Concas *et al. Appl Netw Sci* (2021) 6:39
https://doi.org/10.1007/s41109-021-00377-4

Applied Network Science

RESEARCH Open Access

Chained graphs and some applications

Anna Concas[1*], Lothar Reichel[2], Giuseppe Rodriguez[1] and Yunzi Zhang[2]

*Correspondence:
anna.concas@unica.it
[1] Department of Mathematics and Computer Science, University of Cagliari, via Ospedale 72, 09124 Cagliari, Italy
Full list of author information is available at the end of the article

Abstract

This paper introduces the notions of chained and semi-chained graphs. The chain of a graph, when existent, refines the notion of bipartivity and conveys important structural information. Also the notion of a center vertex v_c is introduced. It is a vertex, whose sum of p powers of distances to all other vertices in the graph is minimal, where the distance between a pair of vertices $\{v_c, v\}$ is measured by the minimal number of edges that have to be traversed to go from v_c to v. This concept extends the definition of closeness centrality. Applications in which the center node is important include information transmission and city planning. Algorithms for the identification of approximate central nodes are provided and computed examples are presented.

Keywords: Network analysis, Multipartition, Chained graph, Central node

Introduction

Complex systems are made up of a collection of objects, that are connected to each other in some manner, and can be modeled as networks. The objects often are referred to as *nodes* or *vertices*, and the connections as *edges*. The nature of the vertices and edges may vary depending on the system being modeled. While networks ignore many properties of the system they model, they nevertheless capture some of its complexity in a way that facilitates the analysis of its properties. Networks can be applied to model systems that arise in social science, telecommunication, transportation, as well as in many other areas; see, e.g., Estrada (2011a), Estrada and Knight (2015), and Newman (2010) for discussions on networks and for many examples of their applications.

A network is represented by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}_{i=1}^n$ denotes a set of vertices or nodes, and $\mathcal{E} = \{e_i\}_{i=1}^m$ is a set of edges between the vertices. Two vertices v_i and v_j, with $i \neq j$, are said to be *adjacent* if there is an edge between them. We consider unweighted connected simple graphs, i.e., connected undirected unweighted graphs without multiple edges and self-loops. Networks are studied by algebraic and computational methods applied to the graphs that represent them. Questions of interest include the determination of the most important vertices and edges of a network, as well as the identification of structural properties. Fundamental topological properties, which will be defined and used in Sect. 2, are bipartivity and, more generally, multipartivity. An m-partite network involves objects that can be split into m disjoint groups V_i, $i = 1, 2, \ldots, m$, called partite sets, with connections occurring only across, but not within, the groups. A survey of

mathematical properties and applications of bipartite graphs in the areas of algebra, combinatorics, chemistry, communication networks, and computer science are provided by Asratian et al. (1998).

The notion of multipartite graphs is required in the definition of chained graphs introduced in this paper. The chained structure characterizes multipartite networks such that edges can occur only between nodes belonging to "subsequent" partite sets V_i and V_{i+1}, $i = 1, 2, \ldots, m-1$, and vice versa, as illustrated in Fig. 1. The definition of chained graphs can be relaxed allowing connections between nodes belonging to the same node subset, as it will be subsequently explained.

The above concepts will be described in detail in Sect. 2, where it will be shown that bipartite graphs are ℓ-chained for some $\ell \geq 2$. This shows that the chained structure is a refinement of bipartivity, since it reveals additional structure of a bipartite graph. The chains provide insight into how the vertices are connected; this structure is not uncovered by bipartivity only.

We also will use chained graphs to identify "central nodes". These are nodes determined by their location in the chain structure, incorporating a different idea of centrality than other centrality measures, such as the degree or the subgraph centrality. A nice introduction to the latter measure is provided by Estrada and Higham (2010); see also Borgatti (2005), Estrada (2011a), and Estrada and Knight (2015) for an overview of other important quantities that describe global properties of a given graph, such as the importance of a particular node within the network, or the ease of traveling from one node to another. With the aim of determining a new centrality measure, called "position centrality", we will first examine the spanning trees associated with a given underlying graph (Bapat 2014; Bondy and Murty 1976). The position centrality of a node will be defined by taking into account the lengths of the paths from it to all the other vertices and it can be computed by using the chained structure determined by the tree rooted at the node. By using this measure, which depends upon a parameter p, one may identify a most "centrally located" node, referred to as a "center vertex", as a vertex with the smallest position centrality. There may be more than one center vertex. For $p = 1$, a center vertex coincides with a vertex with the largest closeness centrality (Newman 2010).

Another application of interest to us is the detection of anti-communities, i.e., subsets $\{\mathcal{S}_i\}_{i=1}^{p}$ of vertices of a graph $\mathcal{G}$ with no or few edges between vertices in each set $\mathcal{S}_i$, but many connections between the node sets $\mathcal{S}_i$ and $\mathcal{V}\backslash\mathcal{S}_i$, $i = 1, 2, \ldots, p$. Once a semi-chained structure has been identified in a graph, the presence of anti-communities can be determined by ascertaining the number of edges among nodes belonging to the same set; see, e.g., the autobahn data set and Fig. 16 in Sect. 6. Community and anti-community detection in networks is an important problem with applications in various fields, including physics, computer science, as well as in the natural and social sciences. Several methods have been developed to identify this kind of structures in networks; see, e.g., Chen et al. (2014) and

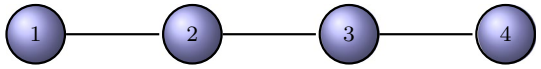

Fig. 1 A chained graph with 4 enumerated node sets

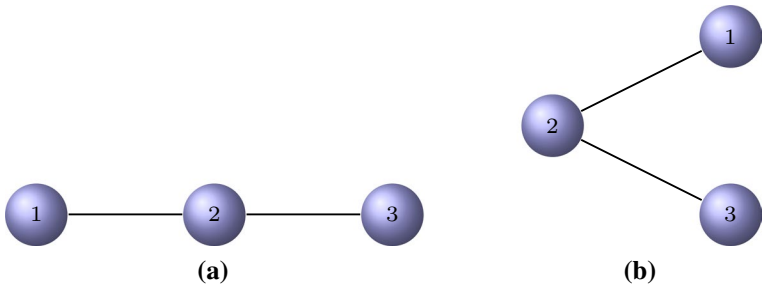

Fig. 2 Example 2: chained graph with vertices v_1, v_2, and v_3. The initial vertex set in (**a**) is $\mathcal{V}_1 = \{v_1\}$, while the initial vertex set in (**b**) is $\mathcal{V}_1 = \{v_2\}$

Raghavan et al. (2007). In Fasino and Tudisco (2017) a spectral method was used to simultaneously detect communities and anti-communities, while in Concas et al. (2020) another approach to identifying anti-communities has been described. We will illustrate the benefit of using the chained structure for this purpose in Sect. 6.

This paper is organized as follows: Sect. 2 introduces notation that will be used in the remainder of the paper and discusses ℓ-chained bipartite graphs. Section 3 describes the structure of the adjacency matrices that are associated to ℓ-chained graphs. The relation between the chain structure and spanning trees is investigated in Sect. 4. Section 5 introduces the notion of position centrality and discusses some applications. Numerical illustrations of ℓ-chained graphs and the identification of approximations of central nodes are described in Sects. 6 and 7 contains concluding remarks.

Some definitions

This section introduces notation and definitions to be used in the sequel. Most of our definitions and terminology follow those in Estrada (2011a), Newman (2010). The adjacency matrix $M = [m_{ij}]_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$ associated with an unweighted undirected simple graph $\mathcal{G}$ with n vertices is symmetric and has the entry $m_{ij} = 1$ if there is an edge between the vertices v_i and v_j, otherwise $m_{ij} = 0$.

Bipartivity, and more generally multipartivity, are interesting structural properties of a graph that provide important information about the network being modeled. There are various characterizations of multipartite graphs (Estrada and Gómez-Gardeñes 2016; König 1916). They can be defined as follows.

Definition 1 A graph $\mathcal{G}$ is said to be *ℓ-partite* if the set of vertices $\mathcal{V}$ that make up the graph can be partitioned into ℓ disjoint non-empty subsets $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ such that every vertex in $\mathcal{V}_i$, for any $1 \le i \le \ell$, is adjacent only to vertices in $\mathcal{V}_j$ for some $j \ne i$, and the number of subsets, ℓ, is as small as possible. A graph is said to be bipartite when $\ell = 2$, and multipartite when $\ell \ge 3$.

Equivalently, the vertices of an ℓ-partite graph can be colored with ℓ colors, so that the vertices at the endpoints of every edge have different colors, and ℓ is the minimal number of colors required (Jensen and Toft 1995).

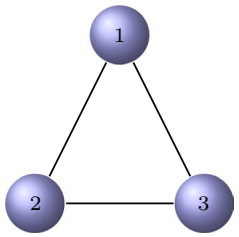

Fig. 3 Example 3: a semi-chained graph

Example 1 The graph on the right-hand side of Fig. 2 is bipartite, and the graph in Fig. 3 is tripartite.

Usually, vertices in distinct subsets $\mathcal{V}_i$ of an ℓ-partite graph model different entities. For instance, users of social bookmarking services, such as Delicious (http://www.delicious.com), put tags on web pages. Users, tags, and web pages can be represented by a tripartite network $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$, in which users define the vertex subset $\mathcal{V}_1$, tags define the subset $\mathcal{V}_2$, and web pages define the subset $\mathcal{V}_3$. This example of tripartite graphs is discussed by Ikematsu et al. (2013).

There are various methods for partitioning the vertex set $\mathcal{V}$ of a bipartite graph $\mathcal{G}$ into unique disjoint non-empty subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, such that every vertex in $\mathcal{V}_1$ is adjacent to a vertex in $\mathcal{V}_2$; see Bondy and Murty (1976) and Concas et al. (2020) for discussions of methods and further references. Assume for the moment that the n vertices in the set $\mathcal{V}$ are enumerated so that the first n_1 of them make up the vertex set $\mathcal{V}_1$ and the remaining $n_2 = n - n_1$ vertices make up the vertex set $\mathcal{V}_2$. Then the adjacency matrix for $\mathcal{G}$ is of the form

$$M = \begin{bmatrix} O & B \\ B^T & O \end{bmatrix}, \tag{2.1}$$

where $B \in \mathbb{R}^{n_1 \times n_2}$, O denotes a zero-matrix of suitable order, and the superscript ^{T} denotes transposition. A bipartite graph with partition sets $\mathcal{V}_1$ and $\mathcal{V}_2$ is said to be *complete* if every vertex of $\mathcal{V}_1$ is adjacent to all vertices of $\mathcal{V}_2$. For complete bipartite graphs, every entry of the submatrix B of the adjacency matrix (2.1) is one. The notion of a complete bipartite graph can be extended to multipartite graphs.

Definition 2 An ℓ-partite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the vertex set $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ partitioned into non-empty disjoint subsets $\mathcal{V}_i$ is said to be *complete* if, for each $1 \leq i \leq \ell$, every vertex in the vertex subset $\mathcal{V}_i$ is adjacent to every vertex in the set $\mathcal{V} \backslash \mathcal{V}_i$.

Complete ℓ-partite graphs are commonly denoted by $\mathcal{K}_{n_1,n_2,\ldots,n_\ell}$, where n_i is the cardinality of the node subset $\mathcal{V}_i$. The adjacency matrix M for $\mathcal{K}_{n_1,n_2,\ldots,n_\ell}$ is of order $n = \sum_{j=1}^{\ell} n_j$ with all entries m_{ij} equal to one, except for the entries of ℓ disjoint diagonal blocks of zeros of orders $n_1, n_2, \ldots, n_\ell$.

