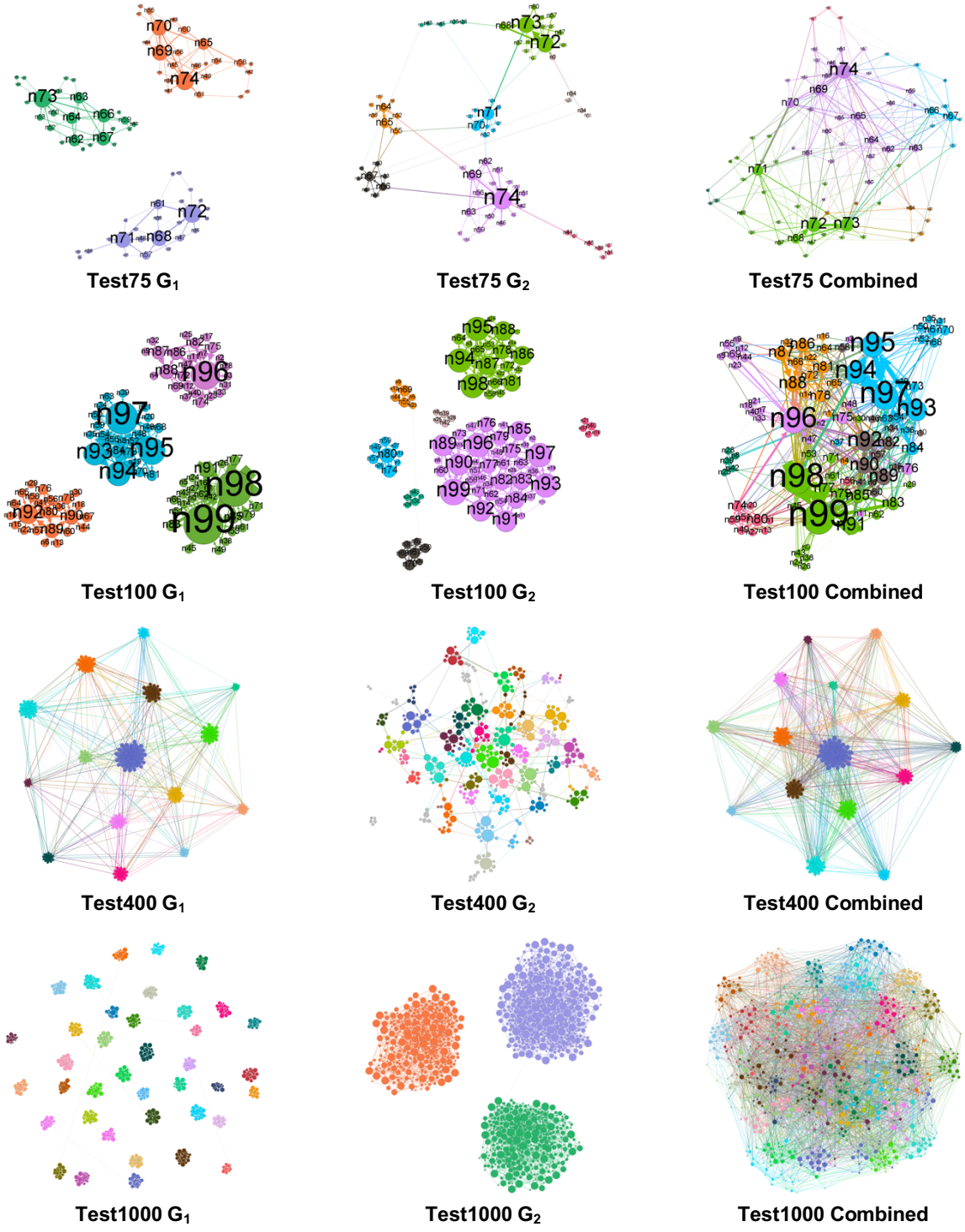


Figure 1. Test Graphs constructed from merge of LFR G_1 and G_2 for |V| = 75, 100, 400 and 1000, respectively. Colors show ground truth communities.

determines the similarity of classification sequences by calculating twice the number of matching pairs of characters M divided by the total number of characters T of both strings, 2*M/T. This metric is 1.0 if the sequences are identical, and 0.0 if they have nothing in common. We then plotted this metric in heat-maps to visualize matches between detected communities and ground-truth values. Since the test graphs have communities from two source graphs, the heat-maps are divided into two parts. The first set of rows represent the first original graph's communities (G_1) and the remaining rows represent the second source graph's communities (G_2).

