A vertex-similarity clustering algorithm for community detection

Antonio Maria Fiscarelli a,b, Matthias R. Brust b, Grégoire Danoyc and Pascal Bouvry b,c

aC2DH, University of Luxembourg, Luxembourg; bSnT, University of Luxembourg, Luxembourg; cFSTC-CSC-ILIAS, University of Luxembourg, Luxembourg

ABSTRACT
Communities in networks are considered to be groups of vertices with higher probability of being connected to each other than to members of other groups. Community detection, then, is a method to identify these communities based on the higher intra-cluster and lower inter-cluster connectivity. Depending on the type and size of the network, detecting such communities can be a challenging task. The method we propose is a degenerate agglomerative hierarchical clustering algorithm (DAHCA) that makes use of the reachability matrix to find a community structure in networks. We tested DAHCA using common classes of network benchmarks as well as real-world networks and compared it to state-of-the-art community detection algorithms. Our results show that it can effectively identify hierarchies of communities, and outperform some of the algorithms for more complex networks. In particular, when communities start to exhibit very low intra-community connectivity, it is the only method that is still able to identify communities.

ARTICLE HISTORY
Received 21 June 2018
Accepted 27 October 2019

KEYWORDS
Community detection; Graph clustering; Social network analysis

1. Introduction

Many complex systems such as social networks (Wasserman & Faust, 1994), the worldwide web (Albert, Jeong, & Barabási, 1999) and biological networks (Jeong, Tombor, Albert, Oltvai, & Barabási, 2000) can be represented using graphs. One of their many properties is the organization into communities. Often, different communities merge and form a hierarchical structure. Also, they differ in size and vertices show different degrees of connectivity. Detecting the community structure of a network can provide access to additional knowledge about the dynamics of the network and its characteristics. For that reason community detection has stimulated many researches in network and computer science.

To tackle this problem, we here propose a novel degenerate agglomerative hierarchical clustering algorithm (DAHCA). DAHCA makes use of the reachability matrix which contains
information about paths connecting vertices. Vertices inside the same community will be more likely to have common paths, so a similarity measure can be used to identify similar vertices and group them into hierarchies of communities.

In this article we empirically demonstrate the better community detection performance and scalability of DAHCA compared to state-of-the-art algorithms on well-known benchmarks from the literature of increasing complexity and size: the Girvan–Newman (Girvan & Newman, 2002), the Lancichinetti–Fortunato–Radicchi and a set of real-world networks that include several social networks such as the Zachary karate club network (Zachary, 1977).

The remainder of this article is organized as follows. The next section proposes an analysis of the related work on community detection algorithms. Section 3 then introduces the main contribution of the article, i.e. a description of the DAHCA, including its complexity analysis. Then experimental results on both artificial and real-world networks are presented in Section 4. Finally, Section 5 provides our conclusions and perspectives for future research.

2. Related work

This section presents some of the existing community detection algorithms. Communities in networks are considered to be groups of vertices with higher probability of being connected to each other (high intra-cluster connectivity) than to members of other groups (low inter-cluster connectivity). Community detection is a method to identify these communities which can be a challenging task depending on the type and size of the network. For that reason, many different methods have been proposed over the last years and contributions came from disciplines such as computer science, applied mathematics, physics, biology and economics.

However, there is no best algorithm. Some algorithms simply perform better or are faster for different types of networks or different applications. Following in this section we present some of the prominent community detection algorithms.

2.1. Community detection algorithms

It is possible to define a taxonomy of community detection algorithms. They can be divided into global and local algorithms. Global algorithms use information about the entire network, while local algorithms use only partial knowledge, for example neighbourhood information.

Hierarchical methods can find hierarchies of communities, where each community is composed of several smaller ones. In particular, agglomerative methods use a bottom-up approach by assigning a different community to each vertex and iteratively merging them together, while divisive methods use a top-down approach by assigning the same community to each vertex and iteratively splitting it.

Another division that can be made is based on the way these algorithms work. Modularity optimization methods make use of the modularity measure, which is linked to the percentage of edges connecting vertices inside the same community. In particular, greedy algorithms use heuristics that try to rearrange nodes into communities by, for example, moving nodes from one community to another when it brings an improvement.
in modularity. Spectral methods make use of the eigenvectors of the modularity matrix to optimize the modularity measure. In particular, the eigenvector corresponding to the second smallest eigenvalue contains information about the community structure of the network. Methods based on random walks make use of the transition matrix: random walkers starts from a certain node and they randomly move to a neighbouring node according to a transition probability. The transition probabilities of each vertex will then gradually converge to stable values, which will be high for edges connecting nodes inside the same community.

