Information Sciences 579 (2021) 574–590

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

APAL: Adjacency Propagation Algorithm for overlapping community detection in biological networks

Osman Doluca [a], Kaya Oğuz [b,*]

[a] Izmir University of Economics, Department of Biomedical Engineering, Turkey
[b] Izmir University of Economics, Department of Computer Engineering, Turkey

ARTICLE INFO

Article history:
Received 21 January 2020
Received in revised form 28 June 2021
Accepted 8 August 2021
Available online 10 August 2021

Keywords:
Overlapping communities
Biological networks
Graph generation

ABSTRACT

We propose a novel method called Adjacency Propagation Algorithm (APAL) which considers the notion that the adjacent vertices are the best candidates for detecting overlapping communities in an undirected, unweighted, nontrivial graph. This is a compact algorithm with a single threshold parameter used to filter the detected communities according to their intraconnectivity property. In this study, APAL was tested rigorously using synthetic generators, such as the widely accepted LFR benchmark, as well as real data sets of yeast and human protein interactions networks. It was compared against the foremost algorithms in the field; the Clique Percolation Method (CPM), Community Overlap Propagation Algorithm (COPRA) and Neighbourhood-Inflated Seed Expansion (NISE). The results show that APAL outperforms its competitors for networks with increases in the number of memberships of the overlapping vertices. Such conditions are often found in biological networks, where a particular protein subunit may form part of several complexes. We believe that this shows the value of the implementation of APAL for protein interaction and other biological networks.

1. Introduction

The relations between the entities of systems form networks that emerge in a wide range of disciplines, such as biology, sociology and computer science. These networks are commonly represented by graphs. Studying their properties is essential to unveil the information contained within these systems.

Although the definition of a community is not universally agreed on since it depends largely on the problem and the system [13], they are usually defined as the subgraphs of a network where the entities are connected densely within the community and sparsely with the outside. A community is *disjoint* if the entities belong to one community only, and is *overlapping* otherwise.

Communities are a common phenomenon in social and biological networks [31]. They emerge in social networks, such as Twitter and Facebook [7,19], in collaboration networks [32] and in biological networks of metabolic pathways [15], of protein–protein interaction networks [40], and of functional brain networks [10]. These have increased in both quantity and size thanks to the advances in computing and the World Wide Web. In accordance with this increase, many algorithms have been proposed for community detection. The reviews of these algorithms span a variety of their properties, such as a very detailed

* Corresponding author.
E-mail addresses: osman.doluca@ieu.edu.tr (O. Doluca), kaya.oguz@ieu.edu.tr (K. Oğuz).

https://doi.org/10.1016/j.ins.2021.08.031
0020-0255/

survey with a focus on techniques designed by statistical physicists [13], a survey with a categorisation based on the definition of the community [11], surveys that focus only on methods for overlapping communities [3,47], a survey for social networks [44] and a recent survey with a multidisciplinary perspective [20].

We propose a novel method, called the Adjacency Propagation Algorithm (APAL), which considers the notion that the adjacent vertices are the best candidates for detecting overlapping communities in social and biological networks. APAL is compact, merely 40 lines long as listed in Algorithm 1, operates with a single intuitive threshold parameter, and is deterministic.

mesure de la densité des connexions à l'intérieur d'une communauté

The strongest point of APAL is the way it makes use of the definition of a community. As will be discussed in Section 3, APAL's definition of a community is based on the *intraconnectivity* property of a graph, that is, how well the vertices within the subgraph are connected to others in the same subgraph. The researchers might have a general idea about the expected intraconnectivity of a community, or form an estimation by sampling subgraphs from the data. APAL detects all subgraphs that have an intraconnectivity value greater than or equal to the threshold value, which is the only parameter the algorithm requires. This allows the users to apply the algorithm intuitively once they have an estimated guess of the intraconnectivity property. The brevity of the algorithm, and its intuitive single threshold parameter makes APAL a good candidate for community detection problems.

The rest of the paper is structured as follows. The next section reviews the existing methods for overlapping community detection. Section 3 gives a detailed account of the proposed algorithm. Section 4 is about the data used and generated for testing and comparing the proposed and existing methods. The results are discussed in Section 5 and the conclusion is given in Section 6.

2. Related work

Graph notation varies slightly in literature. It is common to refer to vertices as nodes, edges as links and the adjacent vertices as neighbours. For consistency, we will be using the following notation. A graph is denoted as $G = (V, E)$ where V is the set of vertices, and E is the set of edges. The adjacent vertices for a vertex $v \in V$ is $N(v)$. The cardinality of the set is expressed with $|V|$.

The major property characterising the existing algorithms is whether they operate on disjoint or overlapping communities. Another characteristic is whether they operate on the vertices or on the edges of the underlying graph. Early approaches worked on disjoint communities, using graph partitioning algorithms, such as [5,22] in which the number of clusters must be known beforehand. Hierarchical clustering was proposed to overcome this restriction by creating a hierarchical clustering tree, or a dendrogram, where vertices are connected by the weight of their connections [14].

The algorithm proposed by Girvan and Newman defined the *edge betweenness* that focuses on the edges communities rather than detecting the edges that are central to communities [16]. The betweenness is defined as the number of shortest paths between the pairs of vertices that run along the edge. The algorithm computes this measure for every edge, removes the one with the highest value, and recomputes the betweenness for the remaining edges until none remain.

The label propagation algorithm detects the disjoint communities in a given graph [38]. Each vertex is initialised with its own unique label. The algorithm then updates the label of each vertex by looking at the labels of its adjacent vertices. It is set to the label with the maximum number of vertices. The ties are picked randomly. This process creates oscillations of labels in bipartite or nearly bipartite subgraphs. This problem is overcome by updating the labels asynchronously and by updating the vertices in random order at every iteration. The algorithm ends when every vertex has a label equal to the maximum number of its adjacent vertices.

Other approaches for disjoint community detection include spectral clustering [34], spectral optimisation [33], genetic algorithms [42] and dynamic algorithms [36].

The definition of overlapping communities bears a striking resemblance to the fuzzy set definition, in which each element has a certain degree of membership to more than one set. There are various approaches that use fuzzy membership to detect communities. Zhang et al. proposed an algorithm [49] based on Newman and Girvan's modularity function for measuring community structure [33] to find the optimal number of communities and use Fuzzy C-Means for the soft-assignment of communities.

As mentioned earlier, some of the approaches work on edges rather than the vertices of the underlying network graph. For any given simple graph, its line graph is formed by a set of vertices which represent the edges in the graph. Evans and Lambiotte define communities as a partition of the edges and identify them by creating a vertex partition of the line graph of the network [12].

Baumes et al. proposed the Iterative Scan (IS) and Rank Removal (RaRe) algorithms that operate in combination [6]. IS expands a seed community by adding or removing vertices until no further improvement in fitness function is possible. The seed communities can be detected by the RaRe algorithm, which identifies a subset of high-ranking vertices.

Gregory has generalised the label propagation algorithm by Raghavan et al. [38] to allow it to detect overlapping communities [18]. The algorithm is called Community Overlap Propagation Algorithm (COPRA) and it retains community identifiers and belonging coefficients, denoted by (c, b), respectively, for each vertex. The belonging coefficient is the strength of the vertex's membership to a community. For all vertices these values are normalised so that they sum up to 1.

At each propagation step t, COPRA sets the labels of the vertices to the union of the adjacent vertices labels and normalise their belonging coefficients via Eq. (1).

$$b_t(c,x) = \frac{\sum_{y \in N(x)} b_{t-1}(c,y)}{|N(x)|} \tag{1}$$

The propagation is applied synchronously, so the label of a vertex at iteration t is based on the labels of its adjacent vertices at iteration $t-1$. At each iteration step, the algorithm removes pairs that have a belonging coefficient below a predefined threshold. This threshold is set to $1/x$, controlled by the only parameter of the algorithm, x, which represents the maximum number of communities to which any vertex can belong. For $x < 2$, the algorithm is the same as the label propagation algorithm by Raghavan et al. [38].

If all of the pairs are below the threshold, then the pair with the greatest belonging coefficient is retained. If there are more than one pair with the greatest coefficient, one is randomly selected. This algorithm is non-deterministic due to this random selection. The normalisation of the belonging coefficients take place after removing the pairs below the threshold.

COPRA, based on the label propagation algorithm, does not converge to a state where the labels are constant between iterations. Therefore, the algorithm stops when there is no change in the minimum number of vertices labelled with each community identifier since the number of identifiers was last reduced. Gregory acknowledges that it is impossible to prove that the iteration at which the algorithm is terminated is always the best solution. [18].

Whang et al. proposed Neighbourhood-Inflated Seed Expansion (NISE) for detecting overlapping communities [45,46]. Their method relies on the initial selection of good seeds and then expanding them based on a metric defined for the community. The algorithm starts by filtering the graph and removing regions that can be easily separated, so that the algorithm can focus on the connected core. The next step is to find the seeds by applying a top-down hierarchical clustering and set the cluster centres as seeds. The seeds are then expanded by a personalised PageRank vector. The final step is the propagation where the communities are expanded to the regions that were removed in the filtering step. It is possible to set the number of seeds, or number of clusters for the algorithm; however, the algorithm may find fewer or more clusters depending on the duplicate clusters. Depending on the seeding and the filtering phase, it is possible that some vertices may not be assigned to a cluster.