For example, Figure 2 shows the heat-map of matching ratios between ground-truth communities of the Test100 graph on the y-axis (4 from the first graph in rows 0-3 followed by 8 from the second graph in rows 4-11) and 12 communities detected by one run of the Label Propagation method on the x-axis. (While this is a square matrix, in general the dimensions can differ depending on the number of algorithmically detected communities.) A perfect detection of a community would look like one dark cell indicating the matching community, with the rest of the row being white (or lighter color for near-perfect matches). For example, in Figure 2, the detected community with x axis index 6 has a high matching ratio with the second graph's ground-truth community with y axis index 4. If there are two similarly dark-colored cells on the same row (e.g., row 10 of Fig. 2), this implies that the row's ground-truth community was detected as two different communities by the community detection method. If there are two dark colored cells in a column (e.g., columns 0 and 10), then a detected community combines (parts of) two ground-truth communities (in this example, from two different source graphs). If all cells in the same row have similar coloration (e.g., row 0, the community detection algorithm was not successful in recognizing the ground truth community represented by that row. If this is the case for all rows of a source graph, then the community structure for that source graph was not detected. By visual comparison of the top and bottom half of the heat-map we can quickly see which source structure was detected. For example, Figure 2 shows that Label propagation focused primarily on the community structure from the second source graph.

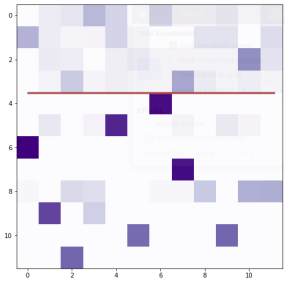


Figure 2. Heat-map

5. Results

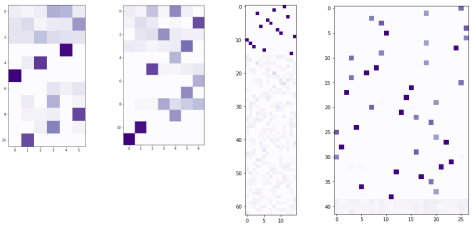
Due to space limitations, only selected heat-maps are provided. To see figure details, zoom in the PDF version of the paper.

5.1. Fast-Greedy

For 3 of the test graphs, Fast-Greedy detected only one community (the full graph). For Test400, the algorithm detected six communities, 5 of which match perfectly with ground-truth communities in the coarser-grained first source graph, and the sixth having a matching ratio of 0.58 with one of the ground-truth communities in the first source graph. Fast-Greedy is a modularity-based algorithm, and modularity suffers from a resolution limit, making densely connected graphs particularly challenging as we see in three of these results.

5.2. Louvain

The Louvain method is also modularity based, but unlike Fast-Greedy, Louvain discovered several communities for each test graph. Six communities are detected in Test75, with 1 perfect match, 4 having high ratios (0.78-0.96) and the lowest matching ratio being 0.59. All of these detected communities belong to the second original graph of Test75 (Figure 3a). Seven communities were found in Test100 (Figure 3b). One detected community is a perfect match, another a 0.95 matching ratio, and others have ratios of 0.78, 0.76 and 0.67. Louvain recovered the complete structure of 15 communities from the first source graph of Test400 with perfect matches; the structure of the second graph is invisible to Louvain (Figure 3c). Louvain found 27 communities in Test1000: 14 of these communities are perfect matches with Test1000's first source graph; and there are also 3 detected communities with matching ratio between 0.96 and 0.98, differing on single nodes. The second source graph's structure is not detected by Louvain (Figure 3d).



(a) (b) (c) (d)

Figure 3. Louvain Heat-maps

5.3. InfoMap

Although InfoMap can analyze directed and weighted graphs, we applied it to the undirected version of the graph for comparison to Louvain. The results were impressively accurate for one of the ground-truth structures in each test graph. For Test400 and Test1000, InfoMap recovered the first source community structure with perfect ratios (Figure 4), but the second is completely ignored. For Test75 and Test100 graphs, community structures from the second source graph are recovered, with low ratios for the first. The discovered communities focus on one of the original graphs.