The following definitions introduce the notions of particular multipartite structures, which will be used in the remainder of the paper.

Definition 3 An undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is said to be ℓ_i-*chained* with initial vertex v_i if the set of vertices can be subdivided into ℓ_i disjoint non-empty subsets

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_{\ell_i} \tag{2.2}$$

such that $v_i \in \mathcal{V}_1$, and all vertices in the set $\mathcal{V}_j$ are adjacent only to vertices in the sets $\mathcal{V}_{j-1}$ or $\mathcal{V}_{j+1}$ for $j = 2, 3, \ldots, \ell_i - 1$, where the chain length ℓ_i is the largest number of vertex subsets $\mathcal{V}_j$ with this property. Moreover, the vertices in $\mathcal{V}_1$ and $\mathcal{V}_{\ell_i}$ are adjacent only to vertices in $\mathcal{V}_2$ and $\mathcal{V}_{\ell_i-1}$, respectively. Vertex sets $\mathcal{V}_j$ with consecutive indices are said to be adjacent.

In the Delicious bookmarking service application mentioned above, vertices in $\mathcal{V}_1$ and $\mathcal{V}_3$ are adjacent only to vertices in $\mathcal{V}_2$. Thus, this vertex partitioning shows that the graph is 3-chained.

Definition 4 The graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is said to be ℓ_i-*semi-chained* with initial vertex v_i if the set of vertices can be subdivided into ℓ_i disjoint non-empty subsets (2.2) such that $v_i \in \mathcal{V}_1$, and all vertices in the set $\mathcal{V}_j$ are adjacent only to vertices in the sets $\mathcal{V}_{j-1}$, $\mathcal{V}_j$, or $\mathcal{V}_{j+1}$ for $j = 2, 3, \ldots, \ell_i - 1$, where the chain length ℓ_i is the largest number of vertex subsets $\mathcal{V}_j$ with this property. Moreover, the vertices in $\mathcal{V}_1$ and $\mathcal{V}_{\ell_i}$ are adjacent only to vertices in $\mathcal{V}_1 \cup \mathcal{V}_2$ and $\mathcal{V}_{\ell_i-1} \cup \mathcal{V}_{\ell_i}$, respectively.

Example 2 Figure 2 displays two chained graphs with three vertices and different initial vertices. In the chained graph displayed in subfigure (a), each vertex set $\mathcal{V}_i$, $i = 1, 2, 3$, contains one node, and the initial vertex is v_1. This gives the chain length $\ell_1 = 3$. The same chain length can be obtained if the initial node is chosen to be v_3. The chained graph in (b) has initial vertex v_2, with $\mathcal{V}_1 = \{v_2\}$ and $\mathcal{V}_2 = \{v_1, v_3\}$, which gives the chain length $\ell_2 = 2$. This example illustrates that the chain length depends on the initial vertex chosen.

Example 3 Figure 3 displays a 1-semi-chained graph, that is not chained in the sense of Definition 3. The semi-chained structure in this example is independent of the choice of the initial vertex.

While chained structure is not so common for graphs, every non-trivial graph is semi-chained. Nevertheless, representing a graph in (semi-)chained form is useful, because this structure is closely linked to anti-communities, which are subsets of vertices, such that there are only few edges between vertices in the same subset, but many edges between vertices in different subsets. Recent discussions on anti-communities and their detection can be found in Concas et al. (2020), Estrada and

Gómez-Gardeñes (2016) , Estrada and Knight (2015), Fasino and Tudisco (2017). We will introduce a density measure for anti-communities, which is similar to the intra-cluster density that allows one to identify clusters or communities; see Estrada (2011b), Estrada and Knight (2015), and Fortunato (2010).

Definition 5 The anti-community score $0 \leq \rho \leq 1$ is the ratio between the number of edges connecting the nodes in the subset and the maximum admissible number of edges between them.

To highlight the role of the anti-community score, we will in the following consider ρ-anti-communities. The sets $\mathcal{V}_i$, $i = 1, \ldots, \ell$, in an ℓ-chained graph are 0-anti-communities, as they have no internal edges, while each set $\mathcal{V}_i$ in a semi-chained graph is a ρ_i-anti-community. When ρ_i is small, $\mathcal{V}_i$ may be considered an anti-community.

Definition 6 The maximal chain length, ℓ, of a graph is defined as

$$\ell = \max_i \ell_i,$$

where the maximum is over all the initial nodes v_i in the vertex set $\mathcal{V}$. When the maximal chain length is considered, the graph is said to be ℓ-chained.

Example 4 The graph $\mathcal{G}$ of Example 2 has maximal chain length $\ell = 3$.

The following notion will be useful in the sequel. It is stronger than (standard) multipartivity, but weaker than complete multipartivity.

Definition 7 An ℓ-partite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the vertex set partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ into non-empty disjoint subsets $\mathcal{V}_i$ is said to be *strongly ℓ-partite* if, for every i, every vertex in the subset $\mathcal{V}_i$ is adjacent to at least one vertex in every subset $\mathcal{V}_j$, $j \neq i$.

The special case of strongly tripartite graphs is applied to community detection by Ikematsu et al. (2013), who refer to these graphs as 3-partite 3-uniform hypernetworks. We also define the notion of strongly ℓ-chained graphs.

Definition 8 An ℓ-chained graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the vertex set partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ into non-empty disjoint subsets $\mathcal{V}_i$ is said to be *strongly ℓ-chained* if, for every i, every vertex in the subset $\mathcal{V}_i$ is adjacent to at least one vertex in the subsets $\mathcal{V}_{i-1}$ (for $1 < i \leq \ell$) and $\mathcal{V}_{i+1}$ (for $1 \leq i < \ell$).

We are interested in strongly chained graphs, because their structure can be identified from the knowledge of the vertex and edge sets of a graph. We note that "standard" chained graphs $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ cannot be uniquely identified from the knowledge of $\mathcal{V}$ and $\mathcal{E}$. Indeed, let the vertex v of a chained graph be connected only to vertices in the vertex set $\mathcal{V}_i$ for some $1 < i < \ell$. Then v may belong to either the vertex sets $\mathcal{V}_{i-1}$ or $\mathcal{V}_{i+1}$.

It is remarkable that an ℓ-chained graph is always bipartite, and vice versa. This property will help us study anti-communities.

Theorem 1 *Let $\mathcal{G}$ be a bipartite graph. Then the graph is ℓ-chained for some $\ell \geq 2$. The partitioning of the node set $\mathcal{V}$ into chained sets is not unique, but the maximal number of chained sets, $\ell_{\max}$, is uniquely determined. Conversely, if a graph is ℓ-chained, then it is bipartite.*

Proof Let the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be bipartite and let $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ be the associated partitioning. It follows that the graph is at least 2-chained. Conversely, let the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be ℓ-chained, i.e., there is a partitioning of the vertex set $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ that satisfies the properties of Definition 3. Then letting

$$\widetilde{\mathcal{V}}_1 = \bigcup_{j \text{ odd}} \mathcal{V}_j \qquad \text{and} \qquad \widetilde{\mathcal{V}}_2 = \bigcup_{j \text{ even}} \mathcal{V}_j, \tag{2.3}$$

shows that the graph $\mathcal{G}$ is bipartite with associated vertex set partitioning $\mathcal{V} = \widetilde{\mathcal{V}}_1 \cup \widetilde{\mathcal{V}}_2$. The unicity of $\ell_{\max}$ follows by recursive subdivision of the sets $\widetilde{\mathcal{V}}_1$ and $\widetilde{\mathcal{V}}_2$, and by a suitable choice of the initial set $\mathcal{V}_1$ in (2.2). □

The property of bipartite graphs shown by Theorem 1 will be further discussed in Sect. 3, where we consider the structure of adjacency matrices for ℓ-chained graphs for $\ell \geq 3$.

We remark that the ℓ-chained structure with $\ell > 2$ gives a finer representation of a bipartite graph, as it provides information on hierarchical connections between nodes that is not contained in the basic notion of bipartivity.

Closed chained graphs

This subsection considers chained graphs that may be cyclic. This kind of graphs are important, e.g., for their connection to n-cubes.

Definition 9 A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is said to be *closed ℓ_i-chained* with initial vertex v_i if the set of vertices can be subdivided into ℓ_i disjoint non-empty subsets

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_{\ell_i}$$

such that $v_i \in \mathcal{V}_1$ and all vertices in the set $\mathcal{V}_j$ are adjacent only to vertices in the sets $\mathcal{V}_{j-1}$ or $\mathcal{V}_{((j+1) \bmod \ell_i)}$ for $j = 1, 2, \ldots, \ell_i$, with $\mathcal{V}_0 \equiv \mathcal{V}_{\ell_i}$, where the chain length, ℓ_i, is the largest number of vertex subsets $\mathcal{V}_j$ with this property. *Closed ℓ_i-semi-chained* graphs can be defined analogously.

We remark that a closed ℓ-chained graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is not ℓ-chained, but may be k-chained for some $k < \ell$. The following example illustrates this.

Example 5 Consider the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in Fig. 4a and define the vertex subsets $\mathcal{V}_i = \{v_i\}$ for $i = 1, 2, \ldots, 6$. This graph is closed 6-chained with initial vertex v_1.

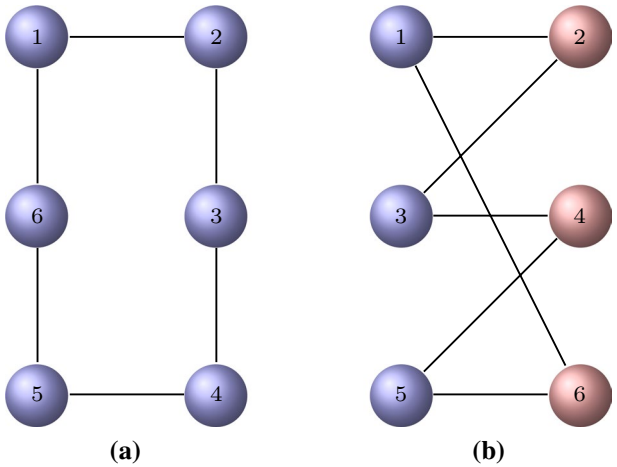

Fig. 4 **a**: a closed chained graph with 6 enumerated node sets (each set contains one node). **b**: the bipartite structure $\widetilde{\mathcal{V}}_1 = \mathcal{V}_1 \cup \mathcal{V}_3 \cup \mathcal{V}_5$ and $\widetilde{\mathcal{V}}_2 = \mathcal{V}_2 \cup \mathcal{V}_4 \cup \mathcal{V}_6$

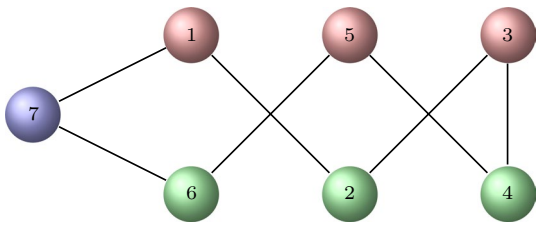

Fig. 5 A closed 7-chained graph is tripartite. A tripartization of the node set is given by $\widetilde{\mathcal{V}}_1 = \mathcal{V}_1 \cup \mathcal{V}_3 \cup \mathcal{V}_5$, $\widetilde{\mathcal{V}}_2 = \mathcal{V}_2 \cup \mathcal{V}_4 \cup \mathcal{V}_6$, $\widetilde{\mathcal{V}}_3 = \mathcal{V}_7$

Define the vertex subsets $\widetilde{\mathcal{V}}_1 = \{v_1\}$, $\widetilde{\mathcal{V}}_2 = \{v_2, v_6\}$, $\widetilde{\mathcal{V}}_3 = \{v_3, v_5\}$, and $\widetilde{\mathcal{V}}_4 = \{v_4\}$. The chain of vertex sets

$$\mathcal{V} = \widetilde{\mathcal{V}}_1 \cup \widetilde{\mathcal{V}}_2 \cup \widetilde{\mathcal{V}}_3 \cup \widetilde{\mathcal{V}}_4$$

shows that $\mathcal{G}$ is a 4-chained graph with initial vertex v_1. The graph also is bipartite. The latter property is illustrated by Fig. 4b.

