Gera and London *Applied Network Science* 2023, **8**(1):51
https://doi.org/10.1007/s41109-023-00575-2

Applied Network Science

RESEARCH **Open Access**

Detecting and generating overlapping nested communities

Imre Gera[1*] and András London[1,2]

*Correspondence:
gerai@inf.u-szeged.hu

[1] Department of Computational Optimization, University of Szeged, Árpád tér 2, Szeged 6720, Hungary
[2] Department of Operations Research and Mathematical Economics, Poznań University of Economics and Business, 61-875 Poznań, Poland

Abstract

Nestedness has been observed in a variety of networks but has been primarily viewed in the context of bipartite networks. Numerous metrics quantify nestedness and some clustering methods identify fully nested parts of graphs, but all with similar limitations. Clustering approaches also fail to uncover the overlap between fully nested subgraphs, as they assign vertices to a single group only. In this paper, we look at the nestedness of a network through an auxiliary graph, in which a directed edge represents a nested relationship between the two corresponding vertices of the network. We present an algorithm that recovers this so-called community graph, and finds the overlapping fully nested subgraphs of a network. We also introduce an algorithm for generating graphs with such nested structure, given by a community graph. This algorithm can be used to test a nested community detection algorithm of this kind, and potentially to evaluate different metrics of nestedness as well. Finally, we evaluate our nested community detection algorithm on a large variety of networks, including bipartite and non-bipartite ones, too. We derive a new metric from the community graph to quantify the nestedness of both bipartite and non-bipartite networks.

Keywords: Nestedness, Community detection, Network science

Introduction

Identifying clusters or communities of nodes in graphs is an important problem in graph-based data mining and network science. The standard methods try to achieve lots of edges within clusters (or communities) and only a few between distinct clusters (Schaeffer 2007), imitating concepts of data clustering in statistics and machine learning (Xu and Tian 2015). In general, this approach works well and provides meaningful clusters for social networks due to some of their widely observed common properties. For instance, the number of triangles in a social network is much larger than in a random graph with similar edge density, they often show heterogeneous degree distribution and have small diameters (McGlohon et al. 2011). Taking these properties into account helps to find the dense parts of the network. Many algorithms rely on maximizing the modularity function, which measures the quality of a given clustering (Newman and Girvan 2004), but there are lots of different approaches. While for clustering, where each node is assigned to exactly one cluster, both bottom-up and top-down type algorithms have been proposed, for community detection, where a node can be a member

of several communities, mostly bottom-up algorithms are used, i.e., smaller initial communities are expanded with new nodes during the community detection process (Bóta et al. 2010).

On the other hand, some social networks and especially technological or transaction networks generally contain fewer triangles and often have tree-like structures (Adcock et al. 2013). Therefore, trying to find disjoint dense parts is inadequate in principle. Moreover, certain bipartite networks, such as pollination networks of plant species and their pollinators or trade networks of countries and their exported/imported goods, show the presence of special structures such as nestedness (Bastolla et al. 2009; Mariani et al. 2019; Uzzi 1996; Wright et al. 1997). That is, the nodes of each side of the bipartite network can be ordered in such a way that the neighborhood of any lower-ranked node contains the neighborhood of any higher-ranked node. Ecological networks often display a nested structure in which specialists species (refer to low degree nodes considering the species' interaction network) interact with generalists (i.e., high degree nodes) species, while generalists interact with each other and with specialists, too (Bascompte 2010). The ecological concept of nestedness was published first by Darlington (1943), and it was formally defined by Atmar & Patterson utilizing graph theoretical concepts (Patterson and Atmar 1986). In the field of economics, the bipartite networks of industrial firms and locations also show a high level of nestedness (Bustos et al. 2012; Saavedra et al. 2009). At the macroeconomic level, world trade can be described by a bipartite graph, where nodes represent either countries or products, and weighted edges between a country and a product represent the ratio related to the total amount of the product imported or exported. Like other economic networks, the world trade network is also highly nested (Ermann and Shepelyansky 2013) with the coexistence of global and regional dynamics in terms of network communities within it Zhu et al. (2014).

Extending the concept of nestedness to unipartite (non-bipartite) networks can be done in various ways, see e.g., Chapter 2 of Mariani et al. (2019) and London et al. (2022). Since perfect nestedness is rarely observed in real-world networks, several metrics have been proposed to quantify the level of nestedness (Payrató-Borràs et al. 2020; Ulrich et al. 2009). In this paper we do not aim to review all the relevant literature in detail, we only refer to the surveys (Csermely et al. 2013; Mariani et al. 2019; Ulrich et al. 2009).

The problem of identifying perfectly nested parts (i.e., nested subgraphs) of a network has received much less attention in the literature, mostly in the context of image processing only. Junttila and Kaski call a binary matrix (that is, a matrix whose entries are either zero or one) fully nested if its rows and columns can be reordered such that the ones are in an echelon form (Junttila and Kaski 2011). They define a binary matrix A fully k-nested if its columns can be partitioned into k pairwise disjoint submatrices, called blocks, each of which is fully nested. Given a matrix A, a natural optimization problem is to find the smallest k such that A is fully k-nested and also to provide a partitioning into k fully-nested parts. The problem can be solved in polynomial time. Note that any $m \times n$ binary matrix can be considered as the incidence matrix of a bipartite network with m and n nodes on its respective sides.

Extending the above definition to non-bipartite graphs can be done in several ways, but much less known about the problem's complexity. For instance, for a graph G and

a fixed bipartite graph H, London, Pluhár and Martin defined the concept of *induced H-avoiding coloring* (London et al. 2022), meaning that the union of any two color classes spans an induced H-free graph, and defined $\chi_H(G)$ as the minimum number of colors in an induced H-avoiding coloring of G. In the case of $H = 2K_2$ (a graph of four vertices with two non-adjacent edges) this coloring realizes a partitioning of G to bipartite, fully-nested clusters. Although determining $\chi_H(G)$ is NP-hard, their approach and those we present in this paper are both applicable to general graphs.

Here we present a novel method that identifies overlapping nested subgraphs and represents them as paths of a directed graph we call a community graph. We also introduce a method that generates bipartite graphs with (any) ground-truth overlapping nested structure, making it possible to generate example nested graphs and test nested community detection algorithms. Since our algorithm detects nestedness in non-bipartite graphs, too, in order to be able to quantify nestedness in any graph, we derive a new metric from the output of our algorithm called *vertex presence*. To measure the "generalist-ness" of a vertex, we derive another metric called *vertex position*.

The rest of the paper is organized as follows. First, we introduce the core definitions we are going to use throughout the paper. In the section "Nestedness and community detection" we introduce an algorithm for detecting overlapping fully nested subgraphs of an arbitrary input graph. We represent the resulting nested community structure with a *community graph* that encodes additional information about the hierarchy and relationship of the nested subgraphs in the network. Then, in "Generation of overlapping communities" section we introduce an algorithm that can generate a class of bipartite graphs that exhibits the nested community structure of the input community graph. This algorithm may also be suitable for testing nestedness metrics. In the "Experiments" section we introduce two new metrics to measure nestedness on both node and graph level, and test our nested community detection algorithm on typical nested and non-nested artificial and real-world graphs. The artificial graphs are either generated using the algorithm introduced in the "Generation of overlapping communities" section, or bipartite nestedness benchmark networks and non-bipartite community detection benchmark networks are utilized. Finally, in "Conclusions" we summarize.

Throughout this paper, $G = (V, E)$ will be a finite and unweighted graph with $|V| = n$ and $|E| = m$ (with no self-edges, i.e., $(i, i) \notin E\ \forall i \in V$), $N(i)$ denotes the neighborhood of node i and $|N(i)|$ its size, i.e., the degree of i. In the case of directed graphs, we will denote the incoming neighbors of a node i with $\mathrm{in(i)} = \{\mathrm{j} : (\mathrm{j}, \mathrm{i}) \in \mathrm{E}\}$ and its outgoing neighbors with $\mathrm{out(i)} = \{\mathrm{j} : (\mathrm{i}, \mathrm{j}) \in \mathrm{E}\}$.

Nestedness

A graph G is *fully nested* if, for any pair of vertices $i, j \in V(G)$ such that j has a higher or equal degree than i, $N(i) \subseteq N(j)$ holds (Mariani et al. 2019). In other words, the vertices of G can be ordered such that the respective neighborhoods (as sets) form a chain. In case of bipartite graphs, the two compared vertices must be in the same (color) class. If perfect or full nestedness holds for one class, then it holds for the other. Figure 1a shows a fully nested graph, while Fig. 1b depicts one that is not fully nested. Observe that the presence of an induced $2K_2$, that is two independent edges (formed by the edges (2, 5), (3, 6) in Fig. 1b, colored in red), is responsible for breaking

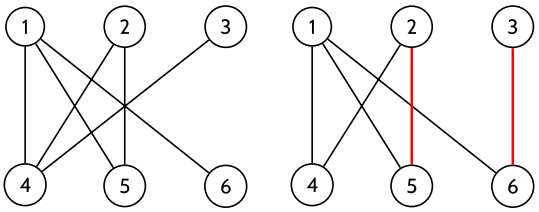

Fig. 1 Examples of (**a**) fully nested and (**b**) partially nested bipartite graphs. The $2K_2$ breaking nestedness is highlighted in red

full nestedness. We want to emphasize the use of *not fully nested* instead of *not nested* as graphs may not be fully nested themselves, but may have fully nested *subgraphs*. In other words, the definition of nestedness may not hold for the entire graph, but it might be true for one or more subsets of vertices.

We distinguish between the definitions of nested *vertices* and nested *graphs*. As a building block to define nestedness of graphs, we first define nestedness of a pair of vertices. The amount (or strength) of nestedness between two vertices can be defined as

$$\text{nest(i, j)} = \frac{\left|\text{N(i)} \cap \text{N(j)}\right|}{\min\left\{|\text{N(i)}|, \left|\text{N(j)}\right|\right\}}. \tag{1}$$

If $\text{nest(i, j)} = 1$, then $N(i) \subseteq N(j)$ or $N(j) \subseteq N(i)$, i.e., the nestedness criterion holds for the given vertices i and j. In the special case of either of the vertices being isolated (where $\min\left\{|N(i)|, \left|N(j)\right|\right\} = 0$), we consider the vertices non-nested and define $\text{nest(i, j)} = 0$.