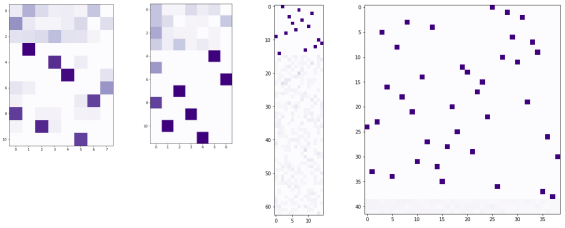


Figure 4. InfoMap Heat-maps

5.4. Label Propagation

The Label Propagation algorithm is non-deterministic, giving varying results. Therefore, we ran the algorithm numerous times and selected the median results. Results show that Label Propagation suffers from the dense community structures in the graphs. For Test75, the best matching community has a 0.93 matching ratio, and the remainder are between 0.5 and 0.84. In this and Test100 the primary matches are to the finer grained second source structure with some mixing of the first source communities. The algorithm generated the clearest results on Test400 by detecting 13 perfect matches out of 15 communities from the first source graph. For Test1000, the algorithm detected 10 perfect matches and 7 matches with ratios above 0.9 out of 39 communities from the first source graph.

5.5. Spin Glass

Spin Glass did not have any perfect matches on Test75: the highest match is 0.93. For Test100 there are two perfectly discovered communities; the rest do not have high matching ratios. The algorithm recovered information partly from both of the underlying community structures of Test100. Figure 5 shows that the highest ratios for each column (detected communities) belong to different structures (rows 0-3 and rows 4-11). While columns 0 and 5 detected by the algorithm match with the communities from the second original graph, others match primarily with the first graph's communities with some mixing of the second. Test400 and Test1000 results are very similar to the Louvain algorithm results: the dominating structure is from the first source graph for both Test400 and Test1000. The results for Test400 matches almost perfectly with all ground-truth communities from the first graph and 9 out of 39 communities are detected perfectly in Test1000.

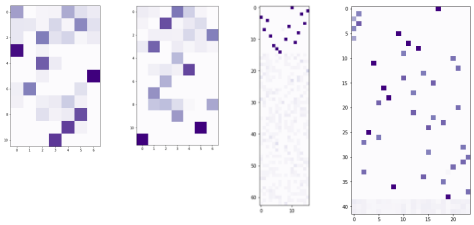


Figure 5. Spin Glass Heat-maps

5.6. Walktrap

Walktrap finds a mixture of structures from each source graph for Test75 (Figure 6), so the ratios for individual matches are not high, reaching a maximum of 0.83 for two matches. For Test100, 6 of the 10 detected communities are a perfect match for the second source structure. There are 15 communities detected in Test400 graph and all of the detected communities match perfectly with the first source structure. Walktrap matches 38 communities in Test1000 graph, all from the first source, which has 39 communities. 37 of the detected communities match with the ground-truth structure perfectly. One detected community (column 0) in Test1000 is a combination of two communities in the first source graph, thereby accounting for all 39 ground-truth communities. Thus, we see mixing of underlying structures in the small graphs but clear detection of one structure in the larger graphs.

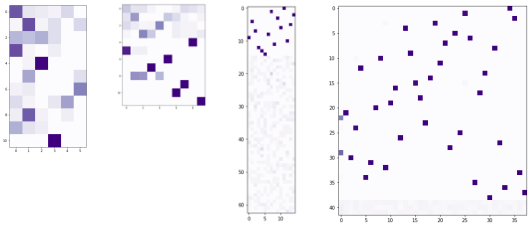


Figure 6. Walktrap Heat-maps

5.7. Clique Percolation

Clique Percolation constructs communities out of overlapping cliques [18]. The Clique Percolation algorithm requires an argument, the clique size.

The first experiment was conducted with clique size 3: communities are based on overlapping triangles. The results did not yield high matching ratios for any of the test graphs (heat-maps are not shown). For Test75, only 5 communities are detected and the highest ratio is 0.75 on a 5-member community. The Test100 results are similar: only 2 communities are detected and the highest ratio is 0.83 on a 5-member community. There are 4 communities detected in Test400; the highest ratio is 0.91, and the rest of the ratios are lower than 0.6. Surprisingly, there are 248 communities detected in Test1000. Only one of the detected graphs has a high ratio with 0.93 on an almost perfect detection; however, the rest of the matching ratios are lower than 0.7. These difficulties may be explained by the fact that in our study, the merging of two source graphs produced dense test graphs containing many overlapping triangles, some of which are purely due to the density of the graph, and hence are random.