One of the most known methods for detecting overlapping communities is the Clique Percolation Method (CPM) by Palla et al. [35]. As the name suggests, it uses cliques to locate overlapping communities. A clique is a subgraph in which every two distinct vertices are adjacent, also referred to as a complete subgraph [9]. A clique is called a maximal complete subgraph when it cannot be part of a larger complete subgraph. On the other hand, the clique may contain smaller cliques made up of k vertices called k-cliques. Two k-cliques are adjacent when they share $k-1$ vertices. Palla et al. define a "k-clique-community" as the union of all adjacent k-cliques. The choice of the k is arbitrarily chosen depending on the definition of a community for a particular network. This choice is dependent on how structured a community is which can be redefined as a ratio of the number of edges present or strong versus all possible within the community. This ratio is in turn used to predict a definition of community or k.

The first step in CPM is to locate the maximal cliques in the network. It is challenging to find all cliques of a graph since it is a non-polynomial problem, but some algorithms find all cliques, such as the Bron-Kerbosch algorithm [8].

The detection of all maximal cliques is followed by the preparation of a clique-clique overlap matrix which indicates the common vertices between two cliques. Using this matrix, any k-clique-community can be obtained for a given k by finding cliques that have at least $k-1$ common vertices.

CPM is realised in the *CFinder* software [1], however it is reported not to terminate in many large social networks [47].

There also exists a wide range of alternative approaches for detecting overlapping communities. These are compiled according to their methodological categories in the following surveys [3,20,47]. Amongst these methods, each has their own advantages and disadvantages.

Compared to these existing solutions, APAL has no oscillations that need to be controlled with stopping conditions, and requires no starting seeds that involve non-polynomial problems, making it much easier to understand and implement. Rather than having multiple parameters that require fine tuning, it operates with the threshold parameter alone, which is compared with the intraconnectivity of a subgraph.

3. Method

3.1. APAL: Adjacency Propagation Algorithm

There is no universally accepted definition of community; it is often defined with reference to the problem and the system in question [13]. APAL defines a community C as a subgraph of a graph $G, C \subseteq G$, in which the intraconnectivity of the community is above a given threshold value, t. The purpose of the Adjacency Propagation Algorithm is to find the overlapping communities given in an undirected, unweighted, nontrivial graph $G = (V, E)$. It rests on the idea that the adjacent vertices are the best candidates for the detection of a community.

Algorithm 1: Adjacency Propagation Algorithm (APAL)

```
1: APAL(G, t):
2: for all v ∈ V do
3:   for all v_n ∈ N(v) do
4:     C_c ← N(v) ∩ N(v_n)
5:     if |C_c| > 0
6:       C_c ← C_c ∪ {v_n, v}
7:       if Intraconnectivity(C_c) ⩾ t then
8:         C ← Evaluate(C, C_c, t)
9:       end if
10:    end if
11:  end for
12: end for
13: return C
14:
15: Intraconnectivity(C_c):
16: k ← 0
17: for all v_c ∈ C_c do
18:   k ← k + |N(v_c) ∩ C_c|
19: end for
20: return k/[|C_c|(|C_c| − 1)]
21:
22: Evaluate(C, C_c, t):
23: J_m = 0
24: C_m = ∅
25: for all C_n ∈ C do
26:   J_c = |C_n ∩ Cc|/|C_n ∪ Cc|
27:   α_c = Intraconnectivity(C_n ∪ Cc)
28:   if C_c ⊆ C_n then
29:     return C
30:   else if C_n ⊂ C_c then
31:     C ← C \ {C_n}
32:   else if J_c > J_m and J_c > t and α_c ⩾ t then
33:     J_m = J_c
34:     C_m = C_n ∪ Cc
35:   end if
36: end for
37: if C_m ≠ ∅ then
38:   C_c = C_m
39: end if
40: return C ∪ {C_c}
```

As listed in Algorithm 1, APAL visits each vertex $v \in V$, and operates on each of its adjacent vertices, $v_n \in N(v)$, to check if $N(v_n)$, and $N(v)$ have any vertices in common. If the candidate community C_c defined by the intersection $N(v) \cap N(v_n)$ is not an empty set, the vertices v_n and v are included into the candidate C_c.

APAL defines the intraconnectivity of a community by its ratio of sum of internal degrees to the maximum possible number of degrees. For a community of n vertices, the sum of internal degrees would be $n(n-1)$ when the subgraph is complete. Therefore, the intraconnectivity value α is defined as,

$$\alpha = \frac{k}{n(n-1)} \tag{2}$$

where k is the sum of internal degrees within the community.

The only parameter required for APAL is the threshold value t. The algorithm compares the intraconnectivity value α of candidate communities to the given threshold in order to continue to operate on them. Since APAL defines a community based on the α values, it computes α for a candidate community and compares it to the threshold value t, in order to accept the subgraph as a community. This gives the user an opportunity to use their own threshold value to define a community for the graph under consideration. The selection of the threshold value is discussed in detail in Section 3.2.

Finding a candidate community C_c is not the only condition for APAL to accept it. If C_c can pass the size and the threshold checks, it is then compared with the existing communities. If the candidate is a subset of or equal to one of the existing communities, then it is discarded. If one of the existing communities is a subset of the candidate, then the existing community is removed from the set of communities. Since a large candidate can be a superset of many existing communities, we keep removing them as we visit all of the communities. Once all are visited, the candidate is joined to the set of communities.

If the candidate community C_c is not a subset or a superset of an existing community C_n, then their union is considered as a temporary candidate to further propagate adjacency. This requires three checkpoints. First, as with all communities, the union of these communities must have an intraconnectivity value greater than the threshold value t. However, such a union should be treated with care. Because APAL checks for subsets and supersets before accepting a candidate community, two very dissimilar communities might create a very large community where the smaller communities might be lost in later iterations of the algorithm. Therefore, APAL uses Jaccard index, J, to quantify the similarity of C_c and C_n which is the ratio of their intersection and their union. For the second check, while the candidate is compared to existing communities, APAL looks for the community with the highest similarity to unite the candidate with. The final check requires that the similarity between the candidate and existing community should be above the threshold value t. The threshold value is set to the expected intraconnectivity of the communities, but having the same range as the Jaccard index and a similar expectancy of a larger value, it is also used in cases where a union is considered. This also avoids the need for APAL defining another parameter which is occasionally used and operationally similar to threshold value.

Consider the bow-tie graph shown in Fig. 1. This simple graph has two overlapping communities, shown in dashed circles, where vertex a is common in both. It should also be noted that the communities are non-complete subgraphs. Let us run APAL on this graph with a threshold value of 0.7.

The algorithm starts with the first vertex in the set V. Assuming that it is a, it finds the vertices that are adjacent to it: $\{b, d, e, g\}$. APAL considers the adjacent vertices of the adjacent vertices, and identifies any common ones. For vertex b, it finds that the adjacent vertices are $\{a, c, d\}$, and their intersection is $\{d\}$. Current vertex and current adjacent vertex are added to the candidate community, $\{a, b, d\}$. Using Eq. (2), the intraconnectivity value can be computed as 1, because these three vertices form a complete subgraph. Consequently, this community is added to the set of communities: $C = \{\{a, b, d\}\}$.

The vertex d will reveal the same community and will be discarded while comparing it to the other communities in C. Either the vertex e or g will add the complete subgraph $\{a, e, g\}$ to set $C : C = \{\{a, b, d\}, \{a, e, g\}\}$.

After completing the first vertex, APAL will move on to the next, say b. The adjacent vertices for b are $\{a, c, d\}$ so APAL will consider all of their adjacent vertices. While the vertex a reveals no new information, vertex c will produce the community $\{b, c, d\}$. When compared to existing community $\{a, b, d\}$, their Jaccard index is $J_c = |\{b, d\}|/|\{a, b, c, d\}| = 0.5$ which is less than the threshold value. The index is smaller for the other community $\{a, e, g\}$. Therefore $\{b, c, d\}$ is added to the set of communities: $C\{\{a, b, d\}, \{a, e, g\}, \{b, c, d\}\}$.

The adjacent vertices for d are $\{a, b, c\}$ and their intersection produces the community $\{a, b, c, d\}$. The intraconnectivity value for this community is $\alpha = (2 + 3 + 2 + 3)/(4 \times 3) = 0.8\bar{3}$ which is greater than the threshold $t = 0.7$. When this community is compared to the existing ones, the communities $\{a, b, d\}$ and $\{b, c, d\}$ will be removed since they are the subsets of this new community: $C\{\{a, b, c, d\}, \{a, e, g\}\}$. Similar steps will take place on the other side of the bow-tie and the final set of communities will be $C\{\{a, b, c, d\}, \{a, e, f, g\}\}$.