The algorithms used for a comparison purpose in this paper are the following: the method proposed by Newman and Girvan (2004) is a hierarchical divisive algorithm that extends the definition of betweenness centrality to edges. Edges connecting communities will have a high edge betweenness and removing them will enhance the community structure of the network (BETW). On the other hand, Clauset, Newman, and Moore (2004) propose a greedy algorithm that makes use of the modularity measure to define communities that have many edges within them and few between them (GREEDY). Furthermore, Raghavan, Albert, and Kumara (2007) use a local technique based on the majority rule to assign vertices to clusters (LAB PROP). The method described in Pons and Latapy (2005) uses random walks to define communities: generally, random walkers tend to stay more in the same community (TRAP). Rosvall and Bergstrom (2008) approach the problem using an information theoretic point of view to discover communities by using the probability flow of random walks (INFOMAP). Finally, the method proposed by Newman (2006) is a spectral method based on the eigenspectrum of the modularity matrix that maximizes the modularity measure (EIGEN).

Our method is different from the others described because it uses a similarity measure, based on the structure of the network, to group vertices. This implies that it is also able to group vertices in the same community when they exhibit very low intra-cluster connectivity.

3. DAHCA: degenerate agglomerative hierarchical clustering algorithm

DAHCA makes use of the reachability matrix which contains information about the total number of paths between vertices. This was initially proposed in Katz (1953) and was defined as

\[ W = \sum_{l=0}^{\infty} (\alpha A)^l = (I - \alpha A)^{-1} \]

where \( A \) is the adjacency matrix, \( I \) is the identity matrix and \( l \) is an integer value that indicates the length of paths considered. The parameter \( \alpha \) is tuned so that longer paths contribute less and the sum converges. In our case we define the reachability matrix as

\[ A^l = \sum_{i=1}^{l} A^i \]

where every entry of \( A^l \) is identified as \( a_{ij}^l \) and it represents the exact number of 1-paths, 2-paths, ..., \( l \)-paths connecting vertex \( i \) with vertex \( j \). We decided to use up to
three-length paths because longer paths would connect vertices in different communities too easily. Numerical tests also confirmed that it performs best for \( l = 3 \). Every vertex is then characterized by its relative row entry in the reachability matrix: vertices belonging to the same community are more likely to have common paths. DAHCA starts by assigning a different cluster to each vertex. It then selects a vertex and computes the Euclidean distances between it and all its neighbours (non-zero entry elements) and assigns it the cluster of the most similar vertex. Ties are broken selecting the vertex that has the most common neighbours. The process iterates until all vertices have been assigned to a cluster. Next, the algorithm merges vertices belonging to the same cluster in a new vertex, after which the reachability matrix is recomputed as follows:

\[
a_{ij}^l = \frac{1}{|C_i| \sum_{k \in C_i} \frac{1}{|C_j| \sum_{h \in C_j}} a_{kh}}
\]

where \( C_i \) and \( C_j \) are the new clusters obtained and \( a_{kh} \) is an element of the adjacency matrix. The new reachability matrix will have number of rows and columns equal to the number of clusters found. Figure 1 shows one iteration of DAHCA. At each step a new cluster assignment will be found. The process iterates until change no longer occurs. This can be seen as a degenerate agglomerative hierarchical clustering: each vertex starts with its own cluster and at each iteration clusters are merged until merging is no longer possible. It is different from a classical agglomerative clustering because more than two vertices can be merged together in one iteration and it does not always end with a single cluster including all vertices.