When clique size is set to 4, Clique Percolation performance improves moderately. This is because the threshold for appearance of random cliques goes up as k increases [2, p. 348]. Clique Percolation detected 7 communities in Test75 with the highest matching ratio of 0.72 (Figure 7). Similarly, for Test100, the highest ratio of 5 detected communities is 0.87. In contrast, on Test400 Clique Percolation detected 2 communities perfectly and 12 communities with high matching ratios between 0.85 and 0.9. Interestingly, it recovered information from both ground-truth structures with high matching ratios. There are 2 perfectly detected communities, one from the first source graph and the other from the second source graph. The rest of the high ratio matches are also shared between both structures. The matching ratios for Test1000 are very low, the highest being 0.4. The method does not produce any high ratio matches for larger clique sizes (k=5 and up).

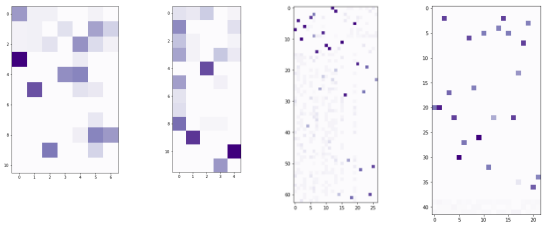
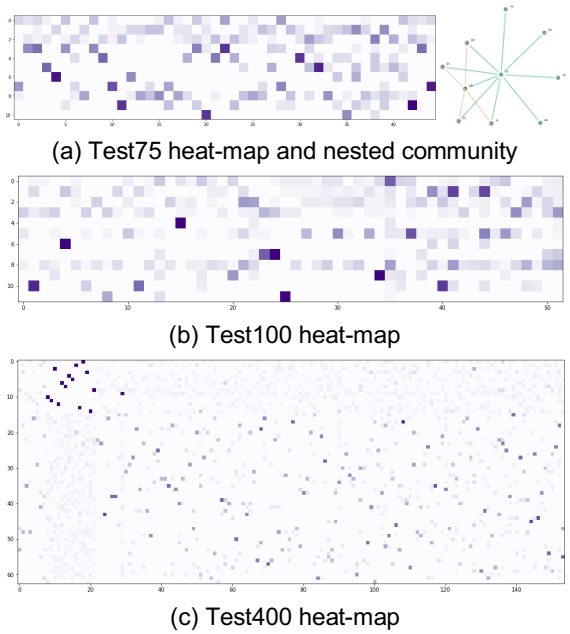


Figure 7. Clique Percolation Heat-maps, k=4

5.8. Link Communities

The linkcomm package induces overlapping communities on nodes by partitioning edges according to their neighborhood overlap. Unlike the other methods studied here, it also detects nested communities. This may partially explain its tendency to produce more communities than the other methods.

Linkcomm detected 45 communities in Test75 (Figure 8a). The matching ratios are not high. There are only 8 detected communities with ratios higher than 0.67. The best matching detected community with a matching ratio of 0.8 contains another community inside it. This is an interesting because this group of nodes belong to a community in the second source graph, but the nested community nodes are a part of a community in the first graph. Thus, linkcomm is able to partially recover structure from both source structures. The results are more accurate on Test100 (Figure 8b). There are 52 detected communities, 5 with perfect matches, and 9 with ratios between 0.67 and 0.87. There are 13 nested communities, again recovering information from both layers of communities.



(a) Test75 heat-map and nested community

(b) Test100 heat-map

(c) Test400 heat-map

Figure 8. Selected Louvain trials

Linkcomm identified 154 communities in Test400, out of which 13 have a perfect match. As the heat-map in Figure 8c suggests, the matches recover information from both original graphs' structures. The first 22 detected communities are from the first source graph, and the remainder from the second, with lower match ratios. Communities detected by this method are generally small in size compared to the ground-truth designations (hence 22 instead of 15 from the first source graph). This resulted in low ratios, but the discovered structure information is useful. It would be possible to obtain larger communities by specifying a different level of cut in the dendrogram. In contrast to other graphs, the algorithm did not find any nested

communities in Test400, although numerous nodes are assigned to multiple communities.