APAL is deterministic, using or depending on no random process. However, the processing order of the vertices depend on the definition of the sets, which can vary because of the way of the input data is presented. If the vertices are provided in the same order, the resulting communities will always be the same. If not, possible differences in the existing communities during the Evaluate function may result in the detection of different communities. We have evaluated how the processing order affects the results in Section 5.2.

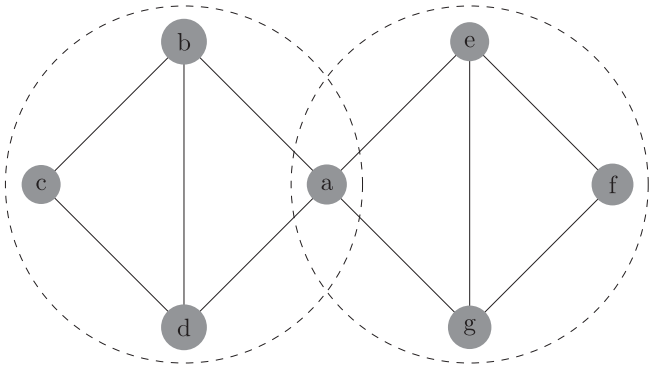

Fig. 1. The bow-tie graph where two communities $\{a, b, c, d\}$ and $\{a, e, f, g\}$ overlap.

APAL operates on each vertex of the given graph and, depending on the threshold value, tries to assign them to a community, but may fail in some cases. If the given graph has isolated vertices that have no neighbours, or if there are vertex pairs that are isolated, APAL would not recognise them as communities. In such cases there might be vertices that are not assigned to any community.

3.2. The threshold value

Choosing the threshold value is critical to the success of the algorithm. If the communities are expected to be highly connected internally, a value closer to 1 would be appropriate, but for looser connections, lower values could be set.

User familiar with the type of network APAL will run on would be able to estimate the intraconnectivity value of the communities. Therefore, they can choose a threshold value that will accept the communities with higher α values. For example, experience shows that biological networks have lower intraconnectivity values, so a threshold value around 0.3 produces good results.

In the absence of such familiarity, different approaches can be taken to the selection of the threshold. For large networks, a statistically sufficient number of random communities could be extracted and their mean intraconnectivity value could be used. While this approach has a preprocessing step that requires community extraction, there are available methods, such as [39], and the results can shed some light on the nature of the network.

For smaller networks, APAL could be run with a range of threshold values, such as from 0.3 to 0.7 with a step of 0.05, since APAL is a compact algorithm with a polynomial running time, as will be discussed in the following section. For each iteration, the resulting communities could be analysed by the user to select the most appropriate threshold.

3.3. Complexity analysis

The input for APAL are the graph $G(V, E)$ and the threshold value t. Let $n = |V|$ be the order, and $m = |E|$ be the size of graph G. The algorithm loops through all n vertices, and all of their adjacent vertices. The number of adjacent vertices is called the degree of a vertex. If there are m edges, then there are $2m/n$ edges on average for each vertex, since each edge is incident to two vertices. The traversing of the vertices and their adjacent vertices takes $O(m)$ time on average.

APAL uses sets to operate on communities using operations such as intersection, union, difference of sets and subset checking. The data structure of the sets determine the running time of these operations. If the sets are implemented using a hash table, look-ups in the set can be achieved in constant time, $O(1)$, on average. The intersection operation can be achieved by looping the elements of the smaller set and looking them up in the larger set, which will be in linear time. Adding a new element to the set takes constant time on average, and the union operation repeats this until all elements of the smaller set is added to the larger set, which takes linear time. The subset operation checks whether all elements of a set exist in the other set. Since look-ups take constant time and the process is repeated for all elements in the set, the running time is linear. Finally, the difference operation has to loop over the set, which is linear time, and check if they exist in the other set, which is constant time.

While all of these set operations take linear time on average, APAL uses some of them with very small sets. For example, on line 6 of Algorithm 1, a union operation is performed using a set made up of the current vertex v and the current adjacent vertex v_n. Since this line always uses these two vertices, its running time is a constant time of only two $O(1)$ operations. In another one, on line 31, the current community is removed from the set of communities. Since it is only one element, this is the same as deleting it, which would take constant time for the look-up and deletion. The last line of the Evaluate function, line 40, also includes a union with a single element, which also takes constant time. Nevertheless, there are operations with sets where larger sets are involved, and most of the time the operations are performed on communities.

The Intraconnectivity function loops through the vertices of the candidate community and finds the number of elements of the intersection of the adjacent vertices of the vertex at hand and the community. Since the candidate community is formed by the intersection of two sets of adjacent vertices, the number of elements are $2m/n$ on average. The intersection operation inside the *for* loop operates on two sets which have $2m/n$ elements on average. Thus the running time of this function is $O(m^2/n^2)$.

The Evaluate function loops through all of the communities and perform subset operations. The number of communities change as the algorithm runs, however, the average size of a community is $2m/n$ and there are n vertices, therefore, the number of communities is $n/(2m/n) = n^2/2m$. Checking subsets, intersection and union operations take linear $2m/n$ time, while calling the Intraconnectivity function takes $O(m^2/n^2)$ time. The running time for the Evaluate function is then $O(n^2/2m \times (2m/n + m^2/n^2)) = O(m/2 + n)$.

Consequently, the operations repeated while traversing all of the vertices and their adjacent vertices take $O(m^2/n^2 + m/2 + n)$ time. Since we have established that traversing takes $O(m)$ time, the running time for APAL is found as $O(m^3/n^2)$. In any graph, the number of edges will be greater than the number of vertices, so it is likely that the number of edges will dominate the running time as the numbers grow. The algorithm is in polynomial time, being slightly faster than cubic time.

4. Data

4.1. Overlapping graph generator

When real data is not available, or when the ground truth for the real data is not known, synthetic data can be generated to test and evaluate the methods. If the generator has parameters, then the properties of the graph can also be adjusted for testing extreme cases.

The Overlapping Graph Generator (OGG) was designed for this purpose and takes the following set of parameters to define the structure of a generated graph with overlapping communities:

- Number of communities, ς
- Average number of vertices in communities, η
- Intraconnectivity, α, defines the connectedness ratio of the community,
- Interconnectivity, ω, is the connectivity between communities,
- Overlapping factor, ξ
- Connect ratio, ζ.

The intraconnectivity parameter, α, is set as the ratio of the number of edges within the community to the all possible edges within that particular community. While the maximum value can be 1 for a community with a complete graph, the minimum value depends on the number of vertices, because the generator first ensures that the community is connected using disjoint sets and union algorithms. If there are η vertices at average, $(\eta - 1)$ edges are required to make the community connected within itself. There has to be at most $\eta(\eta - 1)/2$ edges to make the graph complete. As a result the intraconnectivity of the community is at least

$$\alpha = \frac{(\eta - 1)}{\frac{\eta(\eta-1)}{2}} = \frac{2}{\eta},$$

depending on the average number of vertices in a community. More edges are added until the target intraconnectivity is reached. It should also be noted that the intraconnectivity of each community is different. OGG generates random intraconnectivity values which averages on the target value.

The same approach is used for the interconnectivity of the communities. The communities can be thought of as vertices in a graph, so the same procedure for the intraconnectivity parameter is applied here. In this case, the minimum value is $\omega = 2/\varsigma$ and the maximum, 1. Since any number of edges or vertices shared by two communities may be defined as a single connection, the interconnectivity is simply defined as the ratio of the number of connections between communities to the possible number of community connections.

The interconnectivity is achieved by either adding an edge between the vertices of two communities, or by merging a vertex from one community with a vertex from the other. When merged, their connections are updated to reflect the change. Consider the case in Fig. 2. There are three communities which will be connected using the vertices in dashed circles in Step I. Vertices *a* and *c* are connected by adding an edge, as shown in Step II. The merging of vertices *f* and *k* result in the removal of one, in this case *k*, and then adding its edges to the other vertex, *f*. The decision to choose either merging, or adding an edge is done with the overlapping factor ξ.

The final parameter, connect ratio, ζ, is used to determine the number of vertices used to connect two communities. In Fig. 2, only one vertex was used from each community, but in larger graphs, more vertices can be chosen. The generator adjusts the η parameter to accommodate more vertices if necessary.

A sample graph generated with the number of communities $\varsigma = 4$, the average number of vertices $\eta = 10$, the intraconnectivity ratio $\alpha = 0.9$, the interconnectivity ratio $\omega = 0.5$, the overlapping factor $\xi = 0.6$ and the connect ratio $\zeta = 0.4$ is shown in Fig. 3.

Although OGG uses random processes to generate a graph, the generation process is controlled strictly. To check the quality of the graphs, the expected parameters are compared against the properties of the generated graph. If the parameters are given within the boundaries, OGG successfully generates graphs with the requested properties.