![Figure 1](image.url)

*Figure 1.* This figure shows one iteration of DAHCA. A different cluster is initially assigned to each vertex. Vertices are then iteratively selected and they will be assigned the cluster of the closest vertex according to Euclidean distance. Nodes belonging to the same cluster are then merged in a single vertex.
Algorithm 1: DAHCA

**Input:** Graph $G(V,E)$

**Output:** Communities $C$

1. $A^l \leftarrow \sum_{i=1}^{l} A^i$
2. $A^l_{old} \leftarrow A^l$
3. $it \leftarrow 1$
4. **while** $A^l \neq A^l_{old}$ **do**
5. $A^l_{old} \leftarrow A^l$
6. **for** $i \in 1, \ldots, \text{nrows}(A^l)$ **do**
7. $C_i^l \leftarrow i$ //Assign unique cluster to vertices
8. **end**
9. **for** $i \in 1, \ldots, \text{nrows}(A^l)$ **do**
10. **for** $j \in 1, \ldots, \text{ncols}(A^l)$ **do**
11. $\text{Dist}_j^l \leftarrow \text{Euclidean}(a_i^l, a_j^l)$
12. **end**
13. $C_i^l \leftarrow \text{argmax}_j \{\text{Dist}_j^l\}$
14. **end**
15. $A^l \leftarrow \text{RecomputeMatrix}(A^l, C^l)$
16. $it \leftarrow it + 1$
17. **end**

### 3.1. Complexity

The computational complexity of a community detection algorithm is crucial, especially for large graphs. Given a graph $G = \{V,E\}$ where $V$ is the set of vertices and $E$ is the set of edges, the complexity analysis of DAHCA can be assessed in this way:

- the reachability matrix can be computed in time $O(|V|^3)$ (where $l = 3$ in our case).
- the clustering process can be computed in time $O(K \times |V|) \simeq O(|E|)$.
- the merging process can be computed in time $O(K \times |V|) \simeq O(|E|)$

Notice that the size of $V$ only corresponds to the actual number of vertices during the first iteration. After that, they are merged and the size of $V$ decreases significantly. The overall complexity of DAHCA is then $O(t(|V|^3 + K \times |V| + K \times |V|)) \simeq O(|V|^3)$ where $t$ is the number of iterations. Table 1 shows the time complexity for all algorithms discussed in this work.

| Algorithm | Complexity       |
|-----------|-----------------|
| BETW      | $O(|V||E|^2)$   |
| GREEDY    | $O(|V| \log^2 |V|)$ |
| LAB PROP  | $O(|E|)$        |
| TRAP      | $O(|E||V|^2)$   |
| INFOMAP   | $O(|E|)$        |
| EIGEN     | $O(|E| + |N|)$  |
| DAHCA     | $O(|V|^3)$      |

Table 1. Computational complexity of DAHCA compared to other state-of-the-art algorithms.
4. Experiments

We have run several experiments on different types of network benchmarks to evaluate DAHCA’s performance. First, we have investigated how effectively our method can detect nested communities using the benchmark proposed by Bagnoli, Massaro, and Guazzini (2012). Next, we have compared our method to state-of-the-art algorithms on the Girvan–Newman benchmark (Girvan & Newman, 2002), as well as the Lancichinetti–Fortunato–Radicchi benchmark, a more complex network model that better reflects real-world networks. Finally, our method has been tested on several real-world networks such as the Zachary karate club network (Zachary, 1977), a well-known real-world network used as a benchmark for community detection algorithms.

4.1. Performance measures

The most used metric (Danon, Diaz-Guilera, Duch, & Arenas, 2005; Yang, Algesheimer, & Tessone, 2016) to evaluate community detection algorithms is the normalized mutual information (NMI): it measures the agreement between communities and clusters found by a community detection algorithm (scikit learn, 2017). NMI = 1 corresponds to perfect assignments, while NMI = 0 corresponds to completely independent assignments. The Adjusted Random Index (ARI) measures the similarity of the assignments (scikit learn, 2017). It ranges from −1 to 1, where ARI = 1 corresponds to perfect assignments, ARI values near 0 correspond to bad assignments and negative values of ARI correspond to independent assignments. Completeness (COMP) measures how vertices of a community are assigned to the same cluster, while homogeneity (HOMOG) measures how every cluster contains only vertices of the same community. When all vertices are assigned to the same cluster HOMOG = 0 and COMP = 1, whereas if each vertex is assigned to a different cluster HOMOG = 1 and COMP = 0.

For real-world networks, most of the times, real communities are not known, therefore the discussed metrics cannot be computed. For this reason, the modularity measure (Q) has been used Newman and Girvan (2004): it considers the fraction of edges connecting vertices inside the same community. This quantity will be significantly larger when computed on a network that exhibits a community structure rather than a random graph of the same size and average vertex degree. It ranges from −1 to 1, where Q is positive when the fraction of edges within communities is larger than the expected one.