In terms of the assigned ground-truth communities, linkcomm performed poorly on Test1000: the highest matching ratio amongst 637 (!) detected communities is 0.53 (heat-map not shown). It focuses on much smaller clusters of similar links than the large clusters in the source graphs. Therefore, it holds greater promise for fine-grained analysis., unless one adjusts the dendrogram cutoff. There are 11 nested communities, with weak matching ratios.

Linkcomm can detect nested communities, but the test graphs employed in this study were not constructed to have nested communities, instead being constructed by superimposing two community structures. Nevertheless, linkcomm was able to find incidental nesting where it existed (due, for example, when a finer grained community from one source graph consisted of nodes in a coarser grained community in the other source graph). Even though the algorithm was designed to find multiplicity at the node membership level, it was still able to recover information from multiple layered community structures, especially smaller scale structures.

6. Discussion

Disjoint community detection methods are not able to assign a single node to multiple communities, so in principle cannot recover complete structure information from merged community structures. However, it is interesting to see which structures disjoint community detection techniques actually detect. In general, they tended to recover information from only one of the multiple community structures, ignoring the other. Exceptions included Spin Glass and Walktrap, which constructed communities from two different structures on one of the merged graphs. In general, the methods recovered the information about smaller communities better than large ones (detecting structure in the source graph with the finer partition). The exception was the Test400 graph: the visualization in Figure 1 shows that the merged graph is dominated by the courser structure of the first source graph, and the algorithms tended to mirror this partitioning.

In terms of matching ratios, the most accurate detection amongst disjoint community detection methods was provided by InfoMap, followed by Walktrap (although with weaker ratios on Test75 because it is detecting structures from both graphs). Spin Glass, Label Propagation and Louvain methods also recover some useful community information; however, overall matching ratios are not high on smaller graphs. Asynchronous Fluid and Fast-Greedy had low matching ratios, with Fast-Greedy suffering from low resolution (results were not reported).

Turning to the overlapping community detection methods, Clique Percolation searches for complete k-cliques to form communities; however, the LFR benchmark does not construct communities from k-cliques. This resulted in the Clique Percolation performing badly on small isolated network structures. Its best results were obtained with 4-cliques on Test400, which apparently had clearer clique structure in the source graphs. The Linkcomm method detects the highest number of communities for each graph, which provides us with a fine-grained analysis of community structures, but this also caused its communities to have lower match ratios to the ground-truth communities, which are coarser-grained. This method is able to recover nested community structures from the graph, a significant advantage. Taken together, the detected overlapping and nested communities recovered community structure from both source graphs. Therefore, Linkcomm was the most able to recover overlapping community structure from multiple structures in the test graphs of this study.

7. Conclusions

We studied the ability of several community detection algorithms to analyze the community structure of graphs created by combining multiple different community structures based on the same set of nodes. To create each test graph, two graphs over the same vertices were created with the LFR benchmark, with ground-truth community membership defined for each, and then our test graphs were created by taking the union of the source graph edges. The algorithms were applied to analyze the community structure of the merged graphs, and then results were compared to the ground-truth community membership values using a sequential matching method.

When multiple structures are superimposed, the results on partitioning algorithms show that these algorithms may “lock in” on one of the structures, ignoring the other. InfoMap did particularly well at finding one of the structures. However, if we rely on such algorithms, we may not be finding all the structure that is present. One might therefore apply multiple partitioning methods, hoping that different algorithms will expose different structures, but our results show that they tend to focus on the same structures. In the cases where partitioning algorithms were sensitive to multiple structures, the matching ratios were lower. This may translate into a perception that the detected community structure is weak (e.g., as measured by modularity), but this study shows that we also need to be cautious here: “muddled” results (as seen with Spin Glass and

Walktrap) may mean that there are multiple community structures present rather than that there is little community structure.

The results on the overlapping community detection algorithms show that assumptions about the origin or fine-grained structure of communities matter. Benchmark graphs were not constructed from cliques, so clique percolation performed poorly. Link communities are constructed based on the assumption that it is our relationships rather than node attributes that matter, so a method based on link partitioning (inducing multiple membership on nodes) was better able to recover structure from superimposed relations. However, these tended to be smaller communities leading to low matching ratios to the larger ground-truth communities. The proliferate communities produced by the linkcomm algorithm can be overwhelming, but careful study of these may reveal clues about such superimposed relations.