Theorem 2 *A closed ℓ-chained graph $\mathcal{G}$ is $(\ell/2+1)$-chained if and only if ℓ is even. A closed ℓ-chained graph is $((\ell+1)/2)$-semi-chained if and only if ℓ is odd.*

Proof Let ℓ be even. Then we may partition the closed ℓ-chained graph $\mathcal{G}$ with vertices $v_1, v_2, \ldots, v_\ell$ as

$$\mathcal{V}_1 = \{v_1\}, \qquad \mathcal{V}_j = \{v_j, v_{\ell-j+2}\},\ j = 2, 3, \ldots, \ell/2, \qquad \mathcal{V}_{\frac{\ell}{2}+1} = \left\{v_{\frac{\ell}{2}+1}\right\}.$$

This shows that $\mathcal{G}$ is $(\ell/2+1)$-chained with initial vertex v_1.

If, instead, ℓ is odd, then we define the vertex sets

$$\mathcal{V}_1 = \{v_1\}, \qquad \mathcal{V}_j = \{v_j, v_{\ell-j+2}\},\ j = 2, 3, \ldots, (\ell-1)/2.$$

The remaining vertices, $v_{\frac{\ell+1}{2}}$ and $v_{\frac{\ell+3}{2}}$, are adjacent and make up the vertex set $\mathcal{V}_{\frac{\ell+1}{2}}$. This makes the graph $\mathcal{G}$ $((\ell+1)/2)$-semi-chained with initial vertex v_1. □

Example 6 Consider the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ shown in Fig. 5. Let $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_7$ with $\mathcal{V}_i = \{v_i\}$ for $i = 1, 2, \ldots, 7$. This partitioning shows that the graph is closed 7-chained with initial vertex v_1. The graph also is 4-semi-chained. Moreover, the graph is tripartite with tripartization $\mathcal{V} = \widetilde{\mathcal{V}}_1 \cup \widetilde{\mathcal{V}}_2 \cup \widetilde{\mathcal{V}}_3$, where $\widetilde{\mathcal{V}}_1 = \{v_1, v_3, v_5\}$, $\widetilde{\mathcal{V}}_2 = \{v_2, v_4, v_6\}$, and $\widetilde{\mathcal{V}}_3 = \{v_7\}$.

The following result shows that the facts that the graphs in Figs. 4 and 5 are bipartite and tripartite are not coincidences.

Theorem 3 *Consider a closed ℓ-chained graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with vertex set partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ and initial vertex v_1. Then the graph $\mathcal{G}$ is bipartite if ℓ is even. It is tripartite if ℓ is odd.*

Proof If ℓ is even, then the partition (2.3) produces a bipartite graph; see an example with $\ell = 6$ in Fig. 4. If ℓ is odd, then the partitioning

$$\widetilde{\mathcal{V}}_1 = \bigcup_{\substack{j \text{ odd} \\ j \neq \ell}} \mathcal{V}_j, \qquad \widetilde{\mathcal{V}}_2 = \bigcup_{j \text{ even}} \mathcal{V}_j, \qquad \text{and} \qquad \widetilde{\mathcal{V}}_3 = \mathcal{V}_\ell$$

shows that the graph is tripartite. An example with $\ell = 7$ is illustrated in Fig. 5. □

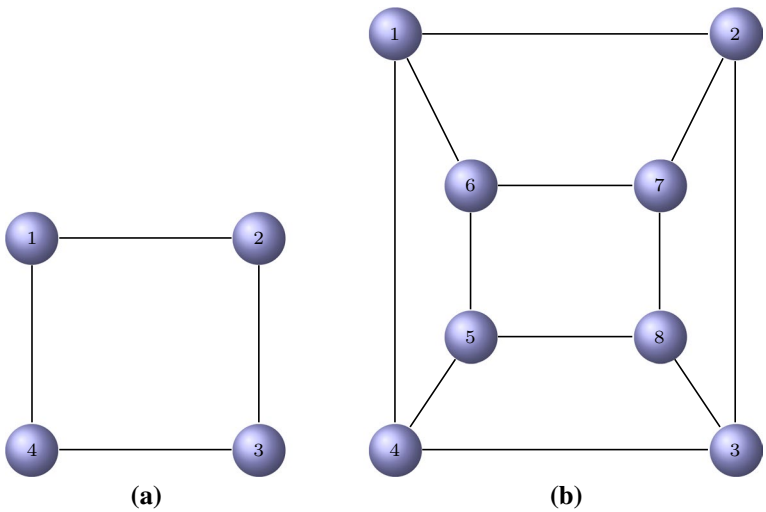

Fig. 6 **a** The 2-cube graph with an enumeration of the nodes. **b** The 3-cube graph with an enumeration of the nodes v_i

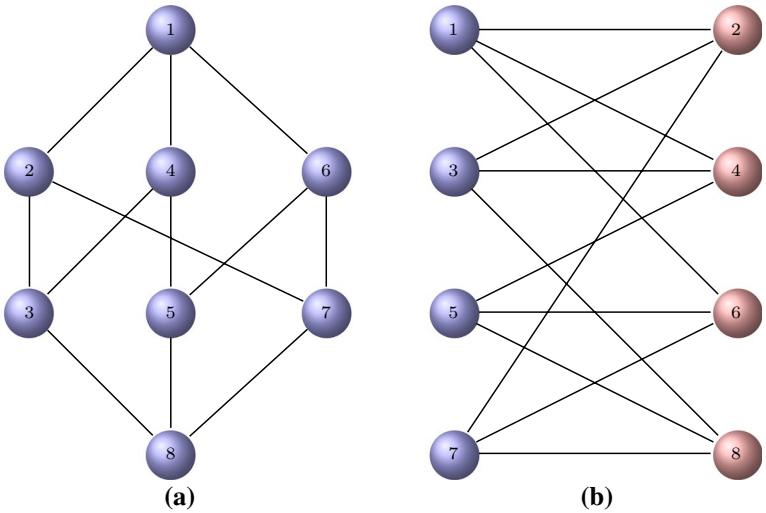

Fig. 7 **a**: the chained structure of the 3-cube graph $\mathcal{G}$. **b**: the bipartite structure of $\mathcal{G}$

Example 7 Regard the 2-cube graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with four vertices $\mathcal{V} = \{v_j\}_{j=1}^4$ displayed in Fig. 6a. The vertices are enumerated so that odd vertices are adjacent to even vertices, and vice versa. The graph $\mathcal{G}$ is bipartite with the partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where the set $\mathcal{V}_1$ contains all vertices with odd index, and $\mathcal{V}_2$ contains all vertices with even index. Moreover, the graph is closed 4-chained with initial vertex v_1, as well as 3-chained with initial vertex v_1. The latter is seen from the chain structure $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$, where $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{v_2, v_4\}$, and $\mathcal{V}_3 = \{v_3\}$.

Example 8 Consider the 3-cube graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ displayed in Fig. 6b. The vertices $v_1, v_2, \ldots, v_8$ are enumerated so that odd vertices are adjacent to even vertices, and vice versa. Hence, the graph $\mathcal{G}$ is bipartite with $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where the set $\mathcal{V}_1$ contains all vertices with odd index, and the set $\mathcal{V}_2$ contains all vertices with even index. The bipartite structure is illustrated in Fig. 7b.

To determine the closed chain structure with initial vertex v_1, we regard the vertex partitioning $\{v_1\} \cup \{v_2, v_4\} \cup \{v_3, v_5, v_7\} \cup \{v_6, v_8\}$, which shows that $\mathcal{G}$ has a closed 4-chained structure with initial vertex v_1. We note that the graph is not strongly chained, as v_8 is not connected to v_1.

The chain structure of the 3-cube gives rise to a different partitioning of the node set $\mathcal{V}$. Define the node subsets $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{v_2, v_4, v_6\}$, $\mathcal{V}_3 = \{v_3, v_5, v_7\}$, and $\mathcal{V}_4 = \{v_8\}$. The vertices in $\mathcal{V}_{i+1}$ are adjacent to the vertices in V_i for $i = 1, 2, 3$. Thus, the graph $\mathcal{G}$ is strongly 4-chained with initial vertex v_1. The chain structure is illustrated by the graph in Fig. 7a.

The above observations can be extended to n-cubes.

Definition 10 A 0-cube is made of just one vertex. An n-cube is composed by 2^n vertices. It is obtained recursively by taking two $(n-1)$-cubes, the first one with vertices v_i, $i = 1, 2, \ldots, 2^{n-1}$, and the second one with vertices v_i, $i = 2^{n-1} + 1, 2^{n-1} + 2, \ldots, 2^n$,

and connecting the vertex v_i in the first cube to the vertex with index $(i \mod 2^{n-1}) + 2^{n-1} + 1$ in the second cube.

The graph of an n-cube, with $n > 3$ is bipartite with $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where the set $\mathcal{V}_1$ contains all vertices with odd index, and the set $\mathcal{V}_2$ contains all vertices with even index. To determine the chain structure with starting vertex v_1, we regard the vertex partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3 \cup \mathcal{V}_4$, where $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{$all vertices adjacent to v_1 with even index$\}$, $\mathcal{V}_3 = \{$all vertices with odd index except $v_1\}$, and $\mathcal{V}_4 = \{$all vertices not adjacent to v_1 with even index$\}$. Thus, the graph is strongly 4-chained with initial vertex v_1. Given the symmetry of an n-cube with respect its nodes, changing the starting node does not modify the number and cardinality of the node sets. An n-cube with $n > 3$ is also closed 4-chained. This structure is not strongly chained. Moreover, it is not unique. It can be determined by considering the above chained node sets and move nodes from $\mathcal{V}_2$ to $\mathcal{V}_4$, without making the set $\mathcal{V}_2$ empty. This discussion leads to the following result.