While Eq. 1 is not ideal for measuring the nestedness of the whole graph (it would require $\approx n^2$ calculations to get an average nestedness value, for example), we can use it to find groups of vertices that form perfectly nested subgraphs—which is our ultimate goal.

Existing methods

Clustering to nested parts

One way to find fully nested subgraphs is to use the incidence matrix to identify submatrices of echelon form (Junttila and Kaski 2011)—this, however, works for bipartite graphs only. Another possibility is to use Eq. 1 and assign vertices to a group where the pairwise nestedness values are equal to 1. This can be done by, for example, performing a $2K_2$-free coloring on the graph (London et al. 2022).

All of these methods exhibit the same problem, though. The graph in Fig. 1b contains two nested subgraphs: one with the vertices $\{1, 2\}$ and another with $\{1, 3\}$. Notice that vertex 1 is present in both fully nested subgraphs, but we can only assign that vertex to a single group in the clustering task. This would create two ambiguous cluster structures: both $\{\{1, 2\}, \{3\}\}$ and $\{\{1, 3\}, \{2\}\}$ are valid clusterings of the same graph. Although this is not necessarily a problem as one will often search for a single clustering, the resulting structure does not encode such overlaps, potentially losing valuable information.

Edge-based nested community detection

We differentiate clustering from *community detection* based on the number of groups a vertex can belong to. We refer to clustering when a vertex can be assigned to a single group only, as in the previous case, and community detection when vertices can belong to multiple, and thus potentially overlapping, groups. Note that in our case, we are looking for a special (or *constrained*) overlapping community structure, where each community is a fully nested subgraph.

One method that avoids the problem of nodes being constrained to a single group is a greedy edge-based community detection algorithm (Gera et al. 2022). This method first assigns community indices to the *edges* of the graph. It calculates nest(i, j) for vertices i and j and if they are nested, the edges of both vertices are assigned to a common community. Otherwise, the edges of i get a different community index than the edges of j. In the end, the communities that a vertex i belongs to will be the union of the communities of its incident edges.

Nestedness in non-bipartite graphs

Since we are focusing on general—not just bipartite—graphs, we need to take into account the connection between a vertex pair when measuring their nestedness. This was not an issue in bipartite graphs, since, by definition, there are no edges between vertices of the same class.

If we use Eq. 1 to measure nestedness between vertices i and j, and there exists an edge $(i, j) \in E(G)$, the two vertices will never be considered fully nested, because $i \in N(j)$ and $j \in N(i)$, but $i \notin N(i)$ and $j \notin N(j)$. Thus, nest(i, j) < 1 when $(i, j) \in E(G)$. This would also mean that the graph in Fig. 2 or even K_n ($n \geq 2$), the complete graph of n vertices, are not fully nested graphs.

The approach we are going to follow when comparing two vertices is to ignore the edge between them if there exists one. Thus, we may use the following equation instead of Eq. 1:

$$\text{nest(i, j)} = \frac{\left|(\text{N(i)} \setminus \text{j}) \cap (\text{N(j)} \setminus \text{i})\right|}{\min\left\{\left|\text{N(i)} \setminus \text{j}\right|, \left|\text{N(j)} \setminus \text{i}\right|\right\}}. \tag{2}$$

Note that considering the existence of edges between nodes when searching for fully nested parts depends on the application. If we are looking for fully nested bipartite subgraphs in graphs that are not necessarily bipartite themselves (e.g., as in Junttila and Kaski 2011; London et al. 2022), the fact that two nodes are connected or not should not be ignored.

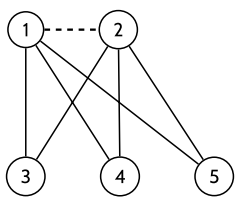

Fig. 2 Example of a non-bipartite, fully nested graph. Nodes 1 and 2 are considered nested, despite them having an edge (dashed line) between them

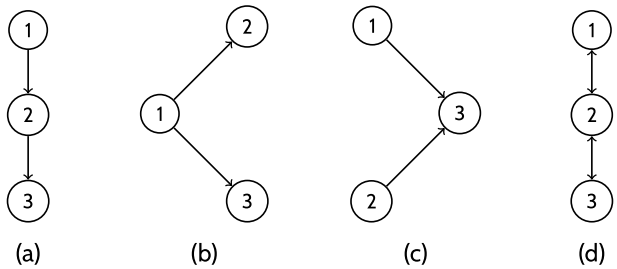

Fig. 3 Cases of nested relations in a community graph. From left to right: a fully nested graph (**a**), a node nested with different nodes (**b**), multiple nodes nested with the same node (**c**), and nodes having equal neighborhoods (**d**)

Nestedness and community detection

In this section, we will present an algorithm that retrieves the nested community structure of the input (bipartite or general) graph. First, we will talk about how the order of nodes inside nested community structures allows us to store more information compared to traditional communities. We use this additional information to construct a so-called *community graph*. Then, we introduce an algorithm that reconstructs not only the nested communities, but the entire community graph from an arbitrary input graph.

While in this work we frequently mention *community detection*, we refer to it as a framework for detecting *overlapping* groups (communities) of vertices that are, in some sense, similar to vertices within the group, while, in the same sense, different to vertices in other groups. Traditionally, this similarity meant vertices being densely connected within a group, while vertices across different groups were less connected. Here, we are looking for overlapping groups (communities) of vertices that form fully nested subgraphs instead of being densely connected. We will call these *nested communities*.

Nested hierarchy from directed graphs

Using Eq. 2 we are able to decide whether two vertices are nested, but the direction of nestedness, i.e., which vertex's neighborhood is a subset of the other, is not considered. This is important because we will use this information to determine the hierarchy of vertices. Knowing the direction of pairwise nestedness, we can use it to create a graph representation that encodes the nested relationships of the entire graph.

To do this, we construct a directed graph, where a directed edge $i \to j$ means $N(i) \subseteq N(j)$. We will refer to this graph as the *community graph*. Before we proceed, we first verify some basic scenarios from Fig. 3.

1 If we have edges $i \to j$ and $j \to k$ (as in Fig. 3a), we get $N(i) \subseteq N(j)$ and $N(j) \subseteq N(k)$. Nestedness is transitive, so this also means $N(i) \subseteq N(k)$. For simplicity, we omit these edges from our community graphs, or equivalently, we work with the *transitive reduction* of the community graph (see the "Community detection algorithm" section for more details).
2 A node can have multiple out-neighbors (Fig. 3b). If we have edges $i \to j$ and $i \to k$, then we get $N(i) \subseteq N(j)$ and $N(i) \subseteq N(k)$. This can be solved by letting both j and k

have all the neighbors of i, but also making sure j and k each have at least one other neighbor the other doesn't

3 A node can also have multiple in-neighbors (Fig. 3c). Here we have edges $i \to k$ and $j \to k$ and get $N(i) \subseteq N(k)$ and $N(j) \subseteq N(k)$. This case can be solved by taking the union of the neighbors of i and j to create the neighborhood of k ($N(k) \supseteq N(i) \cup N(j)$).

4 Finally, nodes may have the exact same neighbors as other nodes (Fig. 3d). This results in edges $i \leftrightarrow j$, and thus in both $N(i) \subseteq N(j)$ and $N(j) \subseteq N(i)$ ($N(i) \equiv N(j)$). To reduce complexity, we will draw a path with bidirectional edges instead of a clique.

It is important to note that a maximal (non-expandable) path of the community graph will represent a nested community or, in other words, a nested subgraph. For example, if the community graph of G is a single P_n (a path of n vertices, as in Fig. 3a), the original graph G is fully nested, whereas if G is fully not nested (i.e., G does not have a single pair of vertices (i, j) where $N(i) \subseteq N(j)$), its community graph will be a graph with no edges.

Community detection algorithm

Now we introduce an algorithm to find overlapping nested communities, that is, fully nested subgraphs of G. The core parts of the detection algorithm reconstruct the community graph from the input graph and then find the community graph's maximal (non-extendable) paths to enumerate the communities. The main steps are the following.

Reconstructing the community graph

To reconstruct the community graph, we first enumerate all nested vertex pairs. Here, instead of greedily performing $\approx n^2$ comparisons, we can use the same trick used in Gera et al. (2022). That is, we do not compare vertices that have no common neighbors, since they are certainly not nested. Instead, we calculate nest(i, k) by first going through $j \in N(i)$, and pick $k \in N(j)$, where $k < i$.[1] This way, i and k have at least one common neighbor (j), so they are potentially nested. In practice, this can save us a lot of computational time (especially in sparse graphs), and the number of discarded comparisons can be large, according to our experience. Since we do not exclude potentially nested vertex pairs, we do not lose any information in this step.

When comparing vertices, we also need to know the direction of nestedness between the vertices, for example, by calculating $\mathrm{sgn}(|\mathrm{N(i)}| - |\mathrm{N(j)}|)$ when $\mathrm{nest(i, j)} = 1$. Once we have done all the comparisons, we build the directed edge list of the nested pairs, where $i \to j$ is an edge if $\mathrm{nest(i, j)} = 1$ and $\mathrm{sgn}(|\mathrm{N(i)}| - |\mathrm{N(j)}|) \leq 0$ (or equally, $N(i) \subseteq N(j)$).

However, the list will contain a lot more edges than we need. Let's revisit the fully nested graph from Fig. 1a for an example. Here $N(3) \subseteq N(2)$ and $N(2) \subseteq N(1)$, but as such, $N(3) \subseteq N(1)$ will also hold. Since we need to find maximal paths in the resulting graph, the transitive $N(3) \subseteq N(1)$ relationship and its corresponding edge are redundant and certainly not part of the maximal path $3 \to 2 \to 1$. To remove them, we perform a transitive reduction on the graph built from the nested edge list. As a result, for all triples $i \to j \to k$ the edge $i \to k$ will be deleted. This significantly reduces the number of

[1] This condition enables us to skip the calculation of nest(k, i), which would have the same result as the previously calculated nest(i, k).

edges to consider in the next step, which greatly improves the performance of the algorithm. This completes the community graph discovery.