4.2. Artificial networks

4.2.1. Benchmarks

First, we have investigated how effectively DAHCA can detect communities and the emergence of nested communities. To do so, we have used a similar benchmark (BMG) as the one described in Bagnoli et al. (2012). Networks in the BMG benchmark have N number of vertices, are divided in G groups and each group is divided into C communities. Vertex connectivity is set to K, this means that each vertex will be connected to exactly K other vertices in the same community. Benchmark graphs have been generated with N = 120, M = 3, C = 2 and K = 5. Every edge is then relinked with probability p_r. If so, the vertex is connected to another vertex in the same community with probability p_c, in the same
group with probability \((1 - p_c)p_g\) or to any vertex in the network with probability \((1 - p_c)(1 - p_g)\). Edges have been relinked with probability \(p_r = 1.0\) and \(p_g = 0.7\), while \(p_c\) was dynamically changed to simulate the emergence of communities (notice that this setting is slightly different from the one presented in Bagnoli et al. (2012)). For \(p_c = 0\) there is no community structure and only the groups are defined, while for \(p_c = 1\) the community structure emerges identifiably.

We also evaluated DAHCA on the Girvan–Newman (GN) benchmark (Girvan & Newman, 2002) and compared it to state-of-the-art algorithms used for community detection. All the algorithms are available in the igraph package (Csardi & Nepusz, 2006). Networks in the GN benchmark have \(N\) vertices that are assigned to \(C\) equally sized communities. Each vertex has a fixed average degree \(z\). A mixing parameter \(\mu\) controls the portion of intra-community edges. For \(\mu = 0\), communities are completely isolated, for \(\mu = 0.5\) vertices will be equally connected to vertices inside and outside their community, while for \(\mu = 1\) vertices inside the same communities are not connected at all. Benchmark graphs have been generated with \(N = 128\), \(C = 4\) and \(z = 16\), while \(\mu\) was dynamically changed.

One drawback of the GN benchmark is that the number of communities and the average degrees are fixed. Thus, it is not an appropriate representation of real-world networks, where these quantities vary. Lancichinetti, Fortunato, and Radicchi (2008) created a benchmark that better reflects the nature of real-world networks: degree distribution \(k\) and number of communities \(C\) are drawn from a power-law distribution. The mixing parameter \(\mu\) is still defined as previously explained. Setting the exponents of the distributions, together with average and maximum values, allows to generate networks with different characteristics. Two different benchmark graphs have been generated: the first one (LFR1) with \(N = 1000\), \(C.min = 10\) and \(C.max = 50\). The second one (LFR2), generating larger communities, with \(N = 1000\), \(C.min = 20\) and \(C.max = 100\). For both benchmarks \(K.avg = 20\) and \(K.max = 50\), while \(\mu\) was dynamically changed. Table 2 shows an overview of all these benchmark with their characteristics.

Lastly, we used the LFR benchmark to generate networks with different sizes, while the other characteristics remained the same, in order to verify how network size affects DAHCA’s performance. In this case \(N \in \{200, 500, 1000, 2000, 3000\}\), \(C.min = 0.02 \times N\), \(C.max = 0.1 \times N\), \(K.avg = 0.02 \times N\) and \(K.max = 0.05 \times N\), while \(\mu\) was dynamically changed. An overview of this benchmark is shown in Table 3.

### 4.2.2. Results

The results for the BMG benchmark are shown in Figure 2. For low values of \(p_c\) DAHCA is able to identify the correct number of groups, while for higher values it is able to identify the correct number of groups as well as the correct number of communities. This shows that DAHCA can effectively identify nested communities.

| #Nodes | Average degree | Directed | Weighted | #Communities | Subcommunities |
|--------|----------------|----------|----------|--------------|----------------|
| BMG    | 120            | 5        | No       | No           | Yes            |
| GN     | 128            | 16       | No       | No           | 4              | No             |
| LFR1   | 1000           | 20       | No       | No           | [10–50]        | No             |
| LFR2   | 1000           | 20       | No       | No           | [20–100]       | No             |
The results for the GN benchmark, instead, are shown in Figure 3. The most used metric for community detection is the normalized mutual information (NMI), but it does not necessarily return zero when the assignment is very poor. This happens when an algorithm assigns each vertex to a different cluster or all vertices to the same cluster. Therefore we decided to compute, at least for this benchmark, completeness and homogeneity to identify when an algorithm returns these naive assignments. For example, the INFOMAP algorithm scores NMI = 0 for high values of $\mu$ (see Figure 3): one cannot say whether it is due to a very bad assignment, a random assignment or just a naive assignment. Using homogeneity and completeness, it scores HOMOG = 0 and COMP = 1 and clearly assigns every vertex to the same cluster. We also decided to use the ARI because, unlike the NMI, it is always independent of the network size and number of communities.