Theorem 4 *An n-cube with $n = 0, 1, 2,$ is strongly $(n + 1)$-chained. It is strongly 4-chained when $n \geq 3$. An n-cube with $n \geq 2$ is closed 4-chained.*

Adjacency matrices for chained graphs

Consider an ℓ-chained graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with initial vertex v_1 and vertex set partitioning $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$. Let n_i be the cardinality of the vertex set $\mathcal{V}_i$ for $i = 1, 2, \ldots, \ell$. Define the matrix $A_i \in \mathbb{R}^{n_i \times n_{i+1}}$ that describes the connections between the vertices in the set $\mathcal{V}_i$ and the vertices in the set $\mathcal{V}_{i+1}$ for $i = 1, 2, \ldots, \ell - 1$. Hence, the entries of A_i satisfy $[A_i]_{jk} = 1$ if there is an edge between vertex v_j in $\mathcal{V}_i$ and vertex v_k in $\mathcal{V}_{i+1}$, and $[A_i]_{jk} = 0$ otherwise. The adjacency matrix M associated with $\mathcal{G}$ is symmetric block tridiagonal with off-diagonal blocks A_i and A_i^T, and has vanishing diagonal blocks

$$M = \begin{bmatrix} O & A_1 & & & & \\ A_1^T & O & A_2 & & & \\ & A_2^T & O & A_3 & & \\ & & A_3^T & \ddots & \ddots & \\ & & & \ddots & O & A_{\ell-1} \\ & & & & A_{\ell-1}^T & O \end{bmatrix}. \tag{3.1}$$

It is known from Theorem 1 that every ℓ-chained graph is bipartite. This also can be seen by applying a suitable permutation matrix P and its transpose to the adjacency matrix M from the left and right, respectively, to obtain

$$PMP^T = \left[\begin{array}{c|c} O & B \\ \hline B^T & O \end{array}\right], \tag{3.2}$$

with $B \in \mathbb{R}^{n_o \times n_e}$, where

$$n_o = \sum_{i=1}^{\lfloor(\ell+1)/2\rfloor} n_{2i-1}, \qquad n_e = \sum_{i=1}^{\lfloor \ell/2 \rfloor} n_{2i}.$$

Here $\lfloor \alpha \rfloor$ denotes the integer part of $\alpha \geq 0$.

Example 9 We illustrate the permutation (3.2) for $\ell = 5$. In this case

$$P = \begin{bmatrix} I_{n_1} & O & O & O & O \\ O & O & I_{n_2} & O & O \\ O & O & O & O & I_{n_3} \\ O & I_{n_4} & O & O & O \\ O & O & O & I_{n_5} & O \end{bmatrix},$$

where I_k is an identity matrix of order k, and

$$PMP^T = \left[\begin{array}{ccc|ccc} & & & A_1 & & \\ & O & & A_2^T & A_3 & \\ & & & & & A_4^T \\ \hline A_1^T & A_2 & & & & \\ & A_3^T & A_4 & & O & \end{array}\right] = \left[\begin{array}{c|c} O & B \\ \hline B^T & O \end{array}\right]. \tag{3.3}$$

This shows that the graph $\mathcal{G}$ associated to the adjacency matrix M is bipartite.

The submatrix B in (3.3) exhibits a particular pattern of zero entries. This suggests the possibility of identifying the strongly chained structure of a graph, whose vertices are in a random order, by first identifying its bipartite structure, e.g., by methods described in Concas et al. (2020), Gleich: MatlabBGL—A Matlab Graph Library. https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl/, and then reordering the vertices to obtain a suitable zero pattern in the submatrix B.

The considered permutation also illustrates that it is not possible to identify a 3-chained graph. Indeed, considering the adjacency and permutation matrices

$$M = \begin{bmatrix} O & A_1 & O \\ A_1^T & O & A_2 \\ O & A_2^T & O \end{bmatrix}, \qquad P = \begin{bmatrix} I_{n_1} & O & O \\ O & O & I_{n_3} \\ O & I_{n_2} & O \end{bmatrix},$$

one obtains

$$PMP^T = \left[\begin{array}{cc|c} O & O & A_1 \\ O & O & A_2^T \\ \hline A_1^T & A_2 & O \end{array}\right] = \left[\begin{array}{c|c} O & B \\ \hline B^T & O \end{array}\right].$$

This shows that the matrix B does not have a zero pattern that would allow one to identify the node partitioning of a 3-chained graph.

If $\mathcal{G}$ is an ℓ-semi-chained graph, then the diagonal blocks of the matrices (3.1) and (3.2) may have some nonzero entries. If there are fewer nonvanishing entries in the diagonal blocks of the matrix (3.2) than in the off-diagonal blocks, then this indicates the existence of an anti-community.

Chained graphs and spanning trees

Many graphs $\mathcal{G}$ are not chained, but their spanning trees are. This section explores the possibility of using the chained structure of a spanning tree to gain insight into properties of the underlying graph.

A spanning tree for $\mathcal{G}$ is a subgraph $\mathcal{T} = \{\mathcal{V}, \mathcal{E}'\}$ that is a tree and contains all the vertices of $\mathcal{G}$; see, e.g., Estrada (2011a), Newman (2010) for further details. In general, $\mathcal{E}' \subsetneqq \mathcal{E}$; if $\mathcal{E}' = \mathcal{E}$, then $\mathcal{G}$ is a tree itself. A spanning tree $\mathcal{T}$ for $\mathcal{G}$ is not uniquely determined by $\mathcal{G}$. In particular, $\mathcal{T}$ depends on the initial vertex, the so-called root, of the tree. A spanning tree for a graph $\mathcal{G}$ with n vertices can be computed in time proportional to n.

Each spanning tree has an ℓ-chained structure: let $\mathcal{V}_1$ contain the root, v_1, of the tree, $\mathcal{V}_2$ the children of the root, and, in general, $\mathcal{V}_{i+1}$ the children of the vertices in $\mathcal{V}_i$ for $i = 1, 2, \ldots, \ell - 1$. The set $\mathcal{V}_\ell$ contains the leaves of the tree at the lowest level. This shows, in particular, that spanning trees are bipartite; cf. Theorem 1. We will use the chained structure of a spanning tree $\mathcal{T}$ for $\mathcal{G}$ to determine an approximated chained structure for $\mathcal{G}$, also in situations when $\mathcal{G}$ is not chained.

Definition 11 Let $\mathcal{T}$ be a spanning tree for the graph $\mathcal{G}$. An ℓ-chained vertex set decomposition for $\mathcal{T}$ is said to be an ℓ-chained vertex set decomposition for $\mathcal{G}$. We will refer to leaves of $\mathcal{T}$ as leaves of $\mathcal{G}$.

Let $\mathcal{D} = \mathcal{E} \setminus \mathcal{E}'$ be the set of the edges in $\mathcal{G}$ that are not in $\mathcal{T}$, and let $C(\mathcal{T})$ denote the graph obtained by adding the edges in $\mathcal{D}$ to the spanning tree $\mathcal{T}$. The graph $C(\mathcal{T})$ coincides with $\mathcal{G}$ and inherits the chain structure of $\mathcal{T}$.

If all the edges in $\mathcal{D}$ are compatible with the chain structure of the spanning tree $\mathcal{T}$, that is, if for each edge $e_k \in \mathcal{D}$, there is an index $2 \leq i \leq \ell - 1$ such that e_k connects a vertex in $\mathcal{V}_i$ to a vertex in $\mathcal{V}_{i-1}$ or $\mathcal{V}_{i+1}$, then the graph $\mathcal{G} = C(\mathcal{T})$ is chained. If an edge in $\mathcal{D}$ connects two vertices that belong to the same node set $\mathcal{V}_i$, the graph is semi-chained. Finally, if an edge in $\mathcal{D}$ connects a vertex in $\mathcal{V}_i$ to a vertex in $\mathcal{V}_{i+j}, |j| \geq 2$, then the graph $C(\mathcal{T})$ is not chained. This observation leads to the following result.

Theorem 5 *A graph is ℓ-chained (semi-chained) if at least one of its spanning trees $\mathcal{T}$ generates a graph $C(\mathcal{T})$ whose edges are compatible with the (semi-)chain structure of $\mathcal{T}$.*

Let the vertex set decomposition $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$ be determined by the chain structure of $\mathcal{G} = C(\mathcal{T})$. Recall that a graph is strongly chained if every vertex in $\mathcal{V}_i$ is connected to at least one vertex in $\mathcal{V}_{i+1}$ and to one vertex in $\mathcal{V}_{i-1}$ for $= 2, 3, \ldots, \ell - 1$. Moreover, every vertex in $\mathcal{V}_1$ (resp. $\mathcal{V}_\ell$) is required to be connected to one vertex in $\mathcal{V}_2$ (resp. $\mathcal{V}_{\ell-1}$). Whether a graph is strongly chained depends on the leaves of the graph.

Theorem 6 *Let $\mathcal{T}$ be a spanning tree of the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, and let $\mathcal{T}$ determine the chain structure $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell$. If $\mathcal{T}$ has leaves connected only to vertices in $\mathcal{V}_{\ell-1}$, then $\mathcal{G}$ is strongly chained. This also holds if $\mathcal{T}$ has leaves connected to vertices in $\mathcal{V}_1$* or in *$\mathcal{V}_2$, but not if there are leaves connected to vertices in both $\mathcal{V}_1$ and $\mathcal{V}_2$.*

Proof The vertices in $\mathcal{V}_\ell$ are leaves. If, in addition to the leaves in $\mathcal{V}_\ell$, there are leaves connected to the root, v_1, but no other leaves, then the graph is strongly chained.

This can be seen by moving the leaves connected to v_1 to a new vertex set $\mathcal{V}_0$ that precedes $\mathcal{V}_1$. This shows that the graph is strongly $(\ell+1)$-chained with chain structure $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_\ell$.

We turn to the situation when, in addition to the leaves in $\mathcal{V}_\ell$, there are also leaves connected to the vertices in $\mathcal{V}_2$. The latter leaves can be moved to $\mathcal{V}_1$, which shows that the graph is strongly ℓ-chained. □

We remark that it is easy to construct examples that illustrate that if $\mathcal{T}$ has a leaf connected to a vertex in $\mathcal{V}_i$ for some $3 \le i \le \ell - 2$, then $C(\mathcal{T})$ is not guaranteed to be strongly chained.

The above discussion leads to Algorithm 1 for determining if a graph with n vertices is (semi-)chained in $O(n^2)$ time steps. We note that the algorithm can easily be parallelized, as each iteration is independent on the others.

Algorithm 1 Determine the (semi-)chain structure of a graph.

Require: Adjacency matrix $A \in \mathbb{R}^{n\times n}$ for a graph $\mathcal{G}$.
Ensure: Spanning tree $\mathcal{T}$ and node sets $\mathcal{V}_i$, $i = 1, 2, \ldots, \ell$, that identify the chain structure of the graph. Returns $\ell = 0$ if the graph is not (semi-)chained.

```
ℓ = 0
for k = 1, 2, ..., n do
    Construct the spanning tree T^(k) starting at vertex v_k.
    Determine the chain structure V_i, i = 1, 2, ..., ℓ^(k), of T^(k).
    if ℓ^(k) > ℓ and C(T_k) is compatible with the chain structure of T_k then
        ℓ = ℓ^(k), T = T^(k)
        Store the sets V_i, i = 1, 2, ..., ℓ.
    end if
end for
```

The following example illustrates that both the partitioning of the vertex set $\mathcal{V}$ of a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and the number of partitions, ℓ, depend on the choice of the root of the spanning tree $\mathcal{T}$ as well as on the spanning tree itself.

Example 10 Consider the graph $\mathcal{G}$ with adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

This graph $\mathcal{G}$ is not chained. The graphs defined by the adjacency matrices

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.1}$$

and

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.2}$$

are spanning trees for $\mathcal{G}$. Regard first the partitioning of the tree (4.1). Starting with vertex v_1, we obtain the vertex subsets $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{v_2\}$, $\mathcal{V}_3 = \{v_3, v_5\}$, and $\mathcal{V}_4 = \{v_4\}$. Thus, $\ell = 4$. If we instead start with vertex v_2, then we get the sets $\mathcal{V}_1 = \{v_2\}$ and $\mathcal{V}_2 = \{v_1, v_3, v_5\}$, $\mathcal{V}_3 = \{v_4\}$, and $\ell = 3$.