Finding nested communities

Now that we have a community graph, we need to retrieve the list of nested communities. To do this, we enumerate the maximal (non-extendable) directed paths in the graph. Listing these paths can be done using any traversal method, such as a depth-first search. Due to the transitive reduction performed in the previous step, this can be accomplished quite quickly.

Here, each directed path represents a fully nested community, with the order of the vertices also encoding hierarchy. For example, if K_n (a clique of n vertices) is the input graph, the community graph will be P_n (a path of n vertices), which will have a single community with all n vertices in it.

Vertex compacting based on neighborhood

There is one edge case that complicates the search for maximal paths, where cycles are created due to bidirectional edges between vertices. To solve this problem, and also improve search performance, we first find vertices with equal neighborhoods and merge them into a single vertex before building the community graph. Isolated vertices are not compacted, and edges between vertices are ignored when checking for neighborhood equality. This means that an isolated K_2 (two nodes with an edge between them), for example, is not compacted. This change has multiple positive effects. First, the community graph is now guaranteed to be a directed acyclic graph (DAG) as there are no other factors that can introduce cycles, making maximal path finding much easier. It also makes the resulting community graph smaller by having it be built from fewer vertices, improving performance. For example, a star graph with any number of nodes will have a community graph of just two nodes and a single edge.

We then find the maximal paths as normal, treating the merged vertices as a single vertex. Finally, we recover the original vertices by expanding the merged vertices and inserting edges in both directions between them.

Figure 4 shows all the steps of the algorithm on an example graph. When enumerating the communities, we traverse the compacted graph (Fig. 4c) and then insert the removed vertices into the paths.

Remarks

Here we pinpoint some key areas in the behavior of the algorithm. The algorithm's pseudocode is available on Fig. 5 and its source code is included in Additional file 1.

1. *Non-bipartite graphs* A major advantage of the algorithm is that it does not exploit any property specific to bipartite graphs. In theory, this could make it directly applicable to any (unweighted) non-bipartite graph. In practice, we need to solve the problem of connected vertices described in " Nestedness in non-bipartite graphs" section. As with Eq. 2 we ignore the connection between two vertices when comparing them. This, combined with vertex compacting, results in the algorithm correctly

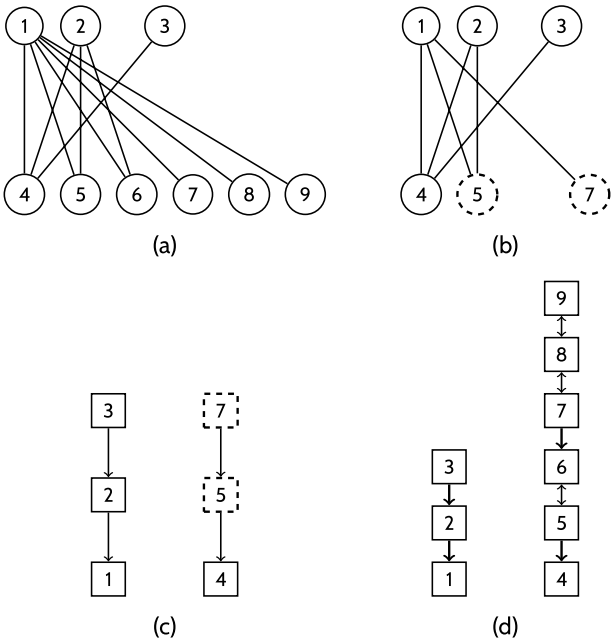

Fig. 4 Steps of the community detection algorithm. Starting from the input graph (**a**), we first compact vertices with equal neighborhoods (**b**), then build the community graph (**c**), and finally reverse the vertex compaction, adding the bidirectional edges

```
nested_pairs ← ∅
for (i in V) {
  for (j in N(i)) {
    for (k in N(j)) {
      if (i ≥ k) continue
      if (nest(i, j) = 1) {
        if (sgn(|N(i)| − |N(j)|) = 1) {
          nested_pairs ← nested_pairs ∪ (j, i)
        } else {
          nested_pairs ← nested_pairs ∪ (i, j)
        }
      }
    }
  }
}
nested_pairs ← transitive_reduction(nested_pairs)
comms ← longest_paths(nested_pairs)
```

Fig. 5 Pseudocode of the nested community detection algorithm. Despite having three nested for loops, the algorithm only iterates over (*i*, *k*) pairs of vertices that have at least one common neighbor

finding nestedness in non-bipartite graphs too (as later demonstrated in " Results on typical examples" section).

2. *On constrained community detection* We also need to make some comments about the community structure detected by the algorithm. The algorithm is designed to detect certain types of overlapping communities (specifically communities that satisfy the constraint of being fully nested), essentially performing a "constrained" community detection. The algorithm is also not a heuristic to detect nestedness. This is due to the fact that we start by enumerating all possible $\approx n^2$ comparisons and then

exclude only those pairs that are guaranteed not to be nested. As a result, all remaining, potentially nested, vertex pairs are compared, and no stochastic elements are included in the process.

3. *Permissive nestedness* As a possible future direction, we would also like to mention the potential of relaxing the requirements of nestedness for communities. So far, we have talked about how the algorithm detects communities that satisfy a certain "constraint". This constraint can be quite strict, as two nodes that share *largely* the same neighborhood (with a few deviations) are considered to be non-nested. There are many metrics that quantify the degree of nestedness of a graph. They allow us to see not only whether a graph is fully nested or not, but also *how much* nested it is. To increase the flexibility of our algorithm, we can similarly allow pairs of vertices that are not fully nested to belong to the same community, *above* a certain nestedness threshold, for example. The algorithm currently does not support this, but it is easy to implement.

Generation of overlapping communities

Previously, we have shown an algorithm that can reconstruct the community graph from an arbitrary input graph. Now, we present an algorithm that is capable of generating bipartite graphs with multiple overlapping fully nested groups of vertices, based on an input community graph. The method also returns the ground truth nested structure, making it suitable for use when benchmarking algorithms that find overlapping nested communities.

The generated structure is more general than in-block nestedness (Solé-Ribalta et al. 2018), where the graph is partitioned into disjoint, fully nested "blocks". Our method is capable of generating not only this structure, making it a more versatile approach.

We believe that the proposed algorithm is useful not only for testing our nested community detection algorithm, but also for creating benchmark data sets for future methods that detect overlapping nested structures.

Benchmark generator algorithm

Now that we have a method for describing nestedness using a directed graph, we will present an algorithm that generates a bipartite graph that satisfies the nested structure described by the community graph. That is, if there is an edge $i \to j$ in the community graph, the resulting graph will have $N(i) \subseteq N(j)$. We will show that the algorithm is capable of generating not only fully nested graphs, but also graphs with overlapping nested communities.

To generate a bipartite graph from a community graph, denoted by G_c, we first perform a topological sorting on G_c, since we need to generate neighbors for each vertex v such that its predecessors (denoted by $\mathrm{in}(v)$) already have their neighbors. To do this, we must assume that the community graph is acyclic, as there must be at least one vertex with no predecessors, which will be the starting vertex. When visiting a vertex, we add all the neighbors of its predecessors to the neighbors of the current vertex and generate a new neighbor for it (if we visit the ith vertex, we can label the new vertex $n + i$). The first part guarantees nestedness, while the new neighbor makes sure that the two vertices do

```
i ← 1
order ← topological_sort(G_c)
while (i ≤ n) {
  v ← order(i)
  N(v) ← { n + i } ∪ ⋃_{j∈in(v)} N(j)
  i ← i + 1
}
```

Fig. 6 Pseudocode of the nested graph generator algorithm

not have the same neighborhood (in which case there should be both a $i \to j$ and a $j \to i$ edge in G_c). Formally, we have the subroutine visible in Fig. 6.

Remarks

1. *Generating random nested community structures* The algorithm we have described so far is used to generate a graph with a given nested structure from an input community graph. To generate random graphs with overlapping nested structures, we can use random input graphs. However, the input graph must be a DAG. This can be achieved, for example, by sampling a random spanning tree of the complete graph K_n and randomly orienting its edges.
2. *Cycles in the community graph* Because the algorithm performs topological sorting and requires the input graph to have a vertex with no predecessors (i.e., one that is not part of a cycle), it is much simpler to work with an acyclic community graph. This prevents us from generating vertices with equal neighborhoods, however, the vertex compacting approach described in section "Community detection algorithm" could be adapted to the generator algorithm to make this possible.
3. *Generated graph sizes* Since the algorithm takes a community graph of size n and generates a neighbor for each vertex, the resulting graph will have exactly $2n$ nodes. This also makes the algorithm incapable of generating bipartite graphs with classes of different sizes—a trivial example is the star graph.
4. *Multiple blocks* We have not touched on whether the input G_c DAG has to be weakly or strongly connected yet. The algorithm can handle community graphs with multiple components and will render a component of the community graph as a component in the generated bipartite graph. For example, a graph with multiple disjoint P_k components (that is, a directed path of k nodes) results in an in-block nested bipartite graph (Solé-Ribalta et al. 2018).
5. *Nested structure of the other class* As mentioned in the description of the algorithm and in remark 2, the input to the algorithm is a DAG that describes the community structure of *one class* of the graph. The other class of the generated bipartite graph is not part of the input community graph (and thus the ground-truth). However, generating the bipartite graph from the community graph of the other class, we get a final bipartite graph that is isomorphic to the one generated using the original input.

Experiments

In this section, we will examine the performance of the nested community detection algorithm from several perspectives. First, we verify that the algorithm is able to detect the nested structures in a few basic examples, then whether it can find all ground-truth communities of the benchmark graphs generated by the algorithm described in the "Generation of overlapping communities" section, fully reconstructing the input community graph. We then examine the community structure of graphs commonly used when benchmarking nestedness metrics. Then, in a separate section, we compare the results of our method with other community detection algorithms to identify key differences in the discovered community structures. Finally, we measure the execution time of our algorithm in the function of node and edge counts in various graphs.