Looking at the NMI, for low values of $\mu$ DAHCA does not perform perfectly, unlike some of the other algorithms. However, for $\mu \in [0.3, 0.6]$ it outperforms GREEDY, INFOMAP, LABPROP and EIGEN. For higher values of $\mu$ it outperforms all other algorithms but BETW. In general, DAHCA obtains better results for higher values of $\mu$ also for the other metrics. Furthermore, it exhibits an interesting behaviour: for $\mu \in [0.75, 1.0]$ there is an increase in performance. One would assume that performance should decrease for $\mu \geq 0.5$ because communities become less evident, but as proved by Lancichinetti and Fortunato (2014) they are actually evident for $\mu$ up to 0.75. Over that range, the number of inter-community edges becomes much higher than the number of intra-community edges (an ‘anti-

### Table 3. Networks of different sizes generated using the LFR benchmark.

| $N$  | $K_{avg}$ | $K_{max}$ | $C_{exp}$ | $C_{exp}$ | $C_{min}$ | $C_{max}$ |
|------|-----------|-----------|-----------|-----------|-----------|-----------|
| 200  | 4         | 10        | -2        | -1        | 4         | 20        |
| 500  | 10        | 25        | -2        | -1        | 10        | 50        |
| 1000 | 20        | 50        | -2        | -1        | 20        | 100       |
| 2000 | 40        | 100       | -2        | -1        | 40        | 200       |
| 3000 | 60        | 150       | -2        | -1        | 60        | 300       |

Figure 2. Clustering over three consecutive iterations. Experiments for $N = 120, M = 3, C = 2, K = 5, p^r = 1.0, p^g = 0.7$. The probability $p^r$ can be found on the x-axis and the number of clusters identified on the y-axis. Each barplot shows the results obtained for a specific $p^r$ value and each single column represents the number of clusters identified at a certain iteration. The two horizontal lines correspond to the total number of groups (3) and the total number of communities (6) in the networks. Experiments have been run 200 times and results averaged.
community’ structure), with $\mu = 1.0$ being the point where there are no more edges inside communities. DAHCA is able to detect anti-communities which explains why DAHCA’s performance increases.

The results for the benchmark LFR1 and LFR2 are shown in Figure 4. Looking at the NMI, for very low values of $\mu$ DAHCA still does not perform perfectly. For $\mu \in [0.2, 0.55]$ it only outperforms GREEDY. From $\mu = 0.6$ it also outperforms LAB PROP. From $\mu = 0.7$ it outperforms INFOMAP and TRAP. DAHCA also outperforms BETW for values of $\mu \in [0.6, 0.7]$, but it gets outperformed again for higher values. Overall, DAHCA obtains the best results for higher values of $\mu$, when communities are not well defined. Results are similar for the benchmark LFR2. All algorithms exhibit a similar behaviour, obtaining slightly better results for lower values of $\mu$ and slightly worse for higher values.

The results for the LFR benchmark with different network sizes are shown in Figure 5. DAHCA shows a similar behaviour for different sizes of networks. The only exception is...

---

**Figure 3.** Experiments on the GN benchmark for $N = 128$, $C = 4$, $K = 16$. The mixing parameter $\mu$ can be found on the $x$-axis and NMI, ARI, completeness and homogeneity on the $y$-axis. Experiments have been run 50 times and results averaged.
$N=200$, reason being the fact that some communities are very small and nodes have low degree, making less accurate the similarity measure used to assign clusters to vertices. This phenomenon is also known as the resolution limit. It is interesting to notice that DAHCA obtains better performance for bigger networks, probably for the same reason.

Figure 4. Experiments on LFR1 and LFR2 benchmarks for $N = 1000$, $C_{\text{min}} = \{10, 20\}$, $C_{\text{max}} = \{50, 100\}$, $K_{\text{avg}} = 20$ and $K_{\text{max}} = 50$. The mixing parameter $\mu$ can be found on the $x$-axis and the NMI on the $y$-axis. Experiments have been run 20 times and results averaged.