Future work should begin by defining and exploring a systematic space of variations in graph structure, including source structures that are both similar to and different from each other, graphs that are constructed in different ways (e.g., via cliques), and with variation in mixing of edges from the two source graphs. Other community detection algorithms can also be included, and experiments can be conducted on larger graphs. The merged graphs experimented in this project are created to resemble the social network structures that occur in real life: experiments can be conducted on real networks with ground-truth information.

10. References

[1] Y.-Y. Ahn, J. P. Bagrow and S. Lehmann, *Link communities reveal multiscale complexity in networks*, Nature, 466: 761-765 (2010).

[2] A.-L. Barabási, *Network Science*, Cambridge University Press, 2016.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, *Fast unfolding of communities in large networks*, J. Statistical Mechanics, P10008 (2008).

[4] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski and D. Wagner, *On modularity clustering*, IEEE Transactions on Knowledge and Data Engineering, 20 (2008), pp. 172-188.

[5] A. Clauset, M. E. J. Newman and C. Moore, *Finding community structure in very large networks*, Phys. Rev. E 70, 066111 (2004), arXiv:cond-mat/0408187.

[6] M. E. Dickison, M. Magnani and L. Rossi, *Multilayer Social Networks*, Cambridge University Press, New York, NY, 2016.

[7] S. Fortunato, *Community detection in graphs*, Physics Reports, 486 (2010), pp. 75-174.

[8] S. Fortunato and M. Barthelemy, *Resolution limit in community detection*, Proc. National Academy of Sciences, 104: 36-41 (2007).

[9] S. Fortunato and D. Hric, *Community detection in networks: A user guide*, Physics Reports, 659: 1-44 (2016).

[10] M. Girvan and M. Newman, *Community structure in social and biological networks*, Proc. National Academy of Sciences, 99: 7821-7826 (2002).

[11] A. T. Kalinka and P. Tomancak, *linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type*, Bioinformatics, 27: 2011-2012 (2011).

[12] J. M. Kumpula, M. Kivelä, K. Kaski and J. Saramäki, *Sequential algorithm for fast clique percolation*, Physical Review E, 78, 026109 (2008).

[13] A. Lancichinetti and S. Fortunato, *Community detection algorithms: A comparative analysis*, Physical Review E, 80, 056117-1-11 (2009).

[14] A. Lancichinetti, S. Fortunato and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Physical Review E, 78, 046110-1-5 (2008).

[15] A. E. Marwick and D. Boyd, *I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience*, New Media & Society, 13: 114-133 (2011).

[16] M. E. J. Newman, *Networks*, Oxford University Press, 2018.

[17] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Physical Review E, 69, 026113-1-15 (2004).

[18] G. Palla, I. Derényi, I. Farkas and T. Vicsek, *Uncovering the overlapping community structure of complex networks in nature and society*, Nature, 435: 814-818 (2005).

[19] F. Parés, D. Garcia-Gasulla, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés and T. Suzumura, *Fluid communities: A competitive, scalable and diverse community detection algorithm*, arXiv:1703.09, 2017.

[20] P. Pons and M. Latapy, *Computing communities in large networks using random walks*, J. Graph Algorithms and Applications, 10: 191-218 (2006).

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

[22] J. Reichardt and S. Bornholdt, *Statistical mechanics of community detection*, Physical Review E, 74, 016110 (2006).

[23] M. Rosvall and C. T. Bergstrom, *Maps of random walks on complex networks reveal community structure*, PNAS, 105: 1118-1123 (2008).

[24] D. D. Suthers, *Applications of cohesive subgraph detection algorithms to analyzing socio-technical networks*, Proc. HICSS-50 (CD-ROM), IEEE, New Brunswick, 2017.

[25] D. D. Suthers, J. Fusco, P. Schank, K.-H. Chu and M. Schlager, *Discovery of community structures in a heterogeneous professional online network*, Proc HICSS-46 (CD-ROM), IEEE, New Brunswick, 2013.

[26] J. Xie, S. Kelley and B. K. Szymanski, *Overlapping community detection in networks: the state of the art and comparative study*, ACM Computing Surveys, 45 (2013), pp. 1-3