4.2. The Lancichinetti–Fortunato–Radicchi benchmark

The Lancichinetti-Fortunato-Radicchi (LFR) benchmark has been proposed by its namesake authors [26] to overcome the unrealistic properties of the then commonly used benchmark by Girvan and Newman (GN) [17]. The GN benchmark had four groups of 32 vertices with an average degree of 16 where the vertices had approximately the same degree. Lancichinetti et al. created a benchmark that can produce networks that have heterogeneous distributions of the vertex degrees. They use power law distributions for the degree and the community sizes. The algorithm uses the following parameters:

- Number of vertices, *N*
- Average degree, $\langle k \rangle$

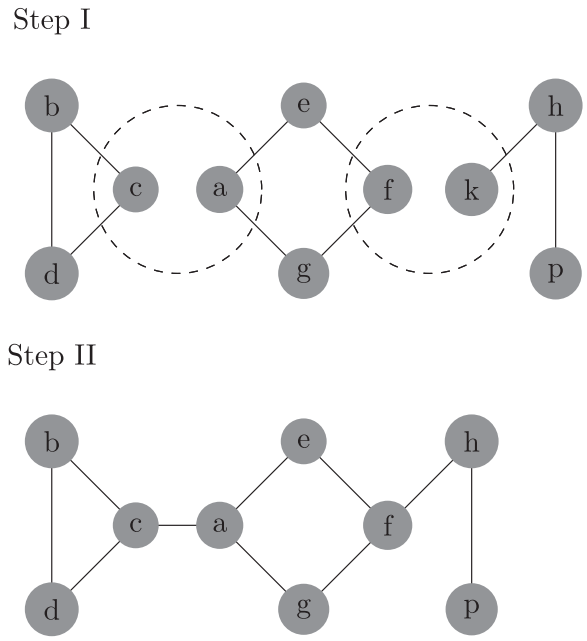

Fig. 2. Connecting with an edge and merging two vertices in steps I and II using the Overlapping Graph Generator.

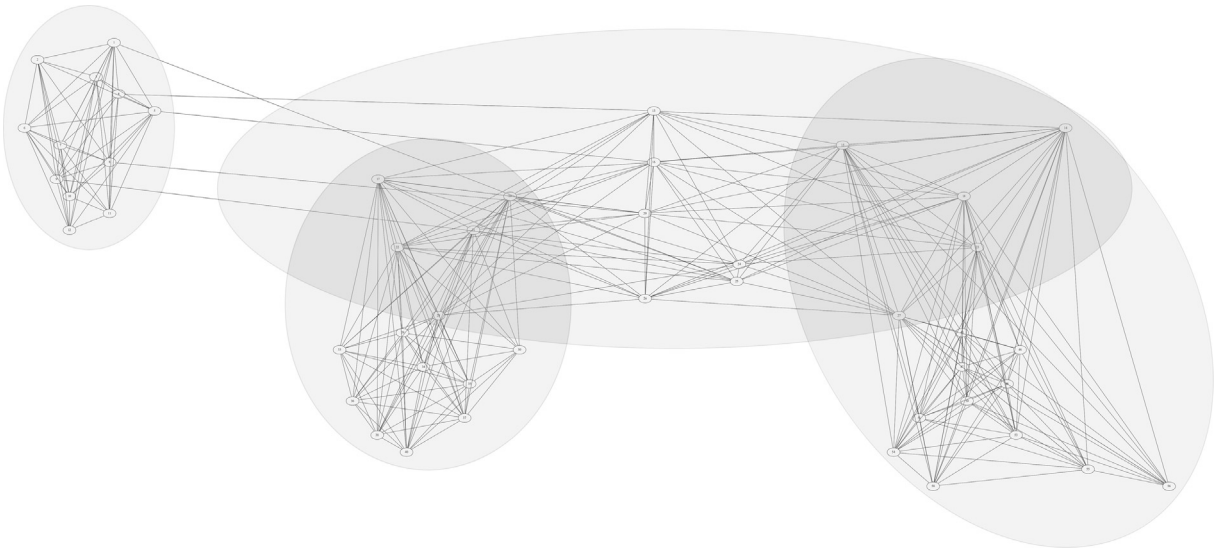

Fig. 3. A graph generated by Overlapping Graph Generator. The shaded circles represent the four clusters, $\varsigma = 4$, defined by the user.

- Maximum and minimum degrees, $k_{\min}, k_{\max}$
- Mixing parameter, μ
- Maximum and minimum community sizes, $s_{\min}, s_{\max}$
- Power law distribution degrees, γ, β

Each vertex gets a degree from a power law distribution with exponent γ and the sizes of the communities are from a power law distribution with exponent β. The maximum and minimum values for the vertex degrees are chosen to respect the average degree, $\langle k \rangle$, while the community sizes are chosen to be $s_{\min} > k_{\min}$ and $s_{\max} > k_{\max}$. Each vertex shares μ fraction of its edges with vertices outside its community.

Lancichinetti and Fortunato extended the benchmark so that it can generate directed and weighted graphs with overlapping communities by assigning a parameter v_i to each vertex v_i for determining the number of communities the vertex belongs to [24]. The realisation of the benchmark is available online for undirected and unweighted graphs with overlapping communities by one of the authors.[1] It uses the o_n and o_m parameters to determine the number of overlapping vertices and the number of memberships of the overlapping vertices, respectively.

Networks created by both OGG and LFR are analysed to extract their properties such as the intraconnectivity, interconnectivity and the number of communities. In order to differentiate these properties from the parameters of the same name used for OGG, we use the subscript notation with the network graph G, such as G_ω for the interconnectivity property.

The intraconnectivity property G_α for a network represented by graph G is defined as the ratio of the number of connections between the vertices in a community to the all possible connections within the community. The intraconnectivity value for each community is computed and the average is assigned as the community intraconnectivity value. The interconnectivity property, G_ω, is defined as the ratio of the number of connections between communities to the number of all possible connections between communities. The property for the number of communities, G_ς, is simply the number of communities in the generated network.

4.3. Yeast protein interactions network

Yeast protein interactome (YPIN) was obtained from various sources as described in [4]. Previously reported protein subunit interactions were collected as follows: 2990 from [48], 2770 from [41] and 58 from [30]. Among these interactions, self-loops and duplications were removed, including reverse directed interactions, yielding a total of 5216 undirected interactions among 2634 protein subunits. The list of complexes and their associated protein subunits were obtained from CYC2008 [37]. Each complex that is formed by the 2634 subunits is regarded as the true community.

A series of elimination processes consisted of three steps; the first is the removal of any interactions in which its subunits are not included in the complexes; the next step is the removal of any subunits in the complexes that was not included in the interactions; the final step is the removal of any complexes with less than three subunits. These steps were performed in cycles until no more omissions were possible. The remaining 111 complexes were regarded as the true communities. Following the elimination process, 435 protein subunits making 508 interactions remained.

Since the final step eliminates any complex with less than three subunits, any community discovered with less than two vertices was also discarded from final communities discovered by COPRA for fair comparison. APAL and CPM do not require such filtering process.

4.4. Human protein interactions network

For human protein interactome, a data set containing both the protein subunit interactions and their associated complexes were obtained from [23]. The data set classifies each association into one of three categories depending on the level of certainty of the association;

- Category 1, where the proteins that are known to form part of a complex,
- Category 2, where the proteins are known to have a related function,
- Category 3, where the association is predicted by computational analysis.

The data was used to generate three different interaction networks with varying degrees of association based on these categories; HPIN1 includes only Category 1 associations; HPIN2 includes Category 1 and Category 2 associations; and HPIN3 includes all associations. The elimination process that was applied to YPIN was also applied to each of these networks yielding a final set of 1506 subunits connected through 5154 interactions and forming 366 complexes in HPIN1, a final set of 1817 subunits connected through 6579 interactions and forming 430 complexes in HPIN2 and a final set of 4337 subunits associated through 20329 interactions and forming 1220 complexes.

5. Results and discussion

In this study, comparisons were made between APAL and COPRA [18], NISE [45,46] and CPM [35] algorithms, using the synthetic and real data sets mentioned in Section 4. These three algorithms were chosen for different reasons: COPRA algorithm, because of its similarity to APAL for its application of propagation and its accurate results; NISE, because it is one of the newer algorithms, and CPM, because it is one of the most commonly used methods available at the time of writing.

The performance of the algorithms has to be compared quantitatively. While metrics based on mutual information are more common, there are many other approaches for comparing two disjoint communities that are listed in studies such as [27,29,43]. However, these metrics have to be extended for the case of overlapping communities.

[1] https://sites.google.com/site/santofortunato/inthepress2

We have adopted the normalised mutual information (NMI) metric that is extended for overlapping communities as detailed in [25]. It has a range of $[0, 1]$ where values closer to 1 are of communities that are more alike, therefore represent better results.

5.1. APAL threshold parameter

APAL has a single threshold parameter, t, that is used in comparison with the intraconnectivity value of a community defined in Eq. (2). Since this value only considers the sum of internal degrees, the connectivity of a candidate community within its own vertices contribute more to the detection process.