We now turn to the spanning tree (4.2). Letting $\mathcal{V}_1 = \{v_1\}$, we obtain $\mathcal{V}_2 = \{v_2\}$, and $\mathcal{V}_3 = \{v_3, v_4, v_5\}$. Hence, $\ell = 3$. If we instead let $\mathcal{V}_1 = \{v_2\}$, then $\mathcal{V}_2 = \{v_1, v_3, v_4, v_5\}$ and $\ell = 2$.

In what follows, we will need the notion of tree branches.

Definition 12 A *branch* for a tree $\mathcal{T}$ is a sequence of vertices starting at the tree root and ending at a leaf. The *length* of a branch is the number of vertices in the branch. A *longest branch* is a branch with maximal length.

A recursive procedure for determining all the longest branches of a tree is presented by Algorithm 2.

Algorithm 2 Determine all the longest branches of a tree.

Require: Tree $\mathcal{T}$ and starting vertex v. At the first call, the starting vertex is the tree root.
Ensure: Length λ of a longest branch and lists L_i, $i = 1, \ldots, m$, of the vertices in each longest branch.

$L_1 = \varnothing$, $m = 1$, $\lambda = 0$
for $c \in \{$children of vertex $v\}$ **do**
  Call Algorithm 2 recursively, passing the same tree $\mathcal{T}$ and the vertex c, and receiving the maximal length $\overline{\lambda}$ and the lists of vertices $\overline{L}_i$, $i = 1, \ldots, \overline{m}$, between c and a leaf.
  if $\overline{\lambda} > \lambda$ **then**
    $L_i = \overline{L}_i$, $i = 1, 2, \ldots, \overline{m}$
    $\lambda = \overline{\lambda}$, $m = \overline{m}$
  else if $\overline{\lambda} = \lambda$ **then**
    $L_{m+i} = \overline{L}_i$, $i = 1, 2, \ldots, \overline{m}$
    $m = m + \overline{m}$
  end if
end for
for $i = 1, \ldots, m$ **do**
  Add vertex v as first element of L_i.
end for
$\lambda = \lambda + 1$

It is natural to seek the root of a spanning tree with the deepest chain structure on a long branch of any of the spanning trees of the graph. A heuristic approach for doing this is described by Algorithm 3.

Algorithm 3 Determine an approximation of the chain structure of a graph.

Require: Adjacency matrix $A \in \mathbb{R}^{n\times n}$ for a graph $\mathcal{G}$ and starting vertex v.
Ensure: Spanning tree $\mathcal{T}$ and node sets $\mathcal{V}_i$, $i = 1, 2, \ldots, \ell$, that approximate the chain structure of the graph.
1: Determine a spanning tree $\mathcal{T}$ for $\mathcal{G}$ starting at vertex v. Let the vertices of $\mathcal{T}$ have a chain structure of length ℓ.
2: **repeat**
3: Determine the longest branches for $\mathcal{T}$ (Algorithm 2).
4: $\overline{\ell} = 0$
5: **for** each longest branch **do**
6: Let c be the last vertex of the branch.
7: Determine a spanning tree $\mathcal{T}^{(1)}$ for $\mathcal{G}$ starting at c, with chain structure of length ℓ_1.
8: **if** $\ell_1 > \overline{\ell}$ **then**
9: $\overline{\ell} = \ell_1$, $\overline{\mathcal{T}} = \mathcal{T}^{(1)}$
10: **end if**
11: **end for**
12: **if** $\overline{\ell} > \ell$ **then**
13: $\ell = \overline{\ell}$, $\mathcal{T} = \overline{\mathcal{T}}$
14: **end if**
15: **until** $\overline{\ell} < \ell$
16: Determine the chain structure $\mathcal{V}_i$, $i = 1, 2, \ldots, \ell$, of the spanning tree $\mathcal{T}$.

Example 11 We have already seen in Example 9 that the chain structure of an undirected simple unweighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ cannot be uniquely determined from the sets $\mathcal{E}$ and $\mathcal{V}$. Here we provide another illustration using spanning trees. The pictures in Fig. 8 show the same graph. The graph in Fig. 8b is obtained by determining a spanning tree for the graph in Fig. 8a, starting from vertex v_9, and then constructing $C(\mathcal{T})$ by adding the missing arcs. Both graphs are strongly 4-chained.

The adjacency matrices for the graphs of Fig. 8 are permutations of each other. The blocks in matrix (3.1) for the two graphs are

$$A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad A_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix},$$

and

$$A_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \qquad A_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

respectively. We notice that the graph $\mathcal{G}$ is in fact 5-chained. This can be seen, for example, by constructing the spanning tree starting from vertex v_2.

Example 12 Consider the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ displayed in Fig. 9. The adjacency matrix for $\mathcal{G}$ is given by

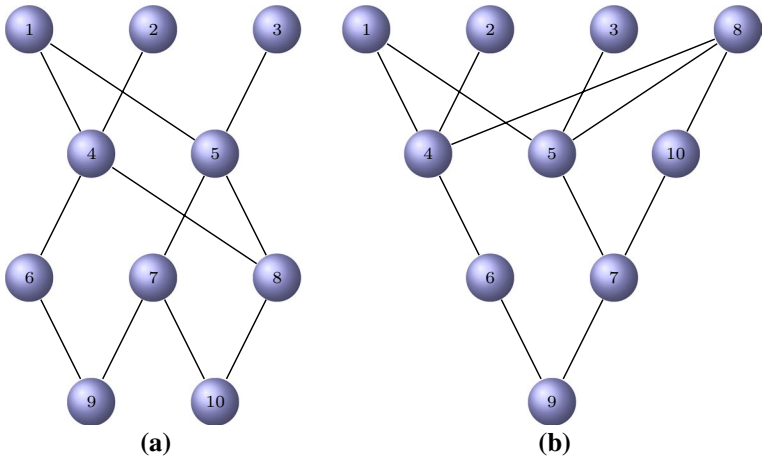

Fig. 8 Example 11: the two figures (**a**) and (**b**) show the same graph. The layouts display different chain structures

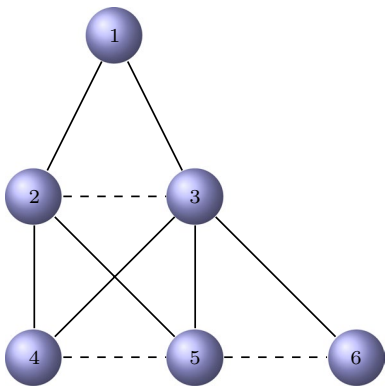

Fig. 9 Example 12: a graph $\mathcal{G}$ without chain structure

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

By removing edges denoted by dashed lines between the vertices v_2 and v_3, v_4 and v_5, as well as between the vertices v_5 and v_6, we obtain the graph $\mathcal{G}'$ with the associated adjacency matrix

$$A' = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

This matrix is of the form (3.1). It follows that the graph $\mathcal{G}'$ is chained. The vertex set for $\mathcal{G}'$ can be expressed as $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$ with $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{v_2, v_3\}$, and $\mathcal{V}_3 = \{v_4, v_5, v_6\}$.

In Example 12, the graph $\mathcal{G}$ is approximated by a 3-chained graph. We conclude that $\mathcal{G}$ is a 3-semi-chained graph with ρ-anti-communities $\{v_1\}$, $\{v_2, v_3\}$, and $\{v_4, v_5, v_6\}$.

Position centrality and some applications

There are many ways to measure the importance of a vertex in a graph; see, e.g., Estrada (2011a), Estrada and Higham (2010), and Newman (2010). These measures often are referred to as centrality measures. In this section, we are interested in determining a most "centrally located" vertex in a graph. We call such a vertex a *center vertex*. For this purpose, we introduce a new centrality measure, which belongs to the class of path-based centrality measures. This class includes closeness and betweenness centralities. In fact, determining the most centrally located nodes is an extension of closeness centrality.

Applications of the detection of a center vertex include:

- Information dissemination: we are interested in determining a vertex (the center vertex) such that information from it can travel to all other vertices in the least amount of time. Here we assume that the travel time is proportional to the number of edges that have to be traversed from a center vertex to the receiving vertices. In the context of social network theory, the importance of a node for spreading information is often associated with the betweenness centrality which assumes that the communication in a network takes place through the shortest paths passing through this node. However, it has been shown that in some circumstances the best spreaders do not correspond to the most highly connected or central nodes. They are often located within the core of the network, identified by using k-shell decomposition analysis; see Kitsak et al. (2010) and the references therein.
- City planning: let the edges of a graph represent the streets of a town. It would be reasonable to allocate a fire station, police station, bus terminal, or hospital at a center vertex of the graph.

Definition 13 Let $\mathcal{T}$ be a spanning tree of the graph $\mathcal{G}$, starting at a vertex v, and let $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ the ℓ-chained structure determined by the tree. The *position centrality* P_p of v in the graph, where $p \in \mathbb{R}$, is defined by

$$P_p(v) = \sum_{k=1}^{\ell-1} k(\#\mathcal{V}_{k+1})^p,$$

where $(\#\mathcal{V}_i)$ denotes the cardinality of the set $\mathcal{V}_i$. We refer to a vertex v_c with the smallest position centrality as a *p-center vertex*.

When $p = 1$, the position centrality is the sum of the lengths of the paths from v to all the other vertices, so its minimization is equivalent to the maximization of the closeness centrality

$$C(v_i) = n\left(\sum_{j\neq i} d(v_i, v_j)\right)^{-1},$$

where $d(v_i, v_j)$ is the distance between v_i and v_j. For $p = -1$, position centrality is equivalent to harmonic mean distance; see Newman 2010, Eq. (7.30). We emphasize that position centrality depends on the chained structure, which contains important information about the network being analyzed.

Using positive p values different from 1 may help select central nodes with different features. A value larger than 1 further penalizes the presence of a large number of long walks, and selects a relatively long ℓ-chained structure, generally with maximal chain length, with sets $\mathcal{V}_k$ containing a small number of vertices. This feature has the interesting side effect of reducing the bandwidth of the adjacency matrix corresponding to the node ordering induced by the chain structure.

On the contrary, $p \in (0, 1)$ reduces the difference between the scores of long and small walks, leading to a shorter chain structure, composed by large node sets.

Example 13 Consider the graph $\mathcal{G}$ displayed in Fig. 10. The position centrality of a vertex in the graph can be computed by using the chained graph starting from this vertex.

To compute the position centrality of vertex v_2, we consider the spanning tree rooted at vertex v_2. We obtain $\mathcal{V}_1 = \{v_2\}$, $\mathcal{V}_2 = \{v_3\}$, $\mathcal{V}_3 = \{v_4\}$, $\mathcal{V}_4 = \{v_1, v_5, v_8\}$, and $\mathcal{V}_5 = \{v_6, v_7, v_9, v_{10}\}$. Since the graph is unweighted, the length between a vertex in $\mathcal{V}_i$ to a vertex in $\mathcal{V}_{i+1}$ is one. It follows that the 1-position centrality of vertex v_2 is

$$P_1(v_2) = 1 \cdot 1 + 2 \cdot 1 + 3 \cdot 3 + 4 \cdot 4 = 28,$$

while $P_5(v_2) = 4828$ and $P_{1/5}(v_2) = 12.02$.