Figure 5. Experiments on the LFR benchmark for $N \in \{200, 500, 1000, 2000, 3000\}$, $C_{\text{min}} = 0.02 * N$, $C_{\text{max}} = 0.1 * N$, $K_{\text{avg}} = 0.02 * N$ and $K_{\text{max}} = 0.05 * N$. The mixing parameter $\mu$ can be found on the $x$-axis and the NMI on the $y$-axis. Experiments have been run 20 times and results averaged.
4.3. Real-world networks

4.3.1. Benchmarks

We also evaluated DAHCA on some real-world networks. In particular, we have decided to focus on social networks. Since the Zachary's karate club network (Zachary, 1977) is the only one for which we know the real communities and they are reliable, it is treated more in detail. The networks used are the following:

- **zachary** is a friendship network of a karate club. It consists of 34 vertices representing members of the club and 78 edges representing ties between members. Edges are undirected and weighted.
- **UKfaculty** is a friendship network of a faculty of a UK university. It consists of 81 vertices representing people and 817 edges representing friendships. Edges are directed and unweighted.
- **mail** is an email dataset. It consists of 184 vertices representing staff's email addresses and 2116 edges representing emails sent. Edges are directed and unweighted.
- **dolphins** is a social network of bottlenose dolphins. It consists of 62 vertices representing individuals from a community of bottlenose dolphins and 159 edges representing frequent associations. Edges are directed and unweighted.
- **jazz** is a collaboration network between jazz musicians. It consists of 198 vertices representing jazz musicians and 2742 edges indicating that two musicians have played in the same band. Edges are directed and unweighted.

An overview of all the networks is available with Table 4. Since some of the algorithms used in this paper only work with undirected graphs, we made all edges for these networks undirected. We also collapsed multiple edges and removed all self-loops.

4.3.2. Results

Zachary's karate club network is known to be a vastly used benchmark for community detection algorithms. Every vertex represents members of the club, with 1 and 34 being the administrator and the director (the leaders of the two communities) while edges represent ties between members. The objective is to find the two groups of people into which the karate club split after an argument between the two leaders. Since we know from a reliable source the real communities, we can treat the network more into details and use all the metrics previously introduced.

Results for all algorithms, including DAHCA, are presented in Figure 6. Nodes belonging to the same communities share the same colour. The algorithms obtain very different results, especially for the number of communities found. For karate, all algorithms are able to identify vertices 1 and 34 as community leaders and assign them to different communities.

|          | #Nodes | #Edges | Directed | Weighted |
|----------|--------|--------|----------|----------|
| karate   | 34     | 78     | No       | Yes      |
| UKfaculty| 81     | 817    | Yes      | No       |
| mail     | 184    | 2116   | Yes      | No       |
| dolphins | 62     | 159    | Yes      | No       |
| jazz     | 198    | 2742   | Yes      | No       |
Figure 6. Communities found by BETW, GREEDY, LAB, TRAP, INFOMAP and EIGEN, together with DAHCA, on Zachary’s karate club network.

communities. Only DAHCA and LAB PROP are able to identify the core sets of vertices \{1,2,3\} and \{33,44\} as nodes with higher connectivity (Girvan & Newman, 2002) and assign them to different communities. To complete the analysis, we also computed normalized mutual information, adjusted randomized index, homogeneity and completeness. Table 5 shows the numerical results of the different algorithms on the Zachary network. In this case, DAHCA is able to outperform only TRAP.

For some of the other networks we do not know the real communities or they are not reliable. This means that the metrics used so far cannot be used. We use instead the modularity measure (Newman & Girvan, 2004) that does not need the real community assignment and is only based on the structure of the network. It is based on the fraction of edges connecting vertices inside the same community: this quantity will be significantly larger when computed on a network that exhibits a community structure rather than a
random graph of same size and average vertex degree. Thus, higher values of modularity imply a better cluster assignment. Results for all algorithms, including DAHCA, on all the networks are showed in Figure 7. Considering the average modularity, DAHCA performs superior on karate, being very close to the algorithm that performs best. On the other networks it does not perform as good.

5. Conclusions and future work

In this paper we have proposed DAHCA, a novel degenerate agglomerative hierarchical clustering algorithm that makes use of the reachability matrix to detect community structures in networks and runs in $O(|V|^3)$. DAHCA was evaluated on different community detection benchmarks and settings. First, it has been tested on the benchmark used by Bagnoli et al. (2012) where networks are organized into groups and each group is organized into communities. We have shown that DAHCA is able to identify both group and community structure. Next we have compared DAHCA to state-of-the-art algorithms on the Girvan–Newman benchmark.