Prior to the comparison with other algorithms, the effect of the threshold parameter on the discovered communities is evaluated by varying the parameters of the synthetic generators, OGG and LFR.

The networks generated by OGG were adjusted using the parameters for the overlapping factor, ξ, the interconnectivity of communities, ω and the intraconnectivity of a community, α. Several networks were generated by different configurations of these parameters by varying them between values $[0.1, 0.9]$ with a step of 0.1. After the generation process, the resulting overlapping, interconnectivity and intraconnectivity properties of these networks were computed and then the networks were used to test the threshold parameter for values between $[0, 1]$ with an increasing step of 0.1.

The threshold parameter t is sensitive to the intraconnectivity property of a network. Fig. 4 shows the performance of APAL as the intraconnectivity of the network increases. As long as the upper bound of the threshold parameter is the intraconnectivity value, APAL continues to yield good NMI scores.

Fig. 5 shows the performance of APAL as t changes while the interconnectivity parameter ω increases. Since APAL defines a community as the sum of internal degrees, it is indifferent to the connections to the other communities.

The LFR benchmark uses the mixing parameter μ which denotes the fraction of the edges outside the community, and the o_n and o_m parameters, which denote the number of overlapping vertices and number of memberships of overlapping vertices, respectively. Just as with the OGG, several networks have been generated by varying these parameters as follows; the number of vertices, N, was set to 1000, the average degree $\langle k \rangle$ was set to $12, k_{\max}$ was set to 24 and the community sizes were set to $s_{\min} = 15$ and $s_{\max} = 25$ for all while the mixing parameter μ varied between $[0.1, 0.45]$ with a step of $0.05, o_n$ is varied between $[0, 500]$ with a step of 100 and o_m is set to values $\{2, 4, 6\}$. The intraconnectivity α, interconnectivity ω and overlapping ξ parameters, as well as the number of communities ς were extracted for these networks. They were tested with APAL by varying the threshold parameter between $[0, 1]$ with a step of 0.1.

The performance of APAL versus increasing mixing parameter μ is plotted in Fig. 6. As μ increases, the performance for APAL drops, albeit not significantly. As the plot shows, the performance decreases for different μ levels and the reason becomes apparent when the properties of these networks are analysed. The intraconnectivity of these networks start at $G_\alpha = 0.423$ and decreases as μ increases where a minimum intraconnectivity value of $G_\alpha = 0.289$ is reached. Since APAL uses the intraconnectivity property, the NMI scores are higher until the threshold t is greater than the intraconnectivity value. This shows the sensitivity of APAL to the intraconnectivity property; the greater the understanding of the average connectedness of the network, the greater the likelihood of setting the threshold value to the best possible value.

Similarly, the interconnectivity property of the network increases in line with the number of overlapping vertices o_n, as shown in Fig. 7. This also results in a decrease in the intraconnectivity property. As long as the threshold t is lower than the intraconnectivity value, APAL is able to reach higher scores.

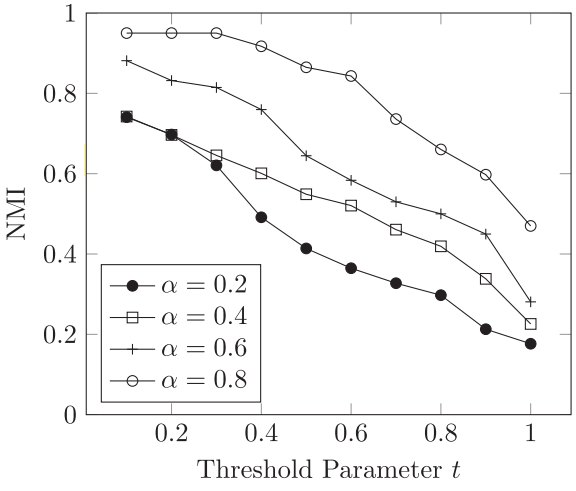

Fig. 4. The performance of APAL against the intraconnectivity parameter α as its threshold parameter t changes for networks generated by OGG with parameters $\xi = 0.3, \omega = 0.2$ and α ranging from 0.2 to 0.8.

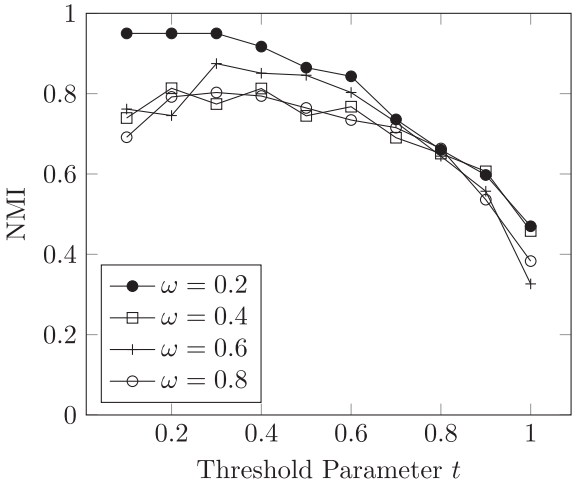

Fig. 5. The performance of APAL against the interconnectivity parameter ω as its threshold parameter t changes for networks generated by OGG with parameters $\xi = 0.3, \alpha = 0.8$ and ω ranging from 0.2 to 0.8.

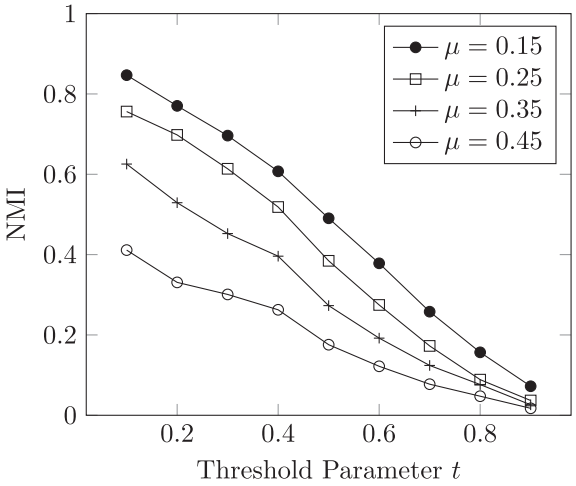

Fig. 6. The performance of APAL as μ is increased from 0.15 to 0.45 for $o_n = 300, o_m = 2$ and $N = 1000$ for networks generated by LFR.

Finally, the number of memberships of the overlapping vertices, o_m, is analysed for $o_n = 300$ and $\mu = 0.3$ as shown in Fig. 8, which has the most dramatic effect on the threshold parameter t. As the number memberships of overlapping vertices increase, so does the interconnectivity value of the generated network. Such increase is also reflected in the intraconnectivity property of the network. For these three LFR networks, the interconnectivity increases from $G_\omega = 0.806$ to $G_\omega = 0.903$ as intraconnectivity decreases from $G_\alpha = 0.338$ to $G_\alpha = 0.178$. These values are also reflected in the NMI scores of APAL since the scores begin to decrease when threshold values become greater than the intraconnectivity value.

The results show that APAL deals with the intraconnectivity in a stable way; as long as the threshold t is below the intraconnectivity property of the network, APAL can detect communities and reach high NMI scores. Although it should be remembered that while individual communities may have different intraconnectivity values, threshold t is applied universally. The results also reveal that the merging process can reduce the number of detected communities for low values of t. Empirically, based on the real dataset YPIN, (defined in Section 4.3) we have chosen a value of $t = 0.35$ as a good balance between the community detection and community merging processes of APAL. However, for a network known to have more densely connected communities, a higher value, such as $t = 0.7$ is more appropriate, and for more loosely connected communities, lower values, such as $t = 0.1$. Just as other community detection algorithms, APAL would also benefit from the structural analysis of the network in question.

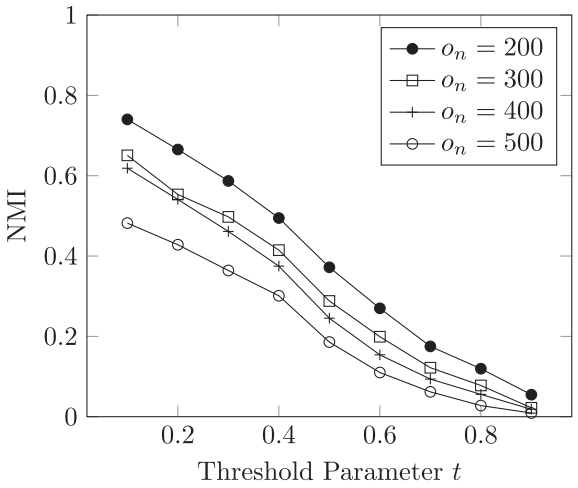

Fig. 7. The performance of APAL as o_n is increased from 200 to 500 for $\mu = 0.3, o_m = 2$ and $N = 1000$ for networks generated by LFR.