We turn to the position centrality of vertex v_4. Letting $\mathcal{V}_1 = \{v_4\}$, we obtain $\mathcal{V}_2 = \{v_1, v_3, v_5, v_8\}$, and $\mathcal{V}_3 = \{v_2, v_6, v_7, v_9, v_{10}\}$. We have

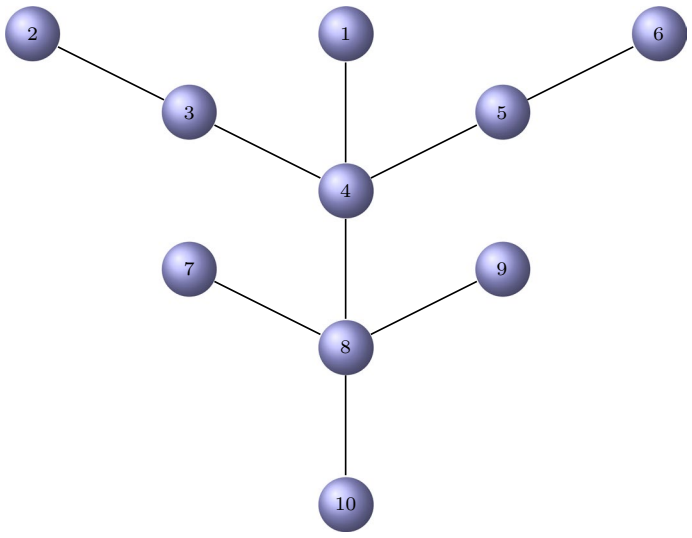

Fig. 10 A spanning tree of $\mathcal{G}$

$$P_1(v_4) = 1 \cdot 4 + 2 \cdot 5 = 14, \qquad P_5(v_4) = 7274, \qquad P_{1/5}(v_4) = 4.08.$$

Similarly, we can compute the position centrality for all the other vertices of the spanning tree. Vertex v_4 has the smallest position centrality score for $p = \frac{1}{5}$ and $p = 1$, while the center vertices for $p = 5$ are v_7, v_9, and v_{10}.

In Example 13, the center vertices lie on one of the longest branches of the spanning tree. It is reasonable to assume that this is typical for many trees. Hence, to approximate the center vertex, instead of evaluating the position centrality for all the vertices, it is more efficient to compute the position centrality for the vertices on the longest branches only. This suggests the iterative procedure described by Algorithm 4. The same approach can also be used for determining the approximate top k p-center nodes, as described by Algorithm 5.

Algorithm 4 Determine an approximate p-center node of a graph.

Require: Adjacency matrix $A \in \mathbb{R}^{n\times n}$ for a graph $\mathcal{G}$, exponent $p \in \mathbb{R}$, starting vertex v, set of nodes $\mathcal{N}$ to be discarded in the search.
Ensure: Approximate p-center node v_c and minimal position centrality $P_c = P_p(v_c)$.

$v_c = v$, $P_c = \infty$
repeat
  done=TRUE
  Determine a spanning tree $\mathcal{T}$ for $\mathcal{G}$ starting at v_c.
  Determine the longest branches for $\mathcal{T}$ (Algorithm 2).
  for each longest branch **do**
    for each node v in the branch **do**
      Determine a spanning tree $\mathcal{T}$ for $\mathcal{G}$ starting at v.
      Compute p-position centrality $P = P_p(v)$.
      if $v \notin \mathcal{N}$ **and** $P < P_c$ **then**
        $v_c = v$, $P_c = P$
        done=FALSE
      end if
    end for
  end for
until done

Algorithm 5 Approximate top k p-center nodes of a graph.

Require: Adjacency matrix $A \in \mathbb{R}^{n\times n}$ for a graph $\mathcal{G}$, exponent $p \in \mathbb{R}$, starting vertex v, integer k.
Ensure: Approximate top p-center nodes v_i and minimal position centralities $P_i = P_p(v_i)$, $i = 1, \ldots, k$.

$\mathcal{N} = \emptyset$
for $i = 1, \ldots, k$ **do**
  call Algorithm 4 with input (A,p,v,$\mathcal{N}$) and output (v_c,P_c)
  $v_i = v_c$, $P_i = P_c$, $\mathcal{N} = \mathcal{N} \cup \{v_i\}$
  $v = v_i$
end for

It may be attractive to identify a tree with the shortest longest branch and then determine a candidate for the central node on a longest branch. We outline this approach, but hasten to add that it is only a heuristic, because a center vertex is not guaranteed to lie on a longest branch.

The following example illustrates that the center vertex depends on the spanning tree.

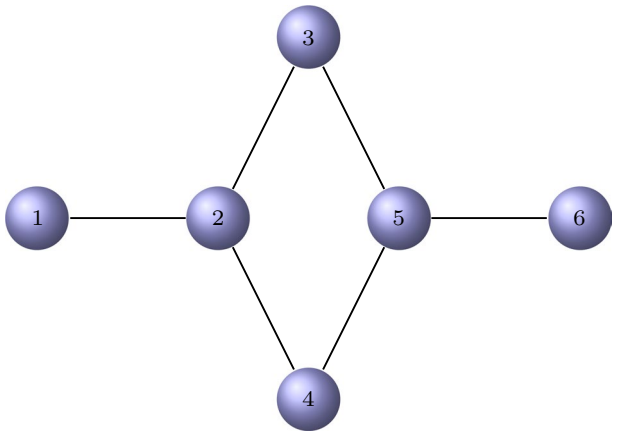

Fig. 11 An undirected and unweighted graph $\mathcal{G}$

Example 14 Consider the undirected and unweighted graph $\mathcal{G}$ displayed in Fig. 11. Two shortest-path trees rooted at vertex v_1 are shown in Fig. 12. The center vertices of the shortest-path tree in Fig. 12a are the vertices v_2 and v_3. Let v_2 be the starting vertex. Then $\mathcal{V}_1 = \{v_2\}$, $\mathcal{V}_2 = \{v_1, v_3, v_4\}$, $\mathcal{V}_3 = \{v_5\}$, and $\mathcal{V}_4 = \{v_6\}$. The position centrality is $P_1(v_2) = 8$.

Let, instead, v_3 be the initial vertex. Then we obtain $\mathcal{V}_1 = \{v_3\}$, $\mathcal{V}_2 = \{v_2, v_5\}$, and $\mathcal{V}_3 = \{v_1, v_4, v_6\}$. The position centrality is $P_1(v_3) = 8$. Both vertices v_2 and v_3 have the smallest 1-position centrality of the vertices in the graph.

Similarly, we find that the center vertex of the shortest-path tree in Fig. 12b are the vertices v_2 and v_4. The position centrality of both these vertices is 8, which is the smallest position centrality of all the vertices.

Numerical experiments

The algorithms discussed in the previous sections were implemented in the MATLAB programming language. Large tests were executed on a Linux virtual machine running on a Cisco UCSB-B480-M5 server based on Intel Xeon Gold 6136 processors. The virtual machine is equipped with 32 cores and 128 Gbyte RAM.

We first illustrate the use of the algorithms on a small graph, namely, the one described in Example 11 and illustrated in Fig. 8. The graphs in Fig. 13 display spanning trees starting at vertex v_2 and at vertex v_6 of the graph $\mathcal{G}$. The dashed lines denote edges that must be added to the tree $\mathcal{T}$ to obtain the graph $C(\mathcal{T})$: it is seen that the added edges are compatible with the chain structure of $\mathcal{T}$, and the chain length is 5. Applying Algorithm 1 confirms that this length is maximal. Hence, $\mathcal{G}$ is 5-chained and all the sets $\mathcal{V}_i$ determined by $\mathcal{T}$ are 0-anti-communities.

By computing the 1-position centrality of all nodes in $\mathcal{G}$, one finds that the corresponding central nodes are v_4, v_5, and v_8. The nodes with the smallest 5-position centrality are v_2 and v_3. Both of them are roots of a tree with maximal chain length; Fig. 13a illustrates this for v_2. We applied Algorithm 3 for approximating the chain structure length

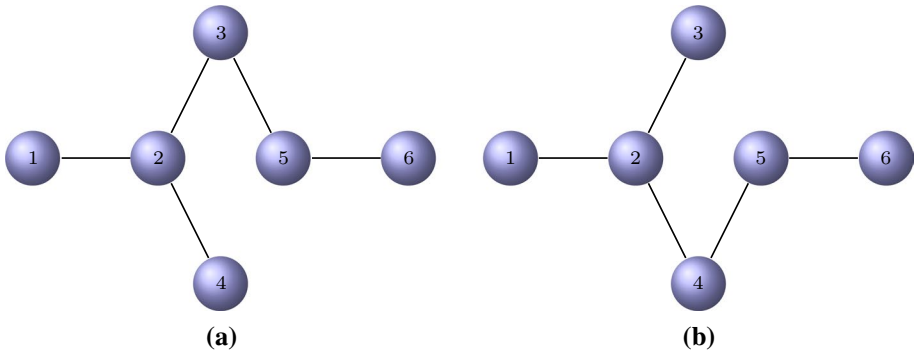

Fig. 12 Two shortest-path trees of node v_1; the center nodes of the (**a**) tree are v_2 and v_3 and the center nodes of the (**b**) tree are v_2 and v_4

of the graph, and Algorithm 4 to approximate its center vertex for $p = 1$. To investigate the global performance of these methods, the algorithms were applied starting from each vertex of the network; the results are displayed in Fig. 14. It can be observed that the chain structure length was not detected for each starting vertex, but the computed approximations are accurate. On the contrary, Algorithm 4 always determined one of the three correct center vertices.

We now analyze the structure of three medium-sized networks, deriving from well known data sets:

- autobahn (1168 nodes, 2486 edges) describes the German highway system network, where the vertices are locations and the edges highways connecting them. It is available at Biological Networks Data Sets of Newcastle University, http://www.biological-networks.org/.
- yeast (2361 nodes, 13828 edges) represents the protein interaction network for yeast: the interacting proteins are connected by edges (Jeong et al. 2001; Sun et al. 2003). It is available at Batagelj and Mrvar (2006).
- geom (7343 nodes, 23796 edges) was extracted from the computational geometry database collaboration network *geombib* by B. Jones (version 2002). Nodes represent authors; the value of the entry (i, j) of the adjacency matrix is the number of papers coauthored by authors i and j. The data set is available at Batagelj and Mrvar (2006). We will use the associated unweighted network.

The autobahn network is connected, but the networks yeast and geom are not. We therefore considered the largest connected component, of 2224 and 3621 vertices, of the latter networks.

As expected, Algorithm 1 reveals all three networks (autobahn, yeast, and geom) to be semi-chained. The maximal chain length of a spanning tree for each of the three graphs is $\ell = 63$, 12, and 15, respectively. The structure of a maximal chain length spanning tree for autobahn (starting at vertex 116) and for geom (starting at vertex 207) are displayed in Fig. 15a,b. The additional edges which define $C(\mathcal{T})$, represented by dashed lines, are compatible with the semi-chain structure of each tree $\mathcal{T}$.