Table 5. Metrics for the different algorithms on the Zachary karate club network.

|       | BETW | GREEDY | LAB PROP | TRAP | INFOMAP | EIGEN | DAHCA |
|-------|------|--------|----------|------|---------|-------|-------|
| NMI   | 0.579| 0.692  | 0.548    | 0.504| 0.699   | 0.677 | 0.512 |
| ARI   | 0.469| 0.680  | 0.467    | 0.333| 0.702   | 0.512 | 0.452 |
| COMPL | 0.431| 0.577  | 0.428    | 0.364| 0.593   | 0.512 | 0.387 |
| HOMOG | 0.885| 0.866  | 0.763    | 0.822| 0.854   | 1     | 0.756 |

Figure 7. Experiments on zachary, UKfaculty, mail, dolphins and jazz networks for all algorithms. From left to right, BETW, EIGEN, GREEDY, INFOMAP, LAB, DAHCA and TRAP. DAHCA is also highlighted. The networks can be found on the x-axis, average modularity can be found on the y-axis. Each barplot represents the results obtained for all algorithms on a specific network. Experiments have been run 50 times and results averaged.
(Girvan & Newman, 2002) and discovered that, even if it does not show optimal results on the simplest networks, it is able to outperform most of the other algorithms for the more complex ones. Then, we tested our method using the Lancichinetti-Fortunato-Radicchi benchmark (Lancichinetti et al., 2008), applying two different settings, where DAHCA is able to outperform some of the algorithms for higher values of $\mu$. We also used the LFR benchmark to generate networks with different sizes in order to verify how network size affects DAHCA’s performance and showed that it remains consistent with network size. Finally, we have selected some real-world networks and compared DAHCA with the other algorithms using the modularity measure. In particular, we have shown the results obtained on the Zachary’s karate club network and discovered that DAHCA is able to assign the two community representatives and core members to different communities.

The major drawback of this method is the computation of the reachability matrix, since it is time consuming and requires the whole network to be accessible beforehand. The next step can be to find a decentralized way to characterize vertices: for example, two vertices can be considered similar if they have many common neighbours and the similarity measure can be redefined according to this definition.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This work was supported by Fonds National de la Recherche Luxembourg [0929115] and is partially funded by the joint research programme UL/SnT-ILNAS on Digital Trust for Smart-ICT.

**ORCID**

Antonio Maria Fiscarelli  [http://orcid.org/0000-0003-0287-4388](http://orcid.org/0000-0003-0287-4388)

Matthias R. Brust  [http://orcid.org/0000-0001-8155-0626](http://orcid.org/0000-0001-8155-0626)

**References**

Albert, R., Jeong, H., & Barabási, A. L. (1999). Internet: Diameter of the world-wide web. *Nature*, 401 (6749), 130–131.

Bagnoli, F., Massaro, E., & Guazzini, A. (2012). Community-detection cellular automata with local and long-range connectivity. *Cellular Automata*, 7495, 204–213.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6), 066111.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal Complex Systems*, 1695. [http://igraph.org](http://igraph.org)

Danon, L., Diaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09), P09008.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821–7826.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., & Barabási, A. L. (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804), 651–654.

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, 18(1), 39–43.
Lancichinetti, A., & Fortunato, S. (2014). Erratum: Community detection algorithms: A comparative analysis [Phys. rev. E 80, 056117 (2009)]. Physical Review E, 89(4), 049902.
Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. Physical Review E, 78(4), 046110.
Newman, M. E. (2006). Finding community structure in networks using the eigenvectors of matrices. Physical Review E, 74(3), 036104.
Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. Physical Review E, 69(2), 026113.
Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. ISCIS, 3733, 284–293.
Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76(3), 036106.
Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences, 105(4), 1118–1123.
scikit learn (2017). User guide-clustering. Retrieved from http://scikit-learn.org/stable/modules/clustering.html
Wasserman, S., & Faust, K. (1994). Social network analysis: Methods and applications. Vol. 8. Cambridge: Cambridge University Press.
Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. Scientific Reports, 6, 30750.
Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. Journal of Anthropological Research, 33(4), 452–473.