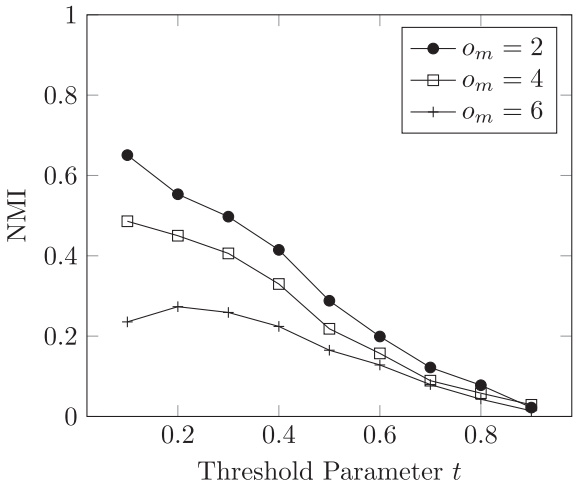

Fig. 8. The performance of APAL for o_m values $\{2, 4, 6\}$ for $\mu = 0.3, o_n = 300$ and $N = 1000$ for networks generated by LFR.

5.2. Alternating vertex order in graph data

While APAL returns deterministic results with a given set of data, alternating the vertex order in that data may change the final list of communities discovered. To evaluate the impact of the order of the vertices, APAL was allowed to run artificially generated and real data with shuffled vertex lists. The vertex order of LFR generated data of 1000 vertices and varying mixing parameter o_m =0, 10, 20, 50, 100, 200 and 300 was repeatedly shuffled and run with APAL for 100 times for each o_m. Each community list output discovered by $\text{APAL}_{t=0.1}$ was compared to the initial output to yield an NMI score. The average of NMI scores between community data discovered by the shuffled vertex orders was above 0.8 for all tests as listed in Table 1. According to these results, increasing o_n, or increasing overlap ratio resulted in decrease in similarity of cluster lists produced by $\text{APAL}_{t=0.1}$. For the real data, the vertex order of YPIN data (defined in Section 4.3) was also shuffled for 100 times, and the average NMI score was determined to be 0.967. It should be noted once again that these results indicate the difference between APAL predictions, not similarity of the APAL predictions to the true communities.

5.3. Comparison using synthetic networks

The two generator algorithms, OGG and LFR benchmark, were used with distinct features to generate overlapping networks in order to test the capabilities of APAL, CPM, NISE and COPRA. APAL was tested with different values for its threshold parameter, $t = 0.1, t = 0.35$ and $t = 0.7$, depending on the case. For the CPM algorithm, the k parameter was set to $k = 3$ and

Table 1
NMI scores calculated to compare the predictions of APAL with a threshold of $t = 0.1$ using LFR generated and real (YPIN) networks. The averages and standard deviations are given for the NMI scores that are calculated between 99 randomly shuffled vertex orders and the unshuffled orders for each graph. For LFR generated data, o_n=0,10,20,50,100,200,300, $k = 12, maxk = 24, \mu = 0.1, minc = 15, maxc = 25, o_m$=2, N = 1000 parameters were used. The table lists the average NMI Scores and the standard deviation, σ of the results.

Graphs	Average NMI Score	σ
LFR $o_n = 0$	0.994	0.0036
LFR $o_n = 10$	0.982	0.0063
LFR $o_n = 20$	0.988	0.0053
LFR $o_n = 50$	0.945	0.0104
LFR $o_n = 100$	0.921	0.0129
LFR $o_n = 200$	0.870	0.0133
LFR $o_n = 300$	0.801	0.0128
YPIN	0.967	0.0204

$k = 4$. The COPRA algorithm was run using its default values, however, we have found out that it detects communities that have a single vertex. These were removed to better reflect its NMI score. The number of seeds for NISE has been set to $k = 15$.

For the networks generated by OGG, the intraconnectivity property was varied to compare the algorithms, as shown in Fig. 9. The performance of APAL and other algorithms was shown to increase with intraconnectivity property of the OGG. While the threshold choice of $t = 0.7$ for APAL resulted in lower NMI scores for most intraconnectivity parameters, the higher end of intraconnectivity rewarded this threshold by dramatically increasing the NMI, indicating a strong association between intraconnectivity and optimum threshold.

When the interconnectivity property is modified, the performance of the algorithms change as in Fig. 10. $\text{APAL}_{t=0.35}$ and CPM_4 produce almost identical results, while CPM_3 fails to get high NMI scores as the interconnectivity increases. For both cases, APAL scores at least as well as COPRA.

The LFR benchmark uses a number of parameters to control the properties of the generated overlapping network. Our results show that the mixing parameter μ, the number of overlapping nodes, o_n, and the number of memberships of the overlapping nodes o_m are the most appropriate parameters to represent the distinction between the algorithms that work on the detection of overlapping communities.

Table 2 lists the NMI scores of the community detection algorithms for the networks generated by the LFR benchmark as the mixing parameter μ is increased for $o_m = \{2, 4, 6\}$. For all networks, N is set to 1000 and $o_n = 500$. The table shows that APAL receives relatively higher NMI scores as the community overlapping increase in comparison to other algorithms. APAL with a threshold value of $t = 0.1$ scores highest in the majority cases, and $t = 0.35$ in the minority. When compared to other algorithms, CPM has the closest scores to APAL, while COPRA and NISE have lower scores, as can be seen in Fig. 11. The analysis of the networks generated by LFR for $o_m = 6$ and increasing values for μ show that their interconnectivity property increases from 0.64 to 0.91, their overlapping property decreases from 0.24 to 0.17, while the intraconnectivity property decreases from 0.4 to 0.24. These results demonstrate APAL's advantage; despite changes in the overlapping and interconnectivity properties of the network change, APAL manages to extract the communities due to consideration of the internal edges of the candidate community.

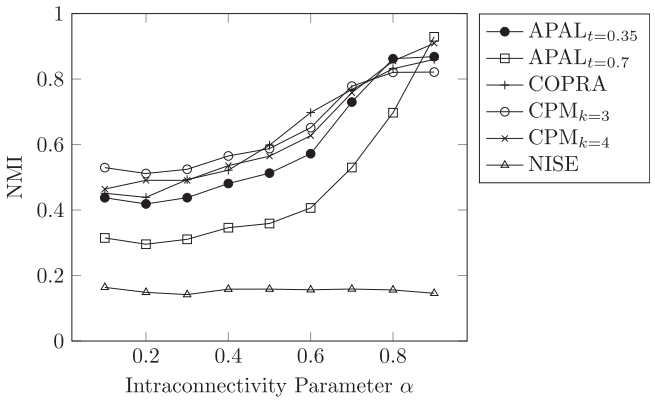

Fig. 9. The performances for APAL with thresholds $t = 0.35$ and $t = 0.7$, COPRA, NISE, and CPM with $k = 3$ and $k = 4$ for overlapping networks generated by OGG using $\xi = 0.3, \omega = 0.2$ while the intraconnectivity parameter α increases.

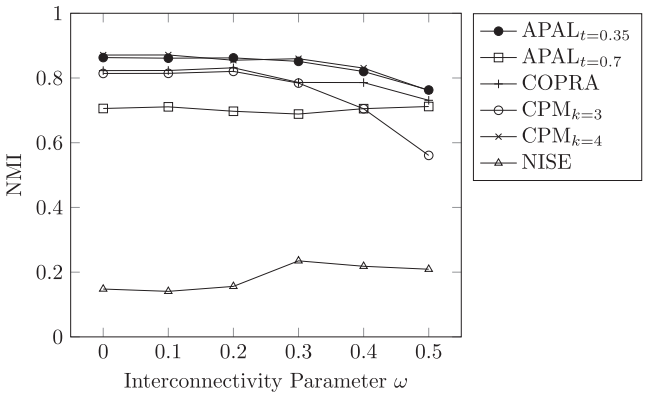

Fig. 10. The performances for APAL with thresholds $t = 0.35$ and $t = 0.7$, COPRA, NISE, and CPM with $k = 3$ and $k = 4$ for overlapping networks generated by OGG using $\xi = 0.3, \alpha = 0.8$ while the interconnectivity parameter ω increases.