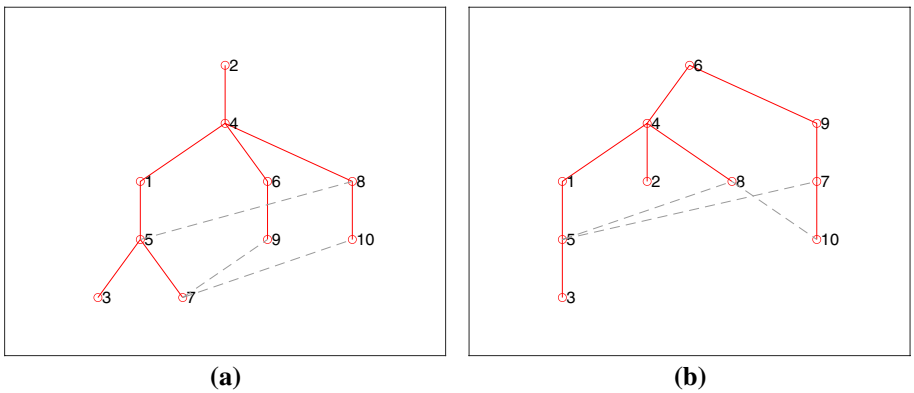

Fig. 13 Example 11: two spanning trees (**a**) and (**b**) with maximal chain structure length. The dashed edges are added to each tree $\mathcal{T}$ to obtain the graph $C(\mathcal{T})$

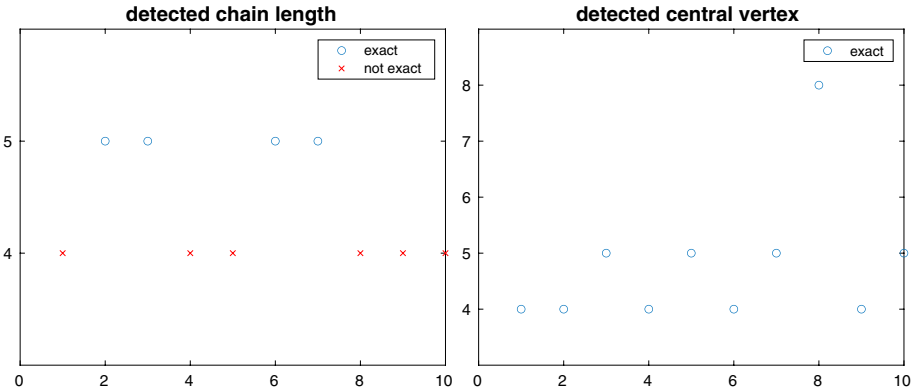

Fig. 14 Example 11: chain length (left) and central vertex (right) detected by Algorithms 3 and 4, respectively, starting from node $v_k, k = 1, \ldots, 10$

Figure 16 displays the adjacency matrix for the autobahn network after applying two particular orderings of the nodes, deriving from the spanning tree displayed in Fig. 15a. By listing the vertices in the same order as they appear in the node sets $\mathcal{V}_i$, $i = 1, 2, \ldots, 63$, we obtain the spy plot in Fig. 16a. In a spy plot, each nonzero entry of a matrix is represented as a dot, and the quantity "nz" on the x-axis denotes the number of nonzeros. The graph exhibits the form reported in (3.1), and shows that this ordering reduces the bandwidth of the adjacency matrix, especially in the presence of a long chain structure. By applying the vertex ordering proposed in Example 9, that is, by listing first the nodes in the sets $\mathcal{V}_i$ with an odd index i and then those with an even index, the adjacency matrix of $\mathcal{G}$ takes the sparsity structure shown in the spy plot in Fig. 16b. It coincides with the form displayed in Eq. (3.3), and shows that the graph is almost bipartite.

In view of the sparsity of the diagonal blocks, this spy plot signals the presence of anti-communities in the network. Indeed, by computing the anti-community score of the node sets $\mathcal{V}_i$ resulting from the application of Algorithm 1 with starting vertex 116, represented in the graph if Fig. 15a, we find that the autobahn network has 48

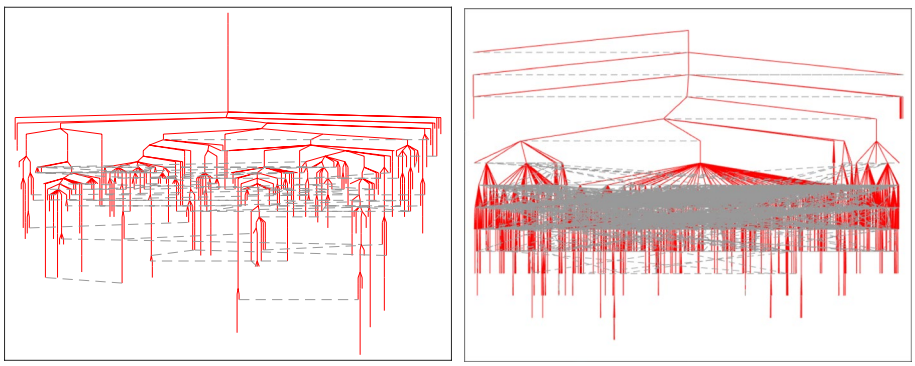

Fig. 15 Maximal chain length spanning trees for the autobahn network (**a**) and the geom network (**b**), starting at vertex 116 and 207, respectively. Each tree $\mathcal{T}$ is completed by the additional edges that define $C(\mathcal{T})$

0-anti-communities (23 including just one vertex) and 15 anti-communities, with maximal score $\rho = 0.07$.

The spy plots for the yeast and geom networks corresponding to the first ordering are reported in Fig. 17a, b, respectively. They clearly show that in both networks there are groups of vertices which do not interact, and that there are no anti-communities.

Figures 18, 19, and 20 depict the results obtained by running Algorithm 3 to approximate the chain structure length, and Algorithm 4 to determine an approximation of the center vertex for $p = 1$. The algorithms are initialized using each node in the network as a starting vertex, in order to investigate their best and worst performances. In real applications, the algorithms should be initialized with a random starting vertex.

The graph in the left panel of each figure shows the maximal chain length We see that for the autobahn network about half of the tests determine the correct value 63, and the other runs obtain the close value 61. For the other two networks, Algorithm 3 is very accurate, missing the correct chain length by one unit in just a few cases.

The graphs (b) in Figs. 18, 19, and 20, report the relative errors in the approximations of the 1-position centrality by Algorithm 4 when compared to the exact result. The (exact) minimal position centrality was computed by Algorithm 1, which identified the following center vertices for the three test networks:

- $v_c = 698$, with $P_1(v_c) = 13954$, for autobahn;
- $v_c = 518$, with $P_1(v_c) = 6914$, for yeast;
- $v_c = 20$, with $P_1(v_c) = 11736$, for geom.

We see that the position centrality was accurately estimated in most cases. The relative error for autobahn exceeds 15% only for a small number of starting vertices, while it is always below 16% for yeast, and 8% for geom.

To illustrate the differences between the center vertex individuated by the position centrality, as defined in Definition 13, and other centrality measures, we consider a real-world data set concerning air transport management.

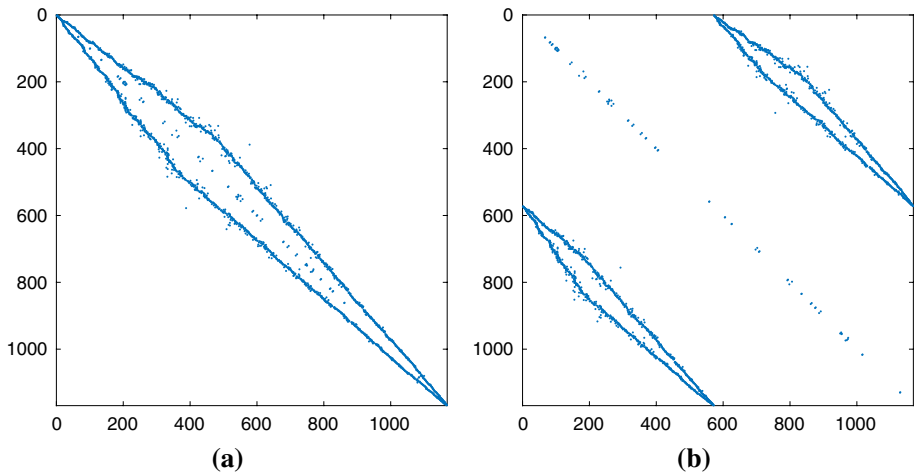

Fig. 16 Spy plots (**a**) and (**b**) for the autobahn network (1168 vertices) after applying two particular node orderings induced by a maximal chain length spanning tree

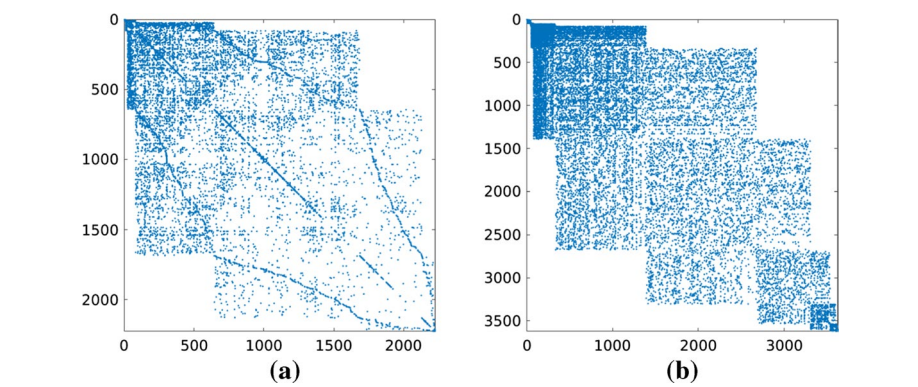

Fig. 17 Spy plots for the yeast (**a**) and geom (**b**) networks, after applying the node ordering induced by the maximal length spanning trees starting at node 569 and 207, respectively

The network is determined by domestic airlines between 164 cities in the 48 contiguous states in the US in 2019. It is reported by the Bureau of Transportation Statistics of the US Department of Transportation (U.S. Department of Transportation). By using cities as nodes and airlines between cities as edges, we construct an undirected and unweighted network with adjacency matrix $A \in \mathbb{R}^{164\times 164}$. We determine the center vertex of this network by three different methods: subgraph centrality, eigenvector centrality based on the computation of the Perron vector, and the position centrality described in this paper.

The idea behind the subgraph centrality, introduced by Estrada and Rodriguez-Velazquez (2005), consists of characterizing the importance of a node in all subgraphs in a network by considering its participation in all closed walks starting (and ending) at it. More precisely, the subgraph centrality for the node i, in a network described by the adjacency matrix A, is the i-th diagonal entry of the exponential of A, that is, it is given by

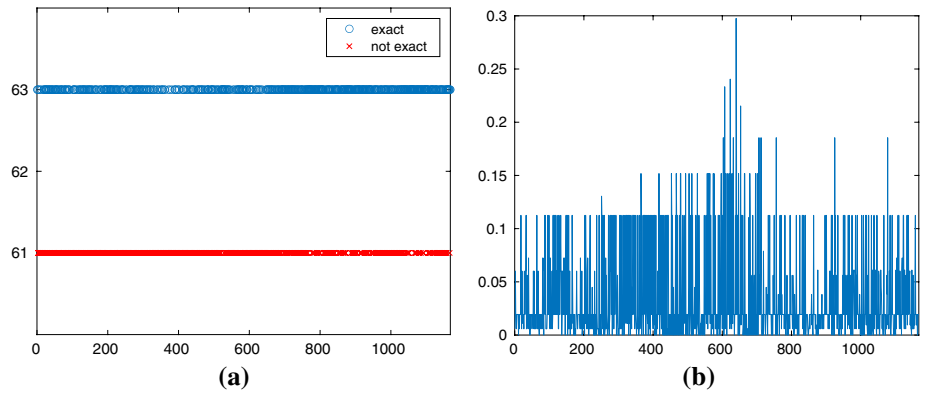

Fig. 18 autobahn network (1168 vertices): chain structure length detected by Algorithm 3 (**a**), and relative error in the minimal position centrality detected by Algorithm 4 (**b**), starting from each node in the network

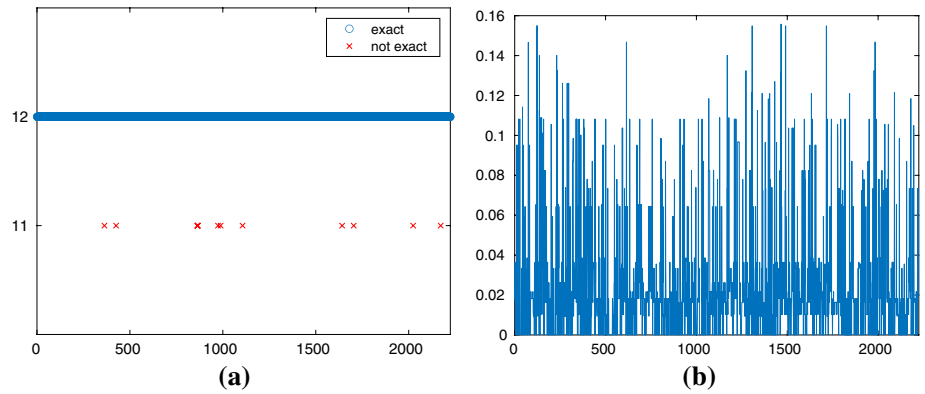

Fig. 19 yeast network (2224 vertices): chain structure length detected by Algorithm 3 (**a**), and relative error in the minimal 1-position centrality detected by Algorithm 4 (**b**), starting from each node in the network

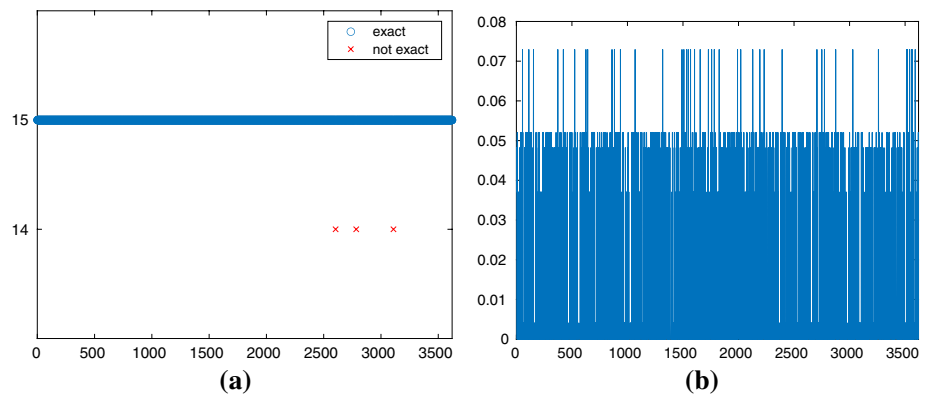

Fig. 20 geom network (3621 vertices): chain structure length detected by Algorithm 3 (**a**), and relative error in the minimal 1-position centrality detected by Algorithm 4 (**b**), starting from each node in the network

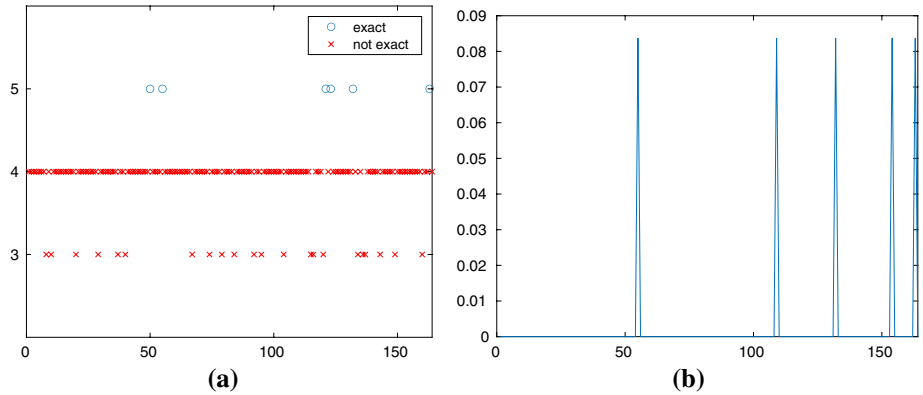

Fig. 21 domestic airlines network (164 vertices): chain structure length detected by Algorithm 3 (**a**), and relative error in the minimal position centrality detected by Algorithm 4 (**b**), starting from each node in the network

Table 1 Central nodes for the networks considered in the paper, according to different centrality measures: degree (deg), betweenness centrality (btwin), PageRank (prank), subgraph centrality (sgcen), eigenvalue centrality (eig), and position centrality P_p, for $p = 1, 5, \frac{1}{5}$

network	deg	btwin	prank	sgcen	eig	P_1	P_5	$P_{1/5}$
autobahn	693	219	693	693	219	698	565	693
yeast	535	138	1338	442	427	518	258	273
geom	20	956	2967	79	956	20	655	43
airlines	79	79	79	104	104	79	84	79

$$\mathbf{e}_i^T \exp(A)\mathbf{e}_i,$$

where $\mathbf{e}_i$ denotes the i-th column of the identity matrix.

The eigenvector centrality was introduced by Bonacich as a measure of the influence a node has in a network (Bonacich 1987). The i-th entry of the principal eigenvector $\mathbf{q}_1$ of the adjacency matrix A of a graph is known as the eigenvector centrality of node i. Typically, $\mathbf{q}_1$ is normalized and, by the Perron-Frobenius theorem, it can be chosen so that all of its components are nonnegative.

The node identified by both the subgraph centrality and the eigenvector centrality is New York City, one of the largest commercial centers of the US. The center vertex determined by the position centrality is located at Las Vegas. Indeed, given its position and connections, it is easy to travel from Las Vegas to any other town.

For completeness, we report in Fig. 21 the results obtained by running Algorithm 3 to approximate the chain structure length, and Algorithm 4 to determine an approximation of the center vertex. We see that in this case Algorithm 3 is not very accurate, but the network is too small for the experiment to be of significance. On the contrary, the center node is determined with high accuracy.

Finally, Table 1 reports the central nodes for the networks autobahn, yeast, geom, and airlines, according to various centrality indices. Position centrality, with $p = 1, 5, \frac{1}{5}$, is compared to the degree of a node, betweenness centrality (Newman 2010), PageRank

(Page et al. 1999), subgraph centrality, and eigenvector centrality. We note that for the autobahn network $P_{1/5}$, the degree, the PageRank, and the subgraph centrality agree in the determination of the center node. For airlines most of the methods agree, with the exception of subgraph and eigenvector centrality, which identify the same node, and P_5. In most cases, different indices select different center vertices, illustrating that they take different features of the network into consideration. We emphasize the fact that the position centrality associates to a center vertex a hierarchy of the nodes, namely, the chain structure, which contains additional strong information about the topology of the network.

Conclusion

The notions of chained and semi-chained graphs, as well as of center nodes, are introduced. Their properties and use to analyze networks are discussed, and algorithms for approximating both the chained structure of a graph and its center nodes are presented.

Acknowledgements
The authors would like to thank the referees for comments that lead to improvement of the presentation. The authors gratefully ackowledge the financial support.

Authors' contributions
All authors collaborated in designing the research and the methodology, performing the analyses, and implementing the algorithms. All authors edited the paper and read and approved the final manuscript.

Funding
A.C. and G.R. were supported by the Fondazione di Sardegna 2017 research project "Algorithms for Approximation with Applications (Acube)"; the INdAM-GNCS research project "Tecniche numeriche per l'analisi delle reti complesse e lo studio dei problemi inversi"; and the Regione Autonoma della Sardegna research project "Algorithms and Models for Imaging Science (AMIS)" [RASSR57257]. L.R. was supported by NSF grant DMS-1720259.

Availability of data and materials
The datasets generated and analyzed during the current study are available from the corresponding author upon request.

Declarations

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

Author details
[1]Department of Mathematics and Computer Science, University of Cagliari, via Ospedale 72, 09124 Cagliari, Italy.
[2]Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA.

Received: 3 January 2021 Accepted: 3 May 2021
Published online: 01 June 2021

References

Asratian AS, Denley TMJ, Häggkvist R (1998) Bipartite graphs and their applications. Cambridge University Press, Cambridge
Bapat RB (2014) Graphs and matrices, 2nd edn. Springer, London
Batagelj V, Mrvar A (2006) Pajek data sets. http://vlado.fmf.uni-lj.si/pub/networks/data/
Biological Networks Data Sets of Newcastle University. http://www.biological-networks.org/
Bonacich P (1987) Power and centrality: a family of measures. Am J Sociol 92:1170–1182
Bondy JA, Murty USR (1976) Graph theory with applications. MacMillan, London
Borgatti SP (2005) Centrality and network flow. Soc Netw 27:55–71
Chen L, Yu Q, Chen B (2014) Anti-modularity and anti-community detecting in complex networks. Inf Sci 275:293–313
Concas A, Noschese S, Reichel L, Rodriguez G (2020) A spectral method for bipartizing a network and detecting a large anti-community. J Comput Appl Math 373, Art. 112306
Delicious. http://www.delicious.com
Estrada E (2011a) The structure of complex networks: theory and applications. Oxford University Press, Oxford
Estrada E (2011b) Community detection based on network communicability. Chaos 21, Art. 016103
Estrada E, Higham DJ (2010) Network properties revealed through matrix functions. SIAM Rev 52:696–714

Estrada E, Gómez-Gardeñes J (2016) Network bipartivity and the transportation efficiency of European passenger airlines. Physica D 323–324:57–63
Estrada E, Knight P (2015) A first course in network theory. Oxford University Press, Oxford
Estrada E, Rodriguez-Velazquez JA (2005) Subgraph centrality in complex networks. Phys Rev 71, Art. 056103
Fasino D, Tudisco F (2017) A modularity based spectral method for simultaneous community and anti-community detection. Linear Algebra Appl 542:605–623
Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
Gleich D. MatlabBGL—A Matlab Graph Library. https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl/
Ikematsu K, Murata T (2013) A fast method for detecting communities from tripartite networks. In: Jatowt A et al (eds) Social informatics. SocInfo 2013. Lecture Notes in Computer Science, vol 8238. Springer, Cham, pp 192–205
Jensen TR, Toft B (1995) Graph coloring problems. Wiley, New York
Jeong H, Mason S, Barabási A-L, Oltvai ZN (2001) Lethality and centrality of protein networks. Nature 411:41–42
Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, Makse HA (2010) Identification of influential spreaders in complex networks. Nat Phys 11:888–893
König D (1916) Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Math Ann 77:453–465
Newman MEJ (2010) Networks: an introduction. Oxford University Press, Oxford
Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: bringing order to the web, Technical Report, Stanford InfoLab
Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76, Art. 036106
Sun S, Ling L, Zhang N, Li G, Chen R (2003) Topological structure analysis of the protein-protein interaction network in budding yeast. Nucleic Acids Res 31:2443–2450
U.S. Department of Transportation. Consumer Airfare Report: Table 1 Top 1,000 Contiguous State City-Pair Markets

Publisher's Note