To evaluate our results and quantify nestedness, we use the NODF (Almeida-Neto and Ulrich 2011), discrepancy (Brualdi and Sanderson 1999) and temperature (using the Binmatnest algorithm) (Ángel Rodríguez-Gironés and Santamaría 2010) nestedness metrics on bipartite graphs. To make it easier to compare these metrics with our community structure, we derive our own nestedness metric from the community graph: the average fraction of communities a vertex is part of, called *vertex presence*. For normalization purposes, this number is multiplied by 2 in bipartite graphs, since perfectly nested bipartite graphs have 2 perfectly nested communities, one for each class. Vertex presence ranges from 0 to 1, where 1 means every vertex is part of every community, i.e., there is only a single community with all vertices in it. When vertex presence is low, it means that vertices are part of few communities while the total number of communities is high. When vertex presence is calculated specifically on the nested community structure, a maximal presence means there is one nested community (path), so the graph is fully nested, and a minimal presence means that every vertex is in its own nested community, having no nestedness in the network at all. Intuitively, a larger value means a vertex is part of a larger portion of the nested communities, which increases the overall nestedness of the network. We also note that since our community detection algorithm works on non-bipartite graphs, unlike the previously mentioned metrics, vertex presence is not restricted to bipartite graphs. Formally, we can obtain vertex presence for a vertex v by calculating

$$\mathrm{pres(v)} = \frac{|\{\mathrm{C} : \mathrm{C} \in \mathcal{C}, \mathrm{v} \in \mathrm{C}\}|}{|\mathcal{C}|}, \tag{3}$$

where $\mathcal{C}$ is the set of nested communities (maximal paths in the community graph). Since the domain of pres(v) will depend on n (its lowest value is $\frac{1}{n}$), we can normalize it into the [0, 1] range, so that 0 means entirely not nested in all cases and 1 still means fully nested. This can be achieved using

$$\overline{\mathrm{pres}}(v) = \frac{n}{n-1}(\mathrm{pres(v)} - 1). \tag{4}$$

For compactness, we will use *average vertex presence* as a metric of the entire *graph* by averaging the normalized vertex presence across all vertices.

As opposed to regular communities, an interesting property of nested communities is that the *position* of a vertex inside a community is informative, too. If a vertex is at the

beginning of a community (path), then its neighborhood is a subset of the other vertices of the community. If, on the other hand, a vertex is at the end of a community, its neighborhood is a superset of the other vertices in the same community. We call the quantification of this *vertex position*, and it can be calculated using

$$\mathrm{pos}(\mathrm{v}, \mathrm{C}) = \frac{\mathrm{i} - 1}{\max\{1, |\mathrm{C}| - 1\}}, \tag{5}$$

where $\exists i : C_i = v$ (v is the ith vertex of C), thus pos(v, C) is only valid if $v \in C$. We perform normalization in the denominator that makes the position of vertices at the beginning of a community 0, even if they are the sole vertex in a community, assuming indexing starts at 1. With this, we can calculate the position of a vertex on all nested communities it is present in, then take its average to get the *mean vertex position* of said vertex.

These two metrics allow us to measure nestedness both on a graph level (by calculating *average vertex presence*) and on a vertex level (through *vertex position*).

Results on typical examples

Before testing the algorithm on generated benchmark examples, we first demonstrate that the algorithm finds basic nested structures. Figure 7a, e show that the algorithm correctly identified the full bipartite graph that has two fully nested parts: nodes in one class belong to the same community. Figure 7b, f show the same concept, but in a special case: the star graph is also considered fully nested, and the algorithm correctly identifies the upper class as fully nested and the single node of the bottom class as another. Figure 7c, g show that the nodes of the fully non-nested bipartite graph are all correctly put into different communities.

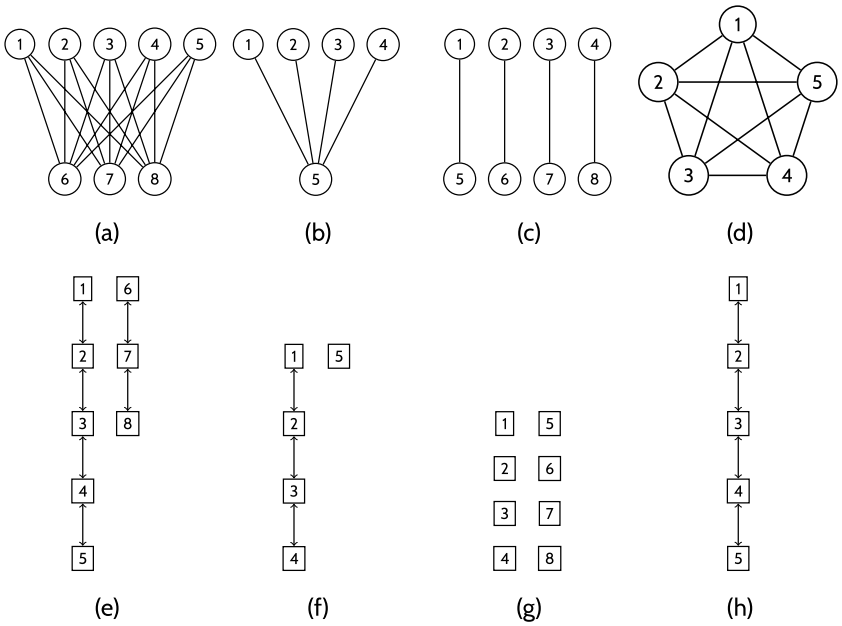

Fig. 7 Graphs showing typical nested configurations (**a**–**d**) and their community graphs (**e**–**h**)

Finally, to demonstrate that the algorithm works with non-bipartite graphs, through Fig. 7d, h we show that all five nodes of the complete graph are correctly classified as a single community.

Results on benchmarks

In this section, we will compare the ground-truth community structure of the generated benchmark graphs with the one found by our algorithm. In order to create a benchmark graph, we use one or more random spanning trees (creating a spanning forest) sampled from complete graphs and orient their edges randomly. These oriented spanning trees will be the community graphs of the benchmark graphs.

The benchmark is set up as follows. We generated 2000 random graphs with ground-truth communities with 1 to 4 blocks (components) and 1 to 60 nodes per block. Let n_b denote the sum of the number of nodes in the input to the generator across all blocks. Note that we know the ground truth for the first n_b nodes as these are the nodes of the input community graph, the rest n_b nodes are generated; this was discussed in more detail in the generator algorithm's "Remarks" section. The generated benchmark graphs are available as Additional file 2.

After generating the benchmark graphs, we run the algorithm on them and compare the first n_b nodes of the result with the known ground truth. Again, we cannot compare the rest due to the limitations of the generator algorithm, however, we do not need to, since nestedness is a symmetric property for bipartite graphs. Furthermore, correctly recovering the ground-truth community structure for the first n_b nodes means that the algorithm is capable of reconstructing the entire community graph on the input.

Our tests show that all community graphs in the benchmark set were correctly recovered and that the resulting community structures were exactly consistent with the ground truths. Figure 8 shows a generated benchmark graph and its detected nested community structure. Even looking at this figure, we can presume that there are many communities, even on smaller graphs, with many of them overlapping over a large part of the vertices.

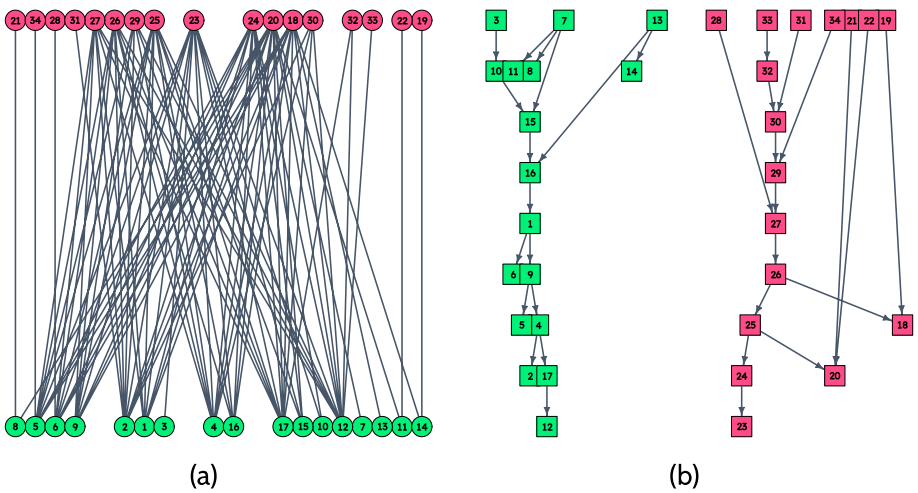

Fig. 8 A generated bipartite graph (**a**) and its detected community graph (**b**). The ground truth is known for the green (bottom) class

Results on real-world networks

Bipartite networks

After validating that the algorithm is capable of reconstructing the community graph, we take a look at real-world networks used to test nestedness metrics. First, we examine the nested structure of ecological networks from Web of Life (Bascompte Lab 2014). We examine the algorithm's output on pollinator (mutualistic) and host-parasite networks. These are two sets of small bipartite networks of species interactions. For an overview of the results, see Table 1.

The first network we examine is `M_PL_069_03` (Kohler 2011) (Fig. 9), a tiny pollination network of seven plant and four hummingbird species created from observations in eastern South America. Vertices 1–7 represent plants, while vertices 8–11 represent hummingbirds. For legibility reasons, we show the vertex IDs instead of species names on the plot and clarify where needed. The NODF value of the network is 75.926 (in the range [0, 100], where 100 means fully nested), and its discrepancy value is 2 (where 0 means fully nested), suggesting that it is indeed a highly nested network. The average vertex presence of the community graph is 0.606.

We can see that the graph is clearly not fully nested, as there are multiple communities (paths) on its community graph in both classes. However, it does have large nested communities that cover most of the vertices in each class with high overlap. In both classes, we have three communities that cover all vertices of that class. In the upper class (plant

Table 1 Computed properties on a subset of the full dataset

	n	m	D	$\overline{C_i^w}$	Q	$\lvert\mathcal{C}\rvert$	$\overline{C_s}$	pres	T_b	T_{nodf}
M_PL_001	185	361	0.02	0.00	0.50	284	3.86	0.04	2.83	14.46
*M_PL_057	997	1920	0.00	0.00	0.53	2000	22.92	0.05	0.79	7.23
*M_PL_058	113	319	0.05	0.00	0.30	277	4.07	0.06	10.13	28.02
*M_PL_069_01	24	29	0.11	0.00	0.43	14	2.93	0.21	34.29	31.07
*M_PL_069_03	11	15	0.27	0.00	0.21	6	3.33	0.57	12.09	75.93
*A_HP_015	10	12	0.27	0.00	0.21	2	5.00	1.00	1.05	75.00
A_HP_016	27	52	0.15	0.00	0.24	11	5.18	0.36	21.82	56.25
A_HP_017	14	19	0.21	0.00	0.26	4	6.00	0.85	4.97	78.26
A_HP_025	58	107	0.06	0.00	0.39	56	3.88	0.12	21.56	25.22
A_HP_026	33	142	0.27	0.00	0.11	36	6.94	0.40	6.46	87.78
Adjnoun	112	425	0.07	0.19	0.28	166	2.57	0.01	NA	NA
celegansneural	297	2359	0.03	0.31	0.39	238	1.66	0.00	NA	NA
dolphins	62	159	0.08	0.30	0.52	65	2.08	0.02	NA	NA
*karate	34	78	0.14	0.59	0.44	33	3.64	0.08	NA	NA
netscience	1589	2742	0.00	0.88	0.96	895	3.38	0.00	NA	NA
power	4941	6594	0.00	0.11	0.93	4256	1.96	0.00	NA	NA
dswomen	32	89	0.18	0.00	0.31	27	2.37	0.12	36.19	48.58
*families	15	20	0.19	0.22	0.40	13	1.92	0.07	NA	NA
les_miserables	77	254	0.09	0.74	0.57	77	5.78	0.06	NA	NA

D: graph density; $\overline{C_i^w}$: mean local vertex transitivity (clustering coefficient); Q: Newman-modularity (based on the multi-level modularity optimization algorithm for finding community structure (Blondel et al. 2008)); $|\mathcal{C}|$: number of nested communities; $\overline{C_s}$: average nested community size; T_b: Binmatnest temperature (0–100, lower = more nested); T_{nodf}: NODF value (0–100, higher = more nested). Graphs marked with an asterisk (*) were directly mentioned and analyzed in the article. NA nestedness values mean the graph is not bipartite, and thus the metric could not be calculated. The full computational results are available in CSV format as Additional file 3

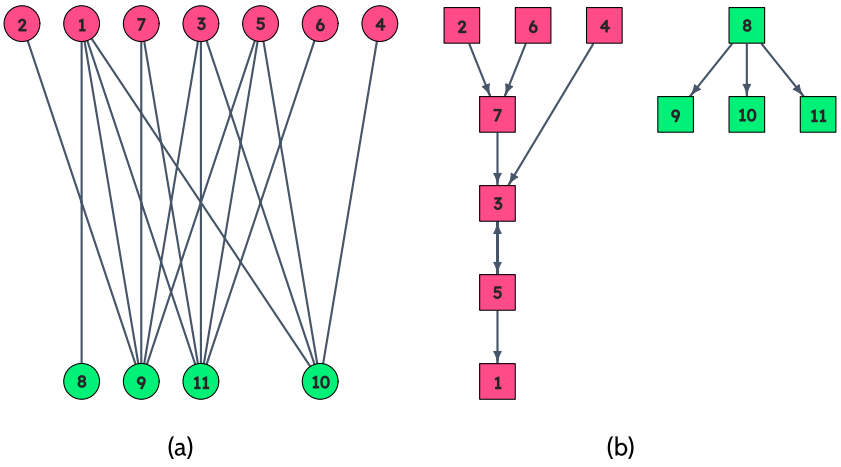

Fig. 9 The original `M_PL_069_03` graph (**a**) and its nested community graph (**b**)

species, purple), the communities cover a larger part of the class with 5 (71%) and 4 (57%) vertices. Three plant species (vertices 3, 5, and 1) also play a key role in these communities, as they are part of all three communities. They represent generalist entities in the network, connected to most vertices of the other class, i.e., most hummingbirds visit them. Vertex 7 is also part of two communities, only vertices 2, 4, and 6 are part of a single community. We can see that they are all visited by only one species of hummingbird.

Looking at the class of hummingbirds, the community structure is simpler because the class has four entities only. Interestingly, we can see that vertex 8 (amazilia versicolor) is part of the three communities but visits only a single plant (vertex 1, aechmea cylindrata), a plant that all other hummingbird species visit, too. This connection alone makes the amazilia versicolor nested with all other species. This is an important aspect of nested networks, where specialist species tend to pick the generalists in the other class.

`M_PL_069_01` (Kohler 2011) (Fig. 10) is a slightly bigger pollination network of 18 plants (vertices 1–18) and 6 hummingbirds (vertices 19–24), with a lower connectance.

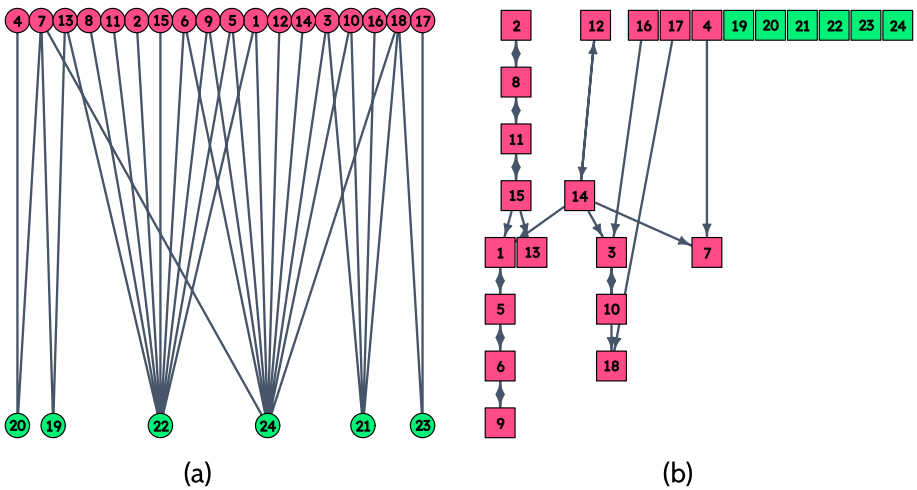

Fig. 10 The original `M_PL_069_01` graph (**a**) and its nested community graph (**b**)

Interestingly, the community structure shows less symmetry in terms of the classes, with eight overlapping plant communities and the six hummingbird species all in their own class. On closer inspection, we can see that some hummingbird species visit largely the same plants as others, but there are always plants that one visits, but the other one does not, and vice versa. For example, vertex 21 (Clytolaema rubricauda) is connected to most of the neighbors of vertex 24 (Thalurania glaucopis), but vertex 16 (Vriesea erythrodactylon) is only connected to 21, creating a $2K_2$.

This graph also highlights the asymmetric nature of nested communities: while we can observe some degree of nestedness (with some paths covering half the vertices) in the class of plants, the class of birds is fully non-nested.

Moving on to larger networks, such as `M_PL_058` (Bartomeus et al. 2008) (community graph visible in Fig. 11), untangling the community structure becomes increasingly more difficult, with many communities (277 over two classes) overlapping each other. However, we can make some important observations on the community graph. For example, the community graph has few zero-degree vertices, which means that most vertices contribute to the overall nestedness of the graph, but they are nested only with *some* other vertices. We can also see that in both large components, there are only a few vertices with high total degrees. In the largest component of the community graph (colored in green, containing bees), there is a bee (of the Andrena genus) with a high in-degree, connecting lots of nested communities, by interacting with 22 plants out of 32. In the second-largest component, which consists of plants, there is a vertex with a high out-degree (a Vicea lutea), being part of a lot of communities at once by having only a single connection to the aforementioned Andrena bee.

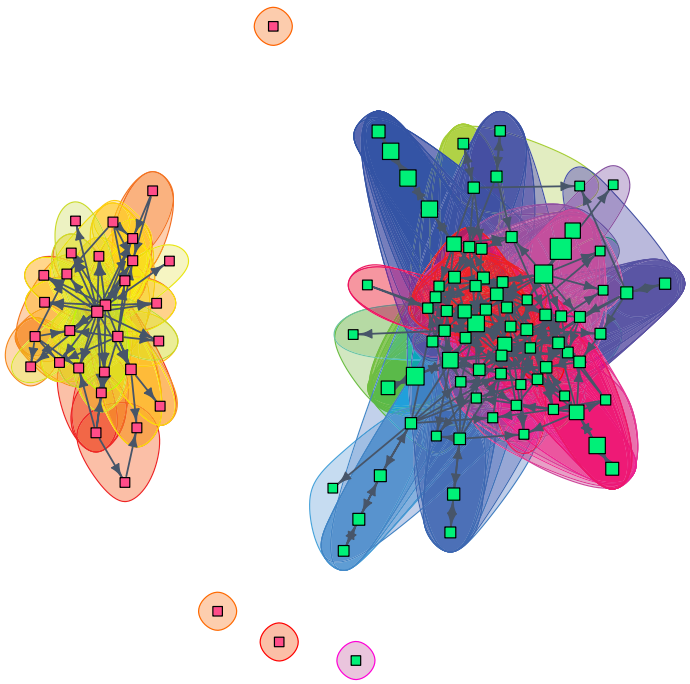

Fig. 11 Community graph of the slightly larger `M_PL_058` graph. Larger vertex sizes correspond to higher in-degrees (including transitive edges)

Such large networks also show that partially nested networks can have a huge amount of nested communities. We expect a fully nested bipartite network to have two communities (one for each class), a fully nested non-bipartite network to have a single community, and a fully non-nested network to have n communities, each vertex belonging to its own community. Partly nested networks, on the other hand, may have more than n communities: `M_PL_057` has $n = 997$ vertices and $m = 1920$ edges but at the same time $2000 > m > n$ communities, with an NODF value of 7.23 and a vertex presence of 0.045. Thus, the number of communities is not a linear function of nestedness. This also means that, unfortunately, the number of communities does not perfectly reflect the network's nestedness.

Host-parasite networks

The host-parasite networks scored an average vertex presence of 0.28 versus 0.188 and an average NODF score of 52.033 versus 30.857 in the pollination set, suggesting that the host-parasite networks are on average more nested.

This set contains a fully nested network: `A_HP_015` (Fig. 12), a network of three rodents (vertices 1–3) and seven parasites (vertices 4–10). We can see that a specialist entity (Microtus oeconomus, vertex 3) interacts only with a generalist species (Amphipsylla marikovskii, vertex 5) and vice versa. The network has a vertex presence score of 1 and a discrepancy of 0, but its NODF value is 75 (where 100 would mean fully nested). Similarly, its temperature score isn't showing perfect nestedness, either, at a value of 1.05 instead of 0.

Nestedness metrics

Examining the relationship between vertex presence and some nestedness metrics, we can see that while vertex presence and NODF behave similarly with few exceptions (Fig. 13a, c), Binmatnest gave small scores (meaning high nestedness) to some graphs with low vertex presence (Fig. 13b). These graphs had low NODF and high discrepancy scores, both suggesting low nestedness. Similarly, NODF gave a score of 75 to a fully nested network in the host-parasite network set, where the discrepancy was 0 and vertex presence was 1 (Fig. 13c, d).

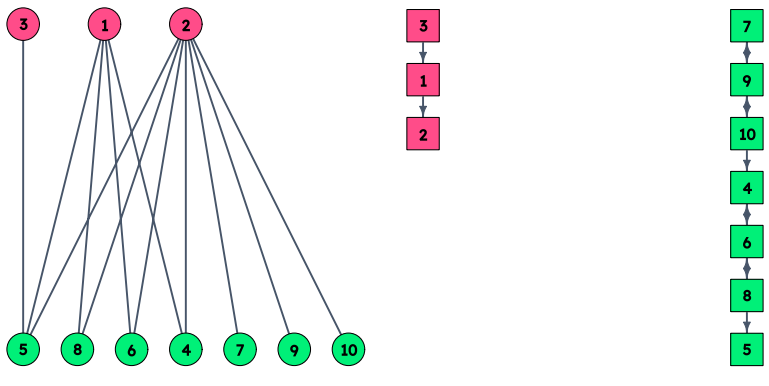

Fig. 12 The `A_HP_015` host-parasite network (**a**) and its nested community graph (**b**)

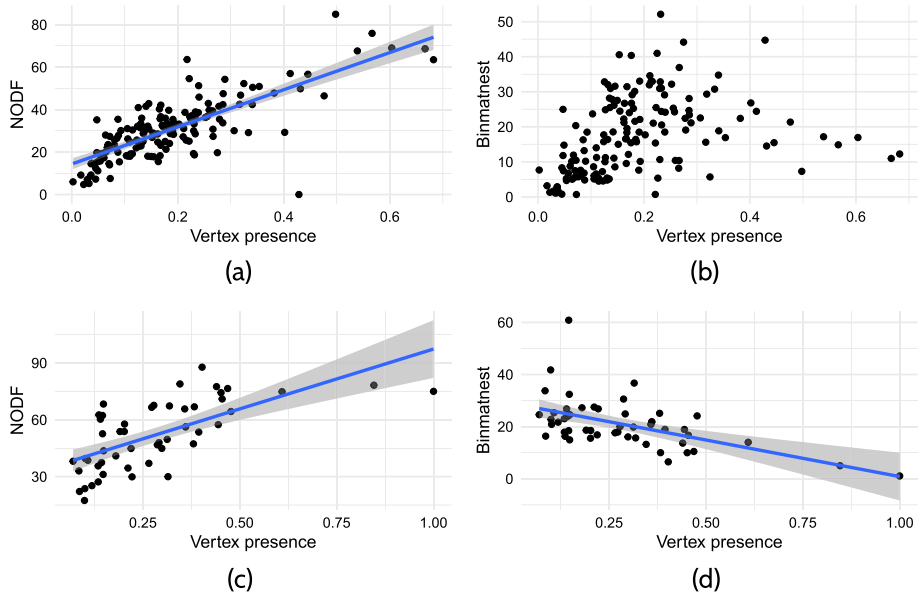

Fig. 13 Comparison of vertex presence with the NODF and Binmatnest nestedness metrics on the pollination (**a**, **b**) and host-parasite (**c**, **d**) network set

Non-bipartite graphs

Now we are going to examine the algorithm's output in some common non-bipartite, mostly social networks. Nestedness in a social network can be interesting in the sense of information exchange and domination. If some i is connected to all acquaintances of j and more, and they get into a conflict, i can spread their position to everyone j knows, potentially dominating j.

A common example used when testing community detection algorithms is Zachary's karate club network (Zachary 1977), visible in Fig. 14a. This is a social network of 34 members of a karate club who interacted outside the club. The club split into two, creating two communities, marked with two colors.

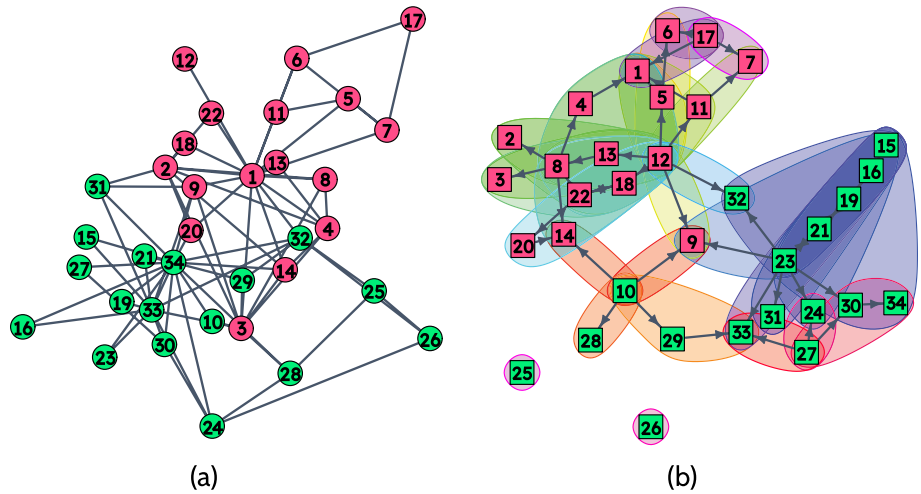

Fig. 14 Zachary's karate club network and its community graph. Vertex colors represent the ground-truth communities

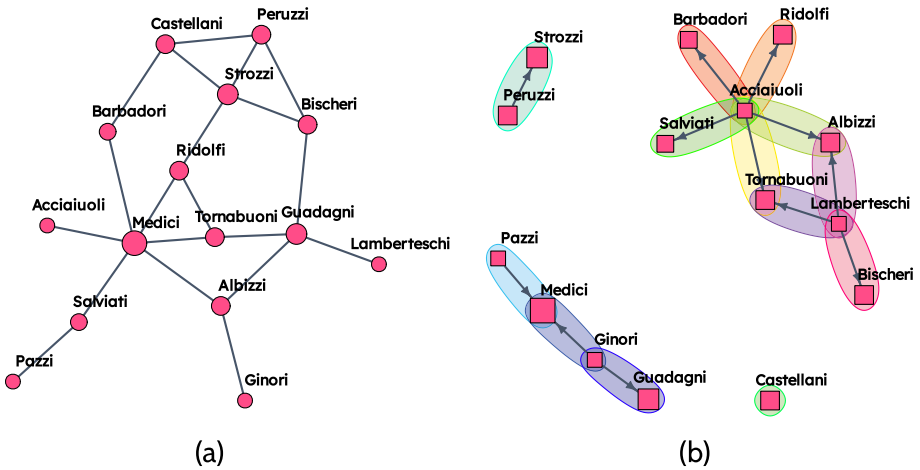

Fig. 15 The Florentine families network and its community graph

Although this is not an ecological network, we can see that there are only three weakly connected components in the community graph (Fig. 14b), two of them being isolated vertices and the third formed by the rest of the graph. Most nested relationships are between vertices of the same color (karate community), except for three nodes. Node 1 (the instructor) is in a nested relationship with 9 other members out of 16 in its karate community, while node 34 (the administrator) is with 7 out of 16. The graph contains 33 nested communities with an average size of 3.67 vertices per community and the mean vertex presence is 0.107. These observations lead us to believe that nestedness does not play a key role in the formation of the network.

The Florentine families network (Breiger and Pattison 1986) (Fig. 15a) contains marriage links between families during the Italian Renaissance. While this network is typically used to demonstrate centrality, the presence of nestedness may be interesting in the sense that families can be in a dominating position if they have connections to all neighbors of another family.

Looking at the community graph in Fig. 15b we can see that there are few cases for this, most of them due to having a single neighbor that is common with one of the central families. For example, the Acciaiuoli family is nested with five others because they have a connection only with the Medici family. This means that if someone could influence the Medici family, they might also be able to influence the Acciaiuoli. Another interesting observation is that the Castellani family is not nested with anyone, so while they have their connections, they are not dominated by any other family. With a low mean vertex presence of 0.128 and a maximum community size of 2, nestedness does not appear to play a key role in this network, either.

Comparison with general community detection algorithms

Now, we compare the nested community structure found by our algorithm with the output of traditional community detection algorithms Linkcomm (Ahn et al. 2010), MOSES (McDaid and Hurley 2010) and CFinder (Adamcsek et al. 2006). These algorithms use different approaches for community detection, Linkcomm being a link partitioning method, MOSES a fuzzy algorithm, and CFinder a clique search algorithm. Note that

while Linkcomm, MOSES and CFinder propose to find communities where vertices within communities are more densely connected, our algorithm defines communities as fully nested subgraphs. While the community definitions are different, we perform these comparisons to highlight the differences between nested and traditional community structures.

Since our algorithm assigns each vertex to a community—vertices that don't belong anywhere else are put into their own communities – we modify the outputs of the community detection algorithms mentioned above so that omitted vertices are also assigned a community. This helps us to level the comparisons and also to avoid making false conclusions based on the number of communities, since a single community is only possible if all vertices are included in it.

Looking at the number of communities and their average sizes (Fig. 16a, b), we can see that MOSES tends to identify fewer but occasionally larger communities, while CFinder created many—sometimes as many as $4n$ communities—, although its average community size was not the smallest in these cases either, meaning that there had to be greater overlap between them. This is confirmed by the vertex presence in Fig. 16c. Vertex presence was low across all graphs and algorithms, which means that overall there is little overlap between the communities. The different community structures detected on Zachary's karate club network are shown in Fig. 17.

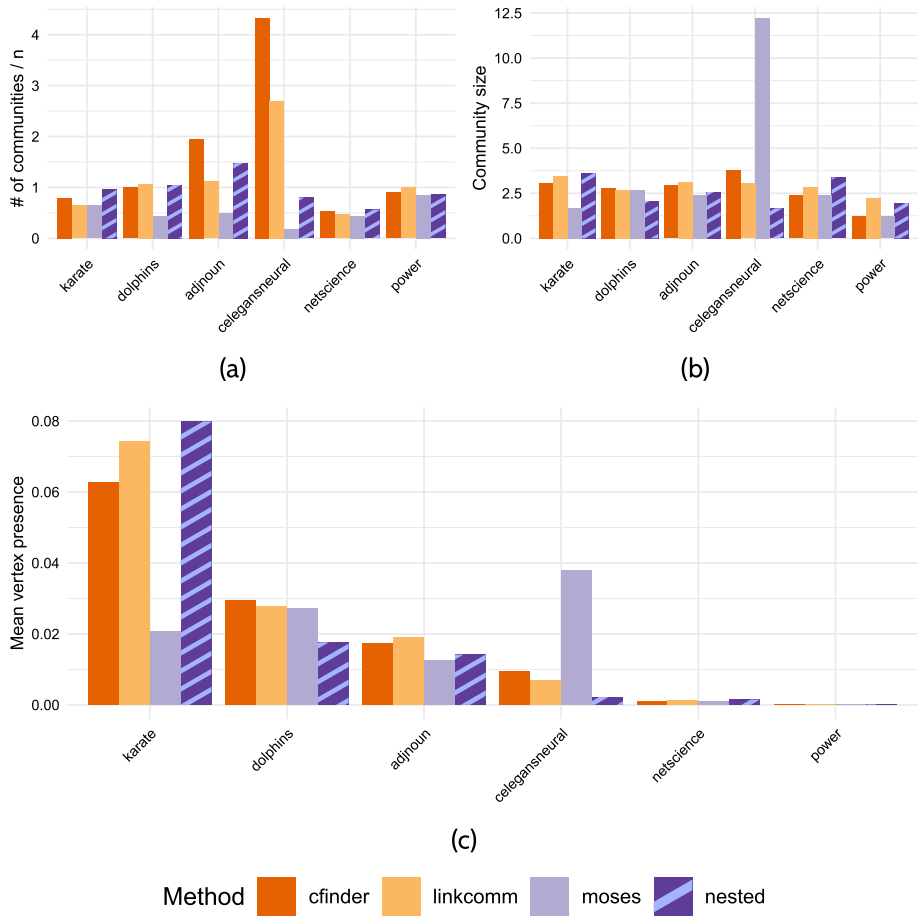

Fig. 16 Comparison of community statistics across algorithms and graphs

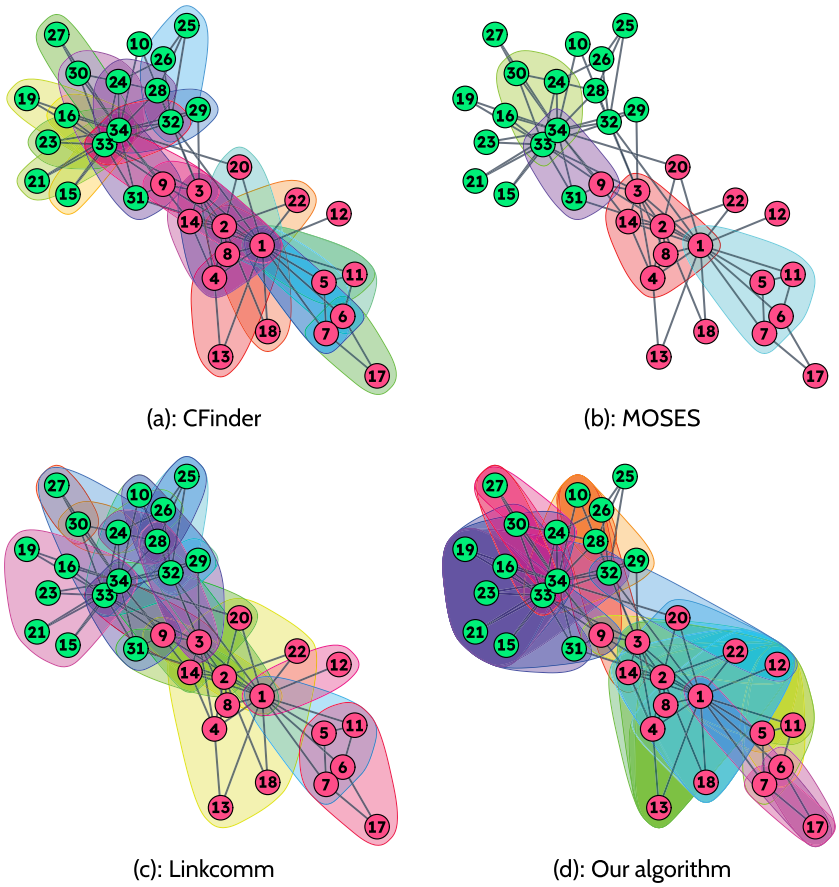

Fig. 17 Community structures detected by different algorithms on Zachary's karate club network. Communities with a single vertex are not plotted

Finally, we also calculate the Generalized Conventional Normalized Mutual Information (Lutov et al. 2019) (GenConvNMI) index to measure Mutual Information between the community structure of our algorithm and the compared other algorithms. The NMI indices shown in Table 2 are high, especially in the case of CFinder. The full results are included in Additional file 3.

Table 2 Generalized Conventional Normalized Mutual Information (Lutov et al. 2019) between the communities of our algorithm and the compared traditional community detection algorithms (higher = more similar)

	NMI_M	NMI_C	NMI_L
adjnoun	0.686	0.911	0.538
celegansneural	0.567	0.678	0.657
dolphins	0.655	0.821	0.676
*karate	0.645	0.699	0.533
netscience	0.941	0.953	0.949
power	0.963	0.974	0.927
dswomen	0.842	0.836	0.530
*families	0.879	0.894	0.724
les_miserables	0.576	0.698	0.617

The subscripts M, C and L correspond to the algorithms MOSES, CFinder and Linkcomm, respectively. Graphs marked with an asterisk (*) were directly mentioned and analyzed in the article

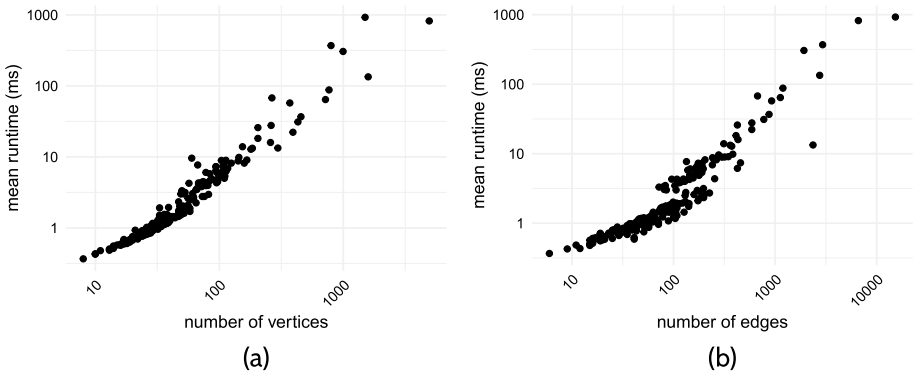

Fig. 18 Average execution times of the nested community detection algorithm's implementation over 100 runs

Implementation and performance benchmarks

Finally, we show that our algorithm is scalable enough to be used on large networks. It was implemented in R (R Core Team 2023) and C++ and its reference implementation, together with the example generator code, is open source.[2] Here, we measure the runtime of our algorithm across all analyzed non-synthetic graphs.

Figure 18 shows average runtimes of 100 executions on each of the analyzed bipartite and non-bipartite networks. The figure shows that while it is not perfectly linear, the algorithm's runtime doesn't scale steeply with the increase of the vertices or edges. We can see that it is capable of achieving runtimes of under a second on graphs with much more than 1000 vertices, or more than 10,000 edges. For the details of the test environment, please refer to "Appendix A".

Conclusions

We introduced a novel constrained community detection algorithm that finds overlapping nested subgraphs of a given input graph and constructs a directed graph, called the community graph, representing this community structure in a compact form. In reverse, we also introduced an algorithm that generates bipartite graphs with any given nested community structure from the input community graph. Derived from the resulting community graph, we also introduced two metrics to measure nestedness in networks on both graph- and vertex levels. We have shown that our community detection method can uncover detailed nested relationships within the input graph. We demonstrated its capabilities through several benchmark networks and real-world networks as well. Finally, we compared our method with multiple commonly used community detection algorithms that are able to find overlapping communities. We showed that our method can reveal a different type of community structure in both bipartite and non-bipartite graphs.

[2] https://github.com/Hanziness/r-nested-comms/releases/tag/v0.2.

Appendix A: Test environment

The performance tests were carried out on an ASUS ExpertCenter computer with an Intel® Core™ 7-10700 CPU @ 2.90GHz (16 cores) CPU and 16 GB of RAM. The system was running Manjaro Linux (kernel version 6.3.5-2-MANJARO (64-bit)) with R 4.3.1 (latest packages as of 2023-07-01), compiled with the Intel Math Kernel Library (MKL). Execution time was measured using the `microbenchmark` package.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1007/s41109-023-00575-2.

Additional file 1. Source code of the community detection package `nested.comms`: This is an R package that can be built and installed using the devtools package. To use it, one can call the `install_local` function from `devtools` on the source zip file. The installed package (`nested.comms`) exposes functions to perform nested community detection (described in the "Nestedness and community detection" section) using the `nested_node_comm` function, and test graph generation ("Generation of overlapping communities" section) through the `generate_benchmark_random` function.

Additional file 2. Generated networks: The 2000 generated networks used in the "Results on benchmarks" section are included in a compressed archive. The final networks are named `gen_id.csv`, their base community graphs are named `base_id.csv`, and the ground-truth community list is included as `truth_id.txt`, where id is the identifier of the generated graph and each line corresponds to one community. Both the final and the base graphs are given in the form of edge lists. The final graph is always a bipartite graph.

Additional file 3. Computational results: A CSV file containing all the computed properties of all the non-synthetic graphs we have analyzed, including graphs and properties omitted for compactness from Table 1.

Acknowledgements
The authors would like to express their gratitude to András Pluhár for his useful insights and to the reviewers for their detailed suggestions and comments. All of them have contributed to a significant improvement of the quality of the article.

Author contributions
IG designed and implemented both algorithms and carried out the experiments using them. AL helped in formalizing and structuring multiple aspects of the paper. The authors jointly wrote and reviewed the manuscript.

Funding
Open access funding provided by University of Szeged. This work was supported by the National Research, Development and Innovation Office-NKFIH Fund No. SNN-135643. This work was supported by the University of Szeged Open Access Fund No. 6267.

Availability of data and materials
The pollination and host-parasite datasets analyzed during the current study are available in the Web of Life repository at www.web-of-life.es. The analyzed non-bipartite networks are available in their respective published articles: Zachary's karate club (Zachary 1977), Florentine families network (Breiger and Pattison 1986). The other graphs not analyzed in detail, but on which we performed calculations: coappearance network of characters in the novel ***Les Misérables*** (`les_miserables`) (Knuth 1993), adjacency network of common adjectives and nouns in the novel ***David Copperfield*** by Charles Dickens (`adjnoun`) (Newman 2006), network representing the neural network of C. Elegans (`celegansneural`) (Watts and Strogatz 1998), social network of frequent associations between dolphins (`dolphins`) (Lusseau et al. 2003), coauthorships in network science (`netscience`) (Newman 2006), network representing the topology of the Western States Power Grid (`power`) (Watts and Strogatz 1998), Davis Southern women social network (`dswomen`) (Davis et al. 1941). The generated random bipartite networks (along with their base community graphs and ground truths) analyzed in "Results on benchmarks" section are included in this published article and its supplementary information files.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 28 April 2023 Accepted: 18 July 2023
Published: 3 August 2023

References

Adamcsek B, Palla G, Farkas IJ, Derényi I, Vicsek T (2006) Cfinder: locating cliques and overlapping modules in biological networks. Bioinformatics 22(8):1021–1023. https://doi.org/10.1093/bioinformatics/btl039

Adcock AB, Sullivan BD, Mahoney MW (2013) Tree-like structure in large social and information networks. In: 2013 IEEE 13th international conference on data mining. IEEE, pp 1–10. https://doi.org/10.1109/icdm.2013.77
Ahn Y-Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466(7307):761–764. https://doi.org/10.1038/nature09182
Almeida-Neto M, Ulrich W (2011) A straightforward computational approach for measuring nestedness using quantitative matrices. Environ Model Softw 26(2):173–178. https://doi.org/10.1016/j.envsoft.2010.08.003
Ángel Rodríguez-Gironés M, Santamaría L (2010) How foraging behaviour and resource partitioning can drive the evolution of flowers and the structure of pollination networks. Open Ecol J 3:1–11. https://doi.org/10.2174/1874213001003040001
Bartomeus I, Vilà M, Santamaría L (2008) Contrasting effects of invasive plants in plant–pollinator networks. Oecologia 155(4):761–770. https://doi.org/10.1007/s00442-007-0946-1
Bascompte Lab (2014) Web of Life: ecological networks database. https://www.web-of-life.es
Bascompte J (2010) Structure and dynamics of ecological networks. Science 329(5993):765–766. https://doi.org/10.1126/science.1194255
Bastolla U, Fortuna MA, Pascual-García A, Ferrera A, Luque B, Bascompte J (2009) The architecture of mutualistic networks minimizes competition and increases biodiversity. Nature 458(7241):1018–1020. https://doi.org/10.1038/nature07950
Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 2008(10):10008. https://doi.org/10.1088/1742-5468/2008/10/p10008
Bóta A, Csizmadia L, Pluhár A (2010) Community detection and its use in real graphs. In: Proceedings of the conference on applied theoretical computer science (MATCOS), pp 95–99
Breiger RL, Pattison PE (1986) Cumulated social roles: the duality of persons and their algebras. Soc Netw 8(3):215–256. https://doi.org/10.1016/0378-8733(86)90006-7
Brualdi RA, Sanderson JG (1999) Nested species subsets, gaps, and discrepancy. Oecologia 119:256–264. https://doi.org/10.1007/s004420050784
Bustos S, Gomez C, Hausmann R, Hidalgo CA (2012) The dynamics of nestedness predicts the evolution of industrial ecosystems. PLoS ONE 7(11):49393. https://doi.org/10.1371/journal.pone.0049393
Csermely P, London A, Wu L-Y, Uzzi B (2013) Structure and dynamics of core/periphery networks. J Complex Netw 1(2):93–123. https://doi.org/10.1093/comnet/cnt016
Darlington PJ (1943) Carabidae of mountains and islands: data on the evolution of isolated faunas, and on atrophy of wings. Ecol Monogr 13(1):37–61. https://doi.org/10.2307/1943589
Davis A, Gardner BB, Gardner MR (1941) Deep South. The University of Chicago Press, Chicago
Ermann L, Shepelyansky DL (2013) Ecological analysis of world trade. Phys Lett A 377(3–4):250–256. https://doi.org/10.1016/j.physleta.2012.10.056
Gera I, London A, Pluhár A (2022) Greedy algorithm for edge-based nested community detection. In: 2022 IEEE 2nd conference on information technology and data science (CITDS), pp 86–91. https://doi.org/10.1109/CITDS54976.2022.9914051
Junttila E, Kaski P (2011) Segmented nestedness in binary data. In: Proceedings of the 2011 SIAM international conference on data mining. SIAM, pp 235–246. https://doi.org/10.1137/1.9781611972818.21
Knuth DE (1993) The Stanford GraphBase: a platform for combinatorial computing, vol 1. AcM Press, New York
Kohler GU (2011) Redes de interação Planta Beija-Flor em um Gradiente Altitudinal de Floresta Atlântica no Sul do Brasil
London A, Martin RR, Pluhár A (2022) Graph clustering via generalized colorings. Theor Comput Sci 918:94–104. https://doi.org/10.1016/j.tcs.2022.03.023
Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: can geographic isolation explain this unique trait? Behav Ecol Sociobiol 54:396–405. https://doi.org/10.1007/s00265-003-0651-y
Lutov A, Khayati M, Cudré-Mauroux P (2019) Accuracy evaluation of overlapping and multi-resolution clustering algorithms on large datasets. In: 2019 IEEE international conference on big data and smart computing (BigComp). IEEE, pp 1–8. https://doi.org/10.1109/bigcomp.2019.8679398
Mariani MS, Ren Z-M, Bascompte J, Tessone CJ (2019) Nestedness in complex networks: observation, emergence, and implications. Phys Rep 813:1–90. https://doi.org/10.1016/j.physrep.2019.04.001
McDaid A, Hurley N (2010) Detecting highly overlapping communities with model-based overlapping seed expansion. In: 2010 International conference on advances in social networks analysis and mining. IEEE, pp 112–119. https://doi.org/10.1109/ASONAM.2010.77
McGlohon M, Akoglu L, Faloutsos C (2011) Statistical properties of social networks. In: Social network data analytics. Springer, pp 17–42. https://doi.org/10.1007/978-1-4419-8462-3_2
Newman ME (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74(3):036104. https://doi.org/10.1103/physreve.74.036104
Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113. https://doi.org/10.1103/PhysRevE.69.026113
Patterson BD, Atmar W (1986) Nested subsets and the structure of insular mammalian faunas and archipelagos. Biol J Lin Soc 28(1–2):65–82. https://doi.org/10.1111/j.1095-8312.1986.tb01749.x
Payrató-Borràs C, Hernández L, Moreno Y (2020) Measuring nestedness: a comparative study of the performance of different metrics. Ecol Evol 10(21):11906–11921. https://doi.org/10.1002/ece3.6663
R Core Team (2023) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna. https://www.R-project.org/
Saavedra S, Reed-Tsochas F, Uzzi B (2009) A simple model of bipartite cooperation for ecological and organizational networks. Nature 457(7228):463–466. https://doi.org/10.1038/nature07532
Schaeffer SE (2007) Graph clustering. Comput Sci Rev 1(1):27–64. https://doi.org/10.1016/j.cosrev.2007.05.001
Solé-Ribalta A, Tessone CJ, Mariani MS, Borge-Holthoefer J (2018) Revealing in-block nestedness: detection and benchmarking. Phys Rev E 97(6):062302. https://doi.org/10.1103/PhysRevE.97.062302

Ulrich W, Almeida-Neto M, Gotelli NJ (2009) A consumer's guide to nestedness analysis. Oikos 118(1):3–17. https://doi.org/10.1111/j.1600-0706.2008.17053.x
Uzzi B (1996) The sources and consequences of embeddedness for the economic performance of organizations: the network effect. Am Sociol Rev 61(4):674–698. https://doi.org/10.2307/2096399
Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. nature 393(6684):440–442. https://doi.org/10.1038/30918
Wright DH, Patterson BD, Mikkelson GM, Cutler A, Atmar W (1997) A comparative analysis of nested subset patterns of species composition. Oecologia 113:1–20. https://doi.org/10.1007/s004420050348
Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193. https://doi.org/10.1007/s40745-015-0040-1
Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(4):452–473. https://doi.org/10.1086/jar.33.4.3629752
Zhu Z, Cerina F, Chessa A, Caldarelli G, Riccaboni M (2014) The rise of China in the international trade network: a community core detection approach. PLoS ONE 9(8):105496. https://doi.org/10.1371/journal.pone.0105496

Publisher's Note