Table 2
NMI scores for APAL with thresholds $t = 0.1$ and $t = 0.35$, COPRA, NISE and CPM with $k = 3$ and $k = 4$ for overlapping networks generated by LFR using $o_m = \{2, 4, 6\}, o_n = 500$ and $N = 1000$ while the mixing parameter μ increases. Higher NMI scores among the choice of parameters for APAL and CPM algorithms are presented in bold.

o_m	μ	$\text{APAL}_{t=0.1}$	$\text{APAL}_{t=0.35}$	COPRA	NISE	$\text{CPM}_{k=3}$	$\text{CPM}_{k=4}$
2	0.1	**0.748**	0.58	4.262E-17	0.0081	0.372	0.609
	0.15	**0.729**	0.568	4.274E-17	0.0081	0.491	0.586
	0.2	**0.645**	0.501	4.263E-17	0.0085	0.330	0.433
	0.25	**0.633**	0.467	8.524E-17	0.0070	0.266	0.390
	0.3	**0.525**	0.360	0.0009	0.0079	0.295	0.27
	0.35	**0.446**	0.331	0.0008	0.0077	0.288	0.209
	0.4	**0.369**	0.249	0.0009	0.0075	0.207	0.143
	0.45	**0.269**	0.184	0.0009	0.0080	0.142	0.106
4	0.1	**0.425**	0.355	0.0024	0.0119	0.295	0.264
	0.15	**0.417**	0.308	0.0009	0.0117	0.198	0.234
	0.2	**0.323**	0.259	0.0009	0.0112	0.183	0.173
	0.25	**0.311**	0.234	0.0004	0.0116	0.234	0.172
	0.3	**0.263**	0.211	0.0004	0.0103	0.175	0.133
	0.35	**0.203**	0.164	0.0009	0.0116	0.166	0.11
	0.4	**0.163**	0.127	0.0004	0.0121	0.133	0.077
	0.45	**0.094**	0.067	0.0004	0.0104	0.056	0.053
6	0.1	**0.275**	0.214	0.0003	0.0118	0.243	0.147
	0.15	**0.177**	0.170	0.0009	0.0123	0.162	0.1
	0.2	**0.167**	0.151	0.0003	0.013	0.15	0.063
	0.25	**0.135**	0.112	0.0009	0.01	0.121	0.041
	0.3	0.087	**0.101**	0.0003	0.0120	0.091	0.013
	0.35	0.063	**0.065**	0.0009	0.0117	0.051	0.028
	0.4	0.046	**0.062**	0.0009	0.0114	0.045	0.008
	0.45	0.014	0.015	0.0003	0.0104	0.011	**0.022**

5.4. Comparison using real graphs

It is established that understanding the structural characteristics of any given network is an important factor for successful community discovery. When compared to other types of networks, aspects of biological, and particularly protein interaction networks, have certain aspects that do not resemble to any other types of networks.

Obtaining the biological data is labour-intensive and yet, biased in methodology and highly prone to error. The produced network is often noisy and rich in false results [2]. A high ratio of the false positive interactions is common, caused by the inability to distinguish between direct and indirect interactions of the proteins through experimentation. At the same time, a significant fraction of the false negatives are omitted due to low sensitivity in the experimental methods. While acquisition of these interactions is costly and labour-intensive process, most methods require each interaction to be tested individually. This leads to omissions of large chunks of data which are still believed to be present in the true network [21]. As a result, the experimentally obtained network is not only a minor representation of the true network but also contains large gaps and noise.

For the applications of the real data, it should be noted that the definition of a community varies widely, and is dependent on the particular application. For protein interaction networks, the protein complexes are often reflected to the network as

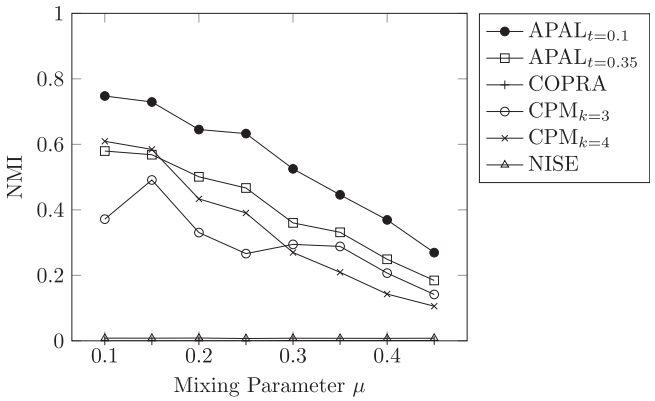

Fig. 11. The performances for APAL with thresholds $t = 0.1$ and $t = 0.35$, COPRA, NISE and CPM with $k = 3$ and $k = 4$ for overlapping networks generated by LFR using $o_m = 2, o_n = 500$ and $N = 1000$ while the mixing parameter μ increases.

Table 3
NMI scores calculated for predictions of different algorithms using human (HPIN) and yeast (YPIN) interactome networks. Known complexes are regarded as true community sets. Table lists NMI scores for both optimum APAL threshold t and where $t = 0.35$. Highest NMI scores for each dataset was presented in bold.

Algorithm	HPIN1	HPIN2	HPIN3	YPIN
$\text{APAL}_{\text{best}}$	0.445	0.419	**0.389**	**0.443**
$\text{APAL}_{t=0.35}$	0.340	0.321	0.200	0.434
$\text{CPM}_{k=3}$	**0.454**	**0.441**	0.316	0.441
$\text{CPM}_{k=4}$	0.333	0.305	0.274	0.300
$\text{CPM}_{k=5}$	0.288	0.287	0.278	n/a
COPRA	0.372	0.342	0.210	0.335

dense subgraphs and thus are regarded as communities. Therein, we regard the known complexes as true communities and compare accordingly [28]. A complex may have several core subunits and other regulatory subunits that bind or release the core depending on the regulatory state of the biological system. These regulatory subunits can often interact with different cores, thus requiring an overlapping network discovery to reveal putative protein complexes [21].

For the evaluation of the community discovery algorithms, two data sets were chosen to generate four graphs where YPIN represent the protein interactions network in yeasts, and HPIN1, HPIN2 and HPIN3 represent human protein interactome at different levels of associations. All algorithms have been tested with the data except NISE, which requires connected graphs. The results are summarized in Table 3.

For the three human interactomes, NMI scores were found to be between 0.45 and 0.39 for APAL. It should be noted that these were obtained for very low t threshold values. This result was not at all surprising since large chunks of interactions are missing in such networks. These NMI scores were also very close to those of CPM_3, with exception of HPIN3 where APAL yields an NMI score well above any other.

In case of YPIN, the similarity between APAL and $\text{CPM}_{k=3}$ is noticeable with both recording an NMI score around 0.44. In all cases $\text{CPM}_{k=3}$ and APAL yielded better results in comparison to COPRA.

Although NMI scores yielded small improvements in these cases, such as in HPIN3, the overall accuracy of the present algorithms is rather unsatisfactory as their NMI scores remained well below 0.5 in all circumstances using biological data. Until the missing portions of the interactomes are restored or the definitions of communities are improved, lower accuracy should be expected.

It should be noted that APAL and other algorithms showed better NMI scores when protein clusters with well-established interactions are considered as the true community, as in HPIN1, in comparison to inclusion of loosely associated proteins, as in HPIN2 and HPIN3. This represents that these algorithms are better suited for the discovery of the core subunits of protein complexes.

6. Conclusion

We proposed a novel overlapping community discovery algorithm based on the propagation of the adjacency list of the vertices, and investigated its advantages and disadvantages in comparison to other well known algorithms, COPRA, NISE and CPM. APAL was tested rigorously using not only synthetic generators, OGG and the LFR benchmark, but also with real data sets of yeast protein and human protein interactions networks.

A basic understanding of the network is sufficient to infer the intraconnectivity property and set the threshold parameter accordingly. Also due to its threshold parameter, APAL compensates missing and low density data that may feature in real

biological networks. As the test results show, for the threshold values much lower than the intraconnectivity property of the network, APAL has performed as well as, if not better than, other algorithms.

APAL is more effective than COPRA, NISE and CPM algorithms, especially in networks where the number of memberships of the overlapping vertices is high. Such situations are often the case in biological networks, where a protein subunit may form a part of several complexes, thus, we believe, showing the value of implementation of APAL for protein interaction and other biological networks will prove useful.

The intuitive and unambiguous nature of the threshold parameter allows users with knowledge about the nature of their network to redefine the community for a particular graph with ease. Additionally, the threshold parameter has more granularity, being a real value between $t = 0$ and $t = 1$, rather than the integer values required for CPM. One cannot use a parameter such as 3.5 for CPM but it is possible to adjust t as precise as possible.

As aforementioned, APAL is a compact algorithm compared to others. It can be seen in Algorithm 1 that APAL has no oscillations needing to be controlled by stopping conditions, is deterministic, and requires no starting seeds. Its one disadvantage is that its simplicity is indirectly proportional with its time complexity when compared to other algorithms. The polynomial running time is slightly inferior to COPRA and CPM.

APAL has many features with potential. For instance, it can be extended to make use of weighted and directed networks. It is also our intention to improve OGG for the generation of more realistic networks to benchmark existing and new overlapping community detection algorithms.

CRediT authorship contribution statement

Osman Doluca: Conceptualization, Methodology, Software, Validation, Resources, Data curation, Writing - original draft, Writing - review & editing. **Kaya Oğuz:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and feedback which have improved the manuscript significantly. We are also indebted to Simon Edward Mumford for his help in language editing and proofreading. No funding has been received for this research.

References

[1] B. Adamcsek, G. Palla, I.J. Farkas, I. Deényi, T. Vicsek, Cfinder: locating cliques and overlapping modules in biological networks, Bioinformatics 22 (2006) 1021–1023, https://doi.org/10.1093/bioinformatics/btl039.
[2] P. Aloy, R.B. Russell, Taking the mystery out of biological networks, EMBO Reports 5 (2004) 349–350, https://doi.org/10.1038/sj.embor.7400129.
[3] A. Amelio, C. Pizzuti, Overlapping community discovery methods: a survey, in: Social Networks: Analysis and Case Studies Lecture Notes in Social Networks, Springer, Vienna, 2014, pp. 105–125.
[4] S. Bandyopadhyay, G. Chowdhary, D. Sengupta, Focs: Fast overlapped community search, IEEE Transactions on Knowledge and Data Engineering 27 (2015) 2974–2985, https://doi.org/10.1109/TKDE.2015.2445775.
[5] E.R. Barnes, An algorithm for partitioning the nodes of a graph, SIAM Journal on Algebraic Discrete Methods 3 (1982) 541–550.
[6] J. Baumes, M.K. Goldberg, M.S. Krishnamoorthy, M. Magdon-Ismail, N. Preston, Finding communities by clustering a graph into overlapping subgraphs, IADIS AC 5 (2005) 97–104.
[7] L. Birdsey, C. Szabo, Y.M. Teo, Twitter knows: Understanding the emergence of topics in social networks, in: 2015 Winter Simulation Conference (WSC), 2015, pp. 4009–4020, https://doi.org/10.1109/WSC.2015.7408555.
[8] C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Communications of the ACM 16 (1973) 575–577, https://doi.org/10.1145/362342.362367.
[9] G. Chartrand, P. Zhang, A First Course in Graph Theory, Dover Publications, 2012.
[10] M.W. Cole, D.S. Bassett, J.D. Power, T.S. Braver, S.E. Petersen, Intrinsic and task-evoked network architectures of the human brain, Neuron 83 (2014) 238–251.
[11] M. Coscia, F. Giannotti, D. Pedreschi, A classification for community discovery methods in complex networks, Statistical Analysis and Data Mining: The ASA Data Science Journal 4 (2011) 512–546, https://doi.org/10.1002/sam.10133.
[12] T.S. Evans, R. Lambiotte, Line graphs, link partitions, and overlapping communities, Physical Review E 80 (2009) , https://doi.org/10.1103/PhysRevE.80.016105 016105.
[13] S. Fortunato, Community detection in graphs, Physics Reports 486 (2010) 75–174, https://doi.org/10.1016/j.physrep.2009.11.002.
[14] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, vol. 1, Springer Series in Statistics New York, NY, USA, 2001.
[15] J. Geryk, F. Slanina, Modules in the metabolic network of e.coli with regulatory interactions, International Journal of Data Mining and Bioinformatics 8 (2013) 188–202, https://doi.org/10.1504/IJDMB.2013.055500.
[16] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (2002) 7821–7826, https://doi.org/10.1073/pnas.122653799.
[17] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (2002) 7821–7826, https://doi.org/10.1073/pnas.122653799.
[18] S. Gregory, Finding overlapping communities in networks by label propagation, CoRR abs/0910.5516 (2009).
[19] M.B. Hutair, I. Kamel, Z.A. Agbari, Social community detection based on node distance and interest, in: 2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT), 2016, pp. 274–279.

[20] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, A. Baig, Community detection in networks: A multidisciplinary review, Journal of Network and Computer Applications 108 (2018) 87–111, https://doi.org/10.1016/j.jnca.2018.02.011.
[21] I. Jurisica, D. Wigle, Knowledge Discovery In Proteomics, CRC Press, 2006.
[22] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, The Bell System Technical Journal 49 (1970) 291–307.
[23] S. Kikugawa, K. Nishikata, K. Murakami, Y. Sato, M. Suzuki, M. Altaf-Ul-Amin, S. Kanaya, T. Imanishi, Pcdq: human protein complex database with quality index which summarizes different levels of evidences of protein complexes predicted from h-invitational protein-protein interactions integrative dataset, BMC Systems Biology 6 (2012) S7, https://doi.org/10.1186/1752-0509-6-S2-S7.
[24] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Physical Review E 80 (2009) , https://doi.org/10.1103/PhysRevE.80.016118 016118.
[25] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New Journal of Physics 11 (2009) 033015.
[26] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E 78 (2008) , https://doi.org/10.1103/PhysRevE.78.046110 046110.
[27] A. Lázár, D. Ábel, T. Vicsek, Modularity measure of networks with overlapping communities, EPL (Europhysics Letters) 90 (2010) 18001, https://doi.org/10.1209/0295-5075/90/18001.
[28] X. Ma, L. Gao, Biological network analysis: insights into structure and functions, Briefings in Functional Genomics 11 (2012) 434–442, https://doi.org/10.1093/bfgp/els045.
[29] M. Meilă, Comparing clusterings–an information based distance, Journal of Multivariate Analysis 98 (2007) 873–895.
[30] J.P. Miller, R.S. Lo, A. Ben-Hur, C. Desmarais, I. Stagljar, W.S. Noble, S. Fields, Large-scale identification of yeast integral membrane protein interactions, Proceedings of the National Academy of Sciences 102 (2005) 12123–12128, https://doi.org/10.1073/pnas.0505482102.
[31] M. Newman, The Structure and Function of Complex Networks, SIAM Review 45 (2003) 167–256.
[32] M.E.J. Newman, The structure of scientific collaboration networks, Proceedings of the National Academy of Sciences 98 (2001) 404–409, https://doi.org/10.1073/pnas.98.2.404.
[33] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2004) , https://doi.org/10.1103/PhysRevE.69.026113 026113.
[34] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in Neural Information Processing Systems, 2002, pp. 849–856. .
[35] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818, https://doi.org/10.1038/nature03607.
[36] P. Pons, M. Latapy, Computing communities in large networks using random walks, in: P. Yolum, T. Güngör, F. Gürgen, C. Özturan (Eds.), Computer and Information Sciences – ISCIS 200. Berlin, Heidelberg: Springer, Berlin Heidelberg, 2005, pp. 284–293. .
[37] S. Pu, J. Wong, B. Turner, E. Cho, S.J. Wodak, Up-to-date catalogues of yeast protein complexes, Nucleic Acids Research 37 (2009) 825–831, https://doi.org/10.1093/nar/gkn1005.
[38] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Physical Review E 76 (2007) , https://doi.org/10.1103/PhysRevE.76.036106 036106.
[39] Y. Su, B. Wang, X. Zhang, A seed-expanding method based on random walks for community detection in networks with ambiguous community structures, Scientific Reports 7 (2017) 1–10.
[40] K. Tarassov, V. Messier, C.R. Landry, S. Radinovic, M.M.S. Molina, I. Shames, Y. Malitskaya, J. Vogel, H. Bussey, S.W. Michnick, An in vivo map of the yeast protein interactome, Science 320 (2008) 1465–1470, https://doi.org/10.1126/science.1153878.
[41] K. Tarassov, V. Messier, C.R. Landry, S. Radinovic, M.M.S. Molina, I. Shames, Y. Malitskaya, J. Vogel, H. Bussey, S.W. Michnick, An in vivo map of the yeast protein interactome, Science 320 (2008) 1465–1470, https://doi.org/10.1126/science.1153878.
[42] M. Tasgin, A. Herdagdelen, H. Bingol, Community detection in complex networks using genetic algorithms, 2007. arXiv preprint arXiv:0711.0491. .
[43] S. Wagner, D. Wagner, Comparing Clusterings: An Overview, Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.
[44] C. Wang, W. Tang, B. Sun, J. Fang, Y. Wang, Review on community detection algorithms in social networks, in: 2015 IEEE International Conference on Progress in Informatics and Computing (PIC), 2015, pp. 551–555, https://doi.org/10.1109/PIC.2015.7489908.
[45] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using seed set expansion, Proceedings of the 22nd ACM International Conference on Information & Knowledge Management CIKM, '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 2099–2108, https://doi.org/10.1145/2505515.2505535.
[46] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using neighborhood-inflated seed expansion, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 1272–1284.
[47] J. Xie, S. Kelley, B.K. Szymanski, Overlapping Community Detection in Networks: The State-of-the-art and Comparative Study, ACM Computational Survey 45 (2013) 43:1–43:35, https://doi.org/10.1145/2501654.2501657.
[48] H. Yu, P. Braun, M.A. Yıldırım, I. Lemmens, K. Venkatesan, J. Sahalie, T. Hirozane-Kishikawa, F. Gebreab, N. Li, N. Simonis, T. Hao, J.-F. Rual, A. Dricot, A. Vazquez, R.R. Murray, C. Simon, L. Tardivo, S. Tam, N. Svrzikapa, C. Fan, A.-S. de Smet, A. Motyl, M.E. Hudson, J. Park, X. Xin, M.E. Cusick, T. Moore, C. Boone, M. Snyder, F.P. Roth, A.-L. Barabási, J. Tavernier, D.E. Hill, M. Vidal, High-quality binary protein interaction map of the yeast interactome network, Science 322 (2008) 104–110, https://doi.org/10.1126/science.1158684.
[49] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A: Statistical Mechanics and its Applications 374 (2007) 483–490, https://doi.org/10.1016/j.physa.2006.